Extending Web Services Datatypes Specification for Different Development Platforms

BY

RAED OMAR AL-ABSI

SUPERVISOR

DR. SAMER HANNA

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master`s Degree in Computer Science

Deanship of Academic Research and Graduate Studies
Philadelphia University

July 2014

جامعة فيلادلفيا

نموذج تفويض

أنا رائد عمر عبد ربه العبسي ، أفوض جامعة فيلادلفيا بتزويد نسخ من رسالتي للمكتبات أو المؤسسات أو الهيئات أو الأشخاص عند طلبها.

التوقيع :

التاريخ :

Philadelphia University

Authorization Form

I am, Raed Omar Al Absi, authorize Philadelphia University to supply copies of my thesis to libraries or establishments or individuals upon request.

Signature:

Date:

Extending Web Services Datatypes Specification for Different Development Platforms

BY

RAED OMAR AL-ABSI

SUPERVISOR

DR. SAMER HANNA

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master`s Degree in Computer Science

Deanship of Academic Research and Graduate Studies
Philadelphia University

July 2014

Successfully defended and approved on _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Examination Committee Signature		**Signature**
Dr. Academic Rank:	, Chairman.	_ _ _ _ _ _ _ _ _ _ _ _
Dr. Academic Rank:	, Member.	_ _ _ _ _ _ _ _ _ _ _ _
Dr. Academic Rank:	, Member.	_ _ _ _ _ _ _ _ _ _ _ _
Dr. Academic Rank:	, External Member.	_ _ _ _ _ _ _ _ _ _ _ _

DEDICATION

To (Allah), who has blessed me beyond belief.

To my father, Omar Al Absi.

To my Mother, my great lover.

To Jasmine, my wife, my life, and best friend.

To Omar and Abd alrahman, my ambitious and loving children's.

To Reem, my loving daughter.

To my brothers (Rami, Mohammed, Ahmad, Ibrahim, Hasan).

To my sisters (Reem and Rasmiah).

To My Friends (Jalal, Mohammad, Sami, Saleem).

To Dr. Samer Hanna, my mentor and friend.

To Dr Seed Al Ghoul , my teacher.

I love you all.

For being the most important part of my dream

For the support, courage, and unconditional love.

Raed Omar Al -Absi

ACKNOWLEDGMENT

(أَفَلَا يَتَدَبَّرُونَ الْقُرْآنَ ۚ وَلَوْ كَانَ مِنْ عِندِ غَيْرِ اللَّهِ لَوَجَدُوا فِيهِ اخْتِلَافًا كَثِيرًا) النساء(82)

A number of people inspired me in carrying out this thesis. It is more or less impossible to acknowledge everyone who helped in some way. But some need to be singled out.

Dr. Samer hanna, my supervisor who make a valuable suggestions which helps in organizing the thesis, His critique also helped forge my ideas in organizing this thesis.

Professor Saeed Al-Ghoul, making critical suggestions which have gone a long way towards reshaping and restructuring my research.

Special mention must also be made of Dr nameer El-Emam and Dr Moayad Al-adhami.

I would also like to extend special acknowledgement to my family, especially my father and my mother, my brothers and all of my friends, for their encouragement and co-operation.

Your prayers and inspiration have been invaluable throughout my research.

Raed Omar Al-Absi

Table Of Contents

List Of Tables

List Of Figures

List Of Abbreviations

ACRONYM / SYNONYM	Meaning
SOA	Service Oriented Architecture
WS	Web Services
XML	Extensible Markup Language
XS	XML Schema
WSDL	Web Services Definition Language
WSA	Web Services Architecture
UDDI	Universal Description, Discovery and Integration
SOAP	Simple Object Access Protocol
CASE	Computer Aided Software Engineering
W3C	World Wide Web Consortium
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
HTTP	Hypertext Transfer Protocol
MEP	Message Exchange Pattern
SW	Semantic Web
OWL	Web Ontology Language
RDF	Resource Description Framework
SAWS	Semantic Annotated Web Service
OWL-S	Ontology Web Language For Services
SWSF	Semantic Web Services Framework
WSMO	Web Service Modeling Ontology
SAWSDL	Semantically Annotating Web Service Descriptions
ASSAM	Automated Semantic Service Annotation with Machine Learning

Abstract

In recent years, Web Services have become an important element in many areas, and the ability to exchange information through Web services is a great example of its role and its benefits and the ability to carry out the functions that may be used in the commercial field, for example, at a high level.The description and the use of the Web services is considered syntactic, this means that the knowledge of semantic Web services themselves are located on the web services user to understand or learn by other means before he decides if he want to use this service or not, and how it will be used.

This thesis is interested in the description of the Semantic Web Services and will be centered on the ambiguity and misunderstanding in the use of the data types that are used in a file written in XML language called WSDL (web service description language) , the description of Web Services is saved in this file. The problem of the ambiguity in the representation of the data types leads to many problems, for example the difficulty of interpreting the data between the service provider and requester and this leads to many errors in the merging the service or its configuration , another example is the difficulties that may encountering the tools or the techniques that developing the web services which works directly with the WSDL file, which is created automatically. thus, there will be inconsistencies in the description of services for the different techniques.

In this thesis, I will give a new way to try to solve this problem by adding semantics descriptions to the data types used in the WSDL file to simplify dealing with this issue in terms of the types of data .

Major scientific contributions to this message:

1. Adding semantic description to the Data types used in the WSDL file.

2. Improving the level of understanding of web services through the Service Description for the Semantic Web.

3. classifying the Data Types into two Categories (simple, derivative).

4. Simple data types are expressed clearly in the WSDL file and do not need to add a semantic description.

5. Derived data types are ambiguity and does not expressed in a clear manner, causing errors for service requester, so it needs a semantic description. Here it is necessary to refer to the service provider to find out the type of data used in this case.

Key Words : *Web Service, WSDL, UDDI, SOAP, XML, Semantic Web, OWL, RDF, Primative DataTypes, Derived DataTypes, Annotation*.

CHAPTER ONE

INTRODUCTION

Service oriented architecture (SOA) is a software paradigm used for creating highly modular and distributed applications. Web services can implement an atomic functions that could be composed into high level business processes(W.Thomas et al,2014). The ability of exchanging data meaningfully through the web service is a great example of the big role for the semantic interoperability to clarifying the benefits of the web services (L.xitong et al,2009). In the recent years, the web services become very important components in certain domains, but the description and the use of these web services is syntactic , that's mean, the semantics of the web services are rely to the users to understand or to earn by other means before he decide whether and how to use the service. Consequently, many opportunities exists to bridge this semantic gap using the application that emerging semantic web and the technologies of the web service for these domains, and the result is enriching and expanding a user's service interactions.

This thesis is interested in the semantic description for the web service , the core of the discussion here is about the misunderstanding and the ambiguity of the using of the datatypes in the web service description language (WSDL) document which written in XML language and containing the web service description.

1.2 Research problem :

A problem that is still facing Web services is that the XML Schema (XS) datatypes system is not expressive enough to produce a comprehendible and unambiguous Web services datatype specification.

The problem of inexpressiveness of the XML Schema based datatype specification had resulted in some other problems such as:

- The difficulty to interpret the marshaled data between service provider and requester. And consequently the difficulty faced by service requesters to understand a Web service. This problem will result in that Web service are integrated or composed in an erroneous way because requesters often misinterpret the datatype requirement for using the underlying services.
- The difficulty faced by the Web service development technologies or tools to produce an understandable data specification inside the auto-generated WSDLs. And hence the inconsistencies of the resulted specifications for different technologies.
- Web services development technologies use its custom datatypes to describe datatypes that are not supported by XS and this may hinder the interoperability and understandability attributes of Web services.

To illustrate the previous problem, let us consider the following example :

In Figure (1.1), we can see a code for generating web service programmed using ASP.Net , from this code we can notice that it is using a List as a datatype

CLASS :

```
namespace WebServiceRaed
{
   public class Employee
   {
      public int ID { set; get; }
      public string Name { set; get; }
   }
}
```

Web Service :

```
public List<Employee> getEmployees()
      {
      List<Employee> I=new List<Employee>();
         I.Add(new Employee{ ID = 1, Name = "RAED" });
         I.Add(new Employee{ ID = 2, Name = "ABSI" });
         return I;
      }
```

Figure (1.1) : Lists programming in ASP.Net – Visual C#

Now look at the next Figure (1.2), it is a part of the WSDL document that having the description for the previous code, we notice here that it is using *Array Of (Object)* as a parameter.

```
<xs:element name="getEmployees">
  <xs:complexType/ >
  </xs:element>
  <xs:element name="getEmployeesResponse">
  <xs:complexType>
  <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="1" name="getEmployeesResult"
type="tns:ArrayOfEmployee/ ">
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  <xs:complexType name="ArrayOfEmployee">
  <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="unbounded" name="Employee" nillable="true"
type="tns:Employee/ ">
  </xs:sequence>
  </xs:complexType>
```

Figure (1.2) : WSDL file for Lists programming ASP.Net – Visual C#

In another example that using a *Array* as a datatype in the Code of the same programming language ASP.Net , in the WSDL file it is using also *Array Of (Object)* as a parameter.

Both of *List* and *Array* are defined in the same way (*ArrayOf*……., *ArrayOf*…….), both of them are defined as an *array datatype*. The question here is how can the user understand which type of data the operations needs, and how can the user distinguish between the *array* datatype and *list* datatype?

1.2 why this problem ? (Motivations)

Web Services have many advantages, such as:

1. Increasing the reusability and consequently reducing the time and cost required to build a Web based distributed application.

2. Facilitating the communication between heterogeneous applications over the Internet.

3. Based on open standards.

WSDL document having a lot of problems because it is containing a large amount of information, the most important problem is the complexity which we trying to solve it among understanding the datatypes in it.

1.3 Research Contributions

The purpose of this research is to meet the following objective:

- ☒ Enhancing Web Service comprehension and understandability by Adding Semantic description to the Web Service datatype specifications for different platforms and IDEs, and to generate a better specification of the Web services input and output messages operations parameters data type.

1.4 Thesis Outline

In addition to this Chapter 1 (Introduction), this thesis consists of 4 chapters, these chapters are organized as follows:

Chapter 2 (WS Background), discusses the main components of Web Service and explains them in detail (such as XML-Schema ,WSDL ,UDDI ,SOAP), then discussing the (Semantic WS), give us some information about semantic web services objectives and technologies, then we will discuss the Literature review, reviews general approaches used to implement and to describe the functionality of Web Service which the client uses to decide if the Web Service is applicable for his needs. Chapter 3 (The Proposed model), presents the proposed method and how to implement our approach on any Web Service. Chapter 4 (Implementation and Evaluation), which will introduce the implementation of the proposed approach taking into account two cases mentioned in the previous chapter, and how to deal with this cases, then I will introduce a case study as an evaluation to my work. Chapter 5 (Conclusion and Future Work), summarizes the main achievements of this thesis, presents the general conclusions and suggests further research directions.

1.5 Summary

In this chapter, we discuss the importance of web services and the challenges facing the understandability and how to solve the ambiguity problem that interested in the datatypes in the WSDL document which having the web services descriptions, and give a simple example to clarify this problem , and we have introduced in this chapter too the importance of this problem and the contribution we aiming to achieve. We proposed an approach based on adding Semantic Annotations to the WSDL file describing the datatype as we will see in the next chapters.

CHAPTER TWO

BACKGROUND

2.1 Introduction

Service Oriented Architecture SOA is a model for organizing and utilizing distributed capabilities that may be under the control of various ownership domains (MacKenzie et al.,2006).The core of SOA is a design that developing software applications from a set of services that exist in a distributed environment, and without the awareness of the implicit implementation of the services (Erl et al., 2005). Services in SOA are well-defined, independent and loosely coupled functionalities with an assurance on their reusability and interoperability.

In service oriented architecture SOA we have two software that communicate with each other, One is a Consumer software and another one is a provider software, Consumer sends a request to the provider and provider sends a response back to the consumer.

2.2 Web Service definition(WS) :

Many Definitions for the web service had been proposed, for example . (Kreger et al., 2001) defined the Web service as "an interface that describes a collection of operations that are network accessible through standardized messages written by XML". In addition, (Huhns et al., 2005) defined the Web Services as "an implementation or understanding of the Service Oriented Architecture (SOA)".additionally (EL Bouhissi et al.,2009) defined the Web Service as "a software components that allow access to functionality via a Web interface network".

Web service is one of the presently well-adopted Web applications. Web service is the software system which enables interactions between machines through a network. It is widely deployed by, yet not limited to business organizations nowadays. The main data format used for Web services is the Extensible Markup Language (XML).

We look at Web services as a way to disclose the functionality of the information system and make it available through standard Web technologies. Using The standard technologies decreasing heterogeneity, so that is it the key that facilitating application integration.(Alonso et al., 2004).

According to the World Wide Web Consortium (W3C) Web service defined as "the application software that enables interactions between machines".(D. Booth et al.,2004). And as we said earlier, it is nowadays widely deployed by business organizations. It enables a service provider to publish its available services on the Internet, while the users can freely search and invoke these services via Internet. WS operates like the traditional Web architecture does it is operates in server-client mode . However, WS not like traditional Web architecture mainly in the feature of the loose coupling between client and server via the use of (XML) format called Extension Markup Language. XML is a general markup language known as “the ASCII of the Web”.

Using XML format, WS creating and sending the important information to its client, and this is done Instead of generating HTML pages. The client could be any application program that can operate the XML data. Transferring data is much shorter in XML format than that in HTML format. Besides, it also allow client to easily carry out post-processing towards the desired data, instead of the HTML page, that are received.(Chang cheer er., 2010).

According to this thesis, Web Services(WS) are defined as a collection of applications (interface application) or a collection of systems (endpoints) interacting with each other by exchanging data and information over networks. Each service has its self-located, self-describing and also self-operational properties.

2.3 Web Services Architecture (WSA)

The Web services architecture (WSA) is a standard architecture for software applications to cooperate using interoperable and reusable functionalities over different platforms. From the definition of the World Wide Web consortium (W3C) a Web service is “a software system designed to support interoperable Machine to Machine interaction over a network”. (D. Booth et al.,2004). The most important principle of the Web services is the interoperability of services though different machines, platforms or frameworks. Every Web service has a well-defined and self-contained functionality which can be invoked by another service. To guarantee the interoperability for these services a set of open standards is used to define the interface of the services as well as the underlying communication between these services. These standards do not care about how and where the services are implemented but they form the basis of the interoperability within the WSA. Several key standards, such as SOAP and WSDL.(Gibson Lam., 2012).

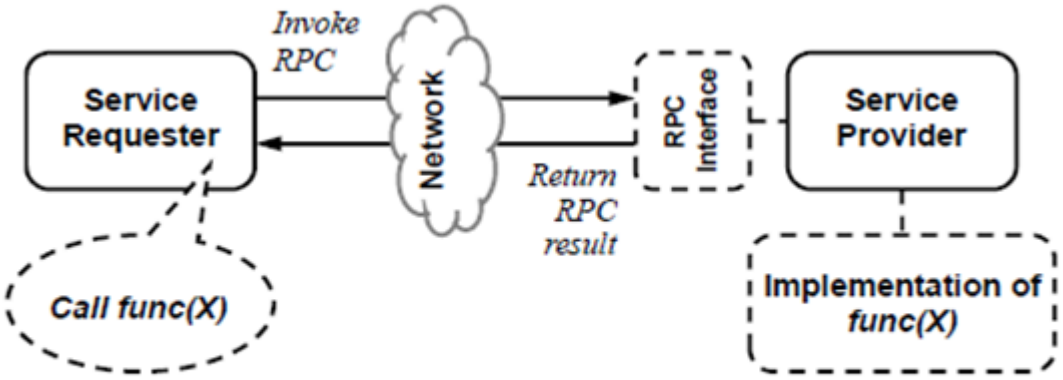
According to (Gibson Lam., 2012) Web services can be used in two different styles:

1. the RPC-oriented style
2. the document-oriented style.

1. ***RPC-oriented style*** *:*

the functionality of a remote procedure call (RPC)implemented and described by the WS. Two roles in RPC will be invoked (service requester) and (service provider). The service requester invokes the operation of the RPC, and the service provider contains the implementation of the operation provided by the RPC and also defines the interface. In Figure(2.1) we show the communication between the requester and the provider. when the service requester invokes the a RPC func(X) from the service provider, it asks the service provider for the interface of the RPC. Then, the service provider exposing

the interface that describes the service, to the requester. Then the requester builds an invocation request of the RPC involving the required parameters and sends the request to the provider. The result of the invocation is go back to the requester. The RPC Web Services are defined and communicated using the open standards of Web Services. Thus, they are reusable and interoperable regardless of the underlying implementation of the operations provided by the RPC. Nevertheless there are an implicit requirement that are common of the RPC interface between the requester and the provider, for example, the name of operation and its parameters. This breaks the loose coupling requirement in Service Oriented Architecture (SOA).



