OWL-SOA: A Service Oriented Architecture Ontology Useful During Development Time And Independent From Implementation Technology

Wagner Arnaut
IBM Software Group
Brazil
warnaut@br.ibm.com

Káthia Oliveira
Univ Lille Nord de France, F-59000
Lille, France
UVHC, LAMIH, F-59313
Valenciennes, France
CNRS, UMR 8530, F-59313
Valenciennes, France
kathia.oliveira@univ-valenciennes.fr

Fernanda Lima
Computer Science Department,
University of Brasilia,
Campus Universitário Darcy Ribeiro,
ICC Centro, Asa Norte,
Caixa Postal 4466, 70910-900,
Brasília, DF, Brasil
ferlima@cic.unb.br

Abstract—**The implementation of service oriented architecture (SOA) has proven to be one of the main initiatives for organizations in gaining necessary agility in the area of information technology. However, in one particular area this process has presented a significant challenge: the implementation of a service catalog. Ontologies have been used to organize services in repositories, as proposed by of OWL-S (Ontology Web Service Language) and WSMO (Web Service Modeling Ontology). Meanwhile, these ontologies have two broad restrictions: they are used on runtime only, and deal only with Web Service technology for the implementation of services. This study proposes an ontology for SOA with an emphasis on the implementation of a service catalog in the development phase, and considering any type of service implementation technology. This ontology was created based on the existing ontologies and literature. The resulting ontology was instanced with services of a large multinational organization**.

Ontology, Service oriented architecture,

I. Introduction

Aiming to increase the operational flexibility and efficiency of the Information Technology (IT) sector, many organizations worldwide have identified service oriented architecture as a main priority (SOA - Service Oriented Architecture) [1]. According to a study conducted by Business Technographics [2], in 1,078 companies in the United States and Europe, one of the major priorities of IT is the implantation of SOA.

When one plans the implantation of SOA, a repository is necessary for storing the services. Service catalogs and repositories can be present in basically two instances: design-time (or development time) and runtime. There are various strategies to create these repositories. The most widely used protocol is the UDDI (Universal Description, Discovery and Integration [3]. Other authors (e.g. [4][5][1][6][7]) propose the use of ontologies to solve this problem—two of the most notable studies being: OWL-S (Ontology Web Service Language) [6] e WSMO (Web Service Modeling Ontology) [7]. However, these studies only approach the search and recovery of services using Web Services, and no other technologies for the implementation of SOA. Besides this, they only consider search and recovery of services during execution (runtime). Nevertheless, we can think about reusing services or part of the service functionality since the design of the new application system. In this context, we need to identify the service (or some of its components) and all related artifacts (from requirements, design and implementation) during the development to make possible the reuse and integration in the design of the new application system.

This article presents a definition of an ontology, denominated OWL-SOA, to support the search and recovery of services in an SOA architecture, considering the services in development time and regardless of implementation technology utilized. This ontology was defined from the integration of the ontologies OWL-S and WSMO, and evolved considering concepts from the literature about SOA.

In the following section (section II) the theoretical reference is presented, taking definitions of ontologies and SOA into account. Section III describes the construction of the intended ontology. Section IV presents the conclusions and future works.

II. SOA and Ontologies in SOA

A. SOA

SOA can be defined as an architecture that provides the capacity to implement the necessary logic to solve a managerial business problem, through the collection of small portions of business process decompositions [8]. According to [9], SOA is an architectural style that introduces a group of characteristics desirable in the architecture of applications.

From Tsai's perspective [10], the SOA architecture is formed by three basic elements: Service Client or Service Requestor, Service Registry, and the Service Providers or

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

Service Brokers. The responsibility of the Service Provider is to develop services with low coupling and to publish them in a service register. When the Service Client needs a service, it locates services available in a runtime environment through the Service Registry. Once the service is located, the Service Client maps a Service Provider. In this context, the Service Repository is of great importance.

The Service Repositories store data pertaining to services published in the Service Registry. The Service Registry in runtime stores the public interface, contracts, and service access policies. The Service Repository contains the said service, as well as all of the auditing and the service log [11].

This repository contains data about all of the available services and can be an element of the SOA implementation [3] or of an application oriented to service. The repository of services can be implemented in two instances: design-time (development time) and runtime [12]. The development repository stores the service constituent files and all of the documentation; the runtime repository contains the service installed and its entire audit log[11].

Different technologies are used to categorize services offering support to search and recovery in an SOA repository, as for example: protocols such as XML (eXtensible Markup Language) and XML Schema, UDDI (Universal Description, Discovery and Integration), the RAS standard (Reusable Asset Specification[13], and ontologies. However, the protocols XML focus only on syntactic and structural aspects, lacking information on the semantic level, and the UDDI merely describes the functionalities of the services registered in a language that is incomprehensible to machines[14].

The RAS standard defines a model for the representation of assets produced throughout the software development process [13]in such a way as to allow its reuse, being amply utilized to describe and indicate software components. Despite its wide use, the RAS standard does not add semantics to its search and recovery of services.

