Implementing Non-functional Service Descriptions in SOAs

Stephan Aier[1], Philipp Offermann[2], Marten Schönherr[2], and Christian Schröpfer[2]

[1] Institut of Information Management, University of St. Gallen,
Mueller-Friedberg-Strasse 8, 9000 St. Gallen, Switzerland
stephan.aier@unisg.ch
[2] Berlin University of Technology, Faculty of Computer Sciences and
Electrical Engineering, Franklinstr. 28/29, 10587 Berlin, Germany
{philipp.offermann,marten.schoenherr,
christian.schroepfer}@sysedv.tu-berlin.de

Abstract. This article describes a framework for extended service descriptions based on OWL-S (Web Ontology Language for Services) focusing on non-functional criteria. Necessary service management tasks will be introduced and extended by corresponding data elements and statements for its automated support. After a short comparative description of several existing approaches to semantic service descriptions the paper addresses the actual extension of OWL-S. Non-functional extensions as service lifecycle elements and Quality of Services (QoS) are added. To extend QoS capabilities, the approach combines the common extension mechanism with UML (Unified Modeling Language) Profile for QoS. A prototype delivers the proof-of-concept for the first part of the extension. The prototype implements SOA-specific authentications and all basic features for a tool-supported service management using extended semantic service descriptions by defining an ontology-based service taxonomy and service annotation.

Keywords: SOA, Protégé, OWL-S, QoS, UML Profile for QoS, service management, ontology, service lifecycle.

1 Service Management as a Key Issue in SOAs

Scientists and practitioners emphasize the potential of SOAs (service-oriented architectures) especially by reconciling business requirements and IT infrastructures. SOA definitions range from a solely technology-driven approach to a new management school approach on how to run the whole enterprise. Gold et al. consider technological aspects focusing on standardized interface descriptions.

> "[A service oriented architecture is] a set of components which can be invoked, and whose interface descriptions can be published and discovered." [1]

McCoy and Natis take into account aspects of stakeholder, granularity, reuse and agility:

D. Draheim and G. Weber (Eds.): TEAA 2006, LNCS 4473, pp. 40–53, 2007.

> "SOA is a software architecture that builds a topology of interfaces, interface implementations and interface calls. SOA is a relationship of services and service consumers, both software modules large enough to represent a complete business function. So, SOA is about reuse, encapsulation, interfaces, and ultimately, agility." [2]

Furthermore issues as service management and optimization are being addressed:

> "SOA is the concept of service-enabling new and existing software; linking internal and external service-enabled software systems; and implementing an enterprise-wide infrastructure to enable, manage, and optimize services use and interaction" [3] (see also [4, 5])

A common understanding of further SOA characteristics are the distributed manner of SOAs, the aspect of service orchestration, loose coupling of applications, and the standardization of interface descriptions [4, 6, 7]. Lubinsky and Tyomkin focus on the business process-driven integration and therefore derive the following three main aspects of an SOA [4]:

- Service descriptions
- Business processes
- Service management

A proper description of services is a fundamental precondition for a service management. While various research activities deal with aspects of functional service description, we will focus on non-functional elements of a service description in order to enable a *service lifecycle management (SLM)* and aspects of *quality of service (QoS)*.

Modeling functional and non-functional information in a machine-readable and semantically enriched way is a basis for a highly automated service management framework. In web services technology, UDDI (Universal Description, Discovery, and Integration) repositories and WSDL (Web Services Description Language) are used for service publication, discovery, and description but do not provide the necessary semantic functionality for service management aspects.

Our approach builds on OWL-S (Web Ontology Language for Services), a well established ontology framework, and existing tools to construct the necessary extensions. The two aspects that need to be covered in the non-functional area are service lifecycle information and offered QoS guarantees by a service. Hence, it is necessary to look at semantic web service description standards in general as well as description standards in the QoS domain.

