Measuring Granularity of Web Services with Semantic Annotation

Muchalintamolee N.

Computer Science Program, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
Email address: nuttida.mu@student.chula.ac.th

Senivongse T.

Computer Science Program, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
Email address: twittie.s@chula.ac.th

Abstract

Web services technology has been one of the mainstream technologies for software development since Web services can be reused and composed into new applications or used to integrate software systems. Granularity or size of a service refers to the functional scope or the amount of detail associated with service design and it has an impact on the ability to reuse or compose the service in different contexts. Designing a service with the right granularity is a challenging issue for service designers and mostly relies on designers' judgment. This paper presents a granularity measurement model for a Web service with semantics-annotated WSDL. The model supports different types of service design granularity, and semantic annotation helps with the analysis of the functional scope and amount of detail associated with the service. Based on granularity measurement, we then develop a measurement model for service reusability and composability. The measurements can assist in service design and the development of service-based applications.

Keywords*: Service granularity; measurement; reusability; composability; semantic Web services; ontology*

1 Introduction

Web Services technology has been one of the mainstream technologies for software development since it enables rapid flexible development and integration of software systems. The basic building blocks are Web services which are software units providing certain functionalities over the Web and involving a set of interface and protocol standards, e.g. Web Service Definition Language (WSDL) as a service contract, SOAP as a messaging protocol, and Business Process Execution Language (WS-BPEL) as a flow-based language for service composition [1]. The technology promotes service reuse and service composition as the functionalities provided by a service should be reusable or composable in different contexts of use. Granularity of a service impacts on its reusability and composability.

Erl [1] defines *granularity* in the context of service design as "the level of (or absence of) detail associated with service design." The service contract or service interface is the primary concern in service design since it represents what the service is designed to do and gives detail about the scope or size of it. Erl classifies four types of service design granularity: (1) *Service granularity* refers to the functional scope or the quantity of potential logic the service could encapsulate based on its context. (2) *Capability granularity* refers to the functional scope of a specific capability (or operation). (3) *Data granularity* is the amount of data to be exchanged in order to carry out a capability. (4) *Constraint granularity* is the amount of validation constraints associated with the information exchanged by a capability.

Different types of granularity impacts on service reusability and composability in different ways. Erl differentiates between these two terms. Reusability is the ability to express agnostic logic and be positioned as a reusable enterprise resource, whereas composability is the ability to participate in

multiple service composition [1]. A coarse-grained service with a broad functional context should be reusable in different situations while a fine-grained service capability can be composable in many service assemblies. Coarse-grained data exchanged by a capability could be a sign that the capability has a large scope of work and should be good for reuse while a capability with very fine-grained (detailed) data validation constraints should be more difficult to reuse or compose in different contexts with different data formats. Inappropriate granularity design affects not only reusability and composability but also performance of the service. Fine-grained capabilities, for example, may incur invocation overheads since many calls have to be made to perform a task [2]. Designing a service with the right granularity is a challenging issue for service designers and mostly relies on designers' judgment.

To help determine service design granularity, we present a granularity measurement model for a Web service with semantics-annotated WSDL. The model supports all four types of granularity and semantic annotation is based on the domain ontology of the service which is expressed in OWL [3]. The motivation is semantic annotation should give more information about functional scope of the service and other detail which would help to determine granularity more precisely. Semantic concepts from the domain ontology can be annotated to different parts of a WSDL document using Semantic Annotation for WSDL and XML Schema (SAWSDL) [4]. Based on granularity measurement, we then develop a measurement model for service reusability and composability.

Section II of the paper discusses related work. Section III introduces a Web service example which will be used throughout the paper. The granularity measurement model and the reusability and composability measurement models are presented in Sections IV and V. Section VI gives an evaluation of the models and the paper concludes in Section VII.