The RAS specification is defined by a basic set of four sections: the Classification section, which defines the set of descriptors to classify an asset, as well as the context in which that asset is relevant; the Solution section, which presents the artifacts that compose the asset; the Usage section, which contains the rules for installation, customization and use of a particular asset; and finally, the Related Assets section, which describes the relationship of this asset with other assets (OMG, 2005). The profile for Web Service contained in the RAS defines that the artifacts presented in the Solution are Requirement, Design, Implementation, and Test types, and that each artifact must contain a WSDL (Web Service Description Language) file that describes it.

B. Ontologies in SOA

Ontologies are explicit formal specifications of terms in a domain and their relationship to other terms [15], containing a set of concepts (entity, objects, domains, processes, goals, and results), properties, relationships, restrictions, or axioms [16].

In the past years there have been increasing attempts to formally shape services using ontologies (e.g. [4][5][1][6][7]), with OWL-S and WSMO being the most notable initiatives in Web Service semantics.

The OWL-S has as its principle the possibility of discovery automation, invocation, composition, interoperability, and monitoring of Web Services, providing the semantic descriptions appropriate to these services [17]. The model OWL-S is structured in three large sub-ontologies [6]: Service profile, which provides a general description of the Web Service defining what the service does; Service model, which describes how to ask for the service and what happens when the service is carried out; and Service grounding, which specifies how the processes of the service model map for the WSDL and define how to interact with the Web Services.

The WSMO was constituted to create an ontology to describe the various aspects of the Semantic Web Service, seeking to solve integration problems[17]. The WSMO is structured into four elements of modeling [18]: Ontologies, which provide the terminology and concepts used for other WSMO elements, describing aspects relevant to the domain; Web Services, which represent the computational entities that can provide access to services that provide some value in a domain; Objectives, which describe the objectives and characteristics of the necessary functionalities by the users; and Mediators, which describe elements that deal with challenges of interoperability among WSMO elements.

Both OWL-S and WSMO provide an extensive ontology for the framework based on the description of Web Services. Web Services is an implementation language well known for SOA; however, it is not the only implementation possible [9]. Moreover, the OWL-S and WSMO do not deal with the perspective of implementation of a service repository in development time.

III. DEFINITION OF OWL-SOA ONTOLOGY

Considering the studies already done in the area of catalog organization and service repository for SOA, two premises were defined:

- to take advantage of existing ontologies (WSMO and OWL-S) that, despite being specific to Web Services and for use in service execution time, have service concepts adequately defined; and
- to seek support in the literature, particularly in the RAS-specified concept in order to complete the necessary knowledge.

Based on these two premises, two ontology engineering methodologies proposed in the literature were integrated (Figure 1): the SABIO [19] as the structure for the main activities in the construction of the ontology; and PROMPT [20] with a sequence of steps to support the integration of the existing ontologies.

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

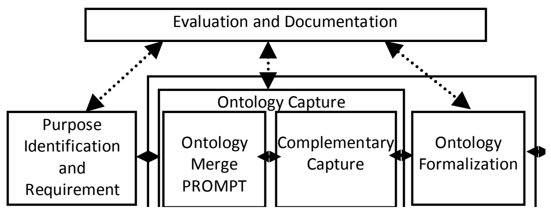


Figure 1. Development Methodology utilized

Thus, the ontology engineering begins with the Purpose Identification and Requirement Specification phase, in which, according to [19] potential scenarios of use of the ontology must be defined as well as competence questions that represent what the ontology must respond or the tasks it can perform. After the Ontology Capture phase, in which the elements of ontology are defined (concepts, attributes, and axioms), the ontology finally must be formalized in an appropriate formal language. Parallel to all the activities, the ontology must be documented and evaluated.

In the beginning of the Ontology Capture phase the PROMPT methodology is applied for the integration of OWL-S and WSMO ontologies.

The PROMPT methodology is based on the two entrance ontologies guiding the process of fusion for one sole exit through a cyclic process that will be presented in section B.

A. *Purpose Identification and Requirement Specification*

The purpose of the ontology is to support the activities of the organization and categorization of services in an SOA architecture — supporting the search and recovery service activities throughout the development time that can be implemented in any technology and not limited to Web Services.

Considering this context, various scenarios of use in which a software architect would traditionally use the ontology, were identified. Some of these scenarios are:

- The software architect in SOA seeks services in the repository that can be reused in a new business process that will be implemented;
- Faced with the non-functional requirement of a business process, the software architecture needs to identify which services were tested and must serve the same non-functional requirement to be able to compose this business process;
- Once a service is located in the repository, the software architect in SOA needs to know how to invoke that service based on the business process. However, the software architect needs to be informed about which public interfaces of that service are operable.

Based on the ontology proposal and the set of defined scenarios, the following competence questions were elaborated to be responded to by the OWL-SOA ontologies:

1) How can a service be decomposed?
2) How is the public interface of a service characterized?
3) In which technology is a certain service available?
4) Which of the business concepts are related to a service?
5) Which of the development assets are correlated to a service?.

B. *Ontology Merge*

Considering that the OWL-S ontology has as its primary objective the search and recovery of services and that the WSMO ontology aims at mediation and integration services, OWL-S was chosen for use as a base for construction of the target ontology denominated OWL-SOA. Whenever faced with concepts of the same significance in both ontologies of entrance, the OWL-S term was used in the OWL-SOA ontology.

