Chapter 2

Literature Review

The rapid growth of internet technologies has unleashed a wave of innovations that change the way business is carried on. Companies are moving their main operations to the web for more automation, efficient business processes and global visibility [7] [73]. This chapter constitutes the technical foundation of this thesis. Therefore, a wide range of diverse topics and their relations are covered and each individual topic contributes to the overall motivation of this work towards a comprehensive approach for QoS based automatic costing of Web services. Also this chapter presents the related works conducted in the area of selection and evaluation of performance of a Web service.

2.1 Basic Technologies

In particular, Web services are introduced as an essential realization of SOA, which is most commonly used in today’s internet applications. Web services are powered by eXtensible Markup Language (XML) and three other core technologies: WSDL, SOAP, and UDDI. These basic platform elements are outlined in this section. Finally, the WSLA is discussed in detail.

2.1.1 Service Oriented Architecture (SOA)

It is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains [60]. In general, entities (people and organizations) create capabilities to solve or support a solution for the problems they have faced in the course of their business. SOA is an architectural paradigm and
discipline that may be used to build infrastructures enabling those with needs and those with capabilities to interact via services across disparate domains of technology and ownership [26].

2.1.2 Web Services

Web services are self-contained and self-describing computational web components designed to support machine-to-machine interaction by programmatic web method calls [106]. Web service represents the result of the effort to address the challenges faced by to Enterprise Application Integration (EAI) or automated business integration, which requires interoperability between a variety of IT resources and applications in an enterprise heterogeneous IT environment. Web services paradigm has emerged as a powerful mechanism for integration, as it combined the best aspects of both component-based development and the web. Web services are light-weight, loosely coupled, platform and language independent components [29].

A variety of definitions about Web services are given by different industry leaders, research groups, and Web service consortia.

World Wide Consortium (W3C) consortium defines a Web service as,

“A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using Hyper Text Transfer Protocol(HTTP) with an XML serialization in conjunction with other Web-related standards” [25].
International Business Machine (IBM) defines Web services as,

“Self-describing, self-contained, modular applications that can be mixed and matched with other Web services to create innovative products, processes, and value chains. Web services are Internet applications that fulfill a specific task or a set of tasks that work with many other Web services in a manner to carry out their part of a complex workflow or a business transaction” [44].

According to Microsoft,

“A Web service is a unit of application logic providing data and services to other applications. Applications access Web services via ubiquitous Web protocols and data formats, such as HTTP, XML, and SOAP, with no need to worry about how each Web service is implemented”. [66].

Hewlett Packard (HP) defines Web services as,

“Modular and reusable software components that are created by wrapping a business application inside a Web service interface. Web services communicate directly with other Web services via standards-based technologies”. [103]

SUN perceives a Web service as an

“Application functionality made available on the World Wide Web. A Web service consists of a network- accessible service, plus a formal description of how to connect to and use the service”. [101].

2.1.3 Web service Architecture

The Web service architecture contains three main components: Web service provider, Web service consumer, and the Web service registry. A Web service provider is
an application which implements the Web service. It implements the business logic, and exposes this business logic through well-defined interfaces. The service provider is in charge of the administration of services within its domain [25] [102]. Once a service provider creates a Web service and its service definition, the service provider then publishes the service within a service registry based on a standard called the UDDI. The implementation of a Web service includes the following steps as shown in Figure 2.1.

**Figure 2.1: Components of a Web service**

1. A Web service needs to be created, and its interfaces and invocation methods must be defined.
2. A Web service needs to be published to one or more intranet or internet repositories for potential users to locate.
3. A Web service needs to be located to be invoked by potential users.
4. A Web service needs to be invoked to be of any benefit.
5. A Web service may need to be unpublished when it is no longer available or needed.
2.1.4 Basic Activities of Web services

The internet is revolutionizing business by providing an affordable and efficient way to link companies with their partners as well as customers. The technical challenges brought forward by the Web service paradigm are related to the Web service life-cycle, which includes a number of activities [7] [12]. It is obvious that, some of these activities will take place at the requester’s site, while other takes place at the broker’s or provider’s site. The first activity is the Web service creation that takes place either by building the service from scratch or composing it by using already existing Web service [9]. The second activity is the Web service description followed by Web service publishing in a registry. The published Web service may also need to be updated or unpublished during its lifetime. Discovery of a Web service can be facilitated by the means of a broker that is expected to support requirements analysis and description of user’s needs, matching of user needs to offer Web service, Web service composition, negotiation and bidding. After a Web service has been discovered and has been decided that it matches the user’s needs, and before it can be used, a number of activities related to contract are usually needed to take place in order to govern the business transaction. Finally, a service can be unpublished if it is no longer available or needed.

