Data Complexity Metrics for XML Web Services

Dilek BASCI, Sanjay MISRA
Department of Computer Engineering, Atilim University, Ankara, Turkey
smisra@atilim.edu.tr

***Abstract*—Web services that are based on eXtensible Markup Language (XML) technologies enable integration of diverse IT processes and systems and have been gaining extraordinary acceptance from the basic to the most complicated business and scientific processes. The maintainability is one of the important factors that affect the quality of the Web services that can be seen a kind of software project. The effective management of any type of software projects requires modelling, measurement, and quantification. This study presents a metric for the assessment of the quality of the Web services in terms of its maintainability. For this purpose we proposed a data complexity metric that can be evaluated by analyzing WSDL (Web Service Description Language) documents used for describing Web services.**

***Index Terms*—Web services, XMl, WSDL, complexity metrics**

I. INTRODUCTION

When the World Wide Web was "invented" in the early 1990's, its primary purpose was the sharing of documents between people, mostly in the scientific and scholarly communities. Since then, the usage of the Web continues to proliferate at an astounding rate.With the emergence of web applications the idea of integrating them as a very loosely coupled software components leads to the development of Web Services [1, 3-5] whose implementation details are hidden behind their interfaces. Web services as a new type of web applications have been promising to provide a common standard mechanism for interoperable integration of disparate systems and have been gaining a great deal of acceptance by different types of parties that are connected to the internet for different purposes.

The increasing popularity and acceptance of the XML Web Services leads the developers of Web service to make research activities to adopt the best practices of web service implementation and to find the ways for managing web services more effectively.

Particularly, when the need for the integration of various applications to build a new Web process arises providing the interoperability between these applications becomes main concern of the developers. In this aspect, the degree in flexibility of Web services has an effect on the overall integration process and should be managed effectively. As the complexity in Web service increases the flexibility of it decreases and, in turn, maintaining the Web service will become more challenger. Further, the data flow between the participants should be properly managed to provide the consistency in data exchanged between the applications via messaging. Hence in the integration process as the number of applications that are exposed as a Web service increases the management of data flow between services becomes a major issue which can affect the overall quality of the resulting Web process if it is not handled carefully.

It is well known that the maintainability is one of the important factors that affect the quality of any kind of software projects. As a loosely coupled software component the development of web service also requires modelling, measurement, and quantification for the ease of maintainability purpose. In the software development life cycle software metrics play an important role since they provide useful feedback to the designers as to the impact of decisions that are made during design, coding, architecture, or specification phases; without such feedback, many decisions are made in ad hoc manner.

Although lots of software metrics have been studied for decades in order to improve the quality of the traditional software products there has been done a few researches for developing metrics for the web services. The lack of research in developing software metrics for the web services is our main motivation to find useful metrics for the assessment of the quality of the web services in terms of maintainability.

Since the degree in complexity has effect on maintaining any kind of software project we studied in developing metrics to measure the complexity of a Web service.The complexity degree of a Web service can be measured by analyzing its WSDL [1, 3-5] document because WSDL provides the description of the Web service to the service requestors. However, the WSDL does not contain information about implementation details of a Web service hence we can only measure the data complexity of a Web service. The data complexity of a Web service can be defined as the complexity of data flowed to and from the interfaces of a Web service and can be characterized by an effort required to understand the structures of the messages that are responsible for exchanging and conveying the data. In this point of view it will help to analyze the structures of the messages for measuring the data complexity of a Web service via the usage of software metrics.

In section 2 we give a brief overview for Web services and the structure of WSDL documents and XML Schema. The related work in developing metrics for Web services is reviewed in section 3. Our proposed metric is demonstrated by examples in section 4. As a part of empirical validation of our newly proposed metric a comparative study with other measures has been done in section 5. Lastly, section 6 provides concluding remarks and a reflection on future work.