Figure(2.1) RPC invocation in the RPC-oriented Web services.

2. ***The document oriented style*** of the Web Services is used to implement a Service Oriented Architecture.

Web service architecture based upon the interaction between three roles :

a. **Service provider** : the Developer and the implementer of the Web Service.

It is a software that providing a Web service, and including the following:

1. The application program
2. The middleware
3. The platform on which they run

The application or the system which introduces the service (J. Jiang et al, 2005). This part plays the major role in a Web Service. When anyone wants to publish a programmed service over network, he must initially program this service using one of the programming languages such as Java, C# or other programming languages that allow the creation of such software (J. Fu et al, 2011). The aim of such software services is to facilitate interaction between clients and Web Services by making them independent from the technologies with which they have been implemented (W. Sun et al, 2009), and also to transfer interaction over network as application-to-application. In this area it should also be said that these services providers cannot publish their services over the network unless they publish a full description called Web Service Description Language, which is automatically generated through the programming language tools such as Apache Axis or .NET (R. Grønmo et al, 2004), used to generate these Web Services itself.

b. **Service requester (Service Consumer)** : distributed application builder (a person).
It is the collection of software that requests the web service from the service provider. Web services requester includes the following :

- The application program
- The middleware
- The platform on which they run

Service requester is the second most important part of the Web Service(Alshraideh F, 2013), defined as client or user looking for an application or any type of operation that he cannot get or apply through his system, so he seeks it via network by using one of the web browsers to find the suitable published service which implements all

of his requirements. (W. Sun et al**,** 2009) and (J. Jiang et al, 2005). In this way, the user can decide whether the service is suitable for his requirements.

The initial and basic step requires the requester to look for the WSDL document that is always attached with the published service. This document has all the information necessary to help the requester decide about the applicability of the Web Service for his needs (A. Bellini et al, 2010).

c. **service registry (service broker)** : Storing the metadata about WS like the name of Provider and the location of the contract.

The location of the service registry is in the centre, where service providers can publish their service descriptions and where service requesters can find those service descriptions.

this registry is an optional component of the WSA because both the service requesters and providers can communicating without it in many situations. For example, the company that provides a service can publish the service description directly to the users of the service in a more than one way, e.g. offering the service as a download from an FTP site.

Figure (2.2) illustrated the interaction between these roles :

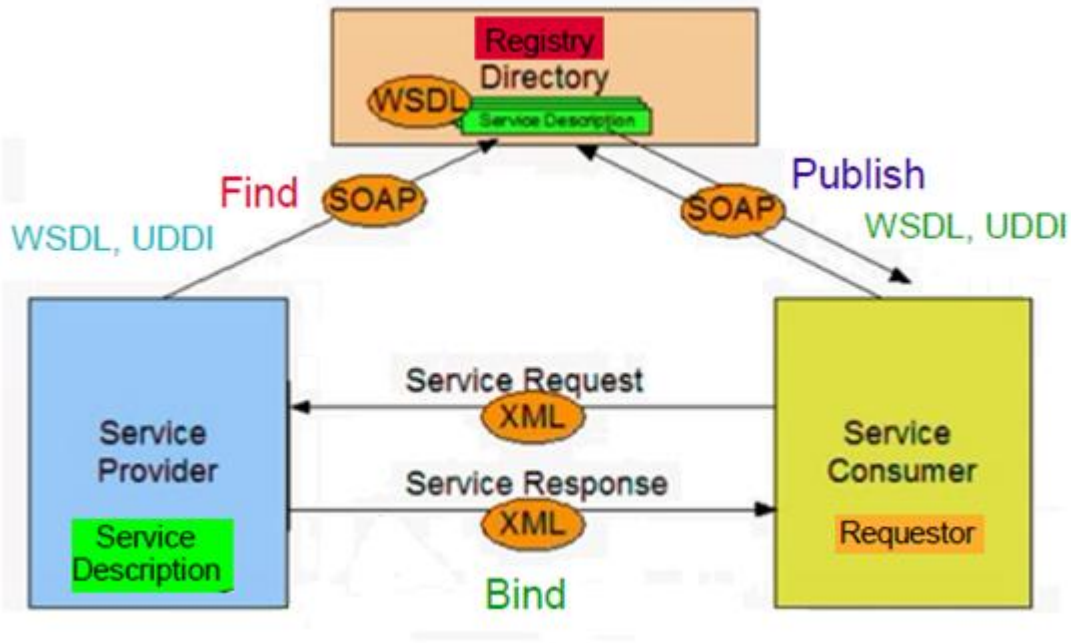


Figure (2.2) Interactions between (service provider),(service requester),(service registry)

The mean characteristic of web service architecture is that is service provider software publishes its service description and this description is placed in a certain directory which called service registry, so all providers will put their service description in that directory and the consumer software can make quires against this directory to find out what services are available and how to communicate with the provider.

WSDL , is simply a language that is used to create service descriptions so before a service description could be placed in a directory it has to be created in this special industry excepted language is called WSDL.

SOAP simple object access protocol , it is a protocol to talk to the directory so service provider will communicate with the directory using SOAP protocol in order to send service description to the directory and consumer will query against this directory using the same protocol as well.

So SOAP is simply a protocol that is used by service provider and consumer software to talk to the directory is as simple as that and both WSDL and SOAP are industry excepted language.

XML, service consumer software will now formulate its massage that needs to sent to the provider software based on the service description.

So ,consumer software will do the query against this data base to find out which services are available and how to communicate with this provider so based on the description returned WSDL language an XML massage will be formed, so this massage that this consumer will be sending to the provider will be written in XML language extensible markup language which is again an industries tendered that is used between two software to communicate with each other , so the tagged based language (looks like go to your browser open up any web page and go to view web page source or view source option or whatever that option is called in your browser and you can see the tag based language ,html language is also a tag based language , XML looks very similar)

consumer software will formulate its massage in XML language , this massage will go to service provider and also we have talked about that this massage will be based on the definition that its residing in this directory , now the service provider will generate its response back to consumer software and this response will be written in XML language as will, again this response will be according to the specification that are stored in service description.

2.4 Web Service Description Language WSDL

The Web Services Description Language (WSDL) is a language based on XML format to describe Web Services (R. Chinnici et al.,2007), WSDL documents are used to describe Web Service specifications such as location, functionality, datatypes of the input and output parameter of the operations which this Web Service provides (S. Hanna et al, 2010)

The requester reading the description that created by the provider if he want to understand the Web service. But if there is no service registry such as RPC-oriented WS, the requester retrieves the WSDL directly from the service provider. In UDDI where is the service broker acting as a service registry(L. Clement et al, 2004), WSDLs can be published to and retrieve from the registry. a list of WSDL files stored at the registry and published by their service providers. The service requester can after that search in the registry and retrieve the appropriate WSDL (Gibson Lam., 2012).

Web Services Description Language (WSDL) is inherently complex and difficult to understand, even for developers. This difficulty and complexity in understanding WSDL is greatly interesting to researchers (W. Sun et al**,** 2009). WSDL defines Web Services as a network of ports that exchange messages between each other as request and response (Requester, Provider) to get port types which are groups of operations. The data format specification and concrete protocol for these ports must be subjected to binding reuse.

Consumers – before they can interact with a Web service- must discover all of the details described above before, and WSDL provides an XML grammar for describing these details. When XML Schema Stopped, WSDL will picked up by providing a way to group the messages into operations and operations into interfaces. (A. Skonnard et

al, 2003) It also define bindings for each interface and protocol combination along with the endpoint address for each one.

All of the information that is necessary to invoke the Web service will be involved in the complete definition of WSDL file. Developers offering WSDL definitions to make it easy for others to access their services.

Everything will be described in the WSDL about the service: operations of the service, the messages of the service, the content of these messages, the grouping way of these operations , and the exposing way for these groups, in terms of network protocols. So that , in the WSDL file everything that another program needs in order to call the service will be presented.

We can see that WSDL document has many parts which are listed as following (W3C, 2008) :

1- Messages .

This element defines an abstract message to serve the input or output of an operation. Messages element consist of one or more part elements, every part is associated with either an element (when using document style) or a type (when using RPC style). Figure(2.3) The basic structure of a message definition is as follows.(A. Skonnard et al, 2003) (* => (zero or more) , ? => (optional)) :

```
<definitions .... >
    <message name="nmtoken"> *
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </message>
</definitions>
```

Figure(2.3) The basic structure of a message definition

Messages According to (W3C, 2008) it is XML-Based format defined as an abstract set of data arranged in message format which implements the data

traveling from one endpoint to another by specifying the structure of the input and output messages.

2- Operations .

XML-Based format defined as message queue , naming a method, or business process that will accept and process the message. It is also defined as the SOAP action and the way the message is encoded (W3C, 2008) .

3- PortType (Interfaces).

This element defining a set of operations, also called an interface in most environments, A portType element may contains zero or more operation elements .(A. Skonnard et al, 2003).

Port Type is an XML-Based format defined as an abstract set of operations mapped to one or more endpoints, and also defined as description of the interface of Web Service (W3C, 2008) .

Figure(2.4) The basic structure of a portType is as follows (*=> (zero or more)):

```
<definitions .... >
    <portType name="nmtoken">
        <operation name="nmtoken" .... /> *
    </portType>
</definitions>
```

Figure(2.4) The basic structure of a portType

Every portType has a unique name to be possible to refer to it from elsewhere in the WSDL definition. Every operation element have a combination of input and output elements; and when you have an output element you can also have a fault element.

4- Binding .

XML-Based format defined as concrete protocol and data formats for the operations and messages for a particular port type (W3C, 2008).

Binding element describeing the concrete details of using a specific portType with a given protocol. The binding element having many extensibility elements as well as a WSDL operation element for each operation in the portType it's describing. (A. Skonnard et al, 2003).

Figure(2.5) The basic structure of a binding element is as follows (* => (zero or more) , ? => (optional)) :

```
<wsdl:definitions .... >
    <wsdl:binding name="nmtoken" type="qname"> *
        <-- extensibility element providing binding details --> *
        <wsdl:operation name="nmtoken"> *
           <-- extensibility element for operation details --> *
           <wsdl:input name="nmtoken"? > ?
               <-- extensibility element for body details -->
           </wsdl:input>
           <wsdl:output name="nmtoken"? > ?
               <-- extensibility element for body details -->
           </wsdl:output>
           <wsdl:fault name="nmtoken"> *
               <-- extensibility element for body details -->
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

Figure(2.5) The basic structure of a binding element

A binding element has a unique name so you can refer to it from elsewhere in the WSDL definition. The binding must also specifying which portType it's describing through the type attribute.

5- Port .

XML-Based format defined as a combination of a binding and a network address ,providing the target address of the service communication (W3C, 2008).

using a single address for binding, the Port element defines an individual endpoint.

Figure(2.6) The basic structure of a Port element is as follows

```
<wsdl:definitions .... >
    <wsdl:service .... > *
        <wsdl:port name="nmtoken" binding="qname"> *
        <-- extensibility element (1) -->
       </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Figure(2.6) The basic structure of a Port element

6- Service

XML-Based format defined as a collection of related end points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions (W3C, 2008) .

Service element defines a collection of endpoints, or ports, that publish a particular binding. (A. Skonnard et al, 2003) . Figure(2.7) The basic structure of the service element is as follows:

```
<definitions .... >
    <service .... > *
        <port name="nmtoken" binding="qname"> *
           <-- extensibility element defines address details -->
        </port>
    </service>
</definitions>
```

Figure(2.7) The basic structure of the service element

Each port should have a name and assign it a binding. Then, inside the port element you use an extensibility element to define the address details specific to the binding.

After discussing WSDL Parts, The main structure of a WSDL document will be as in Figure (2.8) :

```
<definitions>
<types>
   definition of types........
</types>

<message>
   definition of a message....
</message>

<portType>
      <operation>
           definition of a operation.......
      </operation>
</portType>

<binding>
   definition of a binding....
</binding>

<service>
   definition of a service....
</service>

</definitions>
```

Figure (2.8) the main structure of the WSDL document

The combination of SOAP and XML schema used by WSDL to provide web services over the Internet. A client program can read the WSDL file and determine the available functions on the server. Any special datatypes used will be embedded in the WSDL file in the form of XML Schema. And then The client can use SOAP to actually call one of the functions listed in the WSDL.

2.5 Simple Object Access Protocol (SOAP)

(W3C, 2008) defines SOAP as " a lightweight protocol for exchanging structured and typed information in a decentralized and distributed environment". This protocol forms the standard of messaging of Web service Architecture (WSA). SOAP is an XML based messaging protocol which is a extensible , standardized, and human-readable serialization of data.

SOAP provides a message which can be exchanged in a different of transport protocols such the (SMTP) Simple Mail Transfer Protocol (J. Klensin.,2001) and (HTTP) the Hypertext Transfer Protocol (R. Fielding et al.,1999), between two nodes, (the SOAP sender and the SOAP receiver) SOAP messages can exchanged. Exchanging The message can be a one-way, a request/response interaction or a peer-to-peer conversation according to the message exchange pattern (MEP). The SOAP Binding method is the transmission of the message in the underlying protocol of the SOAP message exchange. SOAP messages can be bound to different protocols. The most important used protocol is HTTP because of the popularity of the protocol in the Internet (Gibson Lam., 2012). In Figure(2.9) we can see an example of a SOAP message :