2 Requirements for Non-functional Service Description

In order to support service management activities, like semi-automatic discovery, service level management, and service migration, several types of information need to be modeled within the service description. The following two sections describe requirements for service description regarding information relevant for service life cycle management and QoS guarantees. The lists contain the most obvious points in both categories. However, they can not be regarded as complete. A future-proof

approach must allow for extension of ontological terms used for description. Building on this extensibility, domain-specific models can be build that capture most requirements relevant in the domain.

2.1 Additional Description for Service Lifecycle Management

In the area of lifecycle management, the following information should be covered. The information can be categorized as organizational aspects and technical aspects.

Organizational aspects include information like *service name*, *service category*, *versioning information*, *variant information*, and *links to further business description* of the service. However, the most important organizational aspect for a service lifecycle management is the *lifecycle status* of a service. Possible states are "Planned", "Design", "Test", "Pilot", "Active–intensive maintenance", "Active–regular maintenance", "Sunsetting candidate", "Sunsetting in progress", and "Sunsetted". Especially for the operational management of active services information about the *service provider*, different *responsibilities*, *roles*, *persons* (e.g. for maintenance), and *pricing* (depending on QoS class) are of importance.

Technical aspects include information on the *infrastructure* the service runs on like server name, configuration management ID, etc. and a link to *source code* of the service. For managing *service dependencies* information about other services used as well as services depending on a certain service are necessary.

These statements are not very complex. It will be shown that they can be easily realized as OWL-S extensions.

2.2 QoS Guarantees

Quality of service aspects can be categorized in a *general dimension*, *cost dimension*, *performance dimension*, *reliability dimension*, and other *boundary conditions*.

The *general dimension* includes the overall *QoS-Level*. The service level regarding performance, quality ("Gold", "Silver", "Bronze") are defined in a separate SLA document. Furthermore services belong to a certain *service category* which may be derived in a service domain analysis [8] and a *communication pattern*, e.g. real time or batch.

The *cost dimension* specifies tariff models. Services may be paid, e.g. per period of time, per service call, or for volume of traffic.

The performance dimension includes primarily technical values. Specified values may be service response *time*, *data capacity* of an underlying database (normal/max after extension), *accuracy* of the result of a calculation, *arrival patterns* describing jitter and arrival distributions, and certain *performance ratios* like number of service requests per period of time (throughput of data sets, calculations per time, normal/max after extension) etc.

The *reliability dimension* includes aspects of *functional correctness* of services, their *availability* (business hours, weekdays/time, and incident resolution time), end-user usability, and aspects of *security* like security level, encryption standard, access rights, authenticity etc.

Other *boundary conditions* describe *organizational aspects* like promoters/opponents for certain activities, *cultural aspects* like different languages needed for end-user communication and *normative aspects* like compliance with laws/regulations and certification.

3 Relevant Standards

A number of standards have evolved in the area of semantic service description. A selection of them most relevant from a content and time perspective is being discussed in this section: OWL-S, WSMO (Web Service Modeling Ontology), and WSDL-S (Web Services Description Language – semantically enriched). For the QoS part we will discuss UML (Unified Modeling Language) Profile for QoS.

3.1 OWL-S

OWL-S (Web Ontology Language for Services) [9] is an upper ontology language developed by the semantic web services arm of the DAML (Darpa Agent Markup Language) program [10]. It uses the OWL (Web Ontology Language) ontology language. OWL-S supplies a core set of markup language constructs for describing the properties and capabilities of web services in unambiguous, computer-interpretable form and facilitates the automation of web service tasks including automated service discovery, execution, interoperation, composition, and execution monitoring [9].

OWL-S uses four classes to describe web services: *Service*, *ServiceProfile*, *ServiceGrounding*, and *ServiceModel*. *Service* is a reference point for the other elements. *ServiceProfile* facilitates service discovery and describes functional and non-functional aspects. It is the part where OWL-S can be extended. *ServiceProfile* is therefore described in detail in section 4. Figure 1 depicts all its elements. *ServiceModel* is targeted at in depth analysis once the service has been discovered. It describes in detail how to use the service, the semantics of requests, responses, pre- and post-conditions (effects), as well as optionally even the process. *ServiceGrounding* describes how the actual instance of a service can be accessed, i.e. protocol, message format, serialization, transport, and addressing.

