Adding Domain Model Description for Web Services

Hongyu Li
Beijing Key Laboratory of Information Service Engineering, Beijing Union University, Beijing, China
Email: ldthongyu1@buu.edu.cn

Yuxin Wang
College of Information, Beijing Union University, Beijing, China
Email: xxtyuxin@buu.edu.cn

Jiazheng Yuan
Beijing Key Laboratory of Information Service Engineering, Beijing Union University, Beijing, China
Email: xxtjiazheng@buu.edu.cn

***Abstract*—For helping Web service consumers establish the corresponding relationship between terms of the domain conceptual model and date types in WSDL, this paper proposes a method to embed semantic annotation in the definition of data types in WSDL with XSLT transformation. So Web service consumers can understand better the meaning of the data with this bidirectional relationship and thus further ensure they use Web services correctly. The additional benefit obtained from this method is that Web service annotators are able to find corresponding concept terms not built in the conceptual model so as to help them accomplish annotating conveniently.**

***Index Terms*—Web services, WSDL, conceptual model, semantic annotation, XSLT**

I. Introduction

Web service is a common distributed system. It can integrate data scattered physically. So long as Web service consumers (users) acquire the description file WSDL [1] of Web services, they can use the Web services like local services. Although Web services provide the mechanism of data integration, the prerequisite of data integration is that the Web service consumers fully aware of the data sources. They should understand not only the format of the data items but also the semantics of the data items. Different data sources have different appellations to data of the same semantics, and have different semantics to data of the same appellations. If the Web service mechanism is not expanded to reflect the semantic information of the data, data integration task will be difficult to complete.

The data items described by WSDL can be added semantic information, through the expansion of extension attributes and elements in WSDL. In order not to destroy the format and function of the original WSDL, while making data items annotation, now we usually adopt the mapping between data items and semantic models. The relevant standards have also been proposed, such as WSDL-S [2], SAWSDL [3] standards. The applications of those standards [4]-[8] and the other applications based on semantic annotation [9]-[12] are beginning to appear. The emphasis of these semantic annotations is not further explanation of the meaning of the data types, i.e. the explanation of metadata, but the semantic transformation of the data model, so that data defined are converted into the data represented by a semantic model. The result generated is usually concrete instances of the corresponding semantic model, so that semantic tools can easily deal with the instances further. In practice, the Web service consumers want that there is a way to implement the mapping between the data items in Web service and their semantic meanings. However, these standards do not give the generic method for the bidirectional mapping between them. Paper [13] proposed a method that extracts semantic information and maps the information extracted to CIDOC CRM conceptual model [14]. But the conceptual model is contained in the XSLT transformation file, thus the definition of the conceptual model is not independent, and the transformation function is not general. This paper proposed a semantic annotation method by which the WSDL data format is further explained and a semantic annotation output file is produced. A mapping between the data items defined by WSDL and the domain conceptual model can be reflected clearly in the output file. In order to help WSDL file annotators discover the annotation terms not mapped, the transformation can be used iteratively to improve the result of annotation, finally all of the annotated terms have corresponding concepts in the domain conceptual model.

Corresponding author: YuXin Wang (xxtyuxin@buu.edu.cn)

doi:10.4304/jsw.9.2.417-424

II. Scheme for Annotation and Transformation of Data Type

Both WSDL 2.0 and WSDL 1.0 contain the element *Types* which encloses data type definitions that are relevant for the exchanged messages. For maximum interoperability and platform neutrality, WSDL prefers the use of XML Schema as the canonical type system, and treats it as the intrinsic type system. The element *complexType* in XML Schema is used to define complex data type. The element *simpleType* is generally used to define simple data type. In order to expound the semantics of the data types, additional attributes are added to the definition of the data types. An additional container element for embedding semantic models can be defined, since WSDL already allows extension elements within the element *wsdl:description*.