2 Related Work

Several research has addressed the importance of granularity to service-oriented systems. Haesen et al. [5] proposes a classification of service granularity types which consists of data granularity, functionality granularity, and business value granularity. Their impact on architectural issues, e.g., reusability, performance, and flexibility, is discussed. In their approach, the term "service" refers more to an operation rather than a service with a collection of capabilities as defined by Erl. Feuerlicht [6]0 discusses that service reuse is difficult to achieve and uses composability as a measure of service reuse. He argues that granularity of services and compatibility of service interfaces are important to composability, and presents a process of decomposing coarse-grained services into fine-grained services (operations) with normalized interfaces to facilitate service composition.

On granularity measurement, Shim et al. [7] propose a design quality model for SOA systems. The work is based on a layered model of design quality assessment. Mappings are defined between design metrics, which measure service artifacts, and design properties (e.g., coupling, cohesion, complexity), and between design properties and high-level quality attributes (e.g., effectiveness, understandability, reusability). Service granularity and parameter granularity are among the design properties. Service granularity considers the number of operations in the service system and the similarity between them (based on similarity of their messages). Parameter granularity considers the ratio of the number of coarse-grained parameter operations to the number of operations in the system. Our approach is inspired by this work but we focus only on granularity measurement for a single Web service, not on system-wide design quality, and will link granularity to reusability and composability attributes. We notice that their granularity measurement relies on the designer's judgment, e.g., to determine if an operation has fine-grained or coarse-grained parameters. We thus use semantic annotation to better understand the service. Another approach to granularity measurement is by Alahmari et al. [8]. They propose metrics for data granularity, functionality granularity, and service granularity. The approach considers not only the number of data and operations but also their types which indicate whether the data and operations involve complicated logic. The impact on service operation complexity, cohesion, and coupling is discussed. Khoshkbarforoushha et al. [9] measure reusability of BPEL composite services. The metric is based on analyzing description mismatch and logic mismatch between a BPEL service and requirements from different contexts of use.

3 Example

An online booking Web service will be used to demonstrate our idea. It provides service for any product booking and includes several functions such as viewing product information and creating and managing booking. Figure 1 shows the WSDL 2.0 document of the service. Suppose the WSDL is enhanced with semantic descriptions. The figure shows the use of SAWSDL tags [4] to reference to the semantic concepts in a service domain ontology to which different parts of the WSDL correspond. Here the meaning of the data type named *ProductInfo* is the term *ProductInfo* in the domain ontology *OnlineBooking* in Figure 2, and the meaning of the operation named *viewProduct* is the term *SearchProductDetail*.