We have chosen OWL-S as the basis for our service description approach for two reasons. First of all, it is based on OWL, a well established ontology language. Secondly, there are tools available for working with OWL ontologies as well as with OWL-S service descriptions.

3.2 WSMF/WSMO/WSML

WSMO is another upper ontology for describing web services semantically. It has been submitted to the W3C (The World Wide Web Consortium). WSMO represents a meta-model for web service description and is compatible to MOF (Meta-Object Facility). Basis for WSMO is WSMF (Web Service Modeling Framework) [11]. WSML (Web Services Modeling Language) which is used in WSMO provides a rule-based language for the semantic web [12]. As defined in WSMF, WSMO uses four main components: *ontologies*, *goals*, *mediators*, and *web services*.

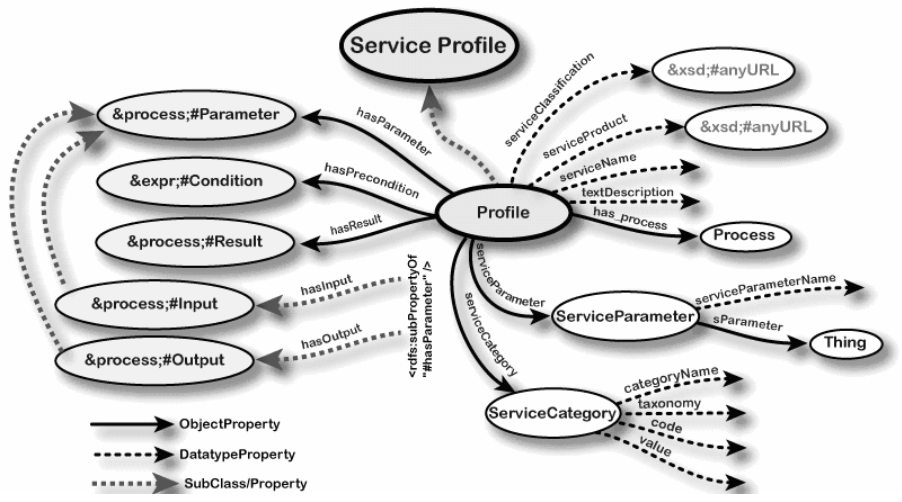

Fig. 1. Overview ServiceProfile [1]

Ontologies can be imported for the description of individual elements. In WSMO, they are used to define an agreed common terminology by providing concepts, and relations between the concepts [13].

Goals describe the functionality and interfaces of the web services from a user perspective. This is the section that can very well be used for discovery by potential service requestors.

Mediators describe the elements mediating between different *ontologies*, *goals*, and *web services*. They refer to external web services providing transformation services. The concept of *mediators* makes WSMO interesting for the description of heterogeneous web services.

Web services are among others described by *non-functional properties* and a *capability*, describing its functionality from a provider's perspective. *Non-functional properties* describe additional information about the web service, e.g. *owner*, *contributor*, *rights*, and *scalability*. They can be defined also for other elements and extended by using terms imported with ontologies.

Like OWL-S, WSMO only contains rather generic elements to describe web services. Some elements for service lifecycle management and QoS elements are included but seem rather ad hoc. That is why OWL-S has been chosen as the basis for the approach described here. However, we assume that a similar approach can be developed based on WSMO using *non-functional properties*.

3.3 WSDL-S

WSDL-S is a standard for a semantically enriched web services description. It is an extension of WSDL (Web Services Description Language). With this incremental approach it has been possible to add semantics to service descriptions without having to redefine the standard. [14]

In WSDL-S, the actual ontology representation can be done with an ontology language chosen by the user, e.g. OWL, WSML, or UML. In WSDL-S semantic

description is done in the following way. Three new elements are added as extensibility elements to WSDL: *category* (extension to *interface*), *preconditions*, and *effects* (extensions to *operation*). Service categorization can be done using *category*. The actual semantic annotation to the service and its elements input, output, operation, precondition, effect, and category is done by referring via URIs (Uniform Resource Identifier) to an externally defined ontology. Two extension attributes specify the association and schema mapping between WSDL elements and ontologies.