The use of the PROMTP methodology implicates in the execution of the following steps:

- List of suggested operations — a study of all the concepts was performed, regarding properties and the relationship between OWL-S and WSMO ontologies. The fusion operations were identified considering the competence questions defined (Table I). Table I contains the desired fusion operation, the identifier of the OWL-S ontology concept, its respective mapping with the identifier of the concept as well as semantic meaning in the WSMO and the final concept in OWL-SOA ontology. The list of fusion operations is composed of fusion (when the concept exists in two ontologies), copy (when there is only one of the two ontologies), or exclusion (when a concept cannot be incorporated in an ontology OWL-SOA) according to the goal defined. For example, the fist operation is a *fusion* from the concepts *Service* from OWL-S and *WebService* from WSMO. It means both concepts were considered similar. The result of this *fusion* operation is the *Service* concept in the OWL-SOA ontology. For the copy operation, we note that we have *complete copy* (for example, operation 7) and *partial copy* (for example, operation 8). The first means that the concept was completely copied in the OWL-SOA ontology, with its all properties (attributes). The second (partial copy) means the concept was copied in the OWL-SOA but some proprieties were not considered since they are not relevant to answer the established competency question and goals.
- Select and achieve the next operation —fusion, copy, or exclusion of each of the elements of the ontology based on what was defined in the list of operations.
- Locate conflicts and update the list of operations – Some conflicts were identified after operations in Table 1 were executed, which brought a second iteration of fusion generating new lines on the fusion table.

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

Table I - LIST OF OPERATIONS FOR FUSION – 1ST INTERACTION.

#	Operation	OWL-S	WSMO	OWL-SOA
Competence Question 1: How can a service be decomposed?				
1	Fusion	Service	WebService	Service
2	Fusion	ServiceCategory	Annotation	ServiceCategory
3	Exclusion	Process	------	------
4	Partial copy	AtomicProcess	------	AtomicService
5	Partial copy	CompositeProcess	------	CompositeService
6	Exclusion	SimpleProcess	------	------
Competence Question 2: How is the public interface of a service characterized?				
7	Complete copy	------	Interface	Interface
8	Complete copy	------	Orchestration	Orchestration
9	Complete copy	------	Choreography	Choreography
10	Fusion	ServiceProfile	Capability	ServiceProfile
11	Fusion	ServiceParameter	Parameter	ServiceParameter
Competence Question 3: In which technology is a certain service available?				
12	Partial copy	ServiceGrounding	------	ServiceGrounding
13	Partial copy	ServiceGrounding	------	WebService
Competence Question 4: Which of the business concepts are related to a service?				
14	Complete copy	------	Ontology	Ontology
15	Complete copy	------	Concept	Concept
16	Complete copy	------	Attribute	Property
17	Complete copy	------	Relation	Relation
18	Complete copy	------	Axiom	Axiom
19	Exclusion	------	Function	------
20	Exclusion	------	Instance	------
21	Exclusion	------	AttributeValue	------
22	Exclusion	------	LogicalExpression	------
23	Exclusion	------	RelationInstance	------
24	Partial copy	------	Goal	Process
Competence Question 5: Which of the development assets are correlated to a service?				
25	Partial copy	------	Non Functional Properties	Non Functional Properties

To answer the competence question 1, fusion operations from 1 to 6 were done. Through the execution of these operations it is evident that a service can be decomposed in one or more atomic or compound services and can be from different categories.

To answer the competence question 2, fusion operations 7 to 11 were done. Through the proceeding concepts of these fusion operations it is possible to identify the public interface of a service and how to invoke it. The Interface concept of WSMO ontology defines how functionality can be obtained and how a certain service is externalized. In accordance with [7], the Interface concept provides two views of operational competence of a Web Service: the Choreography concept, which decomposes a capacity in terms of the interaction with the Web Service; and the Orchestration concept decomposes a capacity in terms of the functionality required by another Web Service. According to [21] and [12] all service had an interface. Therefore the Interface, Orchestration and Choreography concepts are applied to any implementation service technology. Besides this, other fusions were done causing the definition of the profile of a service and in its parameters of entrance and exit.

To answer the competence question 3, the fusion operations 12 and 13 make evident that a service is implemented in a technology platform and one of its possible implementations is through the Web Services technology. To answer totally this competence question, it is necessary to incorporate new concepts referring to the other possible technology platforms for implementation of services. This analysis will be done in the next section.

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

To answer the competence question 4, fusion operations 14 to 24 were done. Only the concepts that represented the basic constituent elements of an ontology, in accordance with [22]were incorporated in the ontology OWL-SOA. They are: Concept, Relation, Property and Axiom. Therefore, the OWL-SOA ontology, considers a strip of the meta-ontology present in the WSMO ontology to facilitate the use of the ontology and avoid the generation of new ontological commitments. The relations of the meta-ontology incorporated in the OWL-SOA ontology are the relationships proposed by [22], because in the WSMO ontology some of these concepts do not have direct relationships to each other. Also with regards to this competence question a partial copy of the Goal concept of the WSMO ontology was done, which represents the process of use of a service. According to [1], the services of an SOA architecture support business processes. Thus, to semantically align with the SOA nomenclature, the Goal concept was named Business Process, in the OWL-SOA ontology, representing the business process of which a certain service is composed.