Description

According to IBM, the service description describes the Web service in such a way as to minimize the amount of shared knowledge and customized programming that is needed to ensure communication between the service provider and the service requester [44].
The description of a Web service is essential for classifying, discovering and using a service. It needs to be understandable by human as well as by machines and to contain functional as well as non-functional requirements [7]. A Web service description is required to be at both semantic and syntactic level. Semantic information must contain details about the service provider, description of what the service does and its characteristics in terms of reliability, security, sequencing of messages, etc. The semantic information enables service requesters to decide whether a service satisfies their needs or not. Also, brokers can use the semantic information to categorize the service. Syntactic information describes how to use the service and may also concern non-functional requirements, e.g. security, by specifying the exact security certificates required or supported by a Web service.

WSDL describes only the syntactic aspects of a service. Essentially, a WSDL document describes how to invoke a service and provides information on the data. An interface specification describes how clients interact with the service. A security policy prescribes who is granted which access right for a particular service. Service providers may specify those who may discover the service using the filter constraint. A service may have multiple descriptions in different vocabularies. Service providers may create their own vocabularies or select a Microsoft BizTalk vocabulary or use a combination of the two. Thus, HP Web service platform supports semantic as well as syntactic description of a Web service. The Service Description Language (SDL) described in the early development stages, promises to address Web service description at both lower and higher level [98].
Discovery

Discovery depends highly on service description, categorization and publishing as well as on the analysis and description of user needs, besides searching and matching techniques. It is a basic activity that has to be supported at a semantic level, as it is important for requesters to describe their needs at a conceptual level and to make sure that their needs are matched with what the service provides [86] [7]. UDDI contains semantic information about companies and syntactic information about services. However, the discovery mechanisms are not so elaborated. Other approaches to Web service discovery include the Hewlett Packard (HP) Web services platform and the Service Request Language (SRL) [98]. The HP Web services platform attempts to address the discovery issue at a conceptual level by allowing users to express their service request as a collection of attribute values. Then, it automatically discovers registered services that have the desired attributes.

Binding and Invocation

Binding refers to the collaboration between services when it is established. This function must support semantic information such as expected quality of service, message ordering, delivery time constraints, priorities, etc. Binding can be done at design time or runtime but due to a static choice of collaborator it is carried out fully dynamic. In the first case, the application knows exactly with which service it will interact and how. In the second case, the designer has encoded a specific query in the application to pass to the broker with respect to a precise collaborating service. However, the details of interaction, e.g. the choice of transport protocol or authentication, are defined when the broker returns the service description at runtime. Finally, in the last case, the application
knows the semantics and the Application Programming Interfaces (APIs) of the service to be used, but queries a broker with a search pattern that allows a set of alternative service providers to be returned and chooses from this list at runtime [12].

2.1.5 eXtensible Markup Language (XML)

According to W3C, eXtensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of the Standard Generalized Markup Language (SGML) [International Standard Organization-ISO 8879]. By construction, XML documents are conforming SGML documents.

XML is a W3C (World Wide Web Consortium) specification that defines a meta-language for describing data. In XML applications, data is described by surrounding it with customizable, text-based tags that give information about the data itself as well as its hierarchical structure. Because XML syntax consists of text-based mark-up that describes the data being tagged, it is both application-independent and human readable [86]. This simplicity and interoperability have helped XML achieve widespread acceptance and adoption as the standard for exchanging information between heterogeneous systems in a wide variety of applications, including Web services. XML forms the basis for all modern Web services, which use XML-based technologies to describe their interfaces and to encode their messages. WSDL, SOAP, and UDDI all use XML-based messaging that any machine can interpret [108] [73].
2.1.6 Web services Description Language (WSDL)

WSDL is an XML-based format for describing Web services maintained by the W3C. Clients wishing to access a Web service can read and interpret its WSDL file to learn about the location of the service and its available operations [86]. In this way, the WSDL definition acts as the initial Web service interface, providing clients with all the information they need to interact with the service in a standards-based way. Through the WSDL, a Web services client learns where a service can be accessed, what operations the service performs, the communication protocols the service supports, and the correct format for sending messages to the service. A WSDL file is an XML document that describes a Web service using six main elements:

- **Port type** – groups and describes the operations performed by the service through the defined interface.
- **Port** – specifies an address for a binding, i.e., defines a communication port.
- **Message** – describes the names and format of the messages supported by the service.
- **Types** – defines the data types (as defined in an XML Schema) used by the service for sending messages between the client and the server.
- **Binding** – defines the communication protocols supported by the operations provided by the service.
- **Service** – specifies the address (URL) for accessing the service.