Summing up, WSDL-S is a standard for semantic description of web services which heavily leverages existing standards and is very flexible with respect to ontology and mapping languages.

3.4 UML Profile for QoS

UML Profile for QoS is a comprehensive framework for modeling QoS requirements and offerings. It extends the reference UML 2.0 meta-model mainly by using the stereotype concept. It allows for the modeling of QoS properties in UML models [15].

UML Profile for QoS uses the following approach. It describes a QoS model specific to the respective domain separated from the actual elements to be annotated. In the actual UML model the elements can be annotated using terms defined in this QoS model. UML profile is based on a QoS meta-model and comprises the three subprofiles *QoS Characteristics*, *QoS Constraints*, and *QoS Levels*.

The QoS model is defined by using Q*oS Characteristics*. Among others, it uses the stereotypes *QoS Characteristic* and *QoS Dimension* to specify respectively quantify aspects of QoS. It is possible to use statistical values (maximum, minimum, range, mean, variance, standard deviation, percentile, frequency, moment, and distribution) to express preferences about the direction when comparing or optimizing parameters. The relationship between several *QoS Characteristics* and elements that are part of one QoS statement can be described by *QoS Context*.

Annotating the elements with QoS requirements/offerings is done with *QoS Values* or with *QoS Constraints*. *QoS Values* specify values for QoS dimensions available at modeling time. *QoS Constraints* describe limitations of *QoS Characteristics* for annotated elements, either by listing the allowed elements or by stating the limits. Three types of constraints exist: *QoS Required*, *QoS Offered*, and *QoS Contract*. For service description in most cases *QoS Offered* will be used. However, using *QoS Required* it is possible to specify constraints the provider has towards the requestor, e.g. invocation/arrival patterns. *QoS Contract* can be used for specifying service level agreements. Different levels supported by a system with regard to QoS can be defined by *QoS Level*. A *QoS Level* is described by an allowed space for the values of the QoS characteristics. The different levels can be used in SLAs.

There are several reasons for choosing UML Profile for QoS for the extension of OWL-S. Firstly, it comes with its own general catalog of QoS characteristics which is neither domain- nor project-specific. Although it is not complete, it is an excellent basis for a common understanding of the most important QoS parameters. Secondly, it can be well integrated with business process modeling, which is the counterpart of the web services matching problem to the service description. For service matching and service level negotiation this offers the advantage of having both the description of offered QoS and the description of required QoS in the same logical format. Thirdly,

compared to other specifications, UML Profile for QoS is quite mature. A lot of other QoS-related work and frameworks where considered already during its first definition. Summarizing, UML Profile for QoS is a comprehensive framework for modeling QoS requirements and offerings and is therefore well suited to add comprehensive QoS capabilities to OWL-S.

4 Extending OWL-S for Non-functional Service Description

The following section describes the proposed extension to OWL-S with respect to service lifecycle management and QoS.

4.1 Extensions for Service Lifecycle Management

Extension of OWL-S happens in the *ServiceProfile*. For the functional description *Parameter*, *Input*, *Output*, *Condition*, *Result*, and *Process* are used. The first five refer to the process description in *ServiceModel*. For the non-functional description the following properties/classes are interesting: *serviceClassification*, *serviceProduct*,

```
<owl:Class rdf:ID="ServiceVersion">
  <rdfs:subClassOf rdf:resource=
  "http://www.daml.org/services/owl-s/1.2/Profile.owl#ServiceParameter"/>
</owl:Class>
<owl:Class rdf:ID="ServiceVersionInfo"/>
<owl:DatatypeProperty rdf:ID="VersionName">
  <rdfs:domain rdf:resource="#ServiceVersionInfo"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="VersionNumber">
  <rdfs:domain rdf:resource="#ServiceVersionInfo"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
```