As shown in Figure 1, the *domain:conceptualModel* part indicating a domain conceptual model is embedded in a WSDL file. The mapping from the data types defined in WSDL to the conceptual model terms(concepts), which is referred as DtoC mapping, can be established and an output file of DtoC mapping will help the Web service annotators make comprehensive semantic annotation. Similarly, the mapping from the conceptual model terms to the data types, which is referred as CtoD mapping, can be built and an output file of CtoD mapping will help the Web service consumers understand the semantics of the data types correctly.

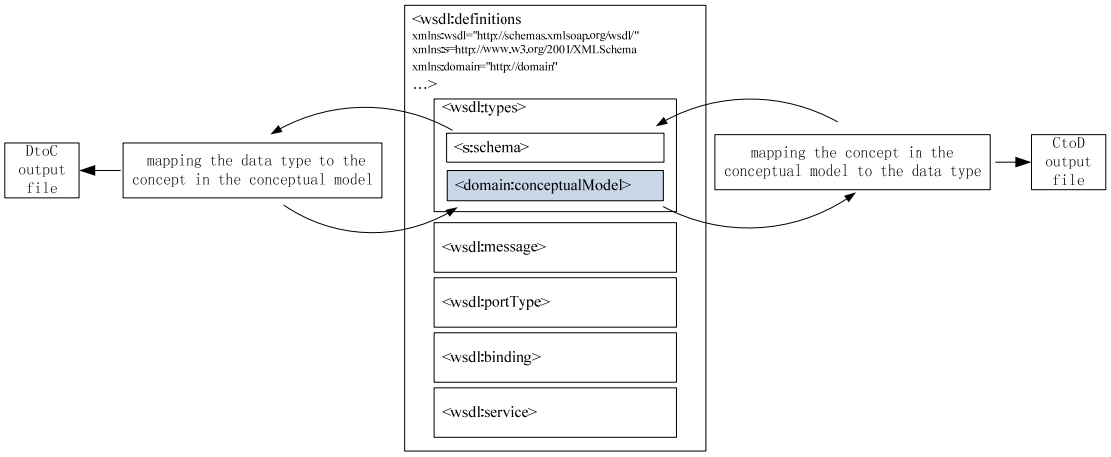

Figure 1. The scheme for annotation and transformation of data types

The semantic annotation is usually implemented by the experts in a domain who need to collaborative with the Web service providers. After a WSDL file have been finished editing by the Web service provider, the extra attributes of the data types defined in the WSDL file are added by the domain experts and then the relationship between these attributes and the concepts in the domain conceptual model can be established. The relationship established above is a direct reflection of the relevance between *s: schema* elements and *domain: conceptualModel* elements in Figure 1. The process of establishing this relationship is also understood as adding semantic information to the data type defined. A specific method will be introduced in subsequent section. The method which does not change with the changes in the conceptual model and data types has the versatility with respect to both CtoD mapping and DtoC mapping and so it has laid the groundwork for transforming itself into a universal service. The Web service consumers will gain the output file after they submit the WSDL file which they have received from a Web service provider to such service. The same solution can be applied to other parts of the WSDL file, such as *message* element, *portType* element and *interface* element in WSDL 2.0. Thus, those parts can also get semantic annotating.

Adding semantic annotations to the data type of WSDL involves multiple parties. The stakeholders, such as domain conceptual model constructors, semantic annotators, Web service providers and Web service users, will be able to co-operate by means of the solution proposed in this paper which provides an efficient mechanism to achieve the semantic description of Web services. The relationship between multiple parties is shown in Figure 2.

