Chapter 2

Web services Preliminaries

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

2.1 Service Oriented Computing

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

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

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

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

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

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

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

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

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

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

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

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

2.2.2 Semantic aspects of SOAP-based Web services

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

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

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

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

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

(a.) Foundational and Domain ontologies

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

### 2.2.3 Semantic description of SOAP-based Web services

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

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

(I.) **OWL-S**

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

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

(III.) **WSDL-S**

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

(IV.) **SAWSDL**

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

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

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

(a.) **Service Discovery**

Web service discovery is a process of finding most suitable service from the repository according to requesters’ requirement. (Singh and Huhns). Web services discovery approaches are classified into four main categories (Zunino and Campo),(Mukhopadhyay

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

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

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

(b.) Service Selection

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

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

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

(c.) Service Composition

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

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

Based on the composition pattern, service composition process is divided into two categories: service orchestration and service choreography (Peltz). Service orchestration represents a business process which coordinates and interacts among the various services, by describing an invocation order of Web services. The standard for Web services orchestration is Web Services Business Process Execution Language (WS-
BPEL), which is widely accepted by the industry. Service Choreography describes collaboration between Web services that focuses on peer-to-peer interaction, where all participating Web services work equally and do not rely on central coordinator. The choreography mechanism is supported by the standard WS-CDL (Web Services Choreography Description Language) \(\text{(Kavantzas et al.)}\).

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

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

Automated services composition can be classified into the Semantic Web and Artificial Intelligence (AI) planning techniques: The Semantic Web allows the representation and exchange of information in a meaningful way, facilitating automated processing of descriptions on the Web. Annotations on the Semantic Web express links between information resources on the Web and connect information resources to formal terminologies. These connective structures are called ontologies, which are a widely accepted state-of-the-art knowledge representation. The extensive usage of ontologies allows semantically enhanced information processing and support for interoperability. AI planning problem can be described as a tuple \(\langle S, S_0, G, A, T \rangle\) \(\text{(Rao and Su)}\), where
S is the set of all possible states of the world; S0 denotes the initial state of the world; G denotes the goal state of the world the planning system attempts to reach; A is the set of actions the planner can perform in attempt to reach a desire goal, and The translation relation \( T = S \times A \times S \) defines the precondition and effects for the execution of each action. There are some shortcomings for using AI planning for Web service composition. A mapping of the states and actions of AI planning problems to the operations of Web services is necessary for Web service composition. One of the basic concepts of Web services is the independence of the interface from the internal processing. A Web service only has to behave according to the specified interface, but the caller doesn’t know about the internal states behind the Web service interface. A planner can only try to derive the state depending on the exchanged messages. This leads to the problems of partial observability of state and ambiguity of state description.

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

### 2.3 Resource Oriented Architecture (ROA)

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

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

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

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

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

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

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

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

Figure 2.4: RESTful Web Services Architecture

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

2.3.3 Semantic aspects of RESTful Web services

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

(I.) Linked Data

The Linked Data provides a set of best practices for publishing and connecting structured data over the Web as an upcoming alternative solution of the Semantic Web. Linked Data is an effort to create a web of data, parallel to the web of documents -
the Web, we know and use today (Berners-Lee, Bizer, and Heath). Complex information may be built aggregating simpler information units, but unlike the current Web paradigm, which conceives complex information as a whole, the information units are individually addressable and linkable (Paganelli et al.). Linked Data describes a method of publishing structured data so that it can be interlinked. It shares information over HTTP as RDF and enables querying over data. Figure 2.5 shows the evolution of Web and position of Linked Data.

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

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

1. Use URI (Berners-Lee, Fielding, and Masinter) as names for things.
2. Use HTTP URIs, so that people can look up those names.
3. When someone looks up a URI, provide useful information, using standards (RDF, SPARQL (Prud’Hommeaux, Seaborne, et al.)).

4. Include links to other URIs, so that they can discover more things (Berners-Lee, Bizer, and Heath).

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

(a.) **Linked Data Services**

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

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

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

According to the REST principles, the resource’s uniform interface is exclusively used to exchange representations. As stated above, the diversity of those representations is the key to differentiating resources and their behaviors. Therefore, a data model that is flexible enough to support those multiple kinds of representations is needed. RDF is such a data model. Being specifically designed for the Web, it uses
URIs to name things, which is consistent with the REST and linked data principles. It is based on graphs that allow representing a wide range of data structures.

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

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

A RDF graph describing Eric Miller is shown in Figure 2.6. The graph shows four sets of linked information: the first being me is a type-of person, second being my

Figure 2.6: An RDF Graph Describing Eric Miller (Manola, Miller, and McBride)
personal title is Dr, third being my mailbox is em@wm.org and the fourth being my full name is Eric Miller. This graph describes Dr Eric Miller is a person who can be reached at em@wm.org. As RDF is semantic, a software application would actually understand this statement and could share it or use it. A RDF graph is written down in the form of RDF triples, where each statement in the graph is a triple (Manola, Miller, and McBride).

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

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

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

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

### 2.3.4 Semantic Description of RESTful Web services

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

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

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

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

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

### 2.3.5 Service Discovery, Selection and Composition for RESTful Web services

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

(a.) **Service Discovery**

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

(b.) **Service Selection**

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

- **QoS**: This represents the approaches that utilize QoS as the parameter to take decision for selection.

- **User Preference**: This represents the approaches that consider user preferences in order to take into account the priority of service requesters.

- **Scalability**: This represents the approaches, which consider numerous characteristics and ranking processes that occur concurrently with accuracy of the results.