Digital Object Identifier 10.4316/AECE.2009.02002

II. WEB SERVICES AND WSDL

As a most widely accepted and successful type of service the XML Web services has become the fundamental building blocks in the movement towards distributed computing on the Internet and providing the platform for integrating diverse applications regardless of where they reside or how they were implemented. By exposing existing applications as XML Web services the developers are provided a way to build new, more powerful applications that use XML Web services as building blocks.Particularly for the business people Web services have become a powerful mean to achieve their business goals as the Web services offer an attractive set of technologies to enable existing legacy systems to be wrappered as Web services and made available for integration with diverse systems. So, for a business person a Web service is a business process or part of a business process that can be made available over a network to internal and/or external business partners to build a new business process or to achieve a business goal. As an example, a company might provide a purchasing order service to the customers, which automatically obtains price information from a variety of vendors, allows the service consumers to select a vendor, submit their orders and then track the shipment until it is received. The vendor company, on the other hand, in addition to exposing its services on the web, might in turn needs to use XML Web services in order to check the customer's credit, to charge the customer's account and to set up the shipment with a shipping company. By using open technology (XML and Internet protocols) and open standards managed by broad consortia such as *OASIS* and the W3C integrating application functionality within an organization or integrating applications between business partners is achieved more rapidly, easily, and cheaply than ever before. Due to the fact that the XML Web Services are based on an open standardized suite of technologies such as eXtensible Markup Language (XML) [6], Hyper Text transport Protocol (HTTP) [1, 2], Simple Object Access Protocol [1, 3-5, 17] (SOAP),Universal Description, Discovery, and Integration [18] (UDDI) and Web Service Description Language[1, 3-5], [16] (WSDL) industry-wide support graded the popularity and importance of this platform.

In section A and B we give a short information about WSDL documents and XML Schema respectively.

A. WSDL

All of the information necessary to invoke a Web service should be clearly described and be made available in a way that the service consumers can easily access and process the information about the service. For this purpose Web Services Description Language provides a model and an XML format for describing Web services. WSDL 1.1 [16] was submitted to the W3C in late 2001, and the W3C Web Service Description Working Group was formed in early 2002 to standardize WSDL. WSDL 2.0 [19] is the standard version of WSDL that includes significant changes and improvements [20].

By creating the WSDL document the owner of the Web service can separate the description of the abstract functionality offered by the service from concrete details of a service description such as “how” and “where” that functionality is offered.

When integration of diverse applications is aimed the exposed web services should be designed to have concise and clear agreement on the common specifications of protocols and data type systems that can be used by diverse set of programming languages in order to work in interoperable manner. The format of the messages exchanged between a service provider and consumer must be well defined so that the message sender can easily construct and the message receiver can process. For the Web service to be easily invoked the WSDL documents provide not only a way for grouping messages into operations and operations into interfaces but also provide a way for defining bindings for each interface and protocol combination along with the endpoint address for each one.Since underlying business and data models used by applications that are indented to be integrated may change over time, in order for accommodation of these changes building a flexible document structure that can be extended will pay off in the future. In [7] the use of WSDL and XML Schema is mandated for describing Web services. By this way the interoperability at the service description layer is ensured.

B. XML Schema

As we mentioned earlier, data flow between applications should be properly maintained to provide consistency in application’s data conveyed by the input and output messages of the Web services in interaction. In order to provide a common data type systems and to represent application’s data the WSDL document is supported by XML Schema which is either imported to or embedded into the WSDL document and encapsulated by <types> construct of WSDL. Representing the application data with a schema requires making strategic decisions that take into consideration some design issues which should be handled at design time, such as performance, extensibility, reusability, data access etc.

In XML context, the data representations are made by designing schemata which can be written in different XML schema languages such as DTD [6], W3C XML Schema [8-11], RELAX [12-13]. W3C XML Schema and DTDs are the most favored schema languages for generating XML documents. However, deploying XML documents is a challenge problem for an application without using supporting schema technology. In order for XML documents to provide a common understanding about data exchanged between applications these XML documents should be validated against the XML schema definition (XSD). The most important tangible aspect of an XML schema is that an XML schema specifies a contract between software applications or between parts of a software application. In large distributed web applications the notion of a contract provides many benefits such as simplifying modularization, resource allocation, testing, and deployment. Using schemas not only provides common understanding about exchanged data but also the ability of easy access methods for XML documents to be validated. With the successful design and implementation of schemas, the developers can have the capability of increasing productivity, improving software reliability, minimizing

development time, and decreasing time to market. Neglecting schemas implies that the schema validators are not used to determine if a given XML document satisfies the desired data transported among applications. In such a case the required check have to be performed by the application programs implying that the application developers have to write lengthy code. All of these considerations imply that, the decision in XML schema design to represent the application's data exchanged may affect the understandability of the exchanged messages described in WSDL document and in turn comprehending of a Web service defined by WSDL document.