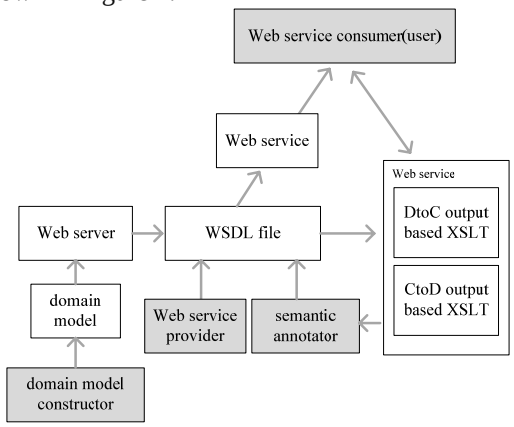

Figure 2. The cooperative relationship among the stakeholders who are shown with the shaded blocks

The constructors of the domain conceptual model are responsible for building the model which contains the complete definition of domain concepts and is generally published through a Web server. The semantic annotators are responsible for semantic annotation of the data types in WSDL, that is, to add semantic information for them.

They also take charge of embedding the conceptual model into the WSDL file. The model embedded is not a complete semantic model, but a simplified model, as long as it can explain the concepts used in the WSDL file. The Web service providers are responsible for the definition of the service interface through WSDL and its implementation. Their expertise is in the areas of computer software, not the field described by the conceptual model. The semantic annotators and the Web service users can adopt and implement the solution proposed in this paper in an open and Web service-based fashion. So semantic annotators using this Web service can get the output of CtoD mapping based XSLT [15] and the Web service users request the same service to get the output of DtoC mapping based XSLT in order to understand exactly the meaning of the data type defined in the WSDL file. Finally, the scheme showed in Figure 2 offers protection from misunderstanding of the data in the Web service whenever it is used by its consumers.

III. Implementation of Annotation and Transformation of Data Types

Because CIDOC CRM [16] is an ontology widely used in the field of cultural heritage and has relatively perfect definition of the domain concepts, so this paper takes it as an example of the semantic annotation method. But the method is not restricted to CIDOC CRM model.

The conceptual model can be embedded into the *Types* element totally, because it allows the elements in it to expand. Relevant elements used to define the data types can explain the semantics of the data types by adding *cidoc:E* attributes. The specific form is as follows.

```
<wsdl:definitions  xmlns:wsdl=http://schemas.xmlsoap.org/wsdl/  xmlns:s=http://www.w3.org/2001/XMLSchema
     xmlns:cidoc="http://cidoc.ics.forth.gr"  …>
  <wsdl:types>
     <s:schema …
          <s:complexType name=" a complex type" cidoc:E=" a concept in the concept model " >
            …
          </s:complexType>
          <s:element name=" a simple type " cidoc:E=" a concept in the conceptual model "/>
           …
     </s:schema>
     <cidoc:conceptualModel> … </cidoc:conceptualModel>
  </wsdl:types>
  …
</ wsdl:definitions >
```

The namespace of relevant conceptual model can be introduced through namespace *cidoc*. The conceptual model, which is also known as semantic model, is described by the element *cidoc:conceptualModel* and its child elements. There is no specific restriction on the establishment of the conceptual model. The real purpose of introducing *cidoc:E* attribute is to establish a corresponding relationship between the data type defined by XML Schema (whether it is complex or simple) and certain concept in the conceptual model. The introduction of the conceptual model is to reflect the mutual relevance of various concepts in the model, thus indirectly reflect the mutual relationship between various data types. This, for the Web service consumers to fully understand the data provided by the service, paves a way of showing its internal relationship. If we want to introduce other conceptual model, we need only to replace the *cidoc* conceptual model and its namespace with the corresponding conceptual model and the namespace. Although the annotation given above is in WSDL 1.0 format, it is also suitable to WSDL 2.0.

After completing the above-mentioned annotation and the introduction of conceptual model, we need also to establish the relationship from the conceptual model to the data type defined by WSDL. Although its reverse relationship is reflected in the *cidoc:E* attribute value, in practice, what Web service consumers really need is a description of bidirectional mapping between the conceptual model and the data type in WSDL, so as to help them find the corresponding WSDL data type from the conceptual model. What follows is how to use XSLT to implement the function of the bidirectional mapping between the conceptual model and the data type in WSDL.

```
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  xmlns:wsdl=http://schemas.xmlsoap.org/wsdl/
     xmlns:cidoc=http://cidoc.ics.forth.gr  …>
   …
   <xsl:template match="//wsdl:types">
          <cidoc:mappingAnnotation>
```

```
                <xsl:apply-templates select=" cidoc:conceptualModel "/>
            </cidoc:mappingAnnotation>
        </xsl:template>
        <xsl:template match="cidoc:conceptualModel">
            <xsl:element name="{name()}"> <xsl:apply-templates/>  </xsl:element>
        </xsl:template>
```