To answer the competence question 5, the fusion operation 25 was done, which concerns the partial copy of the Non Functional Property concept occurring in ontology WSMO. In the WSMO ontology it is possible to verify the non-functional properties of a Web Service. These properties represent the real data and information measured in an execution environment. Through the fusion operation 25, it is possible to address the second scenario of use previously shown, in which the software architect needs to confront the test data achieved in the service with the real data measured in the execution of the service. Meanwhile, to completely answer this question other concepts need to be incorporated, which will be shown in the next section.

Some concepts were not included in the resultant OWL-SOA ontology, because they were not applied to answer the competence questions raised in the specification phase.

C. *Complementary Capture*

The aim of this phase was to aid the answering of competence questions.

To thoroughly answer the competence question 5, regarding the development assets associated to a service, the RAS protocol was used. In the context of the implementation of a service repository in an SOA architecture, a service in OWL-SOA ontology is considered an asset of the RAS specification, given that some RAS specifications define the mapping of an asset with a Web Service in one of its Profiles. Since in the proposal of this ontology an asset is a service, and all assets are artifacts, therefore a service is composed of artifacts. Those artifacts can be of different types: Requirements, Design, Implementation and Test artifacts. Among the possible types of implementation artifacts one type deserves highlighting, due to its frequent occurrence SOA: the components. Finally, in fusion operation 7 the Interface concept was incorporated to the OWL-SOA ontology. Thus, based on the focus of the repository in development time, the whole interface is implemented in a file. This file contains the interface source code. Based on this analysis, the Interface concept was incorporated as a sub-type of the Implementation Artifact concept.

To entirely answer the competence question 3, about in which technology a determined service is available, a search in the literature was done to verify possible technologies for the implementation of services in a SOA architecture. The following have been identified: FIX (Financial Information Exchange) language [23], JMS (Java Message Service) language [24], Web Services, REST (Representational State Transfer)[5], CORBA (Common Object Request Broker Architecture) [25], EJB (Enterprise Java Beans) [26] and SBA (Space-Based Architecture) [27]

In the OWL-S specification, the Service Grounding concept included the information regarding the invocation of the Web Services. As the ontology OWL-SOA is independent of the service implementation language it becomes a super-type of specific concepts for each one of the platform technologies.

The complete ontology model is shown in Figure 2. For reasons of simplicity the concept properties were not shown.

As one can observe in Figure 2, the ontology consists of a central concept denominated Service. This concept contains the basic properties of a service. A compound service is made up of other services. Each service is exposed and accessed from one interface, represented by the relationship between the Service concept and the Interface concept. Besides this, each service can be part of one or more business processes, and each business process can be associated to one or more services. A service is implemented in a Service Grounding that represents the different technology platforms in which a service is implemented. A service is composed of one or more artifacts—it must have at least one implementation type artifact. Finally, a service can be associated to the concepts of the domain of which the service is a part. The ontology concept represents the meta-ontology of the domain, with the domain concepts. Figure 3 shows the meta-ontology of the domain defined in this ontology.

D. *Ontology Formalization*

The OWL-SOA ontology was formalized using the OWL 1.0 language (Ontology Web Language) [28] in Protégé tool with the Racer inference motor (http://www.sts.tu-harburg.de/~r.f.moeller/ racer/).

Different axioms were formalized. Some of them are:

- Each service associated to the Service Grounding FIX type has a Financial Ontology associated to it.

```
<owl:Class rdf:about="#FIX">
<owl:Restriction>
        <owl:onProperty>
         <owl:ObjectProperty
       rdf:ID="supportedByFix"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty
                rdf:ID="importsOntology"/>
            </owl:onProperty>
            <owl:allValuesFrom>
              <owl:Class
```