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
envelope">
        <env:Header>
                <h:mailheader
xmlns:h="http://example.org/mailheader">
                        <h:priority>1</h:priority>
                </h:mailheader>
        </env:Header>
        <env:Body>
                <m:mail xmlns:m="http://example.org/mail">
                <m:author>John Chan</m:author>
                <m:subject>Reminder</m:subject>
                <m:content>Remember the meeting is at
9am!</m:content>
                </m:mail>
        </env:Body>
</env:Envelope>
```

Figure(2.9) An example of a SOAP message

2.6 Universal Description, Discovery and Integration (UDDI)

The idea of UDDI started In 2000,by three companies IBM, Microsoft, and Ariba they starting a project to create standards for discovering, describing, and consuming Web services. The idea was for registries, named as *UDDI registries*, to managing information about service implementations, service providers, and service metadata. Providers, could then publish their information while giving consumers - anyone needing a service - the ability to request the information to find services they needed and to request the information about how the services are consumed. UDDI carry out this interaction.(R. Richards., 2006).

Conceptually, a Provider can register three types of information into a UDDI registry.

- **White pages**

Basic necessary Data about a company, including business name, address, and contact information. The importance of This information it allows consumers to determine your service based upon your business identification. This is similar to searching either the phone number or the business address when you know the name of the business (Enterprise SOA , 2006).

- **Yellow pages**

It describing the service by classification the information. For instance, the phone directory can provide Data to find any restaurant in Amman area. It let consumers to discover any service by its categorization (taxonomy) (Enterprise SOA , 2006) .

- **Green pages**

It allow to describe the service offered by the business . containing a technical information about the supported functions, behavior, and the service access point. (Enterprise SOA , 2006).

2.7 Web services benefits

WS have many technological and business advantages,(S. Hanna et al, 2010) (Altova Inc, 2006) :

- **Interoperability** - Web Services usually working outside of private networks, giving the developers a non-proprietary way to their solutions. Services developed to have a longer life time, and offering a good advantages on investment of the developed service (S. Hanna et al, 2010).

- **Usability** - Web Services publishing the business logic of any systems over the Web, so all of your applications having the freedom to chose the Web Services that they need. Instead of starting from the beginning for each client, you only need to include the additional application of business logic on the side of the client. This allows you to use the language and tools you want to develop services code (S. Hanna et al, 2010), (Altova Inc, 2006).

- **Reusability** - Increasing the reusability of the Web Services and accordingly reducing the time and cost required to build a Web-based distributed application (S. Hanna et al, 2010).

- **Deployability** - Web Services are deployed over all standard Internet technologies.

2.8 Web services development challenges

However, Web Service faces numerous challenges including, but it is not limited to the following (S. Hanna et al, 2010) (Altova Inc, 2006) :

- There is a hindrance in Web services in some respects. Web services use plain text protocols identifying data using a fairly verbose method. And the result of that the Web service requests are more larger than the requests encoded with a

binary protocol. This large size is really an issue in the slow connections, or on the busy connections (Altova Inc, 2006).

- **The trustworthiness problem**: The Service Requester can only see the contract (WSDL) of a Web Service but not the source code. This fact has caused Service Requesters to question the trustworthiness of Web Service because Service Requesters do not trust Web Services that were implemented by others without seeing the source code. (W. Tsai et al, 2005) mentioned that this problem is limiting the growth of Web Service applications and that these applications will not grow unless researchers meet this trustworthiness challenge. (J. Zhang, 2005) stated that the current methods and technologies cannot ensure Web Service trustworthiness and that for Web Services to grow, researchers must address this challenge. (S. Hanna et al, 2010).
- HTTP and HTTPS are simple protocols, but they not concerned with long-term sessions. Typically, the connection may be disconnect while downloading a web page having maybe some images .
- **The selection problem:** Service Requesters have no criteria to choose between Web Services that accomplish the same task. (J. Zhang, 2005) stated that it is a big challenge to choose the most appropriate Web Service from a "sea of unpredictable Web Services". These problems and challenges appeared for more than one reason, one of them is that the WSDL contract of a Web Service describes the operation or the function that a Web Service provides and how to bind to this Service. However, it does not describe the non-functional quality attributes such as robustness, reliability or performance. (S. Hanna and A. Alawneh, 2010).

- There is a problem with HTTP and HTTPS protocols, these protocols are "stateless"—there is no knowledge between the client and the server because there is no data being exchanged between them (no interaction). More specifically, the server will never know that the client is no longer active if a power failure happened when the client makes a request to the server and receiving some information. The server needs to know of what a client is doing and also to determine when a client is no longer active (Altova Inc, 2006).
- **Vulnerability to invalid inputs by malicious Service Requesters**: Since Web Services are advertised in the Internet, any Service Requester can access this Web Service and some of these might be malicious Requesters that aim to do harm. The Web Input manipulation vulnerability is 59.16% of the overall Web Services vulnerabilities (W. YU et al, 2006) and that is why Web Services should be tested against this kind of fault to assess if a Web service is vulnerable to input manipulation attacks in order to increase Web service trustworthiness. (G. Myers, 2004) mentioned that testing that a program does what it is supposed to do is only half the battle, the other half is to test whether the program does what is not supposed to do. In other words, to check if a program is vulnerable to invalid input.

2.9 Semantic Web Services

Researchers have made great efforts on defining rich and machine-understandable descriptions of service behaviors to enable sophisticated service search so that existing services can be organized and utilized more effectively without human interventions(Ke Hao,2013), Tim Berners Lee the inventor of (www) and the director of (W3C), proposed the Semantic Web standards (Berners-Lee et al., 2001) and the concepts of Semantic Web, a world where the information is processed directly and indirectly by the machines. Describing rough data by means of metadata this let them to be unambiguously interpreted by the computer, and the metadata are associated in a well defined meaning to the entities that are involved.

The aims of The Semantic Web (Berners-Lee et al., 2001)(N.Shadbolt et al., 2006) is to make these large amount of information on Web accessible to machines by using annotation of the content of the Web using – a format understandable by machine- RDF, and then, this information integrated through using of the ontology (Dieter Fensel, 2003), which could use Web Ontology Language OWL (L.Deborah et al., 2004). However, these annotations indicate only to static knowledge, and the ontologies are a static descriptions of background knowledge in a specific domain.

2.9.1 Objective of Semantic Web (SW):

- SW let the user to share, find and combine the information to transmit it from one place to another very easily.(B. David,2002) people can use the Web to carry out works such as finding the Italian word for " Winter ", reserving a library book, and searching for the lowest price for a Laptop. However, machines cannot do all of these works without human guidance,

because the web pages are designed to be read by humans, not the machines.(B. David,2002)

- SW making the current web more secure and more usable by showing the information.(B. David,2002).
- It is easy for the users to use the Web for carrying out the works of finding the folders or categories (B. David,2002).
- SW provides the directive for the machine to execute any task by providing the interpreter that can interpret it, (B. David,2002). so more works will be performed by machines like combining, finding, and acting upon information on the web.
- Any task provided by the SW, the Machines can perform it and it involves combining, finding, and acting on the information that is existing on the web.(B. David,2002).

2.9.2 Semantic Web Technologies

One of the big disadvantages of using XML as a data model is that XML files do not convey the meaning of the information contained in the document. XML schema allow making constraints on the format, but not on the meaning of XML data. Exchanging XML files over the Web is possible if the parties participating in the exchange agree having the exact syntactical format (using XML Schema) of the data and the meanings of the expression and structures into XML files. The SW (Berners-Lee et al., 2001) allows representing and exchanging of information in a meaningful way, simplifying automated description processing on the Web.

Ontologies are a connective structures consist of links between the resources of the information that on the Web and connect these resources to a formal terminologies ,

these links are expressed using annotations on the Semantic Web (Dieter Fensel, 2003), Ontologies forms the base of the Semantic Web, allowing machines to understand the information through the links between the resources of the information's and the terms in the ontologies (J. Bruijn et al.,2008). Moreover, ontologies facilitating the interoperation between the resources of the information through the links to the same ontology or links between ontologies.

There are two ontology languages for Semantic Web recommended by W3C, namely *Web Ontology Language OWL* and the *RDF schema*. RDF providing a simple method to represent any kind of metadata and data, and creating the links between the annotation resources and resources with a connection to ontologies on the SW, while OWL is used to define a Web ontologies, that is, conceptualizations of a particular domain. OWL is a language that extends RDFS in an ontology way(J.de Bruijn et al.,2008) .

Two ways we can using them to create semantic annotated web service (SAWS)(M. Keyvan et al., 2012), the first one creating independent framework of the Web service description then link it to the current standards of Web service , in another words searching in OWL-S (Martin et al., 2004) and WSMO (Bruijn et al, 2005) . the second way is adding semantic annotation into the current standards of the Web service , that’s mean searching in WSDL-S(Akkiraju et al, 2005) and SAWSDL (J. Farrell et al.,2007)

2.9.3 The Resource Description Framework (RDF)

The Resource Description Framework (RDF) it is the first language constructed to build the semantic Web, RDF is a language for adding a metadata- machines can read it- to the existing data on the Web.RDF is a framework to publish information on the Web about anything. Anyone can describe the Web resources, such as the creation date,

subject, author and copyrights of any image (D.John et al., 2011).RDF is an XML language format (J.de Bruijn et al.,2008), and from the definition of RDF when we say Resource Description Framework then we have three important parts , first one the Resources which is the essence of the Semantic Web, the Second part is the Description of Resources and this is important for understanding. These descriptions could be features or relations concerned with the resource, the third part is the Framework , and this is mean it provides languages, syntax, models for these descriptions. (D.John et al., 2011).

RDF Schema (RDFS) indicating the combination of RDF with RDF Schema , it is a simple ontology language can defining the vocabularies that can be used with RDF. RDFS Unlike XML Schema, which determine the combinations and order of tags in an XML document, RDFS provides information only about the interpretation of the statements given in an RDF data model. RDF Schema does not say anything about the syntactical aspect of the RDF description. RDFS is an extension of RDF with a vocabulary for classes definition, class hierarchies, property restrictions, and property hierarchies (J.de Bruijn et al.,2008).

2.9.4 Ontology Web Language For Services (OWL-S) :

OWL-S (Martin et al., 2004) determining the upper ontology that describing the Web service properties and capabilities in OWL to facilitating the automation of the Web service tasks, including Web service execution, discovery, interoperation and composition. According to IEEE P1600.1 (2003, March 12): “An upper ontology is limited to concepts that are generic, meta, philosophical and abstract, and thus it is general enough to be address (at a high level) a broad range of domain areas. The Concepts that are specific to given domains will not be included, however, this standard

can provide a set of general concepts and a structures upon which domain ontologies could be constructed (e.g. medical, engineering, financial, etc..)."

According to (M. Keyvan et al., 2012), the upper ontology of service divided into three parts each part provides a basic type of information we must know about the service :

1. ServiceProfile (What the service does)

   ServiceProfile enabling matchmaking and discovery by determining if the service meets its needs. This profile including the nonfunctional and the functional parts of the service. The functional parts including the information of the transformation represented by the inputs and the outputs , and including the changes happened in the state cause by the execution of the service. The nonfunctional parts including the references to existing ontolgies, the quality of the service , the provider information.

2. ServiceModel (how the service works)

   It enabling the invocation of the service, monitoring, recovery and composition. ServiceModel seeing the interactions of the service as a process.

3. ServiceGrounding (How to access the service)

   ServiceGrounding mapping the constructs of the process model to detailed specifications of message protocols, formats , in other words, OWL-S Mapping Atomic process to WSDL operations (inputs and outputs) to WSDL message.

Figure 2.10 illustrates the three parts of the upper onology According to (M. Keyvan et al., 2012) , oval representing an OWL class, and the arc representing the OWL property :

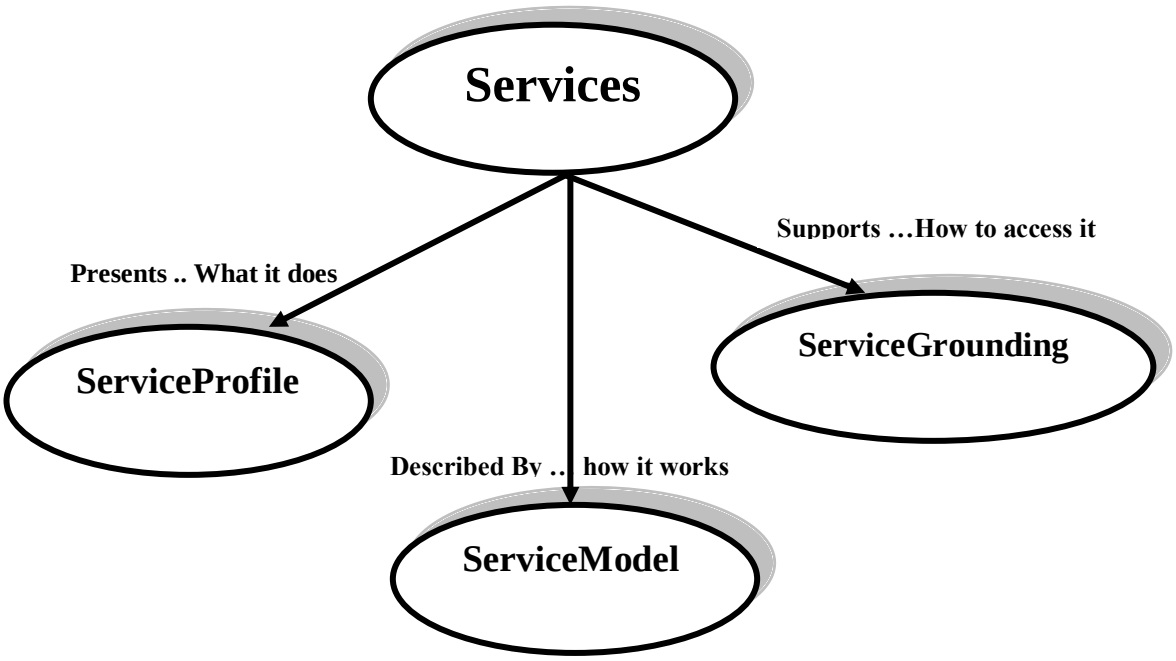


Figure (2.10): Service Ontology in OWL-S

OWL-S differentiate between two types of services, atomic service and composite service. In atomic services there are a single computer program that can accessing the Web, or a sensor, or a device invoked by a request message, and performs the message task and maybe produces a single response to that requester. In atomic services there is no continuous interaction between the user and the service. the complex or composite services are in contrast of the atomic service it is built up from multiple more primitive services it may require more interactions between the requester and the group of services that are being used.

In my thesis I am concerned with using OWL-S , transforming WSDL document into OWL-S , after that adding annotations to this ontology , these annotation to help the requester to understand the web service comes from the provider , and trying to solve WSDL DataType problem using this Semantic web method. In the next chapter we will discussing some of the literature reviews for semantic web services and comparing these studies with the subject of our thesis.

2.10 Literature Review (State Of The Art) :

Many researches and studies talk about web services, this field of study still need more and more publications and articles to cover all sides that inform this field like building web service , security in web services, semantic web services, and so on.

On my thesis , I am concerned with semantic web service , many researchers published many publication discussing semantic web services , and a lot of concepts and expressions rise during these research since 2001 when Tim Berners Lee the inventor of the World Wide Web and director of the World Wide Web Consortium ("W3C") proposed the concepts of Semantic Web (Berners Lee et al., 2001).

2.10.1 Overview

Semantic Web services was and still very important for the researchers in Web Service field because of its important we discussed in last chapter, and many frameworks were constructed in the recent years, Web service description Language Semantic (WSDL-S) (Akkiraju et al, 2005), Web Ontology Language for Services (OWL-S) (Martin et al, 2004) , Semantic Web Services Framework (SWSF) (Battle et al, 2005), Web Service Modeling Ontology (WSMO) (Bruijn et al, 2005), and the implementation of semantic registry semantically annotating Web service descriptions SAWSDL (Farrell and Lausen, 2007).

Both of OWL-S an WSMO Frameworks creating a semantic web that are independent in its description and they link it in the current standards of the web service, while WSDL-S and SAWSDL adding semantic annotation into the current standards of the web service.

In this thesis we will survey some frameworks and researches that enabling semantic web services and discuss it and find the limitations for each one and try to solve it in this thesis.

2.10.2 A Semantic Approach for Transforming XML Data into RDF Ontology

(T. Pham Thi et al, **2013**), proposed an approach that aims to measure the similarity of the duplicated elements in XML schema before transforming , redundancy in data resulted from these duplicated elements in XML schema causing ambiguity in transformation and the result may be not semantically richer than the source document .

This approach transforms XML Data into RDF Ontology since RDF presents data by using graphs of resources, duplicates we mentioned will be transformed into appropriate RDF concepts .

We can summarized the proposed approach in two steps :

1. measuring the semantic similarity of duplicated element, and this is done after the researcher find that there is two factors affect the similarity between duplicated elements, particularly the children, and the ancestor. Solved by combining them.
2. XML Schema to RDF Transformation , and here the researcher follow some rules depends on classes and some properties that created from XML schema.

In This approach the researcher concerned in transforming XML data into RDF ontology and did not measure or examine the effect of transformation on datatypes , and did not solve this problem which I am trying to solve it in my thesis.