The above XSLT segment is the first part, in which the ellipsis denotes that some transformation details are omitted, e.g. some auxiliary namespaces, because they have no influence on the description of the entire transformation method. The XSLT first searches for *wsdl:types* node in the WSDL file. When the node matches, it writes the root node *cidoc:mappingAnnotation* into the output file, then it searches the root node *cidoc:conceptualModel* of the conceptual model further. When the node is matched, it outputs the node and continues searching along the root of the conceptual model. The main objective of this segment is to locate the conceptual model in the WSDL file, then to traverse the entire nodes of the conceptual model, so as to check whether the node traversed matches certain data type in the WSDL file.

```
    <xsl:template match="cidoc:conceptualModel//*">
        <xsl:element name="{name()}">
            <xsl:variable name="elementName" select="name()"/>
            <xsl:if test="starts-with($elementName,'cidoc:E')">
                <xsl:attribute name="rdf:resource">
                    <xsl:for-each select=" //*[@cidoc:E]">
                        <xsl:if test="concat('cidoc:',@cidoc:E)=$elementName">
                            <xsl:text>[</xsl:text>
                            <xsl:value-of select="concat(name(),'/',@name)"/>
                            <xsl:text>]</xsl:text>
                        </xsl:if>
                    </xsl:for-each>
                </xsl:attribute>
            </xsl:if>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

The above template will match all of the subnodes of *cidoc:conceptualModel*. When matching a subnode, the template first outputs the node, then judges whether to add *rdf:resource* attribute for it according to the following condition: If, in the WSDL file, the attribute value of *cidoc:E* for a data type is the same as the current element name, then the template needs to add a *rdf:resource* attribute to the element, the attribute value is "*element name of the data type / the attribute value of the name for this element*", and is bracketed in "[ ]"; if the condition is not met, then the template does not add *rdf:resource* attribute to it. Through the above steps, a concept in the conceptual model can corresponds with WSDL data type items. Because many data types can correspond with the same concept, "[ ]"can be the delimiters of every data type corresponded with. If there are many "[ ]" in a *rdf:resource* attribute value, then this concept corresponds with many data types. The use of "[ ]" is merely for showing the importance of the correspondence. For ease of further processing with conventional multi-attribute valued semantic disposal tool, we can use blank as the delimiter among the multi-attribute values instead of "[ ]". Before *rdf:resource* attribute is outputted, further check has been made on the type of element matched. Because, in the CIDOC CRM model, *cidoc:P* type concept denotes predicate, it has no corresponding WSDL data type, there is no need to add *rdf:resource* attribute for these elements. Here, we only discussed CIDOC CRM predicate. But it does not lose generality. Because, in a conceptual model, there are always certain concepts that are used to describe the conceptual model itself. These concepts does not correspond with outside data, there is no need to add some attributes that reflect the correspondence relationship like predicate in CIDOC CRM. After mapping attributes are added to concepts, the transformation template should be used continually, so that the subelements of the element get the same treatment. After all concepts are traversed, all mapping relationships are added to their corresponding *rdf:resource* attributes.

What follows is an example showing how to establish the mapping relationship between the data types defined and the conceptual model in WSDL and how to

form the mapping relationship between the conceptual model and the WSDL data type through the above XSLT transformation. In the example, the *Types* part in WSDL uses XML Schema to define a complex data type called *painting*, and it contains a more detailed description item called *creator*. The name and structure of this data type denote that it represents a painting (picture), and contains more detailed information of the painting, such as its creator (maker). But the name of the data type and the structure of the definition is not enough for a Web service consumer to acquire the above meaning, i.e. we cannot suppose the Web service consumer to understand the semantics of *painting* and *creator* and their relationship. Thus, domain conceptual model is needed to further annotate the data type. To this end, *cidoc:E* attribute needs to be added to the corresponding elements of *painting* and *creator* to demonstrate the relationship between the data type and certain concept in the conceptual model, and the conceptual model demonstrating the above relationship is introduced in *cidoc:conceptualModel* root element.