4 Granularity Measurement Model

Granularity measurement considers the schema and semantics of the WSDL description. Semantic granularity is determined first and then applied to different granularity types.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description
   targetNamespace="http://localhost:8101/GranularityMeasurement/ wsdl/OnlineBooking#"
   xmlns="http://localhost:8101/GranularityMeasurement/wsdl/ OnlineBooking#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:wsdl="http://www.w3.org/ns/wsdl"
   xmlns:sawsdl="http://www.w3.org/ns/sawsdl">
   <wsdl:types>
      <xs:schema targetNamespace="http://localhost:8101/ GranularityMeasurement/wsdl/OnlineBooking#"
            elementFormDefault="qualified">
         <xs:element name="viewProductReq" type="productId"/>
         <xs:element name="viewProductRes" type="productInfo"/>
            …
         <xs:simpleType name="productId">
            <xs:restriction base="xs:string">
               <xs:pattern value="[0-9]{4}"/>
            </xs:restriction>
         </xs:simpleType>
         <xs:complexType name="productInfo" sawsdl:modelReference="http://localhost:8101/Granularity
               Measurement/ontology/OnlineBooking#ProductInfo">
            <xs:sequence>
               <xs:element name="productName" type="xs:string"/>
               <xs:element name="productType" type="productType"/>
               <xs:element name="description" type="xs:string"/>
               <xs:element name="unitPrice" type="xs:float"/>
            </xs:sequence>
         </xs:complexType>
         <xs:simpleType name="productType">
            <xs:restriction base="xs:string">
               <xs:pattern value="[A-Z]"/>
            </xs:restriction>
         </xs:simpleType>
            …
      </xs:schema>
   </wsdl:types>
   <wsdl:interface name="OnlineBookingWSService"
         sawsdl:modelReference="http://localhost:8101/Granularity
         Measurement/ontology/OnlineBooking#OrderManagement">
      <wsdl:operation name="viewProduct" pattern="http://www.w3.org/ns/wsdl/in-out"
            sawsdl:modelReference="http://localhost:8101/Granularity
            Measurement/ontology/OnlineBooking#SearchProductDetail">
         <wsdl:input element="viewProductReq"/>
         <wsdl:output element="viewProductRes"/>
      </wsdl:operation>
         …
   </wsdl:interface>
</wsdl:description>
```

Figure 1: WSDL of online booking Web service with SAWSDL annotation.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  …
   <owl:Ontology />
   <owl:ObjectProperty rdf:ID="part"/>
     …
   <owl:Class rdf:ID="OrderManagement" />
     …
   <owl:Class rdf:ID="ProductInfo" />
   <owl:Class rdf:ID="HotelInfo" >
      <rdfs:subClassOf rdf:resource="#ProductInfo" />
   </owl:Class>
     …
   <owl:Class rdf:ID="ProductName" >
      <rdfs:subClassOf rdf:resource="#Name" />
   </owl:Class>
   <owl:FunctionalProperty rdf:ID="hasProductID">
      <rdfs:subPropertyOf rdf:resource="#part"/>
      <rdfs:domain rdf:resource="#ProductInfo" />
      <rdfs:range rdf:resource="#ID" />
      <rdf:type rdf:resource="&owl;ObjectProperty" />
   </owl:FunctionalProperty>
   <owl:FunctionalProperty rdf:ID="hasProductName">
      <rdfs:subPropertyOf rdf:resource="#part"/>
      <rdfs:domain rdf:resource="#ProductInfo" />
      <rdfs:range rdf:resource="#ProductName" />
      <rdf:type rdf:resource="&owl;ObjectProperty" />
   </owl:FunctionalProperty>
   <owl:FunctionalProperty rdf:ID="hasProductPrice">
      <rdfs:subPropertyOf rdf:resource="#part"/>
      <rdfs:domain rdf:resource="#ProductInfo" />
      <rdfs:range rdf:resource="#Price" />
      <rdf:type rdf:resource="&owl;ObjectProperty" />
   </owl:FunctionalProperty>
   <owl:FunctionalProperty rdf:ID="hasProductType">
      <rdfs:subPropertyOf rdf:resource="#part"/>
      <rdfs:domain rdf:resource="#ProductInfo" />
      <rdfs:range rdf:resource="#Type" />
      <rdf:type rdf:resource="&owl;ObjectProperty" />
   </owl:FunctionalProperty>
     …
   <owl:Class rdf:ID="SearchProductDetail" />
   <owl:Class rdf:ID="SearchProductInfo" >
      <rdfs:subClassOf rdf:resource="#SearchProductDetail" />
   </owl:Class>
   <owl:Class rdf:ID="SearchRelatedProductInfo" >
      <rdfs:subClassOf rdf:resource="#SearchProductDetail" />
   </owl:Class>
   <owl:Class rdf:ID="GetProductUpdate" />
   <owl:Class rdf:ID="GetProductPriceUpdate" />
   <owl:FunctionalProperty rdf:ID="hasGetProductUpdate">
      <rdfs:subPropertyOf rdf:resource="#part"/>
      <rdfs:domain rdf:resource="#SearchProductDetail" />
      <rdfs:range rdf:resource="#GetProductUpdate" />
      <rdf:type rdf:resource="&owl;ObjectProperty" />
   </owl:FunctionalProperty>
   <owl:FunctionalProperty rdf:ID="hasGetProductPriceUpdate">
      <rdfs:subPropertyOf rdf:resource="#part"/>
      <rdfs:domain rdf:resource="#SearchProductDetail" />
      <rdfs:range rdf:resource="#GetProductPriceUpdate" />
      <rdf:type rdf:resource="&owl;ObjectProperty" />
   </owl:FunctionalProperty>
     …
</rdf:RDF>
```

Figure 2: A part of domain ontology for online booking (in OWL).