2.10.3 An Improved Semantic Annotation Method of Web Services Based on Ontology.

(L. Zhang et al, 2008), presented a new method for semantic annotation , this method based on Ontology, taking advantage of the similarity between WSDL and ontology to annotate services.

This method eliminate some problems that faces similarity calculations between temporary ontology (comes from mapping process of WSDL into OWL to generate the corresponding OWL called (temporary ontology)) and domain ontology like the huge of computational complexity and the lower of efficiency and the decreasing of accuracy.

The proposed method summarized in two steps :

1. Filtering out the related concepts to generate a concept set for the candidates before similarity calculation, here we need to compute just the similarities between a concept and its candidate set .
2. Raising the accuracy of the structural similarity algorithm by Setting different weights to different sub-concepts .

As a result, this method raised the efficiency and accuracy in a greatly manner , and the average accuracy is 82.5%. and we find this method guarantee the settling of the semantic description of services.

We can criticize this work, Suppose one or more of these applications using this method is used by one of the Datatypes not implemented clearly in WSDL, such as char, array, array of objects. Here, the model which implements the Web Service before composition will have ambiguity, but after it composes with others, inevitably the ambiguity will increase, so that this approach is good and will work properly if all of the datatypes of Web Services parameters are represented clearly. If one or more parameters are represented ambiguously, surely it will face missed understanding for

Web Services requesters and developers. Our proposed approach seeks to overcome these challenges and also to reach batter comprehension for Web Service functionality.

2.10.4 Discovery of Semantic Web Services Compositions based on SAWSDL Annotations

(C. Guilherme et al, 2012), presented approach aims to automatic discovery and composition of semantic web services at request time, combining services when one of these services does not satisfy the requirements specified in the discovery request, to identify this composition the researcher used SAWSDL (semantic annotation for WSDL) to implement the proposed approach which called SWScomposer.

SWScomposer depends on the repository of the semantic web services and on the compositions match the characteristics specified in a discovery request , this is achieved by analyzing process between operations and the inputs and the outputs.

We can summarized the proposed approach in four steps :

1. Invoking a single operation provided by the web service
2. Extracting the semantic annotation from the WSDL description
3. Discovering and building the web services compositions and returns them to the web services
4. Web Service returns the compositions to the requester, which can then invoke the web services that integrate them.

This approach different from the last method because it aims to discovering the semantic web service compositions , but didn't solve our problem which we try to solve, so the same challenges of understandability are still present.

2.10.5 Reverse Engineering Existing Web Service Applications

(H. EL Bouhissi et al, **2009**) proposed a novel approach based on reverse engineering specifying web service according to the web service modeling ontology WSMO. This approach is split in two stages:

1. reverse engineering to extract the useful information from the WSDL document.
2. engineering for constructer of the Web Service.

The proposed approach is adding a semantic to the Web Service according to Web Service Modeling Ontology, to facilitate for the Web Service clients to discovery, selection, composition, and also execution of the Web Service.

A reverse engineering approach reduces the effort and cost to build a new Semantic Web Service by adding a semantic layer to an existing Web Service using a description file WSDL without referring to source code. In this case the semantic for Web Services will be built depending on WSDL. As we mentioned earlier, the semantic for Web Service will suffer with the same problems because the WSDL document may contain one or more aforementioned dtatatypes, and that will lead to a misleading semantic for a Web Service, resulting in the Web Service not having a good route for selection, composition, discovery from the users because a user cannot understand the functionality of the Web Service to decide if it is applicable for his purposes.

2.10.6 A framework for deriving semantic web services

WSDL provides the syntactic means for describing web service and very weak language in providing a semantic basics, (B.David et al, 2006) developed a framework that deriving semantic from syntactical description of the web service, this framework construct the ontology for the defined technical services and transforming syntactical web services to semantic web service.

Researchers following four strategies to adopted this framework :

4. theory building
5. scenario analysis
6. observation
7. framework development and evaluation

the developed framework stands on a philosophy and on the concepts of semantic web service, interpreting, scoping and harmonizing the syntactical elements defining the web service , this framework don't help us in solving the problem of datatype in WSDL document, since it just driving the semantic from syntactic web service and dealing with WSDL documents without any care about the mismatching of the parameters datatypes so it will suffer the same problems of ambiguity to select, reverse engineering, reused the Web Services.

2.10.7 Meta-Modeling of Semantic Web Services

This research discussed other manners for dealing with the understandability of the Web Service, called Meta-Model. This manner is proposed by (R. Virgilio et al, 2010) , and it allows interoperability at different levels of abstraction.

The Meta-Model Approach is summarized according to (R. Virgilio et al, 2010) in three levels: a conceptual level, a logical level and a physical level which are illustrated as follows:

1. A conceptual level, proposing a simple conceptual model where a set of constructs properly represents semantic concepts. Each construct is used to properly represent elements of documents, with the same semantics.

2. A logical level, implementing the conceptual model into a logical one. In this case they used the relational model.

3. A physical level, defining the physical design of the logical representation of previous level.

This approach is different from others in the literature, as it provides implementation solution starting with the definition of a meta-representation of the chosen data model at a conceptual level. The main challenges which we attempt to solve are not exceeded and also we must note that the understanding of Web Service functionality by its users depends on the parameters datatypes which are used to implement the Web Service operations. This approach is not effective if the Web Service used one or more of the aforementioned dtatatypes, as that will lead to a misleading comprehension for the Web Service that should be avoided.

2.10.8 ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services

In (H. Andreas et al, 2008), the authors introduce a mapping tool called Automated Semantic Service Annotation with Machine Learning (ASSAM). ASSAM generates OWL-S file from WSDL file, however, it suffers from the following limitations:

First, it introduces a list of different choices to the user to select the most appropriate class that can represent a semantic definition for each datatype in the WSDL file. This list is an unordered (unranked) list. So, the choosing process is difficult for any user.

And the second limitation is that it doesn't provide organization for the available ontologies. And this could make problem if it used in a real Semantic Web service system which could have a huge number of ontologies and concepts.

2.10.9 Summary

In this chapter we discussed a sample of some recent research which aims to understand the Web service functionality , As shown the proposed approaches used Semantic Web models to express Web service descriptions , but they have ignored dealing with the operation input/output parameters datatypes. However the Web service description remains unclear because the description of parameters datatypes is differently expressed from tool to another, so the Semantic description will be different for the same Web Service if programmed at different tools.

CHAPTER THREE

THE PROPOSED MODEL :

Extending Web Services Datatypes Specification for Different Development Platforms

We have seen in last chapters that there is no approach attempt to solve the problem in defining the Web service operations parameters datatypes which causing the inconsistency and the ambiguity in the Web services. All the previous approaches solved the problems of the Web services understanding, reusing and comprehension by using Semantic Web services, but these approaches ignored the important part which is the needed data that must be used to bind with Web services, and this is my thesis talking about.

WSDL mapping abstract messages to a concrete message using a declarative information and the binding will be expressed to determined the port to post or read the messages from. But WSDL is not expressive enough to determined the semantic competitions or the interactions between protocols which we needed for the compositions, OWL-S is in the contrast, it describing the Web services in expressions of their ports and describing the Web service capabilities in expressions of the provided abstract functions , the process model and the grounding which describing how the service interact. So WSDL and OWL-S are complementary to each other : OWL-S give us an abstract information about the operations and about the exchanged information , while WSDL give us how this abstract information mapped into messages .WSDL will be involved in the specification of the OWL-S Grounding to provide the information which will be bind to determine the ports.

The proposed approach is could be accompanied with a tool in order to prove the approaches utility and compare it with other approaches. This approach can answer the major questions of this thesis, that is :

- Can we add a semantic description to the data specification that are produced based on different Web services platforms (such as J2EE and .NET) and also based on different IDEs such as Eclipse, Visual studio, and NetBeans.
- Can we investigate how different web service platform handle the datatype specifications for a certain Web service operation and how we can enhance the specifications to make it more understandable and reusable by requesters.

3.1 The Proposed Model

The proposed model can be explained using the following Figure (3.1) of the main components of this model.

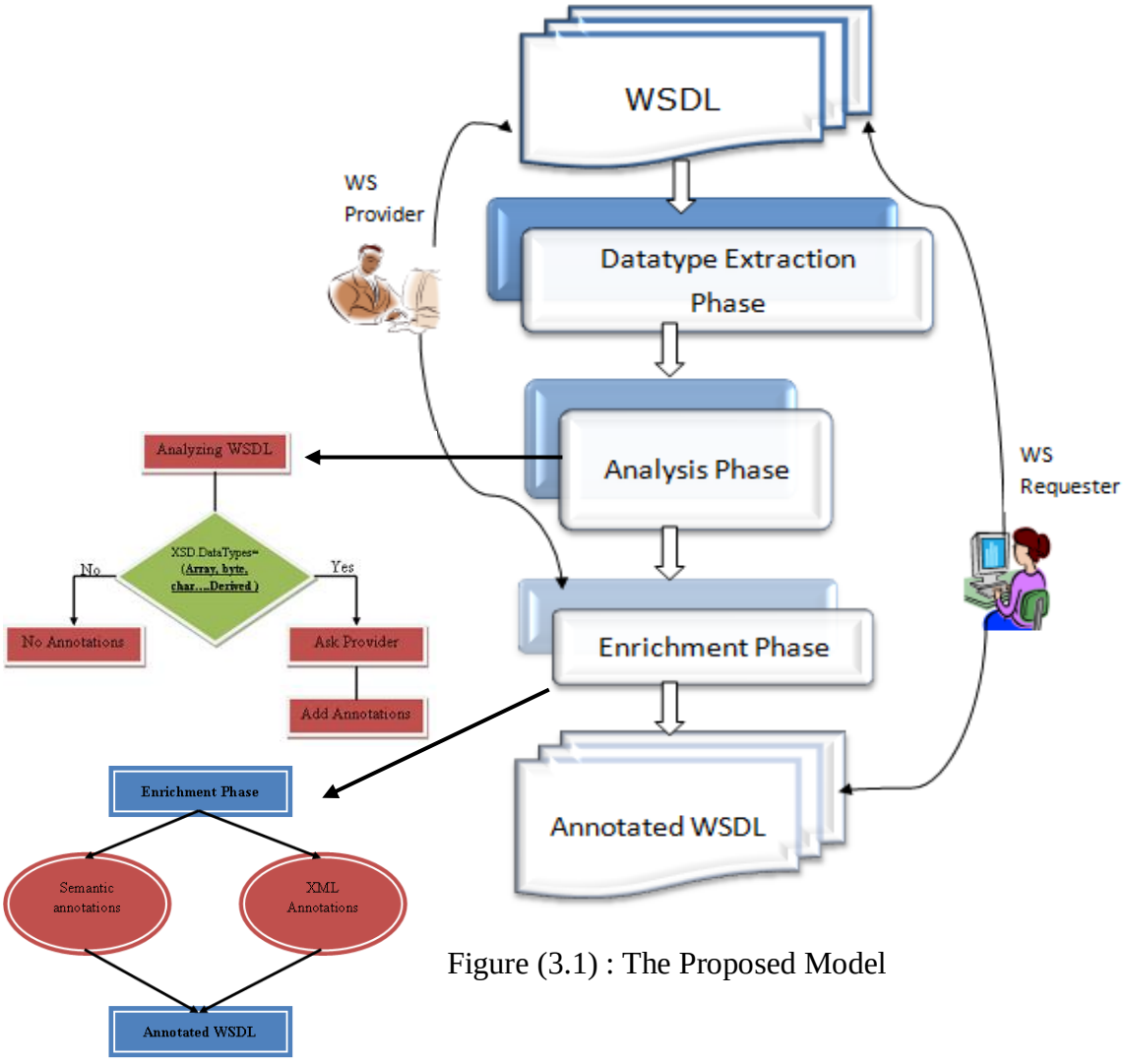


Figure (3.1) : The Proposed Model

Figure (3.1) based on the following abstract phases:

1. Extracting the XS based datatype specification inside a WSDL.
2. Analyzing the specification based on the producing platform and IDE in order to specify the needed enhancements.
3. Enhancing the data specification based on the previous analysis.
4. Producing an enhanced, semantic, understandable and reusable data specification for the Web service described by the analyzed WSDL.

The proposed approach analyzing the WSDL file , then extracting the parameters needed for the Web service and then transforming the WSDL file to Ontology Web Language For Services (OWL-S) document , then adding annotation that make the Web service more clear to the requester.

3.2 Extracting WSDL elements (including Datatypes) :

The first step we have to extract the WSDL document, WSDL documents are compulsorily published with Web Service; the provider cannot publish his own service application until its description (WSDL) generated, so that any developer or user wanting to know more about the operations or services then he can review the provided WSDL document.

There are many ways to extract WSDL document, but here we are looking to make our proposed approach to run automatically when the Web Service client, user, and also developer want to bind with the Web Service and in the final stage give him a clear and simple description for Web Service input/output parameters datatypes. The proposed approach extracts the WSDL document and then extracts WSDL elements and the XSD. Then the approach can distinguish between the input/output parameters datatypes which may need more description and constraints with which do not need.

3.3 WSDL DataTypes Descriptions

Web service provider publishing the application and using parameters , these parameters must clearly appear to the users without ambiguity; because any error in the filling of these parameters will lead to Web Service failure which we always seek to ensure does not happen. Therefore we are proposing an approach Extending Web service Datatypes specification to reach better comprehension and reusing the Web service functionality , which in turn leads users to operation understandability for all Web Services and also to determine all the parameters datatypes which Web Services need.

The W3C XML Schema Datatype Specification defined many datatypes for validating the content of the element and the values of the attribute. These datatypes using for validating only the scalar content of the XML elements, and not the mixed or non-scalar content. The text located between the <opening> and </closing> tags, and the attribute's value are often referred to as scalar data, or it could be a list of scalar data. These datatypes are designed for use in the definition of the XML Schema.

According to W3C , Datatypes divided into two categories :

1. **Primitive** datatypes are those that are not defined in terms of other datatypes, they are the primary dataTypess of the XSD, and acting as a base for defining the other datatypes in XSD. It contains only values and there is no attributes or elements.
2. **Derived** datatypes are those that are defined in terms of other datatypes, they are derived from primitive datatypes and they could be built in or user defined e.g. integer -> built in -> derived from -> decimal datatypes.

We will summarize all the primitive types and their description including a simple restriction used in its specification in the following table (There are the 19 primitive datatypes supported by the XML Schema Datatypes Specification) :

Primitive Types		
Name	**type**	**Description**
String	xs:string	A sequence of Unicode characters
Boolean	xs:boolean	Values (True OR False)
Decimal	xs:decimal	A rational number
Float	xs:float	
Double	xs:double	
Duration	xs:dateTime	An instant in time - known at least to the second and always includes a time zone.
URI	xs:anyURI	A Uniform Resource Identifier Reference. (Absolute OR Relative), may have an optional fragment identifier
Date		A date, or partial date Dates SHALL be valid dates. date is a union of the w3c schema types of date (gYearMonth and gYear).
DateTime	xs:dateTime,	A date, date-time or partial date. If hours and minutes are specified, a time zone shall be populated. and Seconds may be provided or may also be ignored. Dates shall be valid dates.
	xs:date,	
	xs:gYearMonth,	
	xs:gYear	
	xs:gMonth	
	xs:gDay	
	xs:gMonthDay	
Base64Binary	xs:base64Binary	A stream of bytes, Base64 are encoded
HexBinary	xs:hexbinary	represents arbitrary hex-encoded binary data, a set of finite-length sequences of binary octets
QName	xs:qname	QName represents XML qualified names. It is a set of tuples {namespace name, local part}, where namespace name is an anyURI and local part is an NCName.
NOTATION		represents the NOTATION attribute type, it is the set of QNames of notations declared in the current schema
Integer	xs:int	A signed 32-bit integer

Table (3.3) : Primitive types according to W3C (http://www.w3.org/TR/xmlschema-2/)

Next table summarizing all the derived datatypes, these datatypes represented as elements with a child with the name of the defined elements of the type (There are 25 built-in derived datatypes supported by XML Schema Datatypes) :

Derived Types		
Name	**type**	**Description**
NormalizedString	xs: normalizedString	set of strings that do not contain the carriage return, line feed nor tab characters
Token	xs: token	set of strings that do not contain the carriage return, line feed nor tab characters, that have no leading or trailing spaces and that have no internal sequences of two or more spaces
Language	xs: language	the set of all strings that are valid language identifiers
NMTOKEN	xs: NMTOKEN	The set of tokens that match theNmtoken production in XML
NMTOKENS	xs: NMTOKENS	the set of space-separated lists of tokens, of which each token is in the ·lexical space ofNMTOKEN
Name	xs: Name	the set of all strings which match the Name production of XML
NCName	xs: NCName	represents XML "non-colonized" Names
ID	xs: ID	represents the ID attribute type, An ID attribute must have a declared default of #IMPLIED or #REQUIRED
IDREF	xs: IDREF	IDREFS must match Names; each Name must match the value of an ID attribute on some element in the XML document
IDREFS	xs: IDREFS	the set of (finite and non-zero-length sequences) of IDREFs
ENTITY	xs: ENTITY	ENTITIES must match Names; each Name must match the name of an unparsed entitydeclared in the DTD
ENTITIES	xs: ENTITIES	the set of finite, non-zero-length sequences of ENTITYs that have been declared as unparsed entities in a document type definition.
Integer	xs:int	the infinite set {...,-2,-1,0,1,2,...}
NonPositiveInteger	xs: nonPositiveInteger	the infinite set {...,-2,-1,0}
NegativeInteger	xs: negativeInteger	the infinite set {...,-2,-1}
Long	xs: long	an optional sign followed by a finite-length sequence of decimal digits
Int	xs: Int	maxInclusive to be 2147483647 and ·minInclusive to be -2147483648.
Short	xs: short	maxInclusive to be 32767 and minInclusive to be -32768
Byte	xs: byte	maxInclusive to be 127 and minInclusive to be -128
NonNegativeInteger	xs: nonNegativeInteger	the infinite set {0,1,2,...}
UnsignedLong	xs: unsignedLong	
UnsignedInt	xs: unsignedInt	
UnsignedShort	xs: unsignedShort	the value of maxInclusive to be 65535
UnsignedByte	xs: unsignedByte	the value of maxInclusive to be 255
PositiveInteger	xs: positiveInteger	the infinite set {1,2,...}

Table (3.4) : Derived types according to W3C (http://www.w3.org/TR/xmlschema-2/)

Next table illustrated the Atomic datatypes and the derived datatypes :

Primitive Types		
Name	**Derived**	**Atomic**
String		√
Boolean		√
Decimal		√
Float		√
Double		√
Duration		√
URI		√
Date		√
DateTime		√
Base64Binary		√
HexBinary		√
QName		√
NOTATION		√
Derived Types		
Name	**Derived**	**Atomic**
NormalizedString	string	
Token	normalizedString	
Language	token	
NMTOKEN	token	
NMTOKENS	NMTOKENS	
Name	token	
NCName	Name	
ID	NCNAME	
IDREF	NCName	
IDREFS	IDREF	
ENTITY	NCName	
ENTITIES	ENTITY	
Integer	decimal	
NonPositiveInteger	integer	
NegativeInteger	nonPositive	
Long	integer	
Int	long	
Short	int	
Byte	short	
NonNegativeInteger	integer	
UnsignedLong	nonNegative	
UnsignedInt	unsignedLong	
UnsignedShort	unsignedInt	
UnsignedByte	unsignedShort	
PositiveInteger	nonNegativeInteger	

Table (3.5) : Atomic VS Derived DataTypes (D. Vint et al, 2003)

ASP.Net – Visual C# :

☒ **LISTS Specification :**

CLASS :