Fig. 2. Definition of *Service Version* in OWL-S

```
<ServiceVersion rdf:ID="ServiceVersion_10">
  <profile:sParameter>
    <ServiceVersionInfo rdf:ID="ServiceVersionInfo_11">
      <VersionName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Snake</VersionName>
      <VersionNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >5.1</VersionNumber>
    </ServiceVersionInfo>
  </profile:sParameter>
  <profile:serviceParameterName rdf:datatype=
  "http://www.w3.org/2001/XMLSchema#string"
  >ServiceVersion</profile:serviceParameterName>
</ServiceVersion>
<profile:Profile rdf:ID="CalculateRoute_Profile">
  <profile:serviceParameter rdf:resource="#ServiceVersion_10"/>
[…]
</profile:Profile>
```

Fig. 3. Instance of a service description for *CalculateRoute* with details for *ServiceVersion*

Table 1. Additional elements defined for service lifecycle management

Additional service lifecycle parameter	**Explanation**	
Properties/ subclasses	**Data type**	**Explanation**
ServiceVersion	Versioning information	
VersionName	String	Version name described as literal
VersionNumber	Float	Version number x.x
ServiceVariant	Variant information	
Variant	Integer	Variant number
ServiceLifecycle-Status	Lifecycle status of service component	
LifecycyleStatus (subclass of owl:Thing)	(Enumerated instances)	Enumerated instances: "Planned", "Design", "Test", "Pilot", "Active_intensive_maintenance", "Active_regular_maintenance", "Sunsetting_candidate", "Sunsetting_in_progress", "Sunsetted"
ServiceProvider	Service provider information	
ProviderLink	anyURI	Link to external information (name, address, contacts, credentials, etc.) in provider database
ServiceInfrastructure	Infrastructure the service runs on	
ServerID	anyURI	List of Server IDs the service runs on
ResourceID	anyURI	List of Resource IDs the service uses
SourceCodeLink	Link to source code in code repository	
SourceCode	anyURI	Link to source code
ServiceResponsibility	Responsibility for service from business and technical perspective	
BusResponsibility	anyURI	Link to organization/person with business responsibility
TechResponsibility	anyURI	Link to organization/person with technical responsibility
BusinessDescription	Information about business background	
BusDescription	String	Textual description of business background
BusInfLink	anyURI	Link to further information resources
ServicePricing	Pricing information	
PricingModelQ1	anyURI	Link to pricing model for QoS level 1
…	…	…
PricingModelQn	anyURI	Link to pricing model for QoS level n

serviceName, *textDescription*, *ServiceCategory,* and *ServiceParameter*. The first five can be used for the requirements mentioned as they are. The web service can be classified using *serviceClassification* (mapping to an OWL ontology of services, e.g. NAICS) and *serviceProduct* (mapping to an OWL ontology of products, e.g. UNSPSC), as well as *ServiceCategory* (mapping to taxonomies potentially outside of

OWL or OWL-S). Using *serviceName*, a semantic name can be given to a service. Free text descriptions can be represented with *textDescription*.

The element *ServiceParameter* is especially important for the extension. Here the remaining additional service lifecycle characteristics are defined (Table 1). Future extensions also can be realized using *ServiceParameter*.

ServiceParameter consists of the *serviceParameterName*, the actual name of the parameter, defined as literal or URI, and *sParameter*, a link to the value within an OWL ontology. Figure 2 shows the definition of *ServiceVersion* in OWL-S as an example. *VersionName* (type *xsd:string*) and *VersionNumber* (type *xsd:float*) are defined as datatype properties of the class *ServiceVersionInfo* (subclass of *owl:Thing*). Figure 3 shows the *ServiceVersion* information in OWL-S in a service description for a logistics web service *CalculateRoute. ServiceVersion_10* and *ServiceVersionInfo_11* are instances that contain the actual version information "snake" and "5.1".

4.2 Extensions for QoS

Section 2.2 gives a flavor of what the level of complexity is when describing QoS offerings. It shows that a comprehensive and extensible QoS framework that builds on extensive experience needs to be leveraged. UML Profile for QoS is such a framework that suffices these requirements.