The WSDL document that describes a Web service acts as a contract between Web service client and server. By adhering to this contact the service provider and consumer are able to exchange data in a standard way, regardless of the underlying platforms and applications on which they are operating [16].
2.1.7 Simple Object Access Protocol (SOAP)

SOAP is an XML-based protocol from the W3C for exchanging data over Hyper Text Transfer Protocol (HTTP). It provides a simple, standards-based method for sending XML messages between applications. Web services use SOAP to send messages between a service and its client(s). Because HTTP is supported by all Web servers and browsers, SOAP messages can be sent between applications regardless of their platform or programming language. This quality gives Web services their characteristic interoperability [86] [108].

SOAP is a lightweight XML-based protocol that supports RPC (Remote Procedure Calls) and messaging over any network protocol but primary over HTTP. SOAP is the standard mechanism for invoking Web services, and it provides a way to communicate between applications running on different operating systems, programming languages, component models, since it uses an XML-based protocol instead of a binary format [73]. SOAP messages are XML documents that contain some or all of the following elements:

- **Envelope** – specifies that the XML document is a SOAP message; encloses the message itself.
- **Header** (optional) – contains information relevant to the message, e.g., the date the message was sent, authentication data, etc.
- **Body** – includes the message payload.
- **Fault** (optional) – carries information about a client or server error within a SOAP message.
Data is sent between the client and the Web service using request and response SOAP messages, the format for which is specified in the WSDL definition. Because the client and server adhere to the WSDL contract when creating SOAP messages, the messages are guaranteed to be compatible.

### 2.1.8 Universal Description Discovery and Integration (UDDI)

UDDI is a standard sponsored by Organization for the Advancement of Structured Information Standards (OASIS). UDDI defines the registry in which available Web services are stored, indexed, and organized for discovery by the Web service consumer [16]. Often described as the yellow pages of Web services, UDDI is a specification for creating an XML-based registry that lists information about businesses and the Web services they offer. UDDI provides businesses a uniform way of listing their services and discovering services offered by other organizations. Though implementations vary, UDDI often describes services using WSDL and communicates via SOAP messaging [108]. Registering a Web service in a UDDI registry is an optional step, and UDDI registries can be public or private. To search for a Web service, a developer can query a UDDI registry to obtain the WSDL for the service to utilize. Developers can also design their Web services clients to receive automatic updates about any changes to a service from the UDDI registry. It provides a simple framework for describing any kind of Web services. The schema defines four core types of information: business information, service information, binding information, and information about specifications for services.
2.1.9 Web service Level Agreement (WSLA)

A service level agreement (SLA) formally defines the level of service. Practically, an SLA constitutes a negotiated agreement between two parties (i.e., a provider and a consumer) about the consumption or usage of one or more services. Normally, a SLA includes a definition of the involved services, according performance measurements, a problem management description, duties of all parties, warranties, methods of recovery and the termination of the agreement [27] [46].

The WSLA framework was developed by the IBM corporation for specifying, creating and monitoring of the SLA for Web services [22] [57] [73]. A SLA defines the agreement between a service provider and a service recipient for a level of performance for a particular service. The framework allows service providers and their customers to define the QoS service, aspects of a service and their Web services. The WSLA framework consists of WSLA language to specify the SLAs and a run time architecture that consists of several SLA monitoring services, which may also be delegated to others independent third parties, to ensure any evaluation is objective and accurate. The WSLA language offers flexibility on how to describe a service, which make WSLA applicable to any inter-domain management scenarios, despite the fact that it was primarily designed for the management of Web services.

2.2 Related Works

This section presents the related works according to the main areas aligned with the contributions of this thesis: selection of a Web service, assigning weights to non-functional parameters, QoS evaluation, cost calculation and managing the Web service.
2.2.1 Web service Selection and Fixing of Functional Weights

Different approaches were discussed in research investigations for Web service selection to satisfy the customer requirements [3] [77] [83] [92] [103] [104]. But, most of the works concentrated on selection of Web service based on QoS evaluation. Patrick and Haifei presented a token-based approach to quantify the QoS and CoS (Cost of Service) for achieving integrative solutions [77]. Daniel B et al. proposed the basis for analyzing the strengths and weaknesses of the existing approaches as well as the prediction of future potential improvements in Web service selection [24]. Eyhab and Quasy introduced a framework to address the issues relating to the efficient access and discovery of Web services across multiple Universal Business Registries (UBRs) and introduced a novel exploration engine, the Web service Crawler Engine (WSCE) [33] [34] [35].