```
namespace WebServiceRaed
{
   public class Employee
   {
      public int ID { set; get; }
      public string Name { set; get; }
   }
}
```

Web Service :

```
public List<Employee> getEmployees()
      {
      List<Employee> I=new List<Employee>();
         I.Add(new Employee{ ID = 1, Name = "RAED" });
         I.Add(new Employee{ ID = 2, Name = "ABSI" });
         return I;
      }
```

Figure (3.2) : Lists programming in ASP.Net – Visual C#

XML Schema Datatypes (XSD) in WSDL file Corresponding to Lists programming in ASP.Net – Visual C# **:**

```
<xs:element name="getEmployees">
 <xs:complexType/ >
 </xs:element>
 <xs:element name="getEmployeesResponse">
 <xs:complexType>
 <xs:sequence>
 <xs:element minOccurs="0" maxOccurs="1" name="getEmployeesResult"
type="tns:ArrayOfEmployee/ ">
 </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:complexType name="ArrayOfEmployee">
 <xs:sequence>
 <xs:element minOccurs="0" maxOccurs="unbounded" name="Employee" nillable="true"
type="tns:Employee/ ">
 </xs:sequence>
 </xs:complexType>
```

Figure (3.3) : WSDL file for Lists programming ASP.Net – Visual C#

The dataType (List) ASP.Net-Visual C# Platform in WSDL file is defined as (ArrayOf……)

xs:complexType name="ArrayOfEmployee

☒ **Integer & String Specification :**

```
[WebMethod]
    public string Philadelphia()
    {
       return "Philadelphia University Jordan";
    }

    [WebMethod]
    public int add(int x, int y)
    {
       return x + y;
    }
```

Figure (3.4) : Int & String programming in ASP.Net – Visual C#

XML Schema Datatypes (XSD) in WSDL file Corresponding to Int & String programming in ASP.Net – Visual C# **:**

```
< <xs:element name="PhiladelphiaResponse">
 <xs:complexType>
 <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="1" name="PhiladelphiaResult" type="s:string" />
 </xs:sequence>
 </xs:complexType>
 </xs:element>
 <xs:element name="add">
 <xs:complexType>
 <xs:sequence>
  <xs:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />
  <xs:element minOccurs="1" maxOccurs="1" name="y" type="s:int" />
 </xs:sequence>
 </xs:complexType>
 </xs:element> >
```

Figure (3.5) : WSDL file for Integer & String programming ASP.Net – Visual C#

The dataType (Integer and String) in ASP.Net-Visual C# Platform in WSDL file is defined as (Integer … String) no change because it is a simple dataTypes .

xs:element minOccurs="0" maxOccurs="1" name="PhiladelphiaResult" type="s:string

xs: element minOccurs = "1" maxOccurs = "1" name = "x" type = "s:int

Windows Communication Foundation (WCF) :

We have found in the practical side within the building of the web service using WCF that the WCF does not show the complex types and showing just the operations in the WSDL document, and all the data structures are located in the XSD files which are linked to the WSDL document, and by copying the URL's into the browser we can see the complex type definitions.

☒ **Array & String Specification :**

```
[DataContract]
public class Test
{
   [DataMember(IsRequired = true)]
    public ArrTest[] array;
}

[DataContract]
public class ArrTest
{
   public DateTime? range1;
   public string range2;
}
```

Figure (3.6) : Array programming in WCF

XML Schema Datatypes (XSD) in WSDL file Corresponding to Array programming in WCF **:**

```
< <xs:complexType name="ArrayOfArrTest">
  <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="unbounded" name="array" nillable="true"
type="tns:ArrTest"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="InvoiceBalance">
  <xs:sequence>
    <xs:element name="range1" nillable="true" type="xs:dateTime"/>
    <xs:element name="range2" nillable="true" type="xs:string"/>
  </xs:sequence>
</xs:complexType>>
```

Figure (3.7) : WSDL file for Array programming in WCF

The dataType (Array) in WCF Platform in WSDL file is defined as (ArrayOf ……)

xs:complexType name="ArrayOfArrTest

xs: element minOccurs = "0" maxOccurs = "unbounded" name = "array" nillable="true" type="tns:ArrTest

from the same example we can see that the dataType (String) in WCF framework in WSDL file is defined as (String) no change because it is a simple dataTypes .

xs:element name="range2" nillable="true" type="xs:string

And here the problem , the object (ArrayOf ….)does not exist in the code where it is generated from, so if I want to construct a client code from the WSDL file , then the client doesn't know that this (ArrayOf …) is not true object at all.

PHP Web service Programming using (NetBeans IDE) :

☒ **Float Specification** :

During my searching on the internet it was very little persons developing web services using PHP , most of them using C# or ASP , but PHP used more in creating WSDL files as a bottom up approach (WSDL2PHP web service).

```
<?php
    class CR1{
   /**
     * sum1s two numbers.
     * @param float $R1
     * @param float $R2
     * @return float
     */
    public function sum1($R1, $R2) {
      return ($R1+$R2);
    }
   }
 ?>
```

Figure (3.8) : Float programming in PHP

XML Schema Datatypes (XSD) in WSDL file Corresponding to Float programming in PHP **:**

```
<message name="sum1">
      <part name="R1" type="xsd:float"></part>
      <part name="R2" type="xsd:float"></part>
    </message>
    <message name="sum1Response">
      <part name="sum1Return" type="xsd:float"></part>
    </message>
```

Figure (3.9) : WSDL file for Float programming in PHP

The dataType (Float) in PHP Platform in WSDL file is defined as (Float) no change because it is a simple dataTypes .

<part name="R1" type="xsd:float"></part>

part name="sum1Return" type="xsd:float

☒ **Array Specification** :

```
<?php
class PHPARR {
   public $par1 = array();
   public $par2;
   /**
    * Making a PHParray.
     * @param mixed $R1
    * @param mixed $R2
    * @return array
    */
   public function PHPArray ($R1, $R2) {
      return array($R1, $R2);
   }
} ?>
```

Figure (3.10) : Array programming in PHP

```
<message name="PHPArrayIn">
<part name="R1" type="xsd:anyType"/>
<part name="R2" type="xsd:anyType"/>
</message>

<message name="PHPArrayOut">
<part name="return" type="soap-enc:Array"/>
</message>
```

Figure (3.11) : WSDL file for Float programming in PHP

The dataType (Array) in PHP Platform in WSDL file is defined as (**AnyType**)

<part name="R1" type="xsd:anyType"/>

XSD:AnyType :

To store any type not of the primitive types in XML , WSDL using the xsd:anyType, All the primitives dataTypes are derivatives of this Type and it is used as any XML schema complex type.

Java Web service Programming in (Eclipse & Netbeans axis2 support):

☒ **Integer & String Specification** :

```
public abstract class Def{
    public Def() {}
    public int R1;
    private int R2;

    private int[] R3;
  }
  public class Camefrom extends Def{
    public int R4;
    private string R5;
  }
```

Figure (3.12) : Integer & string programming in Java

XML Schema Datatypes (XSD) in WSDL file Corresponding to Integer & string programming in WCF **:**

```
<xsd:complexType name="Def" abstract="true">
    <xsd:sequence>
        <xsd:element name="R1" type="xsd:int"/>
        <xsd:element name="R2" type="xsd:int"/>
        <xsd:element name="R3" minOccurs="0" maxOccurs="unbounded"
type="xsd:int"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Camefrom">
    <xsd:complexContent>
        <xsd:extension Def="ns:Def">
            <xsd:sequence>
                <xsd:element name="R4" type="xsd:int"/>
                    <element name="R5" nillable="true" type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Figure (3.13) : WSDL file for Integer & string programming in Java

The dataTypes (Integer & String) in Java Platform in WSDL file is defined as (int .. String) no change because it is a simple dataTypes

xsd:element name="R4" type="xsd:int

element name="R5" nillable="true" type="xsd:string

☒ **Char Specification** :

```
<xs:complexType name="charExample">
    <xs:sequence>
        <xs:element name="arg0" type="xs:unsignedShort"  minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="charExampleResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:unsignedShort"/>
    </xs:sequence>
</xs:complexType>
```

Figure (3.14) : WSDL file for Char programming in Java

The dataType (char) in Java Platform in WSDL file is defined as (**unsignedShort**)

xs: element name = "return" type = "xs: unsignedShort

in this section we illustrated a comparisons between platforms which building the Web Service (Asp.Net, Java, PHP, WCF, C#, Net Beans, Eclipse) and we found that there are two categories of the dataTypes : primitive DataType and Complex DataTypes (Drived), the first category is simple and the WSDL file can express it without any difficulties, the second category is complex, WSDL file cannot express it as simple as the first category, so WSDL using another Types like (AnyTpe, ArrayOf……, Unsigned….) to build the XML file, These differences create misunderstandings for the Web Services requesters, clients, users and also developers because these datatypes are

not implemented in the same and formal way as we have seen in *char* datatype Figure 3.16. But here in our thesis we suggest to implement our proposed approach on .NET tool as case study.

3.4 Proposed solutions for XSD DataTypes :

In this thesis We proposed more than one solution to extend the web service DataTypes in WSDL file , three solutions are proposed :

1. Using unified modeling language (UML) by a graphical definitions for the web service (discussed by (Alshraideh F, 2013)) .
2. XML Annotations
3. Semantic Annotations using (OWL-S)

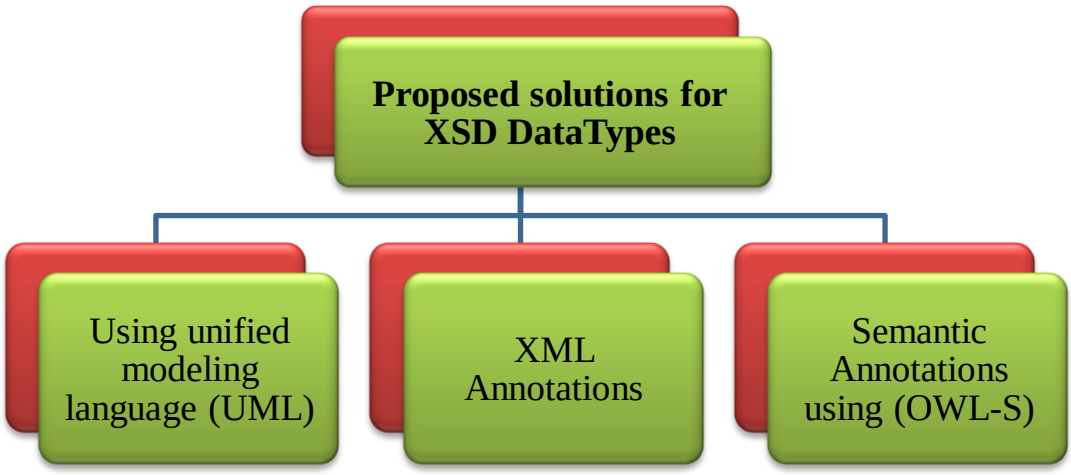


Figure (3.15) : Proposed solutions for XSD DataTypes

Now we can draw the Enrichment Phase in the proposed model clearly assigning the two proposed solutions as in Figure 3.16

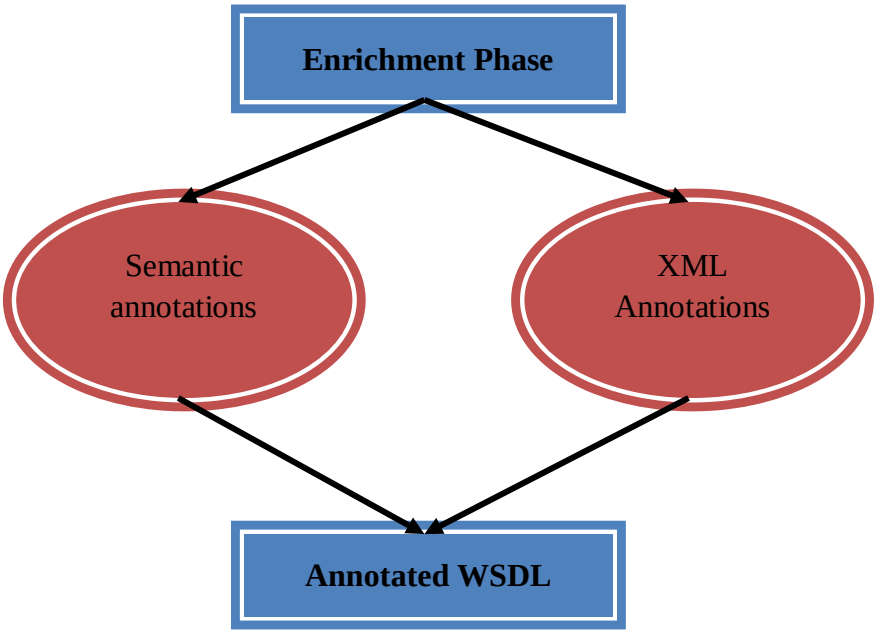


Figure (3.16) : Enrichment Phase in the proposed model

When to make Annotations ?

If the WSDL file have a simple dataTypes then no annotation to be added. The enrichment part will have the same implementation for datatype as it is in original WSDL document with no annotations. Otherwise the approach will back to Web Service provider by sending to him an message as interface, asking him to select from a datatypes list which datatype he given for the operation which written its name in the interface. After the provider select the parameter datatype then the approach can add the selected parameter datatype to the enrichment schema .

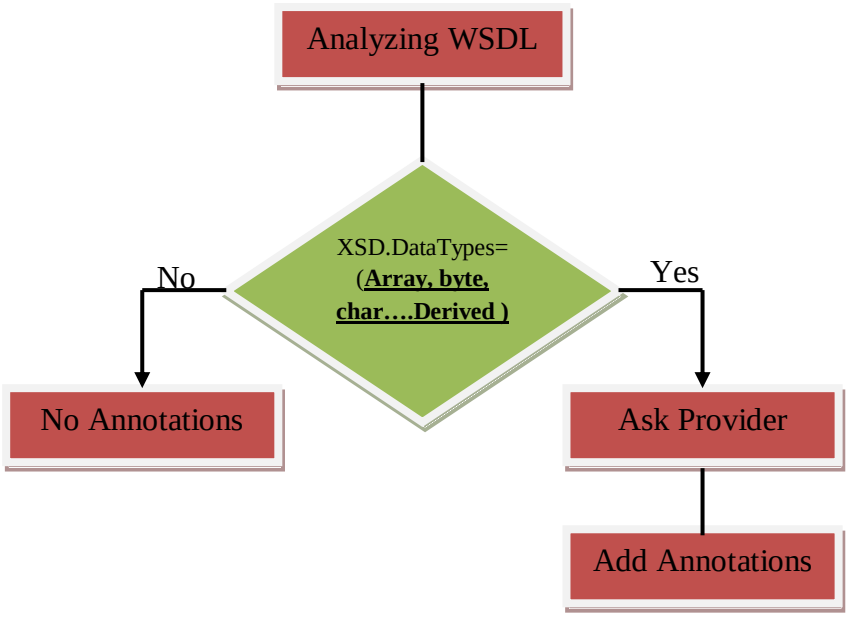


Figure (3.17) : When to make Annotations ?

But what if the service Provider did not answer, or his answer needs more time ?

Here we can make detection against the Web Service, there are many algorithms searching on detecting web service e.g (L.Carolin et.al 2007) he stated that detection could be executed

1. fully automatically without human intervention
2. semi automatically with human feed back
3. manually by human programmer.

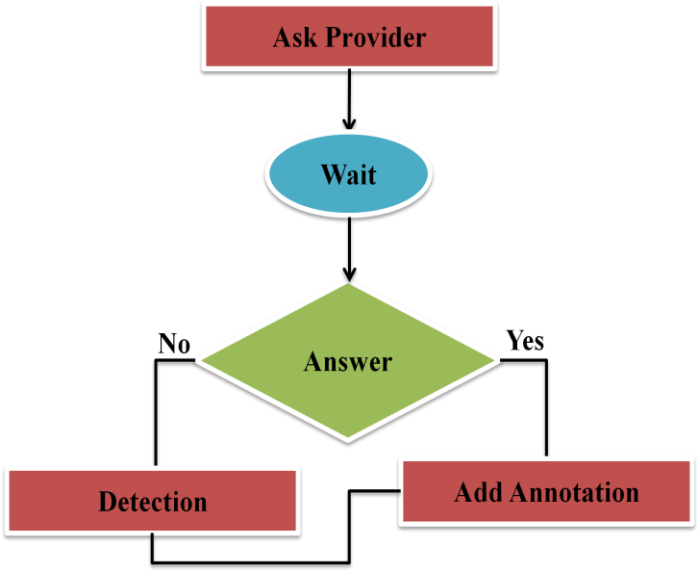


Figure (3.18) : Detecting

3.5 Semantic Annotations :

There are many Semantic languages can help us to annotate the WSDL file , for example we can use Resource Description Framework (RDF), Web Ontology Language (OWL), Semantic Annotations for WSDL (SAWSDL) and so on. We will use OWL-S language, this ontology is built on top of the OWL describes the services of the semantic web, and it is widely used and proposed by W3C. OWL-S language is convenient to the problem of web services inter-operability and composition, for the representation and the description of the web service and the request.