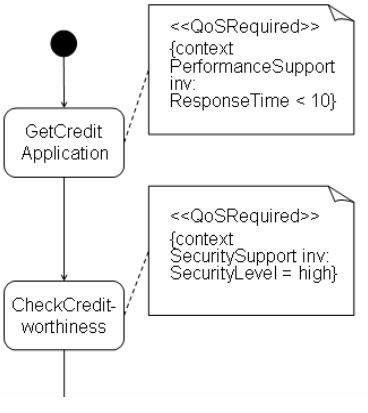

Fig. 4. Example QoS requirements in an UML activity diagram

Hence we propose to use UML Profile for QoS together with OWL-S to bring QoS functionality to web services description. To achieve this, it is not necessary to develop the whole QoS model in OWL-S. It is sufficient to introduce the QoS annotations to the services to be described. The QoS model does not have to be defined in OWL-S. This description remains in UML and can be reused for other service descriptions. This is very much in line with the idea of using the same QoS notation on the business process side as well as on the service description side to facilitate service level negotiation. Introducing the QoS annotations into the OWL-S service descriptions can again be easily done by adding *QoSCharacteristics* as a new

```
<profile:Profile
  rdf:ID="GetCreditApplication_Profile">
  <profile:serviceParameter>
    <QoSCharacteristics rdf:ID="QoSCharacteristics_14">
      <profile:sParameter>
        <QoSStatement rdf:ID="QoSStatement_15">
          <Statement rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
           &lt;&lt;QoSOffered>> {context PerformanceSupport inv: ResponseTime
           &lt; 8}</Statement>
        </QoSStatement>
      </profile:sParameter>
      <profile:serviceParameterName rdf:datatype=
       http://www.w3.org/2001/XMLSchema#string>
       QoSCharacteristics </profile:serviceParameterName>
    </QoSCharacteristics>
  </profile:serviceParameter>
  […]
</profile:Profile>
```

Fig. 5. QoS offering for *GetCreditApplication* in the service description

ServiceParameter. *QoSStatement* is defined as a subclass of *owl:Thing* with the data type property *statement* of the type string. This field contains the QoS constraints in OCL (Object Constraint Language) of the element to be annotated. Figure 4 shows example QoS requirements on the demand side in a UML activity diagram. *ResponseTime* of *GetCreditApplication* is required to be lower than 10 ms. Figure 5 shows the corresponding QoS offering in the service description of *GetCreditApplication*.

5 Prototyping a Service Management

In order to support the mentioned features we have prototyped a service management framework. It consists of a system architecture mainly integrating open source features and a methodology describing how to use the prototype to introduce a service management in an SOA.

5.1 Architecture

The architecture consists of a modeling approach using Protégé (extended by an OWL plug-in) to edit an OWL-S ontology. The Jena framework is being used to integrate and search the models. Furthermore there is a relational database (MySQL) representing the ontology attributes and integrating the UDDI taxonomy. The logic and GUI implements a service management-specific authentication approach differentiating roles as “Service Architect”, “Service Programmer”, and “Business Process Owner”. A role-specific GUI shows relevant issues depending on the roles’ tasks and interests only. For example a programmer needs to have detailed information about finding and binding of services which a business process owner will never need. Figure 6 gives an overview of the prototypical architecture and its three-layered structure.

Protégé is a free, open source ontology editor from Stanford University [16]. Protégé with Protégé-OWL, a plug-in for defining ontologies in OWL also from Stanford University (available at [17]), is used for taxonomy definition. OWL-S Editor is a Protégé plug-in developed at SRI International (available at [18]). It helps to define