QoS based service selection

In 2006, Ruben et al. presented a model for the discovery of Web services which relies on alternative views of Web service functional capabilities, allowing for different trade-offs between the accuracy of discovery results and the efficiency of the discovery process [83]. Lennon and Murphy analyzed the failure of guaranteed QoS in most cases [54]. Liu et al. proposed an open, fair and dynamic QoS computation model for Web services selection through implementation of and experimentation with a QoS registry in a hypothetical phone service provisioning market place application was achieved through secure active users’ feedback and active monitoring [56]. Le et al. presented trust and reputation management method to detect and deal with the false rating of Web services
by their provider, a QoS-based semantic Web service selection and ranking solution with the application [53].

**Influence of functional attributes on Web services selection**

Different approaches have been followed so far, spanning the use of QoS ontology, the definition of ad-hoc methods in some general framework, and the exploitation of optimization algorithms. Considering the use of ontologies, Maximilien and Sing addressed dynamic service selection via an agent framework coupled with a QoS ontology [62]. With this approach, participants can collaborate to determine each other service quality and trustworthiness. The same authors described a service selection process based on a trust model taking into account a shared QoS conceptualization that considers the preferences for qualities when determining the trust value to assign to service instances [62]. Le et al. presented a QoS-based semantic Web service selection and ranking solution with the application of a trust and reputation management method [53]. A quite different approach was proposed by Casati et al. that present a dynamic service selection using a data mining approach on the service conversation logs that can dynamically analyze the logs of various conversations and determine the services that best satisfy the service consumer’s goals [15].

**Approaches using optimization algorithms for selection of Web service**

Other works concern different kinds of optimization algorithms for the selection of concrete services in a composite service [24] [83] [100] [90] [97] [106]. Yu and Lin discussed selection algorithms for multiple QoS attributes defining the problem as a multi-dimension multi-choice 0-1 knapsack one as well as a multi-constraint optimal path problem [102]. Zeng et al. in 2004 presented a global planning approach to select an
optimal execution plan by means of integer programming. They proposed a simple QoS model using the attributes: price, availability, reliability, and reputation [105]. They applied linear programming for solving the optimization QoS matrix formed by all of the possible execution plans to obtain the maximum QoS values. In 2005, Ardagna and Pernici modeled the service composition as a mixed integer linear problem where both local and global constraints were taken into account [8]. Their approach was formulated as an optimization problem handling the whole application instead of each execution path separately. Claro et al. in 2005 proposed the use of multi-objective optimization techniques to find a set of optimal Pareto solutions from which a requestor could choose [19]. Canfora et al. adopted a quite different strategy for optimal selection based on genetic algorithms [13]. An iterative procedure was defined to search for the best solution of a given problem among a constant size population without the need for linearization required by integer programming.

Along with functional aspects advertised in WSDL specifications, non-functional characteristics of Web services such as service performance or reliability represent important characteristics for their actual and potential clients. A set of specifications have been proposed for regulating how non-functional service properties can be expressed and processed. Web service policy framework defines a model for expressing policies which refer to domain-specific capabilities, requirements, QoS parameters and general service characteristics. Policies prescribe how client applications should look like for being able to communicate with the corresponding service [15].
Involvement of non-functional parameter in QoS based Web service selection

Web services can be selected manually by developers at design time or selected automatically by self-reconfigurable Service Oriented Systems (SOSs) at runtime. Regarding design-time selection, the main research efforts are dedicated to QoS modeling and elicitation of requirements to service components. One of the first approaches to the runtime service selection was done by Zeng et al. [105]. The authors considered QoS-driven Web service selection as a global optimization problem and applied linear programming to determine the optimal service composition. The target function was defined as a linear combination of five QoS parameters: availability, successful execution rate, response time, execution cost and service reputation. If global QoS characteristics such as total composition cost are not constrained, the optimal solution to this problem could be found using the Dijkstra algorithm on a graph of available compositions [93] [88] [14] [41]. Ardagna and Pernici modelled service selection as a mixed integer linear programming problem where both local and global constraints were specified [8]. Yu and Lin proposed a model where Web service selection was mapped to the multi-choice multi-dimension 0-1 knapsack problem [102]. In this approach, the practice of offering different QoS levels was taken into account. Gao et al. in 2006, applied integer programming to dynamic Web service selection in the presence of inter service dependencies and conflicts [37].