To make a semantic annotation using OWL language we can use the <Annotation> element :

- Element : Annotation

  This element is placeholder for more than one way of annotations such as <Label> and <Documentation>. specially, it can take the element (xsd:any) See Figure (3.19).

  - ✓ Element **< label >** : it provides a human-readable name for the annotated element , See Figure (3.20).
  - ✓ Element **< Documentation >** it provides a human-readable description for the annotated element, See Figure (3.21).

```
<Annotation>
 Content: ( Label | Documentation | xsd:any )
</Annotation>
```

Figure (3.19) : Header of the Annotation element

```
<owlx:Annotation>
    <owlx:Documentation>Using Semantic OWL ontology</owlx:Documentation>
</owlx:Annotation>
```

Figure (3.20) : Element < Documentation >

```
<owlx:Annotation>
  <owlx:Label>  Semantic  OWL </owlx:Label>
</owlx:Annotation>
```

Figure (3.21) : Element < label >

Now we can use this element to dealing with the types of Datatypes in the WSDL file to make semantic annotations according to the classification we did in our approach.

3.6 Case 1 : primitive datatypes

In this case, the datatypes are implemented in a formal way and the datatypes are implemented as it is without any changes, so there is no need for any enrichment. The new WSDL document generated by our proposed approach will have the same XSD datatypes without any modification to the original WSDL document. The enrichment part will have the same implementation for the datatypes with no changes, as the datatypes are primitive and no need for annotations.

The next example shows how the .NET tool implements *Integer and String* datatypes as a case study and also shows how the proposed approach deals with this case

Example :

Integer & Sting in ASP.Net – Visual C#

```
[WebMethod]
    public string Philadelphia()
    {
       return "Philadelphia University Jordan";
    }

    [WebMethod]
    public int add(int x, int y)
    {
       return x + y;
    }
```

Figure (3.22) : Int & String programming in ASP.Net – Visual

XML Schema Datatypes (XSD) in WSDL file Corresponding to Int & String programming in ASP.Net – Visual C# as Figure (3.23) :

```
< <xs:element name="PhiladelphiaResponse">
        <xs:complexType>
                <xs:sequence>
 <xs:element minOccurs="0" maxOccurs="1" name="PhiladelphiaResult" type="s:string" />
                </xs:sequence>
        </xs:complexType>
 </xs:element>
 <xs:element name="add">
        <xs:complexType>
                <xs:sequence>
 <xs:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />
 <xs:element minOccurs="1" maxOccurs="1" name="y" type="s:int" />
                </xs:sequence>
        </xs:complexType>
 </xs:element> >
```

Figure (3.23) : WSDL file for Integer & String programming ASP.Net – Visual C#

The dataType (Integer and String) in ASP.Net-Visual C# Platform in WSDL file is defined as (Integer … String) no change because it is a simple dataTypes .

xs:element minOccurs="0" maxOccurs="1" name="PhiladelphiaResult" type="s:string

xs: element minOccurs = "1" maxOccurs = "1" name = "x" type = "s:int

The proposed approach will firstly extract the WSDL document and then extract the XSD part, and finally check if the datatype is primitive or not. In this example the approach will skip the third and forth steps of our proposed model because there is no need for any annotations or constraints. The parameter (*Integer OR String*) is given its type *Integer Or String* without any ambiguity. The following steps summarize how the approach working :

Step 1:

Extract the WSDL document for the (public int add(int x, int y) method and for the String Datatype ,which shown in Figure 3.23 .

Step 2:

Extract the parameter datatypes XSD as:

a. < xs:element minOccurs="1" maxOccurs="1" name="x**" type="xs:int"** /> **(**Input parameter).

b. < xs:element minOccurs="1" maxOccurs="1" name="x**" type="xs:int"**/> (Output parameter).

c. < ***xs:element minOccurs="0" maxOccurs="1" name="PhiladelphiaResult" type="s:string*** />

In this phase the approach can be distinguished that these are a primitive Datatypes and don't needs more description because it is simple and clear and the requester can know that it is *Integer Or String* datatypes as it is.

Step 3:

No annotations to be added. The enrichment part will have the same implementation for datatype as it is in original WSDL document with no annotations.

3.7 Case 2 : derived datatypes

In this category, datatypes cannot be addressed until back to the Web Service provider itself. The approach can execute step 1 and step 2 and then checking about the datatype classification. In the previous category (primitive datatypes) the approach can address the problem automatically; but here it stops and asks the Web Service provider about which datatypes the provider specified for Web Service operation parameter datatypes.

Example :

Array & String Programming in WCF

```
[DataContract]
public class Test
{
   [DataMember(IsRequired = true)]
    public ArrTest[] array;
}

[DataContract]
public class ArrTest
{
   public DateTime? range1;
   public string range2;
}
```

Figure (3.24) : Array programming in WCF

XML Schema Datatypes (XSD) in WSDL file Corresponding to Array programming in WCF **:**

```
< <xs:complexType name="ArrayOfArrTest">
        <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="unbounded" name="array" nillable="true"
type="tns:ArrTest"/>
         </xs:sequence>
</xs:complexType>
        <xs:complexType name="InvoiceBalance">
                <xs:sequence>
          <xs:element name="range1" nillable="true" type="xs:dateTime"/>
                 </xs:sequence>
        </xs:complexType>>
```

Figure (3.25) : WSDL file for Array programming in WCF

The dataType (Array) in WCF Platform in WSDL file is defined as (ArrayOf ……)

xs:complexType name="ArrayOfArrTest

xs: element minOccurs = "0" maxOccurs = "unbounded" name = "array" nillable="true" type="tns:ArrTest

And here the problem , the object (ArrayOf ….)does not exist in the code where it is generated from, so if I want to construct a client code from the WSDL file , then the client doesn't know that this (ArrayOf …) is not true object at all.

Both of *list* and *array* are defined in the same way (*ArrayOf…….*, *ArrayOf…….*), both of them are defined as an array datatype. The question here is how can the user understand which type of data the operations needs, and how can the user distinguish between the array datatype and list datatype? So that the proposed model can answer these questions by referring to the service provider itself to determine the specific datatype, and then presenting it for a requester in a simple and clear way. The following steps summarize how the approach working:

Step 1:

Extract the WSDL document for Web Service. The example in Figure 3.27 illustrates this step.

Step 2:

Extract the parameters datatypes XSD as:

a. < xs:complexType name="**ArrayOfArrTest** *"/>*

b. < ***xs:element minOccurs="0" maxOccurs="unbounded" name="array"***

nillable="true" type="tns:ArrTest *"/>*

Step 3:

Now, the approach will back to Web Service provider by sending to him a message as interface, asking him to select from a datatypes list which datatype he given for the operation which written its name in the interface. If the provider didn't answer we can make detection against the web service as mentioned in Fig(3.18). After the provider selects the parameter datatype then the approach can add the selected parameter datatype to the enrichment schema.

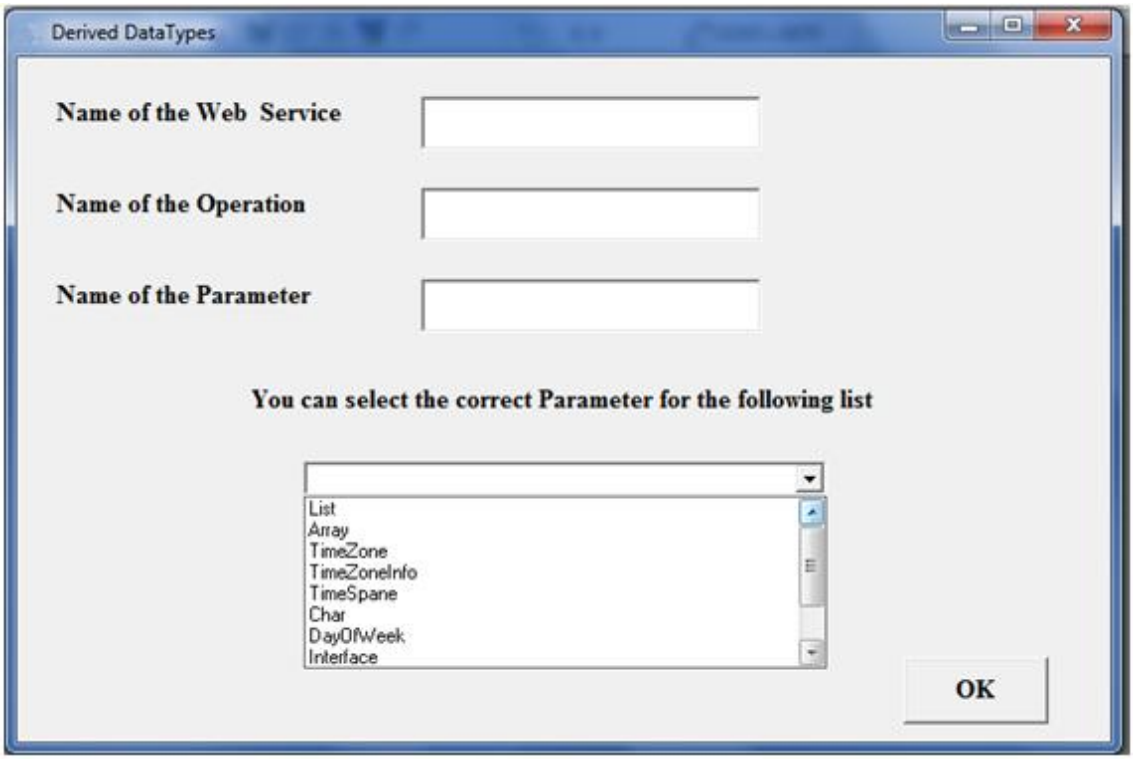


Figure (3.26): an interface providing the DataTypes

Step 4:

Now we can mapping between the chosen parameter and the semantic one , this operation could be done by using if statement or by storing the OWL-S semantic statements in a table to simplified the mapping process.

Step 5:

Adding the new Annotation element to the WSDL file in the right place according to the operation name and the parameters in this operation, thus, a new Annotated WSDL file will be created .

```
<owlx:Annotation>
    <owlx:Documentation>this parameter is Array  </owlx:Documentation>
</owlx:Annotation>
```

Figure (3.27) : Element < Documentation >

This Documentation element provides a human-readable description for the annotated element.

Here In Figure (3.28) an Figure (3.29) we can understand the documentation element how it works, the code in .Net programming is

```
[WebMethod(Description = "Identifying Your DataTypes from the Providers ")]
public List<Employee> getEmployees()
```

Figure (3.28) : Code generated in .Net - WebMethod

The part of WSDL file containing the documentation element here in Figure (3.29)

```
< wsdl:operation name="getEmployees">
    < wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
Identifying Your DataTypes from the Providers
    </wsdl:documentation>
    < wsdl:input message="tns:getEmployeesSoapIn" />
    < wsdl:output message="tns:getEmployeesSoapOut" />
</wsdl:operation>
```

Figure (3.29) : WSDL part for documentation element

3.8 : Summary

In this chapter I have described the problem that this thesis trying to solve, the proposed model and how to use my model in solving this problem using semantic annotations, starting from the analyzing the WSDL file and extracting the datatypes used in it, then classifying of the datatypes used in WSDL files to a primitive and derived datatypes, and presented an example for each one in different platforms, and taking a case study for every one showing how my proposed model deals with it. The part of the WSDL file resulted from the proposed approach gives the user a clear look to the datatypes used in it without any ambiguity or complexity with the help of the provider.

CHAPTER FOUR

IMPLEMENTATION AND EVALUATION

In this chapter I will introduce the implementation of the approach taking into account the two cases mentioned in the previous chapter, and how to deal with this cases, then I will introduce a case study as an evaluation to my work.

4.1 Implementation

The approach running automatically when the provider want to bind the web service to do its process , this approach will be executed according to the following pseudocode in Figure (4.1) :

```
Function Main
    Open " WSDL " file for output
    While not EOF do:
        Read every line in " WSDL "
        Display (Operation Name, XSD_Type)
        If XSD_Type ={String OR Boolean OR Decimal OR Int OR Float OR
           Double OR DateTime … etc } then No changes will done.
        Elseif  XSD_Type={AntType OR ArrayOF* OR UnsignedShort } then
            Call: Function Derived_XSD with Operation Name and  XSD_Type
            Call: Function Annotation with String
    End While
Endfunction
```

Figure (4.1) : the Proposed algorithm in PseudoCode

This algorithm starting in opening the WSDL file for writing and then reading and extracting the Operation name and its DataTypes . we can see here the tow cases in If statements, where is the first IF statement telling us that the primitive datatypes will do nothing in the Annotated WSDL file , but the next IF statement Calls the function

named (Derived_XSD) which ask the provider about the type of the used DataType here as we will see next in Figure (4.2).

```
Function Derived_XSD(string)
        messageBox contains
        "Please Select the Parameter from the following list of DataTypes"
        ListOfDataTypes
        Return string
Endfunction
```

Figure (4.2) : Provider DataTypes Determination

This function ask the service provider to determine the type of the unknown Parameter used in WSDL file , and this is done by a list of Datatypes and just click on the suitable parameter to back to the main function with the known parameter.

Next step is calling the function Annotation as in the following Figure (4.3)

```
Function Annotation( )
        Open tag        Print "owlx:Annotation"
                Open tag        Print " owlx:Documentation"
                        Print  String
                Close tag       Print " owlx:Documentation"
        Close tag       Print " owlx:Annotation"
Endfunction
```

Figure (4.3) : Adding Semantic Annotation to WSDL file

This function opening tags of Annotation element and the opening the documentation tag in it , then it print the Semantic annotation , after that a closing tags are printed to both of the documentation and Annotation elements .

Next we can see a part of Code of implementation in Figure (4.4) :

```
define('DS' , DIRECTORY_SEPARATOR);
        $fieldNameAtr = 'name';
        $fieldTypeAtr = 'type';
        $wsdlPath = 'wsdl.files' . DS . 'WSDLs' . DS;
                $dataTypes = array(
                'integer' => 'integer integer integer integer integer integer integer',
                'nonNegativeInteger' => 'nonNegativeInteger nonNegativeInteger
                nonNegativeInteger nonNegativeInteger ',
                'struc' => 'struc strucstruc struc struc struc struc',
                'string' => 'string string string string string',
                'notype' => 'notype notype notype notypenotypenotype',
                'int' => 'int int int int'
        );
        // for development usage:
        define('DIR_PATH' , $wsdlPath);
```

Figure (4.4) : Cod of implementation

4.2 Evaluation

We will evaluate our work using a case study, Case studies can be particularly very good for understanding how different elements are suitable together and how different elements (implementation, context and other factors) have produced the observed impacts.

Rather than using large samples to examine a limited number of variables, case study methods involve an in-depth, longitudinal examination of a single instance or event.

So that we will illustrate now two examples as a case study of a WSDL files , these files we analyzing them and extracting their services , SOAP bindings(Operation bindings with the input and output binding), PortTypes(Operations and their inputs and outputs), and messages they have(datatypes), then adding the annotations to the unknown DataTypes.