```
<wsdl:types>
 <s:schema …>
        <s:complexType name="Painting" cidoc:E="E22.Man_Made_Object">
            <s:sequence>
                <s:element name="creator" type="s:string" cidoc:E="E21.Person"/>
                <s:element name="timeSpan" type="s:date" cidoc:E="E52.Time-Span"/>
                <s:element name="type" type="s:string" cidoc:E="E55.Type"/>
                <s:element name="preservationPlace" type="s:string" cidoc:E="E53.Place"/>
          …
            </s:sequence>
        </s:complexType>
        …
 </s:schema>
 <cidoc:conceptualModel>
        <cidoc:E22.Man_Made_Object>
            <cidoc:P108B.was_produced_by>
                <cidoc:E12.Production>
                    <cidoc:P14F.carried_out_by> <cidoc:E21.Person/> </cidoc:P14F.carried_out_by>
                </cidoc:E12.Production>
            </cidoc:P108B.was_produced_by>
        </cidoc:E22.Man_Made_Object>
        …
 </cidoc:conceptualModel>
</wsdl:types>
```

The relationship between the concepts E21 and E22 are defined in *cidoc:conceptualModel* showing that E22 is a man-made object and is made by E21 i.e. man. This is a general method in cultural relic's field to use standard conceptual model to describe relationship between concepts, the semantics of which is not ambiguous to field experts. Through the mapping relationship reflected by *cidoc:E* in the complex data type, it is clear: *painting* is a man-made object, and is made by *creator* that has the characteristic of a man.

In order to extract concepts used for annotating, and add *rdf:resource* attribute for them so as to reflect the corresponding relationship between the concepts in the conceptual model and the WSDL data types, we need to run the XSLT transformation program given above. The result is as follows:

```
<cidoc:mappingAnnotation xmlns:s=http://www.w3.org/2001/XMLSchema xmlns:cidoc=http://cidoc.ics.forth.gr
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" …>
  <cidoc:conceptualModel>
        <cidoc:E22.Man_Made_Object rdf:resource="[s:complexType/Painting]">
            <cidoc:P108B.was_produced_by>
                <cidoc:E12.Production rdf:resource="">
                    <cidoc:P14F.carried_out_by>
                        <cidoc:E21.Person rdf:resource="[s:element/creator]"/>
```

```
                                    </cidoc:P14F.carried_out_by>
                                </cidoc:E12.Production>
                            </cidoc:P108B.was_produced_by>
                        </cidoc:E22.Man_Made_Object>
                        …
                    </cidoc:conceptualModel>
                    …
                </cidoc:mappingAnnotation>
```

From the above result which is also known as the CtoD mapping output, we can see that there exist corresponding data types in WSDL for concepts E22 and E21, while there is no relationship for other E entity concepts, there is no need to add *rdf:resource* attribute for P concept because it is a predicate in the conceptual model.

IV. Method of Iterative Annotating

In order to facilitate semantic annotations of WSDL files, the annotation work is usually carried out by iteration. First, add annotations, i.e. *cidoc:E* attributes to data types defined in WSDL, then, establish corresponding conceptual models for them. This process repeats, until the data types needed are all annotated. There are many data types defined in a complex WSDL file. During the annotation process, we usually want to know which concept objects used for annotating are not established in the conceptual model. If, during the above transformation process, those concepts that do not have corresponding relationship in the conceptual model but have used in the annotations of the data types are listed, then the conceptual models will be improved so that all of the annotation concepts have their own positions in the conceptual model. In order to implement this function, we can add an additional template in the above XSLT transformation file. The specific form of the template is as follows:

```
<xsl:template match="s:schema//*[@cidoc:E]">
    <xsl:variable name="cidocName" select="concat('cidoc:',@cidoc:E)"/>
    <xsl:if test="not(//cidoc:conceptualModel//*[name()=$cidocName])">
            <xsl:element name="{name()}">
                    <xsl:attribute name="name"> <xsl:value-of select="@name"/> </xsl:attribute>
                    <xsl:attribute name="cidoc:E"> <xsl:value-of select="$cidocName"/> </xsl:attribute>
            </xsl:element>
    </xsl:if>
</xsl:template>
```

The template matches all of the attributes embracing *cidoc:E* in *Types*, then adds "*cidoc:*" for them as prefixes, so that they become the values of the variable *cidocName*, then traverses the whole conceptual models written in *cidoc:conceptualModel*, so as to determine whether the nodes with the above variable as their element names have never occurred in the models. If this condition is true, this means the annotation has no corresponding node in the conceptual model, i.e. no corresponding concept item, thus this data type and its relevant semantic annotation needs to be outputted, so that conceptual models can be updated continuously, until all concepts are interpreted. In order to distinguish the above output results and the corresponding relationship between concepts and data types, the application of the above template can be put into the template matching "*//wsdl:types*", behind the template *cidoc:conceptualModel*. The specific form is as follows:

```
<xsl:template match="//wsdl:types">
        <cidoc:mappingAnnotation>
                <xsl:apply-templates select="cidoc: conceptualModel "/>
        </cidoc:mappingAnnotation>
        <cidoc:noMapping> <xsl:apply-templates select="s:schema//*[@cidoc:E]"/> </cidoc:noMapping>
</xsl:template>
```

From the above template we will see: the output of CtoD mapping eventually put into *cidoc:mappingAnnotation* element, while concepts not interpreted are listed in *cidoc:noMapping* element. If there is no element in *cidoc:noMapping* element, then the conceptual model constructed is complete, otherwise all

of the concepts listed in *cidoc:noMapping* element need to be interpreted in turn. Using the previous example, the following illustrates the execution result of the template in the *cidoc:noMapping* element.