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

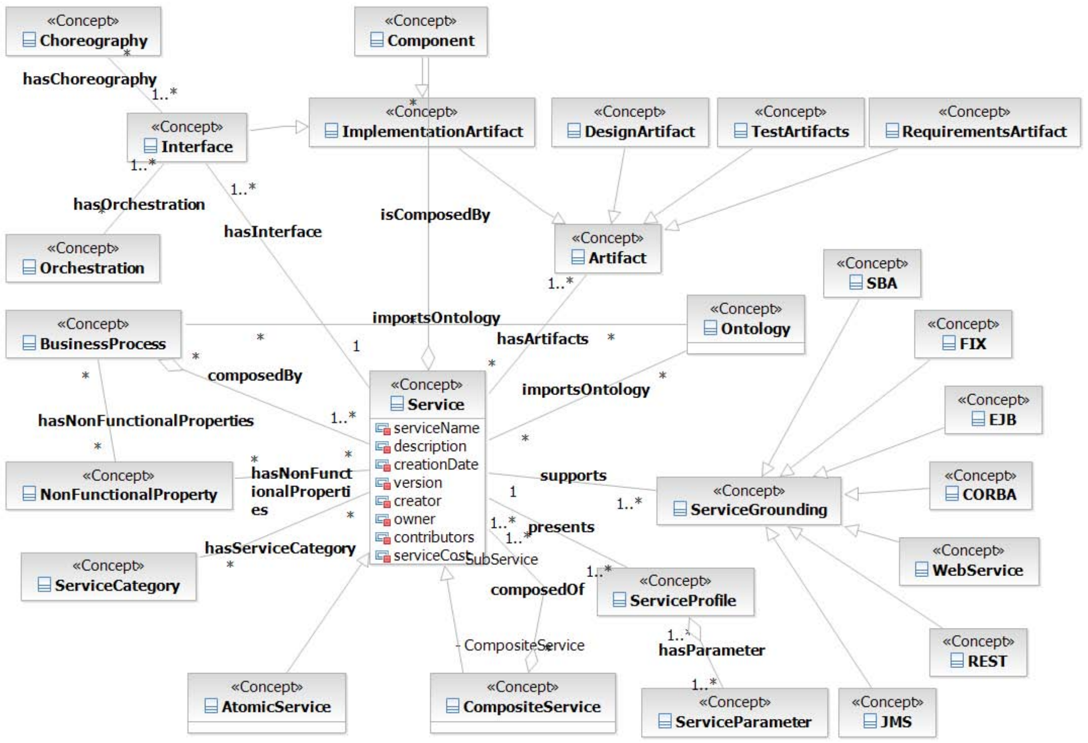


Figure 2. OWL-SOA ontology model

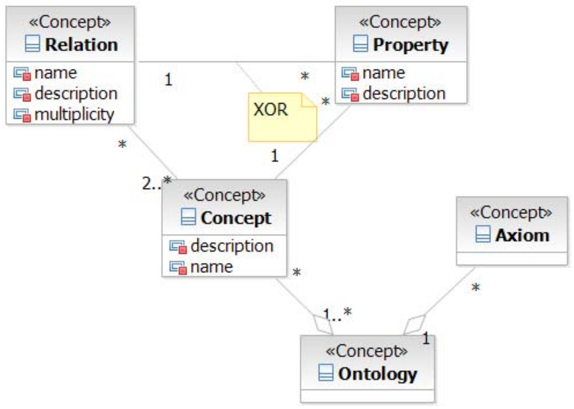


Figure3. Domain meta-ontology model

```
                rdf:ID="FinancialOntology"/>
            </owl:allValuesFrom>
          </owl:Restriction>
        </owl:someValuesFrom>
      </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf>
      <owl:Class
        rdf:about="#ServiceGrounding"/>
    </rdfs:subClassOf>
</owl:Class>
```

- Each service has at least one Implementation Artifact that implements it.

```
<owl:Class rdf:about="#Service">
   <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty
               rdf:ID="hasArtifacts"/>
        </owl:onProperty>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

- Each service is implemented in at least one technology platform, or rather, at least one Service Grounding.

```
<owl:Class rdf:about="#Service">
   <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty
         rdf:ID="supports"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class>
            <owl:unionOf
rdf:parseType="Collection">
              <owl:Class rdf:about="#CORBA"/>
              <owl:Class rdf:about="#EJB"/>
              <owl:Class rdf:about="#FIX"/>
              <owl:Class
rdf:about="#WebService"/>
              <owl:Class rdf:about="#SBA"/>
              <owl:Class rdf:about="#REST"/>
              <owl:Class rdf:about="#JMS"/>
            </owl:unionOf>
          </owl:Class>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

- Each service associated to the Service Grounding FIX type has a Financial Ontology associated to it.

```
<owl:Class rdf:about="#FIX">
<owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty
       rdf:ID="supportedByFix"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Restriction>
            <owl:onProperty>
              <owl:ObjectProperty
               rdf:ID="importsOntology"/>
            </owl:onProperty>
            <owl:allValuesFrom>
              <owl:Class
               rdf:ID="FinancialOntology"/>
            </owl:allValuesFrom>
          </owl:Restriction>
        </owl:someValuesFrom>
      </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf>
      <owl:Class
       rdf:about="#ServiceGrounding"/>
    </rdfs:subClassOf>
</owl:Class>
```

- Each Service implemented with SBA ServiceGrounding has a NonFunctionalProperty with Scalability propriety equals to High.

```
  <owl:Class rdf:about="#SBA">
    <rdfs:subClassOf
       rdf:resource="#ServiceGrounding"/>
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf
         rdf:parseType="Collection">
          <owl:Restriction>
            <owl:someValuesFrom>
              <owl:Class
               rdf:about="#Service"/>
            </owl:someValuesFrom>
            <owl:onProperty>
              <owl:ObjectProperty
               rdf:ID="supportedBy"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Restriction>
            <owl:allValuesFrom>
              <owl:Class rdf:about="#High"/>
            </owl:allValuesFrom>
            <owl:onProperty>
              <owl:ObjectProperty
     rdf:ID="hasNonFunctionalProperties"/>
            </owl:onProperty>
          </owl:Restriction>
          <owl:Class>
            <owl:intersectionOf
               rdf:parseType="Collection">
              <owl:Restriction>
                <owl:onProperty>
                  <owl:ObjectProperty
               rdf:about="#supportedBy"/>
                </owl:onProperty>
                <owl:someValuesFrom>
                  <owl:Class
               rdf:about="#Service"/>
                </owl:someValuesFrom>
              </owl:Restriction>
              <owl:Restriction>
                <owl:allValuesFrom>
                  <owl:Class
               rdf:about="#High"/>
                </owl:allValuesFrom>
                <owl:onProperty>
                  <owl:ObjectProperty
rdf:about="#hasNonFunctionalProperties"/>
                </owl:onProperty>
              </owl:Restriction>
            </owl:intersectionOf>
          </owl:Class>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
```

E. Evaluation and Documentation

During the engineering of the ontology, a dictionary of terms was created for all of the concepts and properties (see some examples in Table II.