In 2006, Wang et al. discussed Web service selection in Web service Modeling Ontology (WSMO) framework. Besides QoS characteristics listed above, the authors considered qualitative QoS factors such as service reputation, accuracy, security and exception handling. Here also, the service selection was based on the assessment of linear
combination of scaled quality parameters [96]. Yang et al. converted QoS characteristics into a form which followed the ascent property [99]. Along with the five QoS attributes considered by Zeng et al. a service matching degree was taken into account [105]. The multiple criteria decision making technique was used to give an overall evaluation of Web service composition. The above solutions depend strongly on weights assigned to each QoS parameter by prospective clients. However, there is no clear mechanism for a client to set up these weights in order to obtain a desired result. Several approaches try to avoid a client involvement into service selection procedure. Hu et al. modeled service selection as the multiple-attribute decision making problem [43]. Four models for determining relative weights of QoS attributes were proposed: subjective, single, objective and subjective objective. Claro et al. extended the QoS-driven selection algorithm proposed by Zeng et al. [19] [105].

The first extension concerned the concept of reputation which was ranked based on the client's knowledge about the service domain. As the second extension, the authors applied a multi-objective genetic algorithm to select the optimal composition without giving any weight to any QoS criterion. Canfora et al. also proposed to use genetic programming for QoS-aware workflow replanning [13]. In 2005, an interesting approach to dynamic Web service selection was studied by Octavio et al. [74]. The approach applied constraint programming for procuring Web services with temporal-aware demands and offers. Bonatti and Festa considered three instances of service selection problem which aimed at optimizing the overall quality of mapping between multiple requests and multiple offers [11]. Mou et al. experimented with expressing client's QoS preferences in a fuzzy way and specified the corresponding choice-making process [70].
In 2009, Alrifai and Risse proposed a more efficient QoS computation approach that based on a heuristic algorithm, which decomposed the optimization problem into small sub-problems [5].

But in these investigations, not much importance was given for domain specific weight for the non-functional parameters to achieve the asserted QoS and thereby satisfy the user’s requirement. Hence in the present research, functionality based fixing of weight for Web services are analyzed.

2.2.2 Quality of Service (QoS)

The importance of Web services has been raised as an enabler of SOA. As a result, most software communities who are related in planning, development and management of SOA have significantly interested in Web services quality.

A generic description of QoS is the ability of a service to guarantee a certain level of quality. This abstract definition of QoS can be applied in different contexts, but also needs an appropriate refinement according to the specific domain such as telecommunication or networking, in which the term QoS was originated. Thus, the first step in this section is a survey of QoS definitions.

The international quality standard ISO 8402 describes quality as,

“The totality of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs” [81].

Quality of service in the field of data networking has been published 1998 by the ITU(International Telecommunication Union) in its recommendation, which comprises the following definition of QoS:

"A set of qualities related to the collective behavior of one or more objects. ”
Zibin et al. defined QoS in the context of the internet as follows:

"The capability to provide resource assurance and service differentiation in a network."

Web service is becoming a major technique for building loosely coupled distributed systems. SOA has been widely employed in e-business, e-government, automotive systems, multimedia services, process control, finance, and a lot of other domains. QoS is usually employed for describing the non-functional characteristics of Web services and employed as an important differentiating point of different Web services. With the prevalence of Web services on the internet, Web service QoS management is becoming more and more important.

As a service could be provided by third parties and invoked dynamically via network, service performance might be varied by the network speed or the number of connected users at a given time. Service Level Measurement Quality (SLMQ) is a set of quantitative attributes which describe the runtime service responsiveness in a view of consumers. This quality factor represents how quickly and soundly Web services can respond which can be measured numerically on system [84]. SLMQ consists of five sub-quality factors; response time, maximum throughput, availability, accessibility, and successability.

**Response time**

Response time refers to duration from the time of sending a request to the time of receiving a response. The response time can be varied by the point of measurement and affected by three types of latency: client latency, network latency and server latency as depicted in Figure 2.2.
*Client latency* refers to the delay time caused by a client system in the whole processing time for a service request. It is a sum of the time taken between ‘a client application requests a service’ event and ‘the request is sent by a client’ event (t1~t2), and the time taken between the ‘response arrives to the client’ event and ‘the application system receives the response’ event (t7~t8).

*Network latency* refers to the time taken on a network for transmitting request message and response message. It is a sum of the time taken between ‘a client sends a request’ event and ‘the Web services server receives the request’ event (t2~t3), and the time taken between ‘the server sends a response’ event and ‘the client receives the response’ event (t6~t7).

*Server latency* is a delay time caused by a server system in the whole processing time for a service request. It is the sum of time taken between ‘the server sends the request’ event and ‘Web services receives the request’ event (t3~t4), ‘the time taken for processing the service’ event (t4~t5), and the time taken between ‘the response is sent by the Web services’ event and ‘the server receives the response’ event (t5~t6).