III. Related Work

In terms of measuring the complexity of a Web service Yu et al [15] studied to compare Web services with other traditional software components by adopting some existing metric to a WSDL document , that were developed for measuring the interface complexities of software components. Some of the metrics presented in [15] are reviewed below.

Argument per Operation (APO): Given total number of arguments (n_a) in total number of operations (n_o) described in WSDL is used for measuring the size of web service and defined as:

$$\text{APO} = \frac{n_a}{n_o} \tag{1}$$

Operations Per Service (OPS): Similar to the OO metric WMC (Weight Method Per Class) operation per service is defined as the total count of operations that are declared by a *PortType* of a Web service and intended to measure the size of a given WSDL document.

Based on these two WSDL documents both Web services have equal APO and OPS which are not sufficient to differentiate these two services in terms of their complexities. Neither the APO nor the OPS metric can capture the difference between these two services since both metrics ignore the complexity of the arguments' data structure. Besides these ignored points, in [15] the schema definition embedded into the example WSDL document for description of arguments' data types is completely ill designed according to W3C XML Schema Language.

IV. Proposed Metric

In general, while searching an appropriate Web service the main consideration of a service consumer is to inspect the list of supported operations hosted in the *portType* construct of WSDL to get an overall feel for what a Web service offers since operations are the focal points of interacting with the service. Because the execution of an operation requires to exchange the requesting and responding messages between the service requestor and the service provider the structure of exchanged messages of the operations such as number of arguments contained, the data types of these arguments is also under consideration. In this point of view understandability of message structures has an important role in comprehending the operations that a Web service offers.

The arguments that an operation takes are contained by the *message* constructs of WSDL and each message construct can host one or more argument definitions. Each argument is defined by one *part* element of a message construct and each *part* element provides either element or type reference via its *element* or *type* attributes to associate the arguments with the components of the XSD schema that consist of element, type etc. definition/declarations in order for defining the data types of these arguments. Since, XSD schema is responsible for defining the data types of the arguments the complexities of these type definitions in XSD will also affect the understandability of the message structures. As defined before the data complexity of a Web service refers the effort required to understand structures of the messages that are responsible for exchanging and conveying the application data. Hence Web service's data complexity can be measured by analyzing the XSD embedded in WSDL and the structures of its requesting and responding messages used by the operations a Web service offers.

In our previous work [14] we presented a complexity metric for the assessment of the quality of XML schema. The complexity of XSD document measured by the presented metric is evaluated by summing up all of its components' complexities. In this evaluation each component's complexity degree is reflected by a weight value which is assigned based on component's internal architecture. The complexity degree of each XSD component will affect the understandability of the message structures since these components are associated with the arguments for describing those arguments' data types. This point has been ignored by the metrics in [15] and becomes our main focus to measure data complexity of a Web service.

By analyzing the structures of the exchanged messages described in WSDL we measured the data complexity of a given Web service via the newly proposed metric introduced in the following subsection.

A. Data Weight of a WSDL (DW(wsdl))

The Data Weight of a given WSDL can be defined as the sum of the data complexities of each input and output messages. By analyzing the message structures which contain the arguments that the operations of a Web service take we can measure each message's data complexity.

Intuitively, one may expect that as the number of operations and the number of arguments increase the understandability of a Web service becomes more difficult. The APO (Arguments per Operation) and OPS metrics presented in [15] were intended to measure the interface complexity of a given Web service. However, we claim that this is not always the case since the arguments may have different data type structures expressed by the components of the XSD schema and may require different effort to be understood [14].

In order to measure the data complexity of a given message we evaluate the data complexities of its part elements that describe the arguments. The data complexity of each part element can be reflected by assigning a weight value which is based on the complexity due to the data structures of arguments. As we presented in our previous

work, based on the internal architectures the complexity degree of the elements, types that are defined/declared in the XSD schema may vary and are assigned a weight value regarding their complexities. Since these XSD components are associated with the arguments to describe the arguments' data structures, it is meaningful to assign a weight value to these arguments which are equal to their associated XSD components' weight values so that the arguments' complexities due to their data structures are reflected.