WSDL Analyzer :

WSDL Example(1) :

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://tempuri.org/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/">
      <s:element name="GetFamilyInfoByDiaryId">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="id"
type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetFamilyInfoByDiaryIdResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
name="GetFamilyInfoByDiaryIdResult" type="tns:FamilyS" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="FamilyS">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="id"
type="s:int" />
          <s:element minOccurs="0" maxOccurs="1" name="DoorCard"
type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="GetFamilyInfoByDiaryIdSoapIn">
    <wsdl:part name="parameters"
element="tns:GetFamilyInfoByDiaryId" />
  </wsdl:message>
  <wsdl:message name="GetFamilyInfoByDiaryIdSoapOut">
    <wsdl:part name="parameters"
element="tns:GetFamilyInfoByDiaryIdResponse" />
```

```
  </wsdl:message>
  <wsdl:portType name="FamilyServiceSoap">
    <wsdl:operation name="GetFamilyInfoByDiaryId">
      <wsdl:input message="tns:GetFamilyInfoByDiaryIdSoapIn" />
      <wsdl:output message="tns:GetFamilyInfoByDiaryIdSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="FamilyServiceSoap"
type="tns:FamilyServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
/>
    <wsdl:operation name="GetFamilyInfoByDiaryId">
      <soap:operation
soapAction="http://tempuri.org/GetFamilyInfoByDiaryId"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="FamilyServiceSoap12"
type="tns:FamilyServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
/>
    <wsdl:operation name="GetFamilyInfoByDiaryId">
      <soap12:operation
soapAction="http://tempuri.org/GetFamilyInfoByDiaryId"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="FamilyService">
    <wsdl:port name="FamilyServiceSoap"
binding="tns:FamilyServiceSoap">
      <soap:address
location="http://www.efamily.cn/WebService/FamilyService.asmx" />
    </wsdl:port>
    <wsdl:port name="FamilyServiceSoap12"
binding="tns:FamilyServiceSoap12">
      <soap12:address
location="http://www.efamily.cn/WebService/FamilyService.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

```
<wsdl:service name="FamilyService">
    <wsdl:port name="FamilyServiceSoap" binding="tns:FamilyServiceSoap">
      <soap:address
location="http://www.efamily.cn/WebService/FamilyService.asmx" />
    </wsdl:port>
    <wsdl:port name="FamilyServiceSoap12" binding="tns:FamilyServiceSoap12">
      <soap12:address
location="http://www.efamily.cn/WebService/FamilyService.asmx" />
    </wsdl:port>
  </wsdl:service>
```

Service ：

Name : FamilyService

Port : FamilyServiceSoap

Port : FamilyServiceSoap12

```
<wsdl:binding name="FamilyServiceSoap" type="tns:FamilyServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetFamilyInfoByDiaryId">
      <soap:operation soapAction="http://tempuri.org/GetFamilyInfoByDiaryId"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

SOAP Binding ：

Name : FamilyServiceSoap

Operation Binding : GetFamilyInfoByDiaryId

Input Binding

Output Binding

```
<wsdl:binding name="FamilyServiceSoap12" type="tns:FamilyServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetFamilyInfoByDiaryId">
      <soap12:operation
soapAction="http://tempuri.org/GetFamilyInfoByDiaryId" style="document" />
      <wsdl:input>
        <soap12:body use="literal" /
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" /
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

SOAP 1.2 Binding ：

Name : FamilyServiceSoap12

Operation Binding : GetFamilyInfoByDiaryId

Input Binding

Output Binding

```
<wsdl:portType name="FamilyServiceSoap">
    <wsdl:operation name="GetFamilyInfoByDiaryId">
      <wsdl:input message="tns:GetFamilyInfoByDiaryIdSoapIn" />
      <wsdl:output message="tns:GetFamilyInfoByDiaryIdSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
```

PortType ：

Name : FamilyServiceSoap

Operation: GetFamilyInfoByDiaryId

Input

Output

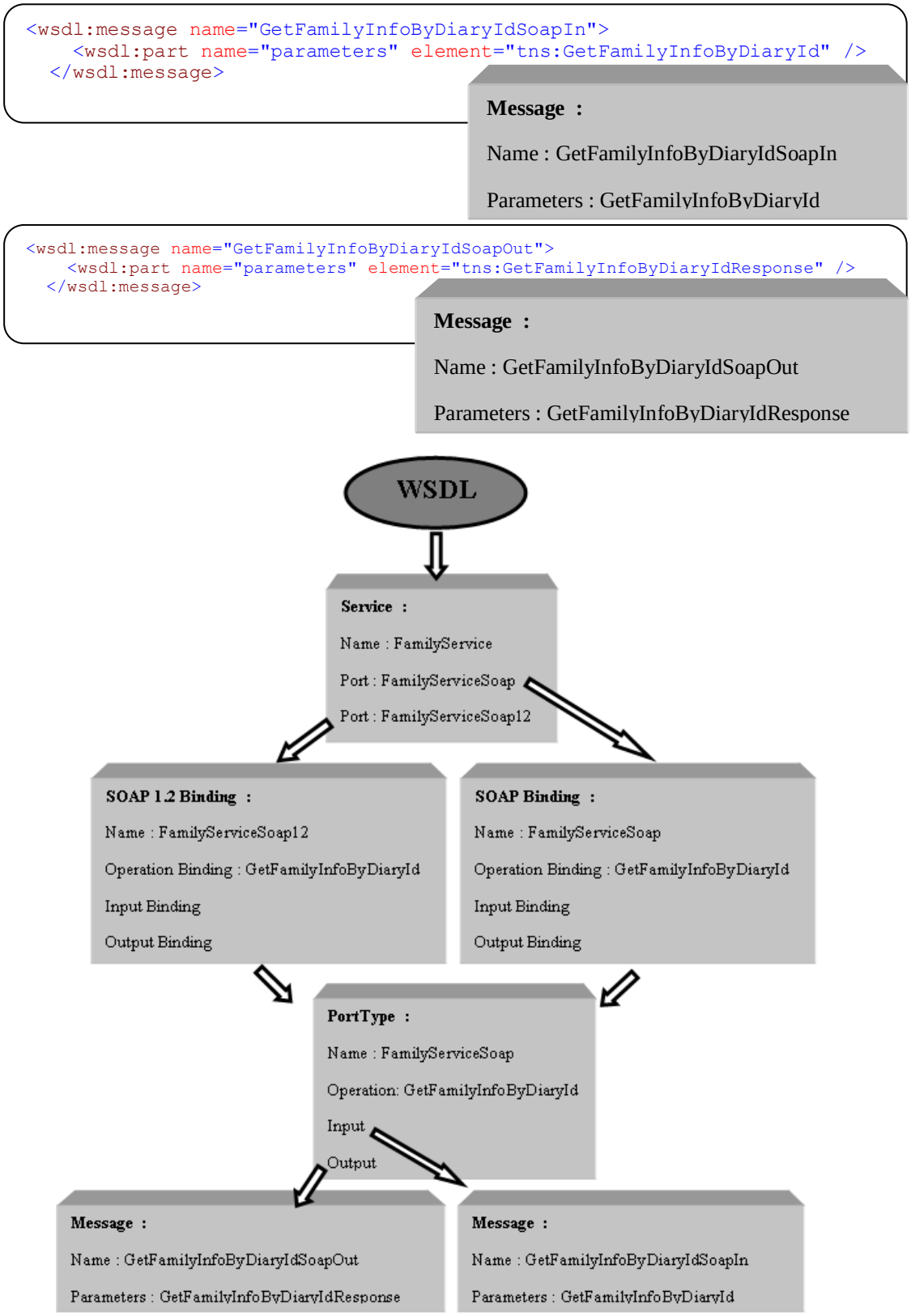


Figure (5.5) : Analyzing Example(1) WSDL file

Form this example we can now extract the XSD types :

Primitive (simple) XSD types		
Element Name		**Type**
GetFamilyInfoByDiaryId	Id	int

Element Name		**Type**
FamilyS	Id	int
	DoorCard	string

Table (4.1) : Primitive and Complex XSD types for Example(1)

All the DataTypes used here is Primitive DataTypes

Here no changes to the WSDL file since there is no derive DataTypes (AnyType, Unsignedshort, ArrayOfObject)

WSDL Example(2) :

```
<?xml version="1.0"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://tempuri.org/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

    <wsdl:types>s:schema elementFormDefault="qualified"
targetNamespace="http://tempuri.org/">
      <s:element name="GetAllNames">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="prefixText"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="count"
type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetAllNamesResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
name="GetAllNamesResult" type="tns:ArrayOfString" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfString">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded"
name="string" nillable="true" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="GetAllNamesSoapIn">
    <wsdl:part name="parameters" element="tns:GetAllNames" />
  </wsdl:message>
  <wsdl:message name="GetAllNamesSoapOut">
    <wsdl:part name="parameters" element="tns:GetAllNamesResponse"
/>
  </wsdl:message>
  <wsdl:portType name="AutoSuggestDoctorNameWebServiceSoap">
    <wsdl:operation name="GetAllNames">
      <wsdl:input message="tns:GetAllNamesSoapIn" />
      <wsdl:output message="tns:GetAllNamesSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
```

```
      <wsdl:binding name="AutoSuggestDoctorNameWebServiceSoap"
type="tns:AutoSuggestDoctorNameWebServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
/>
    <wsdl:operation name="GetAllNames">
      <soap:operation soapAction="http://tempuri.org/GetAllNames"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="AutoSuggestDoctorNameWebServiceSoap12"
type="tns:AutoSuggestDoctorNameWebServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
/>
    <wsdl:operation name="GetAllNames">
      <soap12:operation soapAction="http://tempuri.org/GetAllNames"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="AutoSuggestDoctorNameWebService">
    <wsdl:port name="AutoSuggestDoctorNameWebServiceSoap"
binding="tns:AutoSuggestDoctorNameWebServiceSoap">
      <soap:address
location="http://www.plasticsurgery.com/services/AutoSuggestDoctorNa
me.asmx" />
    </wsdl:port>
    <wsdl:port name="AutoSuggestDoctorNameWebServiceSoap12"
binding="tns:AutoSuggestDoctorNameWebServiceSoap12">
      <soap12:address
location="http://www.plasticsurgery.com/services/AutoSuggestDoctorNa
me.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

```
<wsdl:service name="AutoSuggestDoctorNameWebService">
    <wsdl:port name="AutoSuggestDoctorNameWebServiceSoap"
binding="tns:AutoSuggestDoctorNameWebServiceSoap">
      <soap:address
location="http://www.plasticsurgery.com/services/AutoSuggestDoctorName.asmx" />
    </wsdl:port>
    <wsdl:port name="AutoSuggestDoctorNameWebServiceSoap12"
binding="tns:AutoSuggestDoctorNameWebServiceSoap12">
      <soap12:address
location="http://www.plasticsurgery.com/services/AutoSuggestDoctorName.asmx" />
    </wsdl:port>
  </wsdl:service>
```

Service :

Name : AutoSuggestDoctorNameWebService

Port : AutoSuggestDoctorNameWebServiceSoap

```
<wsdl:binding name="AutoSuggestDoctorNameWebServiceSoap"
type="tns:AutoSuggestDoctorNameWebServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetAllNames">
      <soap:operation soapAction="http://tempuri.org/GetAllNames"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="AutoSuggestDoctorNameWebServiceSoap12"
type="tns:AutoSuggestDoctorNameWebServiceSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetAllNames">
      <soap12:operation soapAction="http://tempuri.org/GetAllNames"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

SOAP binding :

Name : AutoSuggestDoctorNameWebServiceSoap
Operation Bining :
Name : GetAllNames
Input binding : Output binding:

Operation Bining :
Name : AutoSuggestDoctorNameWebServiceSoap12
Input binding : Output binding:

```
<wsdl:portType name="AutoSuggestDoctorNameWebServiceSoap">
    <wsdl:operation name="GetAllNames">
      <wsdl:input message="tns:GetAllNamesSoapIn" />
      <wsdl:output message="tns:GetAllNamesSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
```

PortType :

Name : AutoSuggestDoctorNameWebServiceSoap
Operation:
Name : GetAllNames
Input:
Output:

```
<wsdl:message name="GetAllNamesSoapIn">
    <wsdl:part name="parameters" element="tns:GetAllNames" />
  </wsdl:message>
```

Message :

Name : GetAllNamesSoapIn

```
<wsdl:message name="GetAllNamesSoapOut">
    <wsdl:part name="parameters" element="tns:GetAllNamesResponse" />
  </wsdl:message>
```

Message :

Name : GetAllNamesSoapOut

```
<wsdl:types>s:schema elementFormDefault="qualified"
  targetNamespace="http://tempuri.org/">
      <s:element name="GetAllNames">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="prefixText"
  type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="count" type="s:int"
  />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetAllNamesResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetAllNamesResult"
  type="tns:ArrayOfString" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ArrayOfString">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded" name="string"
  nillable="true" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
```

types :

Name : VerNoticiaResponse
prefixText: xsd : string
count: xsd : int
GetAllNamesResult: xsd : ArrayOfString

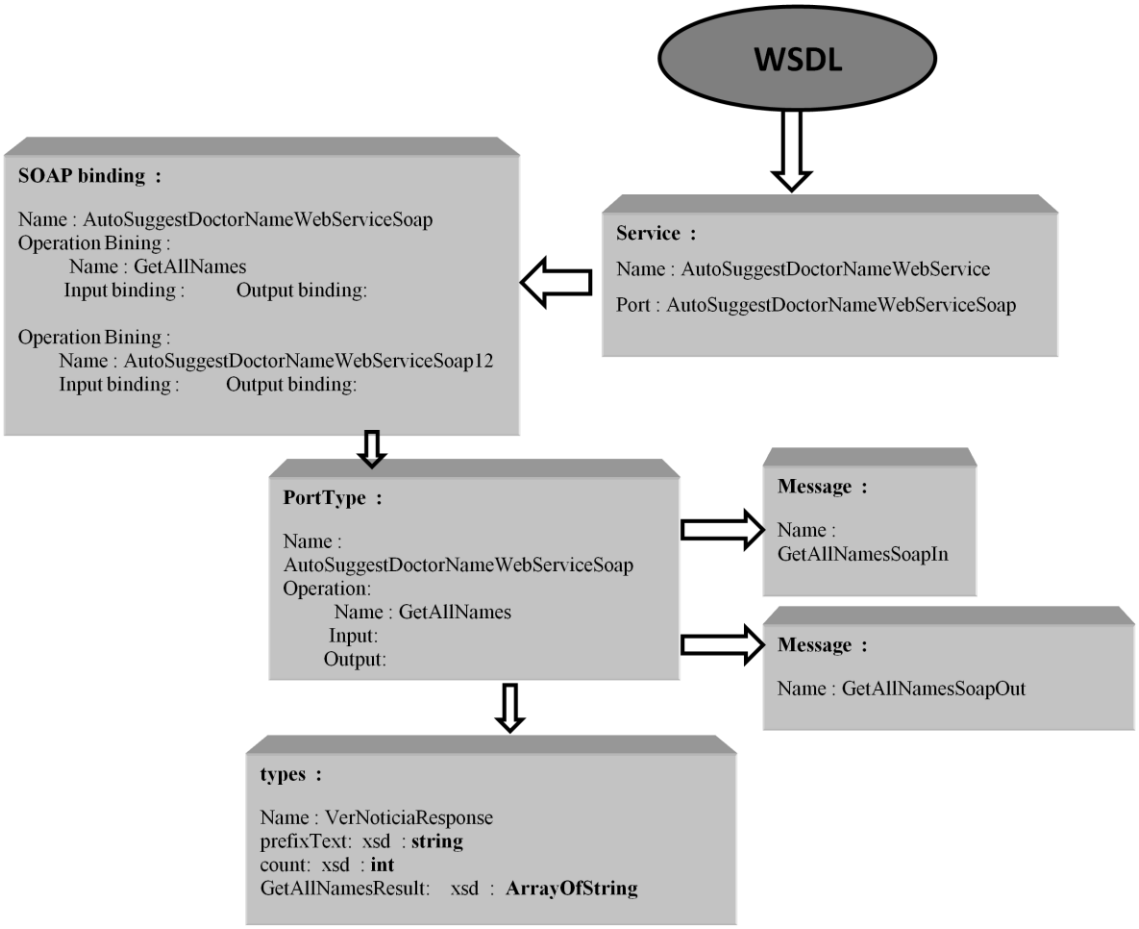


Figure (4.6) : Analyzing Example(2) WSDL file

Form this example we can now extract the XSD types :

Complex XSD types		
Element Name		**Type**
VerNoticiaResponse	prefixText	string
	count	int
VerNoticia		AnyType
GetAllNamesResult		ArrayOfString

Table (4.2) : Primitive and Complex XSD types for Example(2)

We can see here there are primitive and derived DataTypes , No changes to the primitive DataTypes , But the derived DataTypes, we will ask the service provider to determine the intended DataType .

Next step, a semantic annotation will be inserted in the WSDL file clarifying the type of this ataType

Service Provider response : ArrayOf String ➔ Array

The code here in WSDL file will be

<owlx:Annotation>

<owlx:Documentation> ***Array*** *</owlx:Documentation>*

</owlx:Annotation>

Service Provider response : AnyType ➔ Byte

The code here in WSDL file will be

<owlx:Annotation>

<owlx:Documentation> ***Byte*** *</owlx:Documentation>*

</owlx:Annotation>

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

In this thesis we have proposing an approach, the output of this approach is a WSDL document having semantic annotations , these annotations clarifying the datatypes used in the original WSDL document which wasn't understandable for the complex datatypes used in it and consequently we can reach better comprehension for the web service functionality.

5.1 conclusion

Our approach analyzing WSDL document and then extracting the datatypes used, we have divided the datatypes into two categories, primitive datatypes accounted 19 datatypes, and derived datatypes accounted 25 datatypes. Our problem was interested in the derived datatypes, this category causing ambiguity to the WSDL document, and the user will be confused about the right datatypes he should use. Then the approach asking the provider to determine the kind of the datatypes , and the approach adding a semantic annotation helping the user of the web service.

