A Flexible Approach to Service Management-Related Service Description in SOAs

Christian Schröpfer[1], Marten Schönherr[1], Philipp Offermann[1], and Maximilian Ahrens[2]

[1] Faculty of Electrical Engineering and Computer Sciences, Technische Universität Berlin, Germany
{Christian.Schroepfer, MSchoenherr, Philipp.Offermann}@sysedv.tu-berlin.de

[2] Deutsche Telekom Laboratories, Berlin, Germany
Maximilian.Ahrens@telekom.de

Abstract. In order for service-oriented architectures (SOAs) to deliver their true value for the business, e.g. flexibility and transparency, a holistic service management needs to be set up in the enterprise. To perform all the service management tasks efficiently heavy support by automated processes and tools is necessary. This article describes a service description approach that is based on OWL-S (Web Ontology Language for Services) and focuses on non-functional criteria. It starts with the necessary service management tasks and explains non-functional data elements and statements for its automated support. After covering related work it explains the proposed flexible extension to OWL-S. This extension is twofold. Firstly, simple service lifecycle elements are added using the normal extension mechanism. Secondly for adding QoS (Quality of Service) capabilities, the approach combines this extension mechanism with UML (Unified Modeling Language) Profile for QoS. A prototype delivers the proof-of-concept.

1 Introduction

In the last years, a lot of work regarding practical usability of technologies in the SOA (service-oriented architecture) area and especially Web services area has been done. Research work is more and more shifting from the technical areas like reliability and security to the business layer. One of the problems is the operational management – or IT service management – of an actual implemented service-oriented IT landscape in the enterprise. ITIL (IT Infrastructure Library, see `http://www.itil.co.uk/`) is a general and widespread IT service management framework. Being a de-facto standard, many other service management frameworks are based on it [1]. Among others, it covers best practices along two areas, *Service Support* and *Service Delivery* including configuration management, incident management, problem management, change management, release management, service level management, capacity management, availability management, IT continuity management, and financial management [1]. Part of the "IT service management" within the SOA is the "service

component management" which deals specifically with managing the service components, e.g. Web services, during their lifecycle.

Due to the special characteristics of an SOA, its operational management is different from managing mature architectures. Additional requirements need to be covered. In an SOA, the implemented Web services are most likely much more fine granular than "normal" applications. In one landscape, there exist services that offer similar functionality and have different lifecycle stages. A high number of services need to be managed while a high reuse rate is a primary goal. At the same time in order for SOA to deliver its advantages, changing services and their orchestration should be easily possible. When managed without automated processes, tool support, and centralized repositories, these conditions can lead to confusion and chaos. The contrary of the original goals of SOA, among others more flexibility and more efficient IT, would be the outcome. Hence, an effective and efficient service management framework for SOAs is needed that is supported by automated processes and tools. The following SOA-specific functional blocks should be covered: service definition, service deployment lifecycle, service versioning, service migration, service registries, service message model, service monitoring, service ownership, service testing, and service security [2]. They represent SOA-specific functionality in the broader area of the ITIL processes. Reference [3] highlights the importance of service description and in particular non-functional service description for managing SOAs and mentions in addition service discovery, substitution, and composition. Modeling functional and non-functional information in a machine-readable and semantically enriched way is a basis for a highly automated management of SOAs and in a broader sense of IT service management.

This article looks at a flexible service description approach to non-functional information. In Web services technology, UDDI repositories (Universal Description, Discovery, and Integration) and WSDL (Web Services Description Language) are used for service publication, discovery, and description but do not provide the necessary semantic functionality. Compared to the functional area less work has been done in the area of semantically enriched non-functional service description. Hence, this paper especially deals with the latter part. The approach builds on OWL-S (Web Ontology Language for Services). The two aspects that form the basis in the non-functional area are service lifecycle information and QoS (Quality of Service) guarantees offered 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.

The remainder of this paper is organized as follows: In section 2, the requirements for describing services are examined. Section 3 gives an overview over the related work. Section 4 describes the extensions to OWL-S and section 5 the prototype. Section 6 describes the importance of this approach for matching, SLA negotiation, and SLA enforcement.

2 Requirements for Service Description

2.1 Requirements Overview

In order to support the above mentioned 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 lifecycle management and QoS guarantees. Two aspects have to be considered, the content and the type of statements that can be modeled. The lists contain the most obvious points in both aspects. However, they can not be regarded as complete. The available sources, e.g. [3], [4], [5], and [6], describe very different non-functional characteristics. In order to be future-proof, the approach must allow for extension of both ontological terms and structure of statements used for description. Building on this extensibility, domain specific models can be built that capture most requirements relevant for the domain.

2.2 