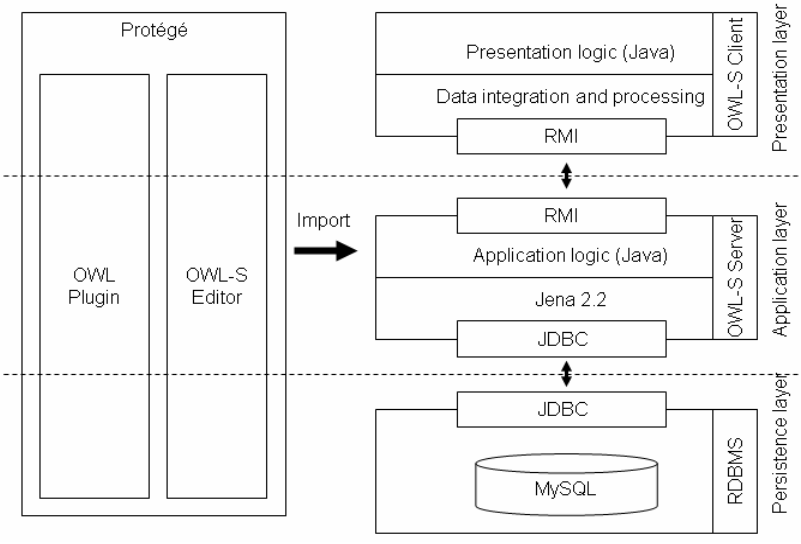

Fig. 6. Overview of service management prototype

services in OWL-S by making available the OWL-S ontology with its predefined elements and a special view on the service, profile, grounding, and process instances. The prototype itself is written in Java. It uses RMI (Remote Method Invocation) for communication between the Java components and builds on Jena, a semantic web service framework, for the semantic support. Jena facilitates the usage of internal and external reasoners and access to the database via RDQL (Resource Description Framework Query Language) [19]. The prototype uses it for interfacing with the database where the semantic description is stored and for performing several operations on the ontology database, in this case MySQL.

5.2 Methodology

The first version of the prototype supports basic tasks for a service management. The following tasks are being implemented:

Taxonomy/Ontology Definition. The mentioned additions to the OWL-S ontology can be made with the OWL Editor adding new *ServiceParameter* and *owl:Thing* subclasses. Later service descriptions and ontology extensions can be done using the OWL file. Also, a taxonomy for the service category field and input/output parameters can be developed with Protégé OWL.

Service Description and Annotation. Service annotation and changes to existing annotations are done with the OWL-S Editor by loading the OWL file that contains the ontology extended by the above mentioned elements. It is possible to import existing WSDL descriptions.

Once the extended OWL-S ontology is loaded the services can be described. For specifying a parameter for a service the predefined *ServiceParameter* has to be used.

There are two ways of doing this. If the parameter contains listed elements, e.g. *ServiceLifecycleStatus*, a link to an existing instance can be used. If the parameter contains an element with free content like a number or a text field (e.g. *ServiceVersion*), a new parameter value instance has to be created.

For the service description Protégé and the web-based GUI can be used. Protégé does not support the service management-specific authentication system. Therefore the use of the prototyped GUI should be preferred.

Apart from the non-functional elements, it is possible to semantically describe the input/output parameters using normal OWL-S functionality and the service parameter ontology defined.

Service Registration. Service registration is done by importing the OWL-S service description into the prototype and its database. This is necessary in the case of changing attributes. The prototype then performs the search activities laid out in the next part**.**

Service Discovery and Review. The main functionality of the prototype is search functionality across the services registered and described. There are several possibilities for performing searches using the additional semantic information:

1. Simple queries – searching for services, input/output parameters, taxonomy expressions, etc. using the full names of these elements
2. Semantic queries for services using their input and output parameters
3. Semantic queries for services that match other services' input or output parameters
4. Semantic queries for services using taxonomy elements
5. Semantic queries using the other additionally defined parameters such as *ServiceVariant*, *ServiceResponsibility*, and *ServiceLifecycleStatus*

Queries can be combined by limiting the search space by an outcome of a previous search operation. For service management in complex environments it is absolutely necessary to support role-specific views combined with access rights management. The architect for example does not necessarily need to know all the details about the pricing scheme. Business owners are not interested in technical details about invocation. Hiding unnecessary information improves usability, reduces the number of errors, and is sometimes a must when it comes to confidentiality. The prototype is currently being extended by this functionality.

The main feature is the generic way of defining/redefining service taxonomy and the permanent annotation of existing and new services. It is a matter of fact that there is no stable service description in complex environments. Therefore the change of taxonomy in a distributed SOA is a must when it comes to a huge number of implemented services, different service lifecycle stages, and an existing role-based service governance approach.