A. Semantic Granularity

When a part of WSDL is annotated with a semantic term, we determine the functional scope and amount of detail associated with that WSDL part through the semantic information that can be derived from the annotation. Class-subclass and whole-part property are semantic relations that are considered. Class-subclass is a built-in relation in OWL but whole-part is not. We define an ObjectProperty *part* (see Figure 2) to represent the whole-part relation, and any whole-part relation between classes will be defined as a subPropertyOf *part*. Then, semantic granularity of a term *t* which is in a class-subclass/whole-part relation is computed by (1):

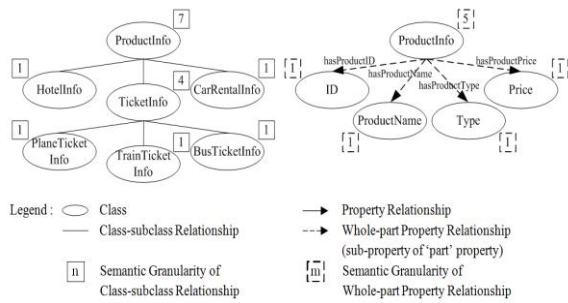

Figure 3: Semantic granularity of *ProductInfo* and related terms.

$$\text{Semantic Granularity}(t) = \text{no. of terms under } t \text{ in either class-subclass relation or whole-part relation, including itself} \quad (1)$$

Using (1), Figure 3 shows semantic granularity of the semantic term *ProductInfo* and its related terms with respect to class-subclass and whole-part property relations. When an ontology term is annotated to a WSDL part, it transfers its semantic granularity to the WSDL part.

B. Constraint Granularity

A service capability (or operation) needs to operate on correct input and output data, so constraints are put on the exchanged data for a validation purpose. Constraint granularity considers the number of control attributes and restrictions (not default) that are assigned to the schema of WSDL data, e.g.,

- Attribute of <xs:element/> such as "fixed", "nullable", "maxOccur" and "minOccur"
- <xs:restriction/> which contains a restriction on the element content.

Constraint granularity *R* of a capability *o* is computed by (2):

$$R_o = \sum_{i=1}^{n} \sum_{j=1}^{mi} Constraint_{ij} \quad (2)$$

where n = the number of parameters of the operation o

m_i = the number of elements/attributes of i^{th} parameter

$Constraint_{ij}$ = the number of constraints of an element/ attribute of a parameter .

In Figure 1, the operation *viewProduct* has two constraints on two out of five input/output data elements, i.e., constraints on *productId* and *productType*. So its constraint granularity is 2.

C. Data Granularity

A WSDL document normally describes the detail of the data elements, exchanged by a service capability, using the XML schema in its <types> tag. With semantic annotation to a data element, semantic detail is additionally described. If the semantic term is defined in a class-subclass relation (i.e., it has subclasses), then the term will transfer its generalization, encapsulating several specialized concepts, to the data element that it annotates. If the semantic term is defined in a whole-part relation (i.e., it has parts), it will transfer its whole concept, encapsulating different parts, to the data element that it annotates.

For a data element with no sub-elements (i.e., lowest-level element), we determine its granularity DG_{LE} by its class-subclass and whole-part relations. For whole-part, if the element has an associated whole-part semantics, we determine the parts from the semantic term; otherwise the part is 1, denoting the lowest-level element itself (see (3)). For a data element with sub-elements, we compute its granularity DG_E by a summation of the data granularity of all its immediate sub-elements DG_{SE} together with the semantic granularity of the element itself (see (4)). Note that (4) is recursive. Finally, for data granularity D_o of a capability o, we compute a summation of data granularity of all parameter elements (see (5)).

$$DG_{LE} = ac_p + \max(1, ap_p) \quad (3)$$

$$DG_E = \sum_{j=1}^{m} DG_{SEj} + ac_p + ap_p \quad (4)$$

$$D_o = \sum_{i=1}^{n} DG_{Ei} \quad (5)$$

where n = the number of parameters of the operation o

DG_E = data granularity of an element with sub-elements/attributes

m = the number of sub-elements/attributes of an element

DG_{SE} = data granularity of an immediate sub-element/attribute of an element

DG_{LE} = data granularity of a lowest-level element/ attribute

ac_p = semantic granularity in the class-subclass relation of an element/attribute, computed by (1)

ap_p = semantic granularity in the whole-part property relation of an element/ attribute, computed by (1).