- **Automatic**: This represents the selection of service from the available services in a transparent way, without involving, bothering, or distracting its user.

(c.) **Service Composition**

Service composition is made possible by using the filtered resources provided through selection. A graph is constructed starting from a resource and then traversing the parent, consumer and producer links recursively. At each page, the descriptions are extracted and converted to RDF to update the graph. This way, a software program that doesn't have the search parameter to access a specific resource could traverse the graph to figure out what other information could be used to present the solution to the user.

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

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

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

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

---

Table 2.1: Comparison of Web services technologies

<table>
<thead>
<tr>
<th>Features</th>
<th>SOAP-based WS</th>
<th>RESTful WS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural model</td>
<td>SOA</td>
<td>REST</td>
</tr>
<tr>
<td>Commu. protocol</td>
<td>TCP, FTP, HTTP</td>
<td>HTTP only</td>
</tr>
<tr>
<td>Security</td>
<td>WS-security specification</td>
<td>HTTP Security</td>
</tr>
<tr>
<td>Service identification</td>
<td>URI, WS-Addressing</td>
<td>URI</td>
</tr>
<tr>
<td>Service description</td>
<td>WSDL</td>
<td>WADL</td>
</tr>
<tr>
<td>Service discovery</td>
<td>UDDI</td>
<td>No centralized registry</td>
</tr>
<tr>
<td>Service Composition</td>
<td>BPEL WS-CDL, User defined</td>
<td>Mashup, User defined</td>
</tr>
</tbody>
</table>
Chapter 2. Web services Preliminaries

- **Five characteristics:** on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service.

- **Four deployment models:** private Clouds, community Clouds, public Clouds, and hybrid Clouds.

- **Three service models:** Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

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

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

### 2.5.1 Cloud Services

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

1. Consumers neither own the hardware on which data processing and storage happens, nor the software that performs the data processing.
2. Consumers have the ability to access and use the service at any time over the Internet.

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

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

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

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

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

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

### 2.5.2 Service Discovery, Selection and Composition for Cloud services

Following section describes the basic concepts of Service discovery, selection and composition of Cloud services.
(a.) **Service Discovery**

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

(b.) **Service Selection**

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

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

### 2.6 QoS Model for Heterogeneous Web services

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

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

\[
Q_{TH}(s) = \frac{#R}{Time - period}
\]  

(2.1)

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

II. Response Time

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

\[
Q_{RT}(s) = ET + WT
\]

(2.2)

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

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

\[
Q(S) = Q_{RT}(S) + Q_{TH}(S)
\]  

(2.3)
Table 2.2: QoS aggregation function

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>QoS Parameters</th>
<th>Aggregation Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Response Time</td>
<td>$Q_{RT}(s) = \sum_{i=1}^{n} Q_{RT}(s_i)$</td>
</tr>
<tr>
<td>2</td>
<td>Throughput</td>
<td>$Q_{TH}(s) = \min_{i=1}^{n} Q_{TH}(s_i)$</td>
</tr>
</tbody>
</table>

### 2.6.1 Calculation of overall QoS Score

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

(I.) **Scaling:**

The QoS parameters could be either positive or negative, thus some QoS values need to be maximized, (i.e., the higher the value, the higher the quality), whereas other values have to be minimized, (i.e., the higher the value, the lower the quality).

To perform this, the scaling phase normalizes each QoS parameter value according to the following formulas. Scaling could be categorized into Positive Scaling and Negative Scaling.

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

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

The equations 2.4 and 2.5 defines the positive and negative scaling values of candidate service ($CS$) as follows.
Chapter 2. Web services Preliminaries

\[
CS_p = \frac{(q - q_{\text{min}})}{(q_{\text{max}} - q_{\text{min}})}, \text{ if } q_{\text{max}} - q_{\text{min}} \neq 0 \quad (2.4)
\]

\[
CS_n = \frac{(q_{\text{max}} - q)}{(q_{\text{max}} - q_{\text{min}})}, \text{ if } q_{\text{max}} - q_{\text{min}} \neq 0 \quad (2.5)
\]

Where, \( q \) is QoS value of respective service, \( CS_p \) is positive scaling, \( CS_n \) is negative scaling, \( q_{\text{min}} \) is minimum QoS value of respective criteria associated with service, \( q_{\text{max}} \) is maximum QoS value of respective criteria associated with service.

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

\[
s(CS, q) = \begin{cases} 
\frac{\text{overall}(CS, q) - q_m(q)}{q_M(q) - q_m(q)} & \text{if } q_{M}(q) - q_{m}(q) \neq 0 \\
1 & \text{otherwise}
\end{cases} \quad (2.6)
\]

\[
s(CS, q) = \begin{cases} 
\frac{q_m(q) - \text{overall}(CS, q)}{q_{M}(q) - q_{m}(q)} & \text{if } q_{M}(q) - q_{m}(q) \neq 0 \\
1 & \text{otherwise}
\end{cases} \quad (2.7)
\]

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

(II.) **Weighting:** This phase computes the overall score taking into account all the involved criteria and the weight assigned to each one of them. The overall QoS score of a single service can be defined by an equation 2.8, which is proposed by (de Oliveira Jr and de Oliveira) as follows.
\[ \text{overallQoS}(S_{qos/R_{qos}} = \sum_{i=0}^{n} (CS_{p_i}/CS_{n_i})) \times W \quad (2.8) \]

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

\[ \text{overallQoS}(CS) = \sum_{\forall (q,v,w) \in R_{qos}} (CS_{p}/CS_{n}) \times W \quad (2.9) \]

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