As a way of verifying the completeness of the ontology, this was instanced with different actual services of a repository of a large organization. This organization has diverse software development labs spread out over various countries using the SOA architecture as a protocol for development architecture. Currently, the development repository has 121 services, 7 being chosen randomly for instancing of the OWL-SOA ontology. Other examples collected from literature and internet were also used.

From the repository of the organization, we collected the following services:

- Order Status - Service that allows the subscribers of the service to obtain information on the status of requests that were made by the client. In addition, displays the details of the situation in each application.

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

TABLE II. EXAMPLE OF THE DICTIONARY

Concept	Description
Service	Mechanism to support access to one or more capabilities, where the access is by completion of a pre-defined interface
BusinessProcess	Sets the time relationships of actions and associated events to interact with one or more services in the implementation of business process.
ServiceProfile	High-level description of the service. This description contains the input parameters, output, preconditions and effects in service.
ServiceCategory	Represents the category the service belongs.
ServiceGrounding	Specifying the details of how an agent can access a service. Typically specify the communication protocol, message format, port number, communication and other specific details of the service
NonFunctionalProperty	Concept that has the technological and non-functional service. The properties do not represent the functional status of a service at runtime. Moreover, the properties have non-functional parameters of quality of service.
ServiceParameter	Concept that defines the parameters that will be used in ServiceProfiles a service.
Interface	This concept identifies how a particular service can be invoked
Orchestration	Concept that decomposes a capability in terms of interaction with the service. Represents the consumer point of view.
Choreography	Meaning that decomposes a capability in terms of functionality required from another service. Represents the service provider point of view.
Component	Identifies which components are stored in the repository that make up a service. These components relate to the implementation artifacts of the service.
Artifact	Artifacts are physical and logical elements of an asset.
RequirementsArtifact	Artifacts containing the requirements of the service.
ImplementationArtifact	Identifies the binaries and source code that provide the implementation of the component and service.
TestArtifact	Identifies the artifacts describing the test cases and test scripts executed for the service.
FIX	Financial Information Exchange - Language for implementation of specific services to the financial market.
JMS	Java Message Service - Technology that provides Java programs the ability to create, send, receive and read an enterprise messaging system.
Web Service	Software system designed to support interoperable machine-interaction in the machine over the network.
CORBA	Common Object Request Broker Architecture - Technology that enables applications to communicate with each other regardless of where they are located and who designed them.
EJB	Enterprise Java Beans - Technology designed to support the construction of enterprise applications through a component architecture processed and managed on the server side.
SBA	Space-Based Architecture - Technology to transform applications developed in multiple layers in linear and dynamically scalable.
REST	Representational State Transfer - REST architectural style for distributed systems over the Internet.
Ontology	Concept representing domain ontologies associated with the ontology of services.

- ServicePack Validator - Service that lists the available ServicePack for each hardware platform. This service also validates the requests for the installation of available ServicePack hardware configurations
- Retrieve Allowed Values for Specific BDS (Business Data Standard) - A service that returns the complete set of allowed values, according to a standard business data set.
- Process Opportunity Service - This service allows the customer to request a financial analysis of contracts. The process of leasing business, evaluating the opportunity and returns the values to the client.
- Composite Document Process Service - Service that allows the customer to request the generation and delivery of business documents. The support process creates and distributes the document in accordance with the method specified by the applicant.
- Order Manager - Service belonging to the domain of solutions of applications. Its primary mission is to be the provider of service creation, modification and execution of purchase orders.
- Recycle Service Fee - A service that defines what the recycling rate to be levied on sales of certain products and carried out in certain geographic areas where recycling rates apply.

Some examples of literature and internet are:

- Trip Reservation Service - travel reservation service online for Austria and Germany destinations.
- CreateIOI - service responsible for the registration of an indication of interest in a particular action.
- Credit Engine - This service analyzes whether the buyer has sufficient credit to complete the transaction. This service should involve the maximum future value of the transaction.

All concepts were instantiated, which shows their utility in describing a service with the ontology.

Based on the instances, we analyzed if the ontology answers each competence question. Figure 4 shows an example of this analysis for the Order Status service:

- Through the relationships and the axioms of the *Service* concept with itself and *Service* with *Componenti,* we can answer the competence question 1. *Order Status* service is an atomic service, so it is not broken down into other services. *Order Status* service is implemented in a component called *OrderStatus*.
- The relationship between the *Service* and *Interface* concepts, and its axioms are essential to address the competence question 2.
- Analyzing the instances we note that *Order Status* service is associated ODS OrderStatus ServiceGrounding which is of type WebService. This analyses answers the competence question 3.