*Figure 2.2: Response time and latency*
Three types of latency and response time can be calculated by the following formulas.

\[
\text{Client Latency} = CL1 + CL2 \quad \text{.......................... (2.1)}
\]

\[
\text{Network Latency} = NL1 + NL2 \quad \text{.......................... (2.2)}
\]

\[
\text{Server Latency} = SL1 + SL2 + SL3 \quad \text{.......................... (2.3)}
\]

\[
\text{Response Time} = \text{Client Latency} + \text{Network Latency} + \text{Server Latency}
\]

**Maximum throughput**

Maximum throughput refers to the maximum amount of services that the service provider can process in a given time period. It is the maximum number of responses which can be processed in a unit time. The following formula expresses the maximum throughput.

\[
\text{Throughput} = \max \left( \frac{\text{No. of Requests Processed by Provider in Measured Time}}{\text{Measured Time}} \right)
\]

\[\text{.......................... (2.4)}\]

**Availability**

Availability of a Web service defines the probability that at a point-in time, the Web service will be operational and able to deliver the requested service. Larger probability values represents that the service is always present and ready for immediate use, while small values indicate that it is difficult to foretell if the service will be available at a given time. The availability of 0.99 means that in every 100 time of unit the Web service is likely to be available for 99 of these. Associated with availability is time-to-repair (TTR). TTR represents the time it takes to repair the Web service that has failed.
Availability is a measurement which represents the degree of which Web services are available in operational status. This refers to a ratio of time in which the Web services server is up and running. As the DownTime represents the time when a Web services server is not available to use and UpTime represents the time when the server is available, availability refers to ratio of UpTime to measured time. Measured Time is time considered for checking the operational status. In order to calculate availability, it is conveniently rather using DownTime than UpTime and it can be expressed as the following formula [84].

\[
\text{Availability} = 1 - \frac{\text{Down Time}}{\text{Measured Time}}
\]  
\[\text{Accessibility} = \frac{\text{Number of Ack Message}}{\text{Number of Requested Message}}\]

Accessibility represents the probability of which Web services platform is accessible while the system is available. This is a ratio of receiving an acknowledge message from the platform when requesting services. That is, it is expressed as the ratio of the number of returned Ack message to the number of request messages in a given time. To increase accessibility, a system needs to be built in expansible architecture.
Successability

Successability is a probability of returning responses after Web services are successfully processed. In other words, it refers to a ratio of the number of response messages to the number of request messages after successfully processing services in a given time [84]. ‘Being successful’ means the case that a response message defined in WSDL is returned. In this time, it is assumed that a request message is an error free message.

\[
\text{Successability} = \frac{\text{Number of Response Message}}{\text{Number of Requested Message}}
\]

\[\text{(2.7)}\]

2.2.3 Evaluation of QoS

Since 2002, Menasce highlighted the need for QoS definition, specification and evaluation in Web service from the perspective of both service provider and service user [39] [87] [67] [52] [63]. In 2003, Kang et al. from the W3C working group summarized the key requirements of QoS for Web service [45]. A review paper on the state-of-art in the field has been presented by Ludwig in 2003[58]. Recently, several research works dealt with the definition of QoS languages for Web service-based applications. This includes HP’s Web services Management Language (WSML) and framework, IBM’s Web service Level Agreement (WSLA) language, the Web services Offer Language (WSOL) as well as approaches based on Web service (WS) -Policy [58].

The importance of QoS in the area of service-oriented systems has been initially discussed in early 2000 by several researchers by leveraging the knowledge from earlier networking-related QoS attributes [59] [20] [66] [78] [103]. Additionally, application and
business related QoS attributes have been addressed to cope with the need for providing quality attributes of loosely coupled distributed services. Up to now, QoS issues in Service Oriented Computing (SOC) have received a lot of attention; in particular, QoS-aware service selection and composition have been core areas of research [1] [68] [80].

**Ontologies for Web service specification**

Web services Offering Language (WSOL) is one of the first approaches introduced for specifying Web service non-functional properties [71] [91]. It allows service providers to describe such characteristics of their service offerings as performance, price or simple access rights. SLAng is another approach in this direction [50]. SLAng aims at facilitating different levels of QoS abstraction and identifies two types of contracts: horizontal contracts refer to contracts between coordinated peers while vertical contracts refer to contracts between parties and their underlying infrastructure. In contrast to WSOL, this approach also supports specification of service client's responsibilities and formalization of mutual service client obligations. Finally, WS-Policy framework is a more recognized effort towards a flexible and open language for specifying how client applications should communicate with a particular Web service [94]. Multiple efforts have been done towards precise ontological description of Web service quality characteristics. Chen et al. designed a QoS ontology which consists of three layers: a property definition layer, used for presenting the property's domain and range constraints; a metrics layer, which provides measurement details; and a profile layer, which is used for QoS matching [17].