By summing the weight values of each part elements belonging to the message construct of WSDL the data complexity of a message can be evaluated. Consequently, the total data complexity of the WSDL can be evaluated by summing up the data complexities of all of its messages. Hence we introduce Data Weight of WSDL (DW(wsdl)) metric to capture data complexity of WSDL documents, i.e. the effort required for understanding the data types of the arguments taken by the operations that a Web service provides.

Definition 1: DW(wsdl) Given a WSDL document having n_m number of messages in total, Weight of WSDL metric is defined as

$$DW(wsdl) = \sum_{i=1}^{n_m} C(M_i) \quad (2)$$

where $C(M_i)$ is the complexity value of i^{th} message and can be evaluated by :

$$C(M) = \sum_{j=0}^{n_p - 1} wp_j \quad (3)$$

where n_p is the total number of <part> elements encapsulated by a given message construct, and wp_i is the weight value of i^{th} <part> definition of the messages exchanged.

As we stated earlier the arguments contained by the part elements of the messages are associated with the element definition/declarations or type definitions of the XSD schema. Since each element or type definition in the XSD schema is assigned to a weight value reflecting their complexity degrees [14] and is associated to the arguments, the wp_i has the same weight value with the associated element or type definitions of XSD schema. That is

wp = **we**, if part has element reference to the schema element declaration (4)

= **wt**, if part has type reference to the schema element definition (5)

= 0, if the message has no <part> element. (6)

where ***we*** is the weight value [14] of the associated element declaration and, ***wt*** is the weight value of the associated type definition in the XSD schema embedded in or imported to WSDL. As we presented in [X] the complexities of each type definition in the schema are different according to their structures. So,

wt = **wc**, if the type is complex type definition (7)

= **ws**, if the type is simple type definition (8)

***Example1*:** In listing 1 and listing 2 we only show the description of one operation and its request and respond messages of the real life WSDL documents to demonstrate the calculation of DW value. We also evaluate the APO, OPS values for these two WSDLs. In listing 1 the operation *GetGSInformation* has two arguments namely *ID* and *Password* for its input message and one argument named *Response* for its output message. The data types of these three arguments are defined by the *type* attribute of part elements of its messages constructs and have a XML Schema type "string". In order to evaluate DW(wsdl) we first calculate the complexities of its each message by (3,4,5,8):

C("*GetGSInformationRequest*") $= \sum_{j=0}^{1} wp_j$

$= wp_{ID} + wp_{Password} = wt_{ID} + wt_{Password} = ws_{string} + ws_{string}$

$= 1 + 1 = 2$

C("*StringResponse*") $= \sum_{j=0}^{0} wp_j$

$= wp_{Response} = wt_{Response} = ws_{string} = 1$

```
....
<message name="GetGSInformationRequest">
                    <part name="ID" type="xsd:string" />
                    <part name="Password" type="xsd:string" />
</message>
<message name="StringResponse">
                    <part name="Reponse" type="xsd:string" />
</message>
...
<portType name="GSNPortType">
                    <!-- Information Operation -->
                    <operation name="GetGSInformation">
                              <input
message="tns:GetGSInformationRequest" />
                              <output
message="tns:StringResponse" />
                    </operation>
...
```

Listing 1.

The weight value for a part element is assigned based on the weight value of the argument it describes and the argument described by part element will have a weight of its associated type of XML Schema. As stated in [14] the weight values for built-in simple types of XML Schema are assigned to 1.Hence, the arguments that have built-in simple type have a weight value of 1. Then the data complexity value for the input message is found as 2 and for the output message it is found 1. Overall the data complexity of the WSDL in listing 1 is evaluated by (2):

$$DW(wsdl) = \sum_{i=1}^{2} C(M_i)$$

= C("*GetGSInformationRequest*")+C("*StringResponse*")