In Figure 1, the input *viewProductReq* of the operation *viewProduct* has no sub-elements or semantic annotation, so its granularity as a DG_{LE} is 1 (0+max(1, 0)). In contrast, the output *viewProductRes* is of type *productInfo* which is also annotated with the ontology term *ProductInfo*. From the schema in Figure 1, this output has four sub-elements (*productName*, *productType*, *description*, *unitPrice*). Each sub-element has no further sub-elements or semantic annotation, so its granularity as a DG_{LE} is 1 as well. In Figure 3, the semantic term *ProductInfo* has three direct subclasses and three indirect subclasses as well as four parts. The granularity of the output data *viewProductRes* as a DG_E would be 16 (i.e., ((1+1+1+1)+7+5). Therefore data granularity D_o of the operation *viewProduct* is 17 (1+16).

D. Capability Granularity

The functional scope of a service capability can be derived from data granularity and semantic annotation. If large data are exchanged by the capability, it can be inferred that the capability involves a big task in the processing of such data. We can additionally infer that the capability is broad in scope if its semantics involves other specialized functions (i.e., having a class-subclass relation) or other sub-tasks (i.e., having a whole-part relation).

Capability granularity C_o of a capability o is then computed by (6):

$$C_o = D_o + ac_o + ap_o \quad (6)$$

where D_o = data granularity of the operation o

ac_o = semantic granularity in the class-subclass relation of the operation o, computed by (1)

ap_o = semantic granularity in the whole-part property relation of the operation o, computed by (1).

From the previous calculation, data granularity of the operation *viewProduct* in Figure 1 is 17. This operation is annotated with the semantic term *SearchProductDetail*. In Figure 2, this semantic term is a generalization of two concepts *SearchProductInfo* and *SearchRelatedProductInfo*, so the capability *viewProduct* encapsulates these two specialized tasks. The semantic term *SearchProductDetail* also comprises two sub-tasks *GetProductUpdate* and *GetProductPriceUpdate* in a whole-part relation. Therefore capability granularity of *viewProduct* is 23 (17+3+3).

E. Service Granularity

The functional scope of a service is determined by all of its capabilities together with semantic annotation which would describe the scope of use of the service semantically. Service granularity S_w of a service w is computed by (7):

$$S_w = \sum_{i=1}^{k} C_{oi} + ac_w + ap_w \quad (7)$$

where k = the number of operations of the service w

C_o = capability granularity of an operation o

ac_w = semantic granularity in the class-subclass relation of the service w, computed by (1)

ap_w = semantic granularity in the whole-part property relation of the service w, computed by (1).

In Figure 1, the online booking service is associated with the semantic term *OrderManagement*. Suppose the term *OrderManagement* has no subclasses but comprises eight concepts (i.e., parts) in a whole-part property relation. So its service granularity is the summation of capability granularity of the operation *viewProduct* (i.e., 23), capability granularity of all other operations, and semantic granularity in class-subclass and whole-part property relations (i.e., 1+9).

It is seen from the granularity measurement model that semantic annotation helps complement granularity measurement. For the case of the operation *viewProduct*, for example, the granularity of its capability can only be inferred from the granularity of its data if the operation has no semantic annotation. However, by annotating this operation with the generalized term *SearchProductDetail*, we gain knowledge about its broad scope such that its capability encapsulates both specialized *SearchProductInfo* and *SearchRelatedProductInfo* tasks. The additional information refines the measurement.