**Domain specific and independent functionalities**

A QoS evaluation approach by Bleul and Weiss additionally supported service packaging and included an ontology for defining functional relations among various QoS
metrics [10]. Michael et al. designed a set of ontologies which aimed at minimizing ambiguities in QoS definition [65]. This set included four ontologies for representing QoS requirements, QoS measurements, traceability and quality management. Another stream of works focused on organizational and technical aspects of QoS data sharing. Bianchini et al. present an ontology-based approach to service classification and discovery [9]. In this work, services were organized into categories on the basis of their functional descriptions and then enriched with QoS features. The discovery process in this approach was a sort of cascade hybrid matching. Firstly, a preliminary functionality search was used to filter a set of candidate services. Secondly, a quality-driven search was applied to the filtered set. Both general-purpose and domain-specific quality parameters were supported that they were relatively stable in time. The approach by D’Ambrogio aimed at aligning current Web services standards with quality modeling [21]. More specifically, the author proposed an extension of WSDL with QoS ontology.

Description of each QoS factor included a value, a unit of the value, a source of measurement (measured, assumed, predicted, measured or undefined), a type of statistical value (maximum, minimum, mean, variance, etc.) and a preferred direction of the value such as increasing, decreasing and undefined. One of the advantages of this model was its simplicity. However, the approach does not cater for extensibility. Along with the methods for describing quality of Web services, the models for expressing user QoS requirements and preferences have been designed. Dobson et al. in 2005, proposed a model for specifying QoS-related service requests with the help of QoS ontologies [28]. Le et al. presented a method for expressing QoS service selection criteria and environmental conditions via description logics and Horn rules [53]. In 2002, Maximilen
and Singh introduced a three-layered ontology for defining QoS of semantic Web services which were collected by distributed agents [61]. This work opens research on collecting user feedback and specifying QoS of Web services on the empirical basis. In such approaches, quality of Web services was defined based on how a given service behaved in the past rather than how it has been advertised.

**Different models for QoS evaluation**

Different methods and tools can be used to predict or analyze QoS of Web services. The main approaches followed so far in the literature were outlined as follows:

Stochastic models are widely used in the performance and reliability evaluation of computer and communication systems [68] [80]. Different kinds of models have been proposed including single queuing systems, queuing networks, timed stochastic Petri nets, Markov and non-Markov stochastic processes. Various types of model can be applied to represent specific characteristics of classes of systems, such as queuing networks for congestion systems and timed stochastic Petri nets for concurrent systems, Markov model to study reliability and availability. Models can be classified in two main categories: analytical and simulation models depending on the analysis approach. An analytical model of a system is represented by a set of variables and parameters to represent system components and a set of equations that correspond to their interactions.

In the field of service computing many architectures and algorithms were proposed to improve the performance of the Web service by evaluating the QoS and by presenting the non-functional characteristics of the Web services.

Ran S was one of the first who has proposed a QoS model and a UDDI extension for associating QoS information to specific Web services [81]. The model comprised of
QoS categories, such as runtime-related QoS, transactional QoS attributes and several other categories. QoS information of a Web service is included when publishing a service in the UDDI registry. Unfortunately, Ran does not describe whether QoS information can be updated once it is published; therefore, leading to static QoS information that quickly becomes obsolete. Additionally, it did not specify how runtime-related QoS attributes are calculated or monitored.

**Functional weights for QoS evaluation**

In 2002, Mani and Nagarajan, from IBM discussed various factors that influence the service such as QoS requirements, bottlenecks affecting performance of Web services, approaches of providing service quality and transactional services [59]. Similarly, Zeng et al. in 2004 employed five generic QoS properties such as execution price, execution duration, reliability, availability, and reputation for dynamic Web service composition [105]. Also, this work addressed the issue of selecting Web services for the purpose of their composition in a way that maximizes user satisfaction expressed as utility functions over QoS attributes, while satisfying the constraints set by the user and by the structure of the composite service.