$= 2 + 1 = 3$

```
.....
<wsdl:types>
   <s:schema elementFormDefault="qualified"
targetNamespace="http://terraserver-usa.com/LandmarkServer/">
  <s:element name="GetLandmarkTypes">
      <s:complexType />
  </s:element>
  <s:element name="GetLandmarkTypesResponse">
     <s:complexType>
        <s:sequence>
```

```
            <s:element minOccurs="0" maxOccurs="1"
name="GetLandmarkTypesResult"
type="tns:ArrayOfLandmarkType" />
          </s:sequence>
      </s:complexType>
  </s:element>
 <s:complexType name="ArrayOfLandmarkType">
       <s:sequence>
         <s:element minOccurs="0" maxOccurs="unbounded"
name="LandmarkType"   type="tns:LandmarkType" />
       </s:sequence>
</s:complexType>
<s:complexType name="LandmarkType">
       <s:sequence>
          <s:element minOccurs="1" maxOccurs="1"
name="ShapeType" type="tns:ShapeType" />
          <s:element minOccurs="0" maxOccurs="1" name="Type"
type="s:string" />
       </s:sequence>
</s:complexType>
<s:simpleType name="ShapeType">
    <s:restriction base="s:string">
        <s:enumeration value="Null" />
        <s:enumeration value="Point" />
        <s:enumeration value="PolyLine" />
        <s:enumeration value="Polygon" />
   </s:restriction>
</s:simpleType>
....
</wsdl:types>
<wsdl:message name="GetLandmarkTypesSoapIn">
    <wsdl:part name="parameters"
element="tns:GetLandmarkTypes" />
 </wsdl:message>
 <wsdl:message name="GetLandmarkTypesSoapOut">
    <wsdl:part name="parameters"
element="tns:GetLandmarkTypesResponse" />
 </wsdl:message>
......
<wsdl:portType name="LandmarkServiceSoap">
  <wsdl:operation name="GetLandmarkTypes">
     <wsdl:input message="tns:GetLandmarkTypesSoapIn" />
     <wsdl:output message="tns:GetLandmarkTypesSoapOut" />
  </wsdl:operation>
......
</wsdl:portType>
.....
```

Listing 2.

In listing 2the operation *GetLandmarkTypes* is described in the second real life WSDL document. While the part element in the input message of the operation associates the input argument with the complex typed GetLandmarkTypes element of the XSD schema, the output message's part element associates the complex typed GetLandmarkTypesResponse element of XSD schema for defining the output arguments namely *ShapeType* and *Type*. The complexity values for each message evaluated by (3,4,5,7):

$$C(\text{“}GetLandmarkTypesSoapIn\text{”}) = \sum_{j=0}^{0} wp_j$$

$$= wp_{parameters} = \mathbf{we}_{tns:GetLandmarkTypes} = wc_{complexType} = 1$$

$$C(\text{“}GetLandmarkTypesSoapOut\text{”}) = \sum_{j=0}^{0} wp_j$$

$$= wp_{parameters} = \mathbf{we}_{tns:GetLandmarkTypesResponse}$$

$$= wc_{tns:GetLandmarkTypesResponse} = 6$$

The method for calculating a complexity value for an element declaration in XSD schema was presented in [14]. Based on the equations in [14] the data complexity i.e weight value for the element declaration *tns:GetLandmarkTypes* of XSD which is associated for the input argument's description of the operation in the WSDL file shown in listing 2 is found as 1.Similarly, the data complexity value for the element declaration *tns:GetLandmarkTypesResponse* in XSD which is associated for the output arguments description of the operation is found as 6 by the equations in [14].Hence the data complexity value for WSDL document in listing 2 is:

$$DW(wsdl) = \sum_{i=1}^{2} C(M_i)$$

$$= C(\text{“}GetLandmarkTypesSoapIn\text{”}) + C(\text{“}GetLandmarkTypesSoapOut\text{”})$$

$$= 1 + 6 = 7$$

The APO, OPS values for both WSDLs are found as 3.

As seen in the example 1, even both WSDLs have equal APO, OPS values, DW(wsdl) values of them are not equal due to diversity in the complexities of the arguments' data structures.