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

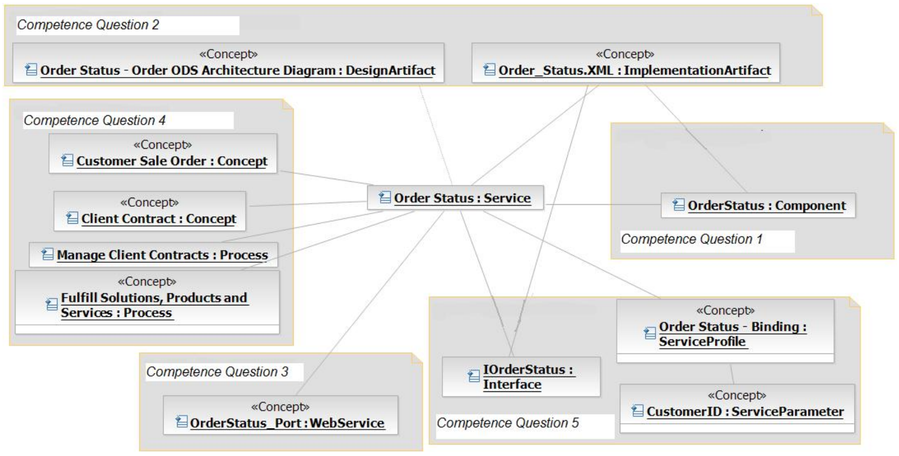
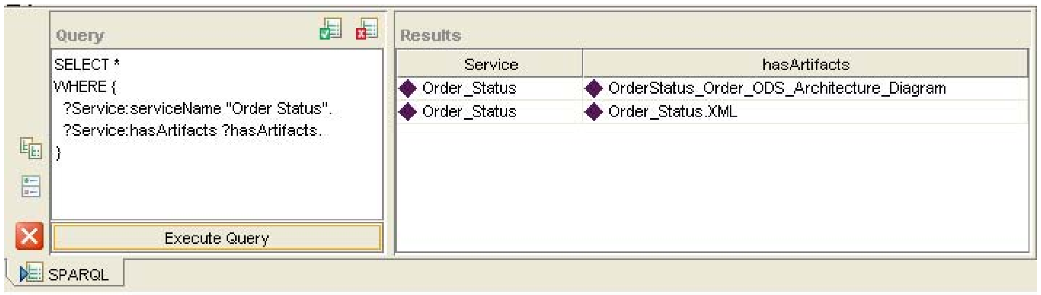

Competence Question 2
«Concept»
Order Status - Order ODS Architecture Diagram : DesignArtifact
«Concept»
Order_Status.XML : ImplementationArtifact
Competence Question 4
«Concept»
Customer Sale Order : Concept
«Concept»
Client Contract : Concept
Manage Client Contracts : Process
«Concept»
Fulfill Solutions, Products and Services : Process
Order Status : Service
OrderStatus : Component
Competence Question 1
«Concept»
Order Status - Binding : ServiceProfile
IOrderStatus : Interface
«Concept»
CustomerID : ServiceParameter
Competence Question 5
Competence Question 3
«Concept»
OrderStatus_Port : WebService

Figure 4. Instances of the Order Status service



```
SELECT *
WHERE {
  ?Service:serviceName "Order Status".
  ?Service:hasArtifacts ?hasArtifacts.
}
```

Service	hasArtifacts
Order_Status	OrderStatus_Order_ODS_Architecture_Diagram
Order_Status	Order_Status.XML

Figure 5. Query for competence question 5.

- *Order Status* service has several business concepts related to it. One of them is the business concept: Customer Sales Order. *Order Status* service also makes up the business process *Manage Client Contracts*. Therefore, those instances can answer the question of competence number 4.
- *Order Status* service has a relationship with project artifact called '' and with an implementation artifact called *Order Status.XML*. They represent, therefore, the assets related to the development Order Status service. Moreover, the *Component* concept also denotes a kind of artifact of implementation. Thus, the Order Status service also has the *OrderStatus* component as an artifact of implementation, answering the question of competence number 5.

Finally, we wrote some queries in SPARQL for each competence question. For instance, Figure 5 shows the query for competence question 5.

IV. CONCLUSION

This article presented the definition of an ontology to support the registering of services in SOA repositories to provide support for the search and recovery of the services in development time in any implementation technology.

Future studies involve: the creation of a service repository in accordance with an OWL-SOA ontology; the evolution of the OWL-SOA ontology for specific domains, integrating the service ontology with an ontology of a specific business domain; and the development of an ontology to contemplate an ontology of business process following the Business Process Modeling Notation (BPMN) standard [29].

REFERENCES

[1] J. Gilman, "Dynamic Service Discovery and Mediation in the Insurance Industry", Proceedings of the 2006 ACM Symposium on Applied Computing SAC '06, 2006.

[2] G. Leganza, M. Gilpin, J. Hopperman, L.M. Orlov, J. Stone, J., "Why Is SOA Hot in Government? Forresters Business Technographics" , 2006.