5 Reusability and Composability Measurement Models

As mentioned in Section I, reusability is the ability to express agnostic logic and be positioned as a reusable enterprise resource, whereas composability is the ability to participate in multiple service composition. We see that reusability is concerned with putting a service as a whole to use in different contexts. Composability is seen as a mechanism for reuse but it focuses on assembly of functions, i.e., it touches reuse at the operation level, rather than the service level. We follow the method in [7] to first identify the impact the granularity has on reusability and composability attributes and then derive measurement models for them. Table 1 presents impact of granularity.

For reusability, a coarse-grained service with a broad functional context providing several functionalities should be reused well as it can do many tasks serving many purposes. Coarse-grained data, exchanged by an operation, could be a sign that the operation has a large scope of work and should be good for reuse as well. So we define a positive impact on reusability for coarse-grained data, capabilities, and services. For composabilty, we focus at the service operation level and service granularity is not considered. A small operation doing a small task exchanging small data should be easier to include in a composition since it does not do too much work or exchange excessive data that different contexts of use may require or can provide. So we define a negative impact on composability for coarse-grained capabilities and data. For constraints on data elements, the bigger number of constraints means finer-grained

restrictions are put on the data; they make the data more specific and may not be easy for reuse, hence a negative impact on both attributes.

Table 1: Impact of granularity on Reuse

Granularity Type	Reusability	Composability
Service Granularity	↑	-
Capability Granularity	↑	↓
Data Granularity	↑	↓
Constraint Granularity	↓	↓

A. Reusability Model

Reusability measurement is derived from the impact of granularity. It can be seen that different types of granularity measurement relate to each other. That is, service granularity is built on capability granularity which in turn is built on data granularity, and they all have a positive impact. So we consider only service granularity in the model since the effects of data granularity and capability granularity are already part of service granularity. The negative impact of constraint granularity is incorporated in the model (8):

$$Reusability = S_w - \sum_{i=1}^{k} R_{oi} \quad (8)$$

where S_w = service granularity of the service w

R_o = constraint granularity of the operation o

k = the number of operations of the service w.

A coarse-grained service with small data constraints has high reusability.

B. Composability Model

In a similar manner, we consider only capability granularity and constraint granularity in the composability model because the effects of data granularity are already part of capability granularity. Since they all have a negative impact, we represent composability measure in the opposite meaning. We define a term "uncomposabilty" to represent an inability of a service operation to be composed in service assembly (9):

$$Uncomposability = C_o + R_o \quad (9)$$

where C_o = capability granularity of the operation o

R_o = constraint granularity of the operation o.

A fine-grained capability with small data constraints has low uncomposability, i.e. high composability.

6 Evaluation

We apply the measurement models to two Web services. The first one is the online booking Web service which we have used to demonstrate the idea. It is a general service including a large number of small data and operations. Its scope covers viewing, managing, and booking products. Another Web service is an online order service which has only a booking-related function. The two Web services are annotated with semantic terms from the online booking ontology which describes detail about processes and data in the online booking domain. Table 2 shows details of some operations of the two services including their capabilities, data, and semantic annotation.

For the evaluation, a granularity measurement tool is developed to automatically measure granularity of Web services. It is implemented using Java and Jena [10]0 which helps with ontology processing and inference of relations.

Table 3 presents granularity measurements and reusability scores. The online booking service is coarser and has higher reusability. It is a bigger service with wider range of functions, exchanging more data, and having a number of data constraints. It is likely that the online booking service can be put to use in various contexts. On the other hand, the online order service is finer-grained focusing on order management. The two services are annotated with semantic terms of the same ontology, and additional semantic detail helps refine their measurements.

Table 4 presents granularity measurements and uncomposability of the operations annotated with the semantic term *UpdateOrder*. The operation *editOrderItem* of the online order service has coarser data and capability compared to the three finer-grained operations of the online booking service, and therefore it is less composable.

7 Conclusions

This paper explores the application of semantics-annotated WSDL to measuring design granularity of Web services. Four types of granularity are considered together with semantic granularity. The models for reusability and composability (represented by uncomposability) are also introduced.

As explained in the example, semantic annotation can help us derive the functional contexts and concepts that the service, capability, and data element encapsulate. Granularity measurement which is

traditionally done by analyzing the size of capability and data described in standard WSDL and XML schema documents can be refined and better automated.