V. Empirical Validation

Empirical validation proves the practical utility of a new metric. For the validation of the proposed metric we analyzed 56 real example WSDL files written in WSDL 1.1. The analyzed WSDLs are collected by searching some well known sites [21] that list publicly available web services. Additionally, to verify the soundness and robustness of our metrics a comparative study with APO and OPS metrics [15] has been done as a part of empirical validation process and the statistic (see appendix A table1) referring these comparison results has been collected. Further we have drawn two graphs to depict these comparison results.

We have demonstrated in section 4, how to calculate the DW values of two WSDL documents to measure the data complexities of Web services. As seen in example 1although the two WSDL documents have equal values for the APO and OPS metrics our metrics can better differentiate these two WSDLs in terms of reflecting the data complexity of their related Web services. The graphs (see figure 1 and figure 2 in appendix A) depict the comparison results between DW and OPS metrics and that of DW and APO metrics respectively. From the first graph it can be observed that our claim stated in section 3 is supported. What we claimed is that as the OPS value increases the complexity of a Web service may not always increase accordingly. Similarly, from the second graph we can see that a Web service having higher APO value may not have higher complexity. On the other hand DW metric can better reflect the data complexity of a Web service than APO and OPS metrics as depicted in both graphs. This is due to the fact that both APO and OPS metrics focus only on the number of operations that are offered by Web service and number of arguments that the offered operations take and do not take into account the complexity caused by the complexities of the arguments data structures. This complexity is captured by our DW metric as observed from both graphs.

VI. Concluding Remark and Future Work

As a loosely coupled software components Web services allow integration of heterogeneous systems in diverse domains including business-to business, business-to-consumer and enterprise applications due to the fact that their flexible nature. For providing universal interoperability between these systems flexibility and complexity degree of Web services are main concerns of the Web service developers during integration process. Particularly, the data flow between interacting applications should be maintained carefully in order to get common agreement on the exchanged data. One of the factors that play a role in the quality of a Web service in terms of maintainability is its data complexity and can be measured by analyzing the messages that are responsible for conveying application data and exchanged by the operations of a Web service. Since the operations offered by a Web service and the messages they exchange are described in WSDL documents, inspecting WSDL documents will help to measure the data complexity of a Web service via the usage of software metrics.

In this study we have presented metric that focuses on the messages describing the arguments of the operations for the assessment of Web service's data complexity. By considering the complexity degree of each argument's data structure which is ignored by the metrics [15] we have evaluated measures that can provide useful feedback for the developers of Web service in the development life cycle.

In order to validate our presented metric and to prove its usefulness empirical validation process has been done and statistic that shows the comparison results of our metric with the others are collected through analyzing publicly available WSDL documents written in WSDL 1.1.

As a future work we aimed to validate our metrics theoretically as well, since practical evaluation and formal validation is the necessity in order for any newly proposed software metric to be reliably applied. For this purpose we will examine our metric against a practical framework [22] and apply the basics of Measurement Theory [23] (MT).

REFERENCES

[1] T.Erl, Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services, Prentice Hall Publischers.(2004)

[2] D.Gourley, B. Totty, HTTP: The Definitive Guide, O'Reilly Publischers(2002)

[3] E. Newcomer, G. Lomow, Understanding SOA with Web Services, Addison Wesley Professional(2004)

[4] E. Cerami, Web Services Essentials, Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, O'Reilly Publischers(2002)

[5] S.Weerawarana,F.Curbera, F. Leymann, T. Storey, D.F.Ferguson, Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More, Prentice Hall Publishers(2005).

[6] http://www.w3.org/TR/1998/REC-xml-19980210

[7] WS-I Basic Profile Version 1.0 http://www.ws-i.org/Profiles/BasicProfile-1.0-2004- 04-16.html

[8] http://www.w3.org/TR/xmlschema-1/

[9] http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[10] http://www.w3.org/TR/2001/PR-xmlschema-0-20010330/

[11] http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[12] ISO/IEC, Information Technology–Text and Office Systems–"Regular Language Description for XML (RELAX) – Part 1: RELAX Core, 2000.D TR 22250-1".

[13] http://www.xml.gr.jp/relax.

[14] D.Basci,S.Misra,"Complexity Metric for XML Schema Documents" ,Proceeding of 5thinternational Workshop on SOA and Web Services,OOPSLA2007

[15] Y.Yu,J.Lu, J.Fernandez-Ramil, P. Yuan,"Comparing Web Services with other Software Components", Proceeding of IEEE International Conference on Web Services (ICWS 007).