[3] G. A. Lewis, E. Morris, S. Simanta, L. Wrage, "Common Misconceptions about Service-Oriented Architecture". 6th International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07), 2007.

[4] X. Wang, T. Vitvar, V. Peristeras, A. Mocan, S.K. Goudos, K. Tarabanis, "WSMO-PA: Formal Specification of Public Administration Service Model on Semantic Web Service Ontology". Hawaii International Conference on System Sciences, HICSS-40, 2007.

[5] M. Paul, S. K. Gosh, "Service Oriented System Modeling and Application: An Approach for Service Oriented Discovery and Retrieval of Spatial Data", Proceedings of the international workshop on Service-oriented software engineering, 2006.

[6] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, 2007. "OWL-S: Semantic markup for web services (technical overview)", 2007. (http://www.daml.org/services/owl-s/1.1/overview/; accessed on 29/11/2007).

[7] D.Roman, H. Lausen, U. Keller, "D2v1.3. Web Services Modeling Ontology – WSMO" , 2006. (http://www.wsmo.org/TR/d2/v1.3/20061021/; accessed on 29/11/2007).

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.

[8] T.Erl, Service-Oriented Architecture: Concepts, Technology and Design. Prentice Hall, 2005.

[9] M.Korotkiy, J.Top, "Onto <--> SOA: From Ontology-enabled SOA to Service-enabled", Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, 2006.

[10] W. T., Tsai, "Service-Oriented System Engineering: A New Paradigm", IEEE International Workshop on Service-Oriented System, 2005.

[11] M.Amar, N. Muhammad, "Service Oriented Architecture (SOA) Registry. Report of Blekinge Institute of Technology", 2007. (https://idenet.bth.se/servlet/download/element/36064/SOA+Registry.pdf) (accessed on 08/09/2008).

[12] T.Erl, "SOA: Principles of Service Design. Prentice Hall" , 2008.

[13] OMG, RAS – Reusable Asset Specification version 2.2, 2005. (http://www.omg.org/cgi-bin/doc?formal/2005-11-02; accessed on 11/10/2008).

[14] A.Haller, E.Cimpiam, A. Mocan, E. Oren, C. Bussler, "WSMX - A Semantic Service-Oriented Architecture". International Conference on Web Services, 2005.

[15] T. R.Gruber, "A Translation Approach to Portable Ontology Specification", Knowledge Acquisition, 5, pp. 199-220, 1993.

[16] M.Grüninger, M.S. Fox, "Methodology for the Design and Evaluation of Ontologies", Technical Report, University of Toronto, 1995.

[17] R.Lara, D.Roman, "A. Polleres, D.Fensel, A Conceptual Comparison of WSMO and OWL-S", ECOW, 2004.

[18] V.Peristeras, K.Tarabanis, "Reengineering the public administration modus operandi through the use of reference domain models and Semantic Web Service Technologies", AAAI Spring Symposium, The Semantic Web meets eGovernment (SWEG), Stanford University, 2006.

[19] R.A.Falbo, "Experiences in Using a Method for Building Domain Ontologies". Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering, SEKE'2004, International Workshop on Ontology in Action, 2004.

[20] N.Noy, M.Musen, "Prompt: Algorithm and Tool for Automated Ontology Merging and Alignment. Stanford Medical Informatics", Stanford University, 2000.

[21] M. P. Papazoglou, "Service-Oriented Computing: Concepts, Characteristics and Directions", Proceedings of WISE, 2003.

[22] R.A Falbo, F.B. Ruy, R.D. Moro, "Using Ontologies to Add Semantics to a Software Engineering Environment", 17th International Conference on Software Engineering and Knowledge Engineering, pp. 151–156, 2005.

[23] Z.Duan, S. Bose, C. Shoniregun, P. Stirpe, A. Logvynovskiy, "SOA Without Web Services: A Pragmatic Implementation of SOA for Financial Transaction Systems",. IEEE International Conference of Services Computing, 2005.

[24] SUN Microsystems, Java Message Service Specification – version 1.1, 2002. (http://java.sun.com/products/jms/docs.html; accessed on 12/07/2008).

[25] OMG, 2000. CORBA Services Specification. Externalization Service Specification 1.0. (http://www.omg.org/technology/documents/corbaservices_spec_catalog.htm; accessed on 12/07/2008).

[26] SUN Microsystems, Enterprise Java Beans Specification 3.0, 2006. (http://java.sun.com/products/ejb/docs.html; accessed on 12/07/2008).

[27] D. Xu, X. Bai, G. Dai, A Tuple-Space-Based Coordination Architecture for Test Agents in the MAST Framework, Second International Symposium on Service-Oriented System Engineering, 2006.

[28] M.Dean, G. Schreiber, S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. Mcguinness, P. Patel-Schneider, L.Stein, "OWL Web Ontology Language Reference. W3C Recommendation", 2004. (http://www.w3.org/TR/owl-ref/) (accessed on 20/11/2008).

[29] OMG, BPMN – Business Process Modeling Notation Specification version 1.0. OMG Available Specification, 2006.

Authorized licensed use limited to: UNIVERSITY OF ESSEX. Downloaded on November 01,2024 at 15:54:06 UTC from IEEE Xplore. Restrictions apply.