Table 2: Part of Service Detail and Semantic Annotation

Operation		Input Data Type		Output Data Type	
Name	***Annotation***	***Name***	***Annotation***	***Name***	***Annotation***
Online booking web service					
newCart	Insert Order	userId	ID	orderId	ID
addProduct ToCart	Update Order	addProduct	OrderItem	process Result	Status
delete Product FromCart	Update Order	delete Product	OrderItem	process Result	Status
editProduct Quantity InCart	Update Order	editProduct Quantity	OrderItem	process Result	Status
view Product InCart	Search OrderItem ByOrderID	orderId	ID	orderItem List	-
reservation	EditOrder	reserved Order	ID	process Result	Status
Online order web service					
createOrder	Create Order	order Request	Order	order Response	Status
edit OrderItem	Update Order	editOrder ItemInfo	Order	orderItem Response	Status
submit Order	EditOrder	orderId	ID	order Response	Status

Table 3: Granularity and Reusability

Service Name	Granularity				Reusability
	$\sum R_o$	$\sum D_o$	$\sum C_o$	S_w	$S_w - \sum R_o$
OnlineBookingWSService	48	143	184	194	146
OnlineOrderWSService	10	47	62	72	62

Table 4: Service Granularity and Uncomposability of Operations Annotated with UpdateOrder

Service Name	Operation Name	Granularity				Uncomposability
		R_o	D_o	C_o	S_w	$C_o + R_o$
Online Booking WSService	addProduct ToCart	4	15	18	-	22
	DeleteProduct FromCart	3	14	17	-	20
	editProduct Quantity InCart	4	15	18	-	22
Online Order WSService	editOrderItem	3	19	22	-	25

For future work, we aim to refine the domain ontology and WSDL annotation. It would be interesting to see the effect of annotation on granularity, reusability, and composability when the WSDL contains a lot of annotations compared to when it is less annotated. Since annotation can be made to different parts of WSDL, the location of annotations can also affect granularity scores. Additionally we will try the models with Web services in business organizations and extend the models to apply to composite services.

References

[1] T. Erl, SOA: Principle of Service Design, Prentice Hill, 2007.

[2] T. Senivongse, N. Phacharintanakul, C. Ngamnitiporn, and M. Tangtrongchit, "A capability granularity analysis on Web service invocations," in Procs. of World Congress on Engineering and Computer Science 2010 (WCECS 2010), 2010, pp. 400-405.

[3] W3C (2004, February 10) OWL Web Ontology Language Overview [Online]. Available: http://www.w3.org/TR/2004/REC-owl-features-20040210/

[4] W3C (2007, August 28) Semantic Annotations for WSDL and XML Schema [Online]. Available: http://www.w3.org/TR/2007/REC-sawsdl-20070828/

[5] R. Haesen, M. Snoeck, W. Lemahieu and S. Poelmans, "On the definition of service granularity and its architectural impact," in Procs. of 20th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2008), LNCS 5074, 2008, pp. 375-389.

[6] G. Feuerlicht, "Design of composable services," in Procs. of 6th Int. Conf. on Service Oriented Computing (ICSOC 2008), LNCS 5472, 2008, pp. 15-27.

[7] B. Shim, S. Choue, S. Kim and S. Park, "A design quality model for service-oriented architecture," in Procs. of 15th Asia-Pacific Software Engineering Conference (APSEC 2008), 2008, pp. 403-410.

[8] S. Alahmari, E. Zaluska, D. C. De Roure, "A metrics framework for evaluating SOA service granularity," in Procs. of IEEE Int. Conf. on Service Computing (SCC 2011), 2011, pp. 512-519.

[9] A. Khoshkbarforoushha, P. Jamshidi, F. Shams, "A metric for composite service reusability analysis," in Procs. of the 2010 ICSE Workshop on Emerging Trends in Software Metrics (WETSoM 2010), 2010, pp. 67-74.

[10] Apache Jena [online]. Available: http://incubator.apache.org/jena/, Last accessed: January 30, 2012.