[16] http://www.w3.org/TR/wsdl

[17] http://www.w3.org/TR/soap/

[18] http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[19] http://www.w3.org/TR/wsdl20/]

[20] http://www.w3.org/TR/2004/WD-wsdl20-20040803/#migration

[21] http://www.xmethods.com/ , http://www.webservicelist.com/

[22] C. Kaner , "Software Engineering Metrics: What do they Measure and how do we know?", Proceedings of 10th International Software Metrics Symposium. Metrics 2004.

[23] L.C. Briand, S. Morasca, V.R. Basily, "Property based Software Engineering Measurement", IEEE Transactions on SE,1996, vol. 22, 1, pp.68-86.

Appendix A.

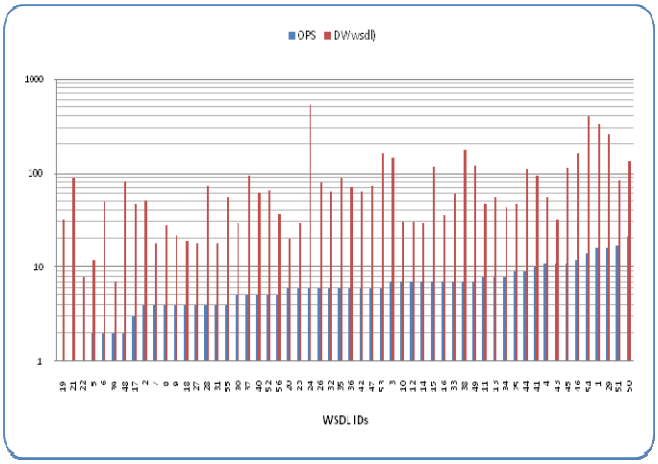

Figure 1. Comparison result between DW and OPS metrics. The graph is drawn in logarithmic scale with base 10 and WSDL files are ordered according to the number of operations. The WSDL IDs are listed in table 1.

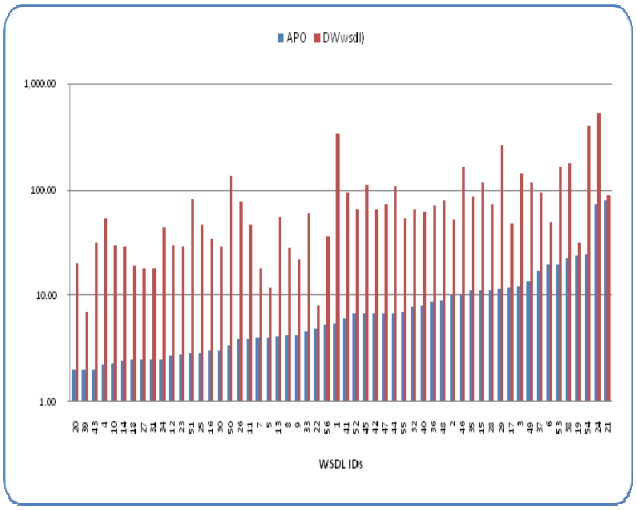

Figure 2. Comparison result between DW and APO metrics. The graph is drawn in logarithmic scale with base 10 and WSDL files are ordered according to APO metric value. The WSDL IDs are listed in table 1.

TABLE 1. THE STATISTIC EVALUATED BY ACCORDING TO COMPARISON RESULTS BETWEEN DW, APO AND OPS METRICS FOR 56 WSDL DOCUMENTS AVAILABLE ON THE WEB. THE *ID* COLUMN REFERS THE WSDL SHOWN IN FIGURE 1 AND FIGURE 2, THE COLUMN LABELED *LINK* PROVIDES WEB ADDRESSES OF WSDLS, AND , #A IS THE NUMBER OF ARGUMENTS OF WEB SERVICES DESCRIBED BY THE LISTED WSDLS.