As a summary the approach is based on the following :

1. analyzing WSDL document and extracting all the datatypes used in it.
2. Dividing the extracted datatypes into two categories (primitive datatypes and derived datatypes). Table (5.1) shows the primitive ones and Table (5.2) shows the derived ones.

Primitive Datatypes : This Category includes the datatypes can easily be understood by service requester. Table (5.1) .

Derived Datatypes : This Category causing the ambiguity for thr web service because it is difficult to understood by the requester, so according to our approach we should ask the provider to determine the type of it . Table (5.2).

Table (5.1) : Primitive Datatypes

Name	type
String	xs:string
Boolean	xs:boolean
Decimal	xs:decimal
Float	xs:float
Double	xs:double
Duration	xs:dateTime
Uri	xs:anyURI
Date	
DateTime	xs:dateTime,
	xs:date,
	xs:gYearMonth,
	xs:gYear
	xs:gMonth
	xs:gDay
	xs:gMonthDay
Base64Binary	xs:base64Binary
HexBinary	xs:hexbinary
QName	xs:qname
NOTATION	
Integer	xs:int

Table (5.2) : Derived Datatypes

Name	type
NormalizedString	xs: normalizedString
Token	xs: token
Language	xs: language
NMTOKEN	xs: NMTOKEN
NMTOKENS	xs: NMTOKENS
Name	xs: Name
NCName	xs: NCName
ID	xs: ID
IDREF	xs: IDREF
IDREFS	xs: IDREFS
ENTITY	xs: ENTITY
ENTITIES	xs: ENTITIES
Integer	xs:int
NonPositiveInteger	xs: nonPositiveInteger
NegativeInteger	xs: negativeInteger
Long	xs: long
Int	xs: Int
Short	xs: short
Byte	xs: byte
NonNegativeInteger	xs: nonNegativeInteger
UnsignedLong	xs: unsignedLong
UnsignedInt	xs: unsignedInt
UnsignedShort	xs: unsignedShort
UnsignedByte	xs: unsignedByte
PositiveInteger	xs: positiveInteger

3. Adding Semantic Annotation according to the datatypes category where :
    - Primitive datatypes : no changes will be done , it will be remain as it is in the annotated WSDL document.
    - Derived datatypes : asking the provider , then adding annotation for the new WSDL document.

5.2 Future Work

The main contributions of this thesis is Adding Semantic description to Web Service datatype specifications for different platforms and IDEs and Enhancing Web Service comprehension and understandability. There are several possible future research directions that could be extended from this thesis such as :

1. In this thesis we use OWL(Ontology Web Language) as a semantic language, so it's a good thing and more useful in future to use another semantic languages such as SAWSDL, WSMO, and so on.
2. Enhancing the approach to Work in a backward direction (WSDL to Code), although it is a big direction, but it could increasing the understandability for the web service.
3. Merging between Semantic annotations and UML (Unified Modeling Language) to Enhance the Web Service comprehension and understandability.
4. We are depending on ASP.Net in this thesis , the future work can use another programming languages such as Visual basic , C#, Java, etc.

References

References

Aaron Skonnard, (2003). Understanding WSDL, Northface University available at : http://msdn.microsoft.com/en-us/library/ms996486.aspx.

Alexandre Bellini, Antonio Francisco do Prado, Luciana Aparecida and Martinez Zaina, (2010). Top-Down Approach for Web Services Development, Fifth International Conference on Internet and Web Applications and Services.

Andreas Heb, Eddie Johnston and Nicholas Kushmerick, (2008). (ASSAM) A Tool for Semi-Automatically Annotating Semantic Web Services, in Proceedings of 12th International Conference on Web Technologies, pp. 470–475.

Arthur Barstow, Mark Burstein, James Hendler, Vincent Marcatt, David Martin, Drew McDermott, Deborah L. McGuinness, Sheila McIlraith and Jeff Pollock, (2004). OWL Web Ontology Language for Services (OWL-S), W3C Member Submission 22 November 2004. Available at: http://www.w3.orgl SubmissionlOWL-S.

Che-Wei Chang, (2010). Realization of resource efficient embedded web service using representational state transfer (REST) packing and Roll-Back streaming XML (RBSTREX) parser. Multimedia University, Malaysia.

Dan Vint, (2003). XML Schema - DataTypes Quick Reference. available at : http://www .xml.dvint.com.

David Bell, Sergio de Cesare, Nicola Iacovelli, Mark Lycett and Antonio Merico, (2006). A framework for deriving semantic web services, Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex UB8 3PH, UK, Springer Science + Business Media, LLC.

David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris and David Orchard, (2004). Web Services Architecture, W3C Working Group Note 11 February 2004, available at : http://www.w3.org/TR/ws-arch.

David Booth, (2002). Fundamentals of the Semantic Web, W3C Fellow / Hewlett-Packard Paris.

Deborah L. McGuinness and Frank van Harmelen, (2004). OWL web ontology language overview, W3C Recommendation 10 February 2004, Available at : http://www.w3.org/TR/owl-features.

Dhanya Aravind, Passarawarin Supthaweesuk and Weider D, (2006). Yu. Software Vulnerability Analysis for Web Services Software Systems. Proceedings of the 11th IEEE.

Dieter Fensel, (2003). Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, 2nd edition. Springer-Verlag, Berlin.

Erin Cavanaugh, Altova Inc, (2006). Web services: Benefits, challenges, and a unique, visual development solution, Altova. Inc. l 900 Cummings Center, Suite 314-T l Beverly, MA, 01915-6181.

Gibson Lam, (2012). Extending the Web Services Architecture (WSA) for Video Streaming, The Hong Kong University of Science and Technology.

Glenford Myers, Corey Sandler and Tom Badgett, (2004). The Art of Software Testing, ISBN 0-471-04328-1, John Wiley. Neumann, P. Principled assuredly trustworthy compostable architecture.

Guilherme C. Hobold and Frank Siqueira, (2012). Discovery of Semantic Web Services Compositions based on SAWSDL Annotations, IEEE 19th International Conference on Web Services.

Gustavo Alonso, Fabio Casati, Harumi Kuno and Vijay Machiraju, (2004), Web Services concepts, Architectures and applications, Springer Verlag, ISBN 3-540-44008-9.

Heather Kreger, (2001). Web Services Conceptual Architecture, International Business Machines Corporation (IBM) group.

Houda El Bouhissi and Mimoun Malki, (2009). Reverse Engineering Existing Web Services Applications, 16th Working Conference on Reverse Engineering, IEEE /WCRE.2009.35.

Jia Zhang and Liang-Jie Zhang, (2005). Editorial Preface: Web Services Quality Testing, International Journal of Web Services Research, April-June 2005.

Jicheng Fu, Wei Hao, Farokh B. Bastani, and I-Ling Yen, (2011). Model-Driven Development: Where Does the Code Come From?, Insights Learned From a Case Study, Fifth IEEE International Conference on Semantic Computing.

Joel Farrell, Carine Bournez, Tomas Vitvar and Jacek Kopecky, (2007). Semantic Annotations for WSDL and XML Schema, W3C Candidate Recommendation IEEE,. Available at: http://www.w3.orgITRisawsdll. January 2007.

John Domingue, Dieter Fensel and James A, (2011). Hendler. Handbook of Semantic Web Technologies, Springer Heidelberg Dordrecht London New York, © Springer-Verlag Berlin Heidelberg.

John Klensin, (2001). Simple Mail Transfer Protocol, IETF, availale at : http://www.ietf.org/rfc/rfc2821.txt.

John McGovern. Enterprise Service-Oriented Architectures, (2006). Working with Registry and UDDI. pp 151-188.

John T. E. Timm and Gerald C. Gannod, (2007). Specifying Semantic Web Service Compositions using UML and OCL, IEEE International Conference on Web Services (ICWS 2007)0-7695-2924-0/07 $25.00 ©, IEEE.

Jos de Bruijn, Christoph Bussler, John Domingue, Dieter Fensel, Martin Hepp, Uwe Keller, Michael Kifer, Birgitta König-Ries, Jacek Kopecky and Michael Stollberg, (2005). Web Service Modeling Ontology (WSMO), W3C Member Submission 3 June 2005. Available at: http://www.w3.orgiSubmissionlWSMO.

Jos de Bruijn, Dieter Fensel, Uwe Keller, Michael Kifer, Holger Lausen, Reto Krummenacher, Axel Polleres and Livia Predoiu, (2005). The Web Service Modeling Language – WSML, W3C Member Submission 3 June 2005 Available at: http://www.w3.org/Submission/WSML.

Juanjuan Jiang and Tarja Systa, (2005). UML-Based Modeling and Validity Checking of Web Service Descriptions, Proceedings of the IEEE International Conference on Web Services (ICWS'05).

Ke Hao, (2013). Semantic Search of Web Services. University of California, IRVINE.

Ken Laskey, Francis McCabe, Peter F Brown and Rebekah Metz, (2006). OASIS Reference Model for Service Oriented Architecture V 1.0. OASIS, available at :http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf.

Keyvan Mohebbi , Suhaimi Ibrahim and Norbik Bashah Idris, (2012). Contemporary semantic Web service frameworks : an overview and comparisons, International Journal on Web Service Computing (IJWSC), Vol.3, No.3, September 2012.

Letz Carolin, vorgelegt von and aus M¨unster, (2007). Web Service Detection in Service-oriented Software Development: A Semantic Syntactic Approach.

Luc Clement, Andrew Hately, Claus von Riegen and Tony Rogers, (2004). UDDI Version 3.0.2. OASIS, UDDI Spec Technical Committee, http://uddi.org/pubs/uddi_v3.htm.

MacKenzie Matthew, Laskey Ken, McCabe Francis, Brown Peter and Metz Rebekah, (2006). Reference Model for Service Oriented Architecture 1.0 , Committee Specification 1, 2 August 2006.

Michael Huhns and Munindar P, (2005). Singh. Service-Oriented Computing: Key Concepts and Principles, IEEE Internet Computing, (Vol. 9, No. 1), January-February 2005.

Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall, (2006). The semantic web revisited, IEEE Intelligent Systems, 21(3):96–101, May/June 2006.

Pham Thi Thu Thuy, Young-Koo Lee and Sungyoung Lee, (2013). A Semantic Approach for Transforming XML Data into RDF Ontology, © Springer Science, Business Media New York, 2013 .

Rama Akkiraju, Joel Farrell, John Miller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth and Kunal Verma, (2005). Web

Service Semantics (WSDL-S), W3C Member Submission, November 2005. Available at: http://www.w3.org/Submission/WSDL-SI.

Robert Richards, (2006). Universal Description, Discovery, and Integration (UDDI), Pro PHP XML and Web Services, DOI 10.1007/978-1-4302-0139-7_19, © .

Roberto Chinnici, Jean-Jacques, Arthur Ryman and Sanjiva Weerawarana, (2007). Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C, availale at : http://www.w3.org/TR/wsdl20.

Roberto De Virgilio, (2010). Meta-Modeling of Semantic Web Services, IEEE International Conference on Services Computing.

Roy Gronmo, David Skogan, Ida Solheim and Jon Oldevik, (2004). Model-driven Web Service Development, International Journal of Web Services Research, 1(4), Oct-Dec 2004.

Roy Thomas Fielding, JIM GETTYS, JEFFREY C. MOGUL, HENRIK FRYSTYK NIELSEN, LARRY MASINTER, PAUL J. LEACH and TIM BERNERS-LEE, (1999). Hypertext Transfer Protocol HTTP/1.1. IETF, available at :http://www.ietf.org/rfc/rfc2616.txt.

Samer Hanna and Ali Alawneh, (2010). An Approach of Web Service Quality Attributes Specification, Communications of the IBIMA Journal (ISSN: 1943-7765) .

Steve Battle, Abraham Bernstein, Harold Boley, Benjamin Grosof, Michael Gruninger, Richard Hull and Michael Kifer, (2005). Semantic web services framework (SWSF) overview, World Wide Web Consortium, Member Submission SUBM-SWSF-20050909, September 2005.

Thomas Erl, (2005). Service-Oriented Architecture: Concepts, Technology & Design, Prentice Hall p. 792. ISBN 0-13-185858-0.

Thomas Weise Steffen Bleul, M. Brian Blake and Steffen Bleul, (2014). Semantic Web Service Composition: The Web Service Challenge Perspective, Web Services Foundations, Springer, 2014.

Tim Berners-Lee, James Hendler, and Ora Lassila, (2001). The semantic web. Scientific American, 284(5):34–43, May 2001.

W3C, (2008). XML schema part 2 : Datatypes, Second edition, W3C Recommendation, available at : http://www.w3.org/TR/2008/WD-xmlschema11-2-20080620, December 2008.

Wei-Tek Tsai, Yinong Chen and Ray Paul, (2005). Specification-Based Verification and Validation of Web Services and Service-Oriented Operating Systems, 10th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 05), Sedona, pp. 139 – 147, February 2005.

Weijun Sun, Shixian Li Defen Zhang and YuQing Yan, (2009). A Model-driven Reverse Engineering Approach for Semantic Web Services Composition, World Congress on Software Engineering, DOI 10.1109/WCSE.2009.403.

Xitong Li, Yushun Fan, Stuart Madnick, and Hongwei Zhu, (2009). An Approach to Composing Web Services with Context Heterogeneity, IEEE International Conference on Web Services.

ZHANG Lei, YANG Xiaoying, YUAN Yanni and SUN Bo, (2008). An Improved Semantic Annotation Method of Web Services Based on Ontology, ISECS International Colloquium on Computing, Communication, Control, and Management. IEEE.

ملخص

في السنوات الأخيرة، أصبحت خدمات الويب Web Services عنصر مهم في كثير من المجالات، والقدرة على تبادل المعلومات من خلال خدمات الويب هو مثال عظيم على دورها وفوائدها وقدرتها على تنفيذ وظائف موحدة قد تستخدم في المجال التجاري مثلا على مستوى عال.

إن وصف خدمات الويب واستخدامها يعتبر نحوي (syntactic)، بمعنى أن معرفة دلالات خدمات الويب (semantic) تقع عاتقها على مستخدم خدمة الويب لكي يفهم أو يتعلم بوسائل أخرى قبل أن يقرر هل سيستخدم هذه الخدمة أم لا، وكيف سيكون استخدامها.
هذه الرسالة تهتم في الوصف الدلالي لخدمات الويب semantic description for the web service وسيكون محورها عن الغموض وسوء الفهم في استخدام أنواع البيانات التي يتم استخدامها في ملف مكتوب بلغة XML حيث يتم حفظ وصف خدمات الويب فيه ويسمى web service description language (WSDL) .

مشكلة الغموض في تمثيل أنواع البيانات تؤدي إلى مشاكل عديدة منها صعوبة تفسير البيانات بين مزود الخدمة provider وطالبها requester وهذا يؤدي إلى أخطاء في دمج أو تكوين الخدمة ومن المشاكل أيضا الصعوبة التي قد تواجهها أدوات أو تقنيات تطوير خدمات الويب التي تعمل مباشر معل ملف WSDL الذي ينشأ تلقائيا وبالتالي ستكون هناك تناقضات في وصف الخدمات لمختلف التقنيات.

سنقدم في هذه الرسالة طريقة جديدة لمحاولة حل هذه المشكلة من خلال إضافة دلالات تفسر أنواع البيانات المستخدمة في ملف WSDL لتبسيط التعامل مع هذا الملف من ناحية أنواع البيانات.

المساهمات العلمية الرئيسية لهذه الرسالة :
١. إضافة وصف دلالي semantic description لأنواع البيانات المستخدمة في ملف WSDL .
٢. تحسين مستوى فهم خدمات الويب من خلال هذا الوصف الدلالي لخدمة الويب.
٣. تقسيم أنواع البيانات إلى مجموعتين (بسيطة ، مشتقة)
٤. أنواع البيانات البسيطة تكون واضحة في ملف WSDL ولا حاجة لإضافة وصف دلالي لها.
٥. أنواع البيانات المشتقة غامضة لا يعبر عنها بطريقة واضحة ، تسبب أخطاء لطالبي الخدمة وهي تحتاج للوصف الدلالي. وهنا لا بد من الرجوع لمزود الخدمة لمعرفة نوع البيانات المستخدمة في هذه الحالة.

توسيع مواصفات أنواع البيانات لخدمات الويب لمختلف الأنظمة المتطورة (لغات البرمجة)

تقديم

رائد عمر عبد ربه العبسي

إشراف

د. سامر حنّا

قدمت هذه الرسالة استكمالاً لمتطلبات الحصول على درجة
الماجستير في علم الحاسوب

عمادة البحث العلمي والدراسات العليا
جامعة فيلادلفيا

تموز 2014