6 Conclusion

Sustainability as the most important characteristic of an integrated architecture needs to be considered in an SOA, too. Changes in the design- and runtime of an SOA will

be represented by changes in service taxonomy – changes regarding service management requirements by changes in the specific OWL-S ontology. The effort of handling changes and the methodology of staying up-to-date in the annotation using the actual taxonomy needs more than known web service standards offer. The contribution of the work related to this article is a hands-on approach to service descriptions that is extensible regarding additional future requirements. The article shows that it is possible to build a semantically enriched service repository with OWL-S that supports several tasks that are the basis for higher level service management activities. Therefore it is evolutionary and a compatible upgrade of the existing web service description standards.

The current prototype will be extended regarding several issues. On the conceptual level, the integration with a UDDI registry has to be improved. With this respect, the role of WSDL-S for integration has to be examined. In addition, further work is necessary to check whether similar extensions can be made with WSMO.

The presented approach is extendable. A valuable field for further research on the business level is therefore a more structured examination of the content and the statements related to service lifecycle management and QoS. For real world applicability, it is important to have ready-to-use de facto standardized QoS models and extensions to OWL-S.

References

1. Gold, N., Knight, C., Mohan, A., Munro, M.: Understanding Service-Oriented Software, pp. 71–77. IEEE Computer Society Press, Los Alamitos (2004)
2. McCoy, D., Natis, Y.: Service-Oriented Architecture: Mainstream Straight Ahead. Gartner Research (2003)
3. New Rowley Group: Building a more flexible and efficient IT infrastructure - Moving from a conceptual SOA to a service-based infrastructure (2003), http://www.newrowley.com/reseach.html
4. Lubblinsky, B., Tyomkin, D.: Dissecting Service-Oriented Architectures. Business Integration Journal, 52–58 (2003)
5. Roth, P.: Moving to A Service Based Architecture. Business Integration Journal, 48–50 (2003)
6. Sleeper, B., Robins, B.: The Laws of Evolution: A Pragmatic Analysis of the Emerging Web Services Market. The Stencil Group, San Francisco (2002)
7. Weinreich, R., Sametinger, J.: Component Models and Component Services: Concepts and Principles. In: Council, W.T., Heinemann, G.T. (eds.) Component-Based Software Engineering: Putting Pieces Together, pp. 22–64. Addison Wesley, Boston (2001)
8. Aier, S.: How Clustering Enterprise Architectures helps to Design Service Oriented Architectures. In: IEEE SCC2006, IEEE, Chicago, USA (2006)
9. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic markup for Web services (2006), http://www.ai.sri.com/daml/services/owl-s/ 2/ overview/
10. DAML: DAML Services (2006), http://www.daml.org/services/owl-s/
11. Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. In: Electronic Commerce: Research and Applications, pp. 113–137 (2002)

12. Feier, C., Domingue, J.: D3.1v0.1 WSMO primer. DERI (2005), http://www.wsmo.org/TR/d3de.1/v0.1/
13. de Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., Stollberg, M.: Web Service Modeling Ontology (WSMO) - W3C Member submission 3 June 2005 (2005), http://www.w3.org/Submission/WSMO/
14. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., Verma, K.: Web service semantics - WSDL-S - W3C member submission 7 November 2005 - Version 1.0 (2005), http://www.w3.org/Submission/2005/SUBM-WSDL-S-20051107/
15. OMG: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms - OMG available specification - Version 1.0 - formal/06-05-02. OMG (2006), http://www.omg.org/cgi-bin/apps/doc?formal/06-05-02.pdf
16. Welcome to Protégé. Stanford Medical Informatics (2006), http://protege.stanford.edu/
17. What is Protégé-OWL? Stanford Medical Informatics (2006), http://protege.stanford.edu/overview/protege-owl.html
18. The OWL-S Editor (2004), http://owlseditor.semwebcentral.org/
19. Jena - A Semantic Web Framework for Java. sourceforge.net, http://jena.sourceforge.net/