ID	Link	#A	OPS	APO	DW(wsdl)
1	*http://ws.xwebservices.com/XWebBlog/V2/Blog.asmx*	89	16	5.56	334
2	http://www.geoservicios.com/V2.0/sgeo/sgeo.asmx	40	4	10.00	52
3	http://www.banguat.gob.gt/variables/ws/BDEF.asmx	88	7	12.57	144
4	http://localhost/ogsa/services/GLCProcessService	25	11	2.27	54
7	*http://www.billyclark.com/DesktopModules/FotoVisionDNN/PhotoService.asmx*	16	4	4.00	18
8	*http://tripleasp.net/Services/ShowCode.asmx*	17	4	4.25	28
9	*http://services.nirvanix.com/ws/Authentication.asmx*	17	4	4.25	22
5	*http://in2test.lsi.uniovi.es/sqlmutationws/SQLMutationWS.asmx*	8	2	4.00	12
6	*http://service.ecocoma.com/shipping/fedex.asmx*	39	2	19.50	50
10	http://www.mathertel.de/AJAXEngine/S02_AJAXCoreSamples/CalcService.asmx	16	7	2.29	30
11	http://del.eterio.us/blog/editposts.asmx	31	8	3.88	47
12	*http://services.test.musiccue.net/rapidcueapplication/WorkManager.asmx*	19	7	2.71	30
13	http://www.oorsprong.org/websamples.arendsoog/ArendsoogbooksService.wso	33	8	4.13	55
14	*http://services.test.musiccue.net/rapidcueapplication/SubmissionManager.asmx*	17	7	2.43	29
15	http://www.esendex.co.uk/secure/messenger/soap/InboxService.asmx	78	7	11.14	117
16	*http://www.simulation.fr/seq/SaintEtiQ.asmx*	21	7	3.00	35
17	http://quisque.com/fr/chasses/blasons/search.asmx	36	3	12.00	48
18	http://rangiroa.essi.fr:8080/dotnet/evaluation-cours/EvaluationWS.asmx	10	4	2.50	19
19	http://sws-challenge.org/shipper/runner	24	1	24.00	32
20	http://www.wubingstudy.com/WebService/Messages.asmx	12	6	2.00	20
21	http://www.iperformonline.com/WebServices/employee/AddUpdateEmployee.asmx	81	1	81.00	89
22	*http://www.thomas-bayer.com:80/axis2/services/BLZService*	5	1	5.00	8
23	*http://services.test.musiccue.net/rapidcueapplication/SecurityProvider.asmx*	17	6	2.83	29
24	http://www.saiasecure.com/webservice/shipment/soap.asmx	450	6	75.00	532
25	http://acims9.acims.arizona.edu/PublicationDB/DEVSPubs.asmx	26	9	2.89	47
26	http://www.multispeak.org/interface/30j/10_OA_EA.asmx	23	6	3.83	79
27	*http://webservices.daelab.net/temperatureconversions/TemperatureConversions.wso*	10	4	2.50	18
28	http://gw1.aql.com/soap/sendsmsservice.php	45	4	11.25	73
29	http://www.esendex.com/secure/messenger/soap/ContactService.asmx	189	16	11.81	262
30	http://knucklehead.cs.umb.edu/vsowmya/lantetwebservice.asmx	15	5	3.00	29
31	http://www.devhood.com/services/timelog/timelog-service.asmx	10	4	2.50	18
32	http://soap1.hopewiser.com:8003	47	6	7.83	64
33	http://pc218.cgk.affrc.go.jp/PMTypeService/MainEntry.asmx	33	7	4.71	60
34	http://www.cts.com.pl/webservices/rt_info.asmx	20	8	2.50	44
35	http://itplaza.jeju.go.kr/rpt_ws/Rpt_Ws_FD.asmx	66	6	11.00	88
36	http://demo.soapam.com/services/FedEpayDirectory/FedEpayDirectoryService	52	6	8.67	70
37	http://itplaza.jeju.go.kr/itplazaweatherservice/ItplazaWeatherService.asmx	86	5	17.20	94
38	http://www.inphoto.cz/Service/PhotoServer.asmx	156	7	22.29	179
39	http://www.elguille.info/NET/WebServices/HolaMundoWebS.asmx	4	2	2.00	7
40	http://www.secureattachment.com/webservices/sadownload.asmx	40	5	8.00	61
41	http://www.sipeaa.it/wset/ServiceET.asmx	62	10	6.20	94