In 2009, Alrifai and Risse proposed an efficient service composition approach by considering both generic QoS and domain-specific QoS properties [5]. Yutu Liu et al. presented open, fair and dynamic QoS computation model for Web services selection through implementation of a QoS registry in a hypothetical phone service application [104]. However, these investigations did not guarantee the QoS level which is asserted at the time of Web service selection. Zibin et al in 2010 presents distributed QoS evaluation of real world Web services by studying the performance of real-world Web services
In this work, to study the performance of real-world Web services, several large-scale evaluations on real-world Web services are used and the QoS datasets are publicly released.

### 2.2.4 Calculation of Cost based on QoS

Web services technologies have over a period of time replaced expensive Electronic Data Interchange (EDI) links and have come to stay as the backbone for collaboration between various enterprises. While currently majority of Web services are available free, over a period of time increased business dependencies of customers on Web services are resulting in demands for better quality of service. Investments in newer technologies to improve quality of service results in increasing cost of service, which needs to be offset by revenues. While service providers seek predictability in revenues, customers and users of the Web services seek flexibility in pricing by not being charged for services not used and service features that are not provided. QoS thus becomes a key determinant of pricing in Web services. In most cases the customer of the Web service thinks that he is paying for the service that is undelivered. To find the optimal correlation between the QoS and cost, some cost effective methods were proposed [6] [84].

In 2007, Al-Masri and Mahmoud evaluated computation of QoS by considering the properties like response time, throughput, availability, accessibility, interoperability analysis, and cost by calculating the Web service relevancy function [3]. Antonova in 2010 evaluated an algorithm that allows clients to select the Web service with an optimal correlation between quality and price [6]. Szu YL et al presented a cost effective planning graph approach for large scale Web service composition [90]. In this study, they proposed a novel cost-effective Web service composition mechanism. This work utilizes
planning graph based on backward search algorithm to find multiple feasible solutions and recommends a best composition solution according to the lowest service cost. Gunther et al. focused on the challenges associated with composing and pricing Web services [42]. George et al. proposed a framework for pricing Web services based on the QoS features provided, the underlying cost of service and the level of predictability of revenues that accrue to the service provider based on the transaction volumes of various services [38]. Their work suggested a subscription-based pricing for commoditized Web services, transaction-based pricing for channelized Web services and risk-based pricing for customized Web services.

These investigations towards evaluation cost do not focus on the actual QoS of Web service. In this thesis, the cost based on the offered QoS are dynamically evaluated thereby satisfy both the customer and provider of the Web service in quality and cost.

2.2.5 Management Decision for Further Development

Web service management, mostly addressed by some academic institutions, as well as major IT community such as OASIS, and IT Companies such as HP, Computer Associate (CA), and IBM that need new solution in order to improve the efficiency of their applications. These organizations have put forward a number of protocols that includes Web service Distribution Management (WSDM) [47], Web service Manageability (WSM) and Web service Level Agreement (WSLA) protocol [57].

Monitoring the QoS and managing the Web service to achieve the expected guarantee in WSLA is proposed in various research works. The quality and usage of Web services is controlled and monitored via a set of management mechanisms. Dan et al. described a framework for providing customers of Web services and differentiated levels
of service through the use of automated management and SLAs [22]. Qi et al. focused on investigating the different research problems, solutions, and directions to deploying Web services that are managed by an integrated Web service management system [79].

Chrysostomos and Dimitris discussed a monitoring system to check violations in the pre-agreed metric values of SLAs [18]. QoS measurement of Web services has been used in the SLA in IBM’s WSLA framework by Ludwig et al. They described a novel framework that relied on services that may be subscribed dynamically and on-demand and the management based on WSLA [57]. Alexander and Heiko proposed a SLA management system that measured and monitored the QoS parameters, checked the agreed-upon service levels, and reported violations to the authorized parties involved in the SLA management process [2]. Several such Web service management protocols do address some specific aspects of Web service management such as differentiation of service offering the use of Web service technology as a communication and integration protocol for distributed system management, and the management Web service itself [47] [27] [95]. However, no related work addresses the integration of management issues such as configuration of Web service, usage analysis and management of Web services during runtime, as addressed in this thesis work.

In the present research work, WSLA based automated management is proposed to take management action against the actual QoS measure and the performance of individual non-functional parameters.
2.3 Summary

This chapter has presented the basic technologies that are used with the implementation of Web services. The related approaches in the field of selection, evaluation, costing and monitoring of Web services are discussed in this chapter. But these approaches for evaluating the QoS and cost of the Web service fails to satisfy the customer needs because these methods does not evaluate the QoS based on the functionality of the Web service. Here the customer satisfaction is very less because the customer always feels that he is paying for the quality that is undelivered. So in this research work, a functionality based QoS and cost evaluation method was proposed to satisfy both the customer and provider in quality and cost.