```
<cidoc:noMapping>
        <s:element name="timeSpan"  cidoc:E="cidoc:E52.Time-Span"/>
        <s:element name="type"  cidoc:E="cidoc:E55.Type"/>
        <s:element name="preservationPlace"  cidoc:E="cidoc:E53.Place"/>
</cidoc:noMapping>
```

The conceptual model described in *cidoc: conceptualModel* element does not contain the definitions of some concepts, such as E52.Time-Span, E55.Type and E53.Place. But when the element of *Painting* is annotated, the above three concepts are used. The user of Web service is likely to lead to misunderstanding because the elements annotated are often easily misreading. In order to avoid the case, the concepts for annotating must be described at least once in the conceptual model. The function of the template given above is to list all of the concepts which are not defined in the conceptual model.

The *name* and *cidoc:E* attributes of an element in *cidoc: noMapping* element can be thought of as the output which reflect the DtoC mapping. The full list of the elements in *cidoc: noMapping* obtained before the embedded conceptual model is established entirely reflects the correspondence between data types and concepts, and so helps the constructor of the conceptual model determine the range of concepts in the model.

V. Conclusion

The accuracy of data understanding plays a key role in data integration and system merge. One of the functions for annotating data types defined in WSDL is to facilitate Web service consumers to understand the semantics of each data type. The semantic annotation can be implemented through the forming of the corresponding relationship between the data types and the concepts in certain field conceptual mode. When Web service consumers use data obtained, they not only need to understand the relationship between the data types of these data and some conceptual models, but also hope to discover the relationship between concepts and data types from these conceptual models in complex applications, that is, the bidirectional mapping relationships between the concepts and the data types facilitate further understanding of the data. The XSLT-based transformation mapping method given in this paper let semantics annotators of WSDL data types and user of WSDL files, while annotating or reading the data types defined in WSDL, acquire the corresponding relationship between the concepts and the data types by embedding the domain conceptual model into the WSDL file, and introducing additional semantic attributes into the definitions of the data types and does not destroy the original WSDL data type definitions. And the semantics annotators can acquire concept list of uncompleted interpretations, so that they can know where the conceptual models need to be improved further.

Acknowledgment

This work is supported by the grant from National Natural Science Foundation of China (No. 61271369).

References

[1] Roberto Chinnici, Jean-Jacques Morea, Arthur Ryman, Sanjiva Weerawarana, *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, W3C Recommendation, 2006.

[2] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. T. Schmidt, A. Sheth and K., Verma, *Web Service Semantics - WSDL-S*, Technical report, W3C Member Submission, 2005.

[3] Farrell Joel, Lausen Holger, *Semantic annotations for WSDL and XML schema*, W3C recommendation, 2007.

[4] PMK Gordon, CW Sensen, "Creating Bioinformatics Semantic Web Services from Existing Web Services: A Real-World Application of SAWSDL," *Web Services, 2008. ICWS '08. IEEE International Conference*, pp. 608 - 614, 2008.

[5] Kashif Iqbal , Marco Luca Sbodio , Vassilios Peristeras , Giovanni Giuliani, "Semantic Service Discovery using SAWSDL and SPARQL," *Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid*, pp. 205-212, 2008.

[6] GC Hobold, F Siqueira, "Discovery of Semantic Web Services Compositions Based on SAWSDL Annotations," *Web Services (ICWS), 2012 IEEE 19th International Conference*, pp.280 - 287, 2012.

[7] Dengping Weia, Ting Wanga, Ji Wangc, Abraham Bernsteinb, "SAWSDL-iMatcher: A customizable and effective Semantic Web Service matchmaker," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, pp. 347-504, December 2011.

[8] Martha Varguez-Moo, Francisco Moo-Mena, Victor Uc-Cetina, "Use of Classification Algorithms for Semantic Web Services Discovery," *Journal of Computers*, vol. 8, No 7: Special Issue: Advances in Internet Technologies and Applications, pp. 1633-1634, 2013.

[9] R. Wang, S. Ganjoo, J. Miller, and E. Kraemer, "Ranking-Based Suggestion Algorithms for Semantic Web Service Composition," in *Proceedings of 2010 IEEE International Workshop on Web in conjunction with the 2010 IEEE International Conference on Web Services (ICWS)*, pp. 606-613, July 2010.

[10] Matthias Klusch, Patrick Kapahnke, Ingo Zinnikus, "Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer," *The Semantic Web: Research and Applications*, vol. 5554, pp. 550-564, 2009.

[11] Yingjie Song, Rong Chen, Yaqing Liu, "A Non-Standard Approach for the OWL Ontologies Checking and Reasoning," *Journal of Computers*, vol. 7, No 10: Special

Issue: Special Issue: Advances in Information and Computers, pp. 2454-2461, 2012.

[12] Bo Hu, Zhi-Xue Wang, Qing-Chao Dong, "A Novel Context-aware Modeling and Reasoning Method based on OWL," *Journal of Computers*, vol. 8, No 4, pp. 943-950, 2013.

[13] Yuxin Wang, Hongyu Li, "Annotating WSDL by CIDOC CRM," *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS'09)* , vol. III, pp.1389-1392, 2009.

[14] Nick Crofts, Martin Doerr, Tony Gill, Stephen Stead, Matthew Stiff, *Definition of the CIDOC Conceptual Reference Model Version 4.2*, The International Committee for Doc-umentation of the International Council of Museums (ICOM-CIDOC), Paris, 2005.

[15] James Clark, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, 1999.

[16] Martin Doerr, "The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata," *AI Magazine*, vol. 24, pp. 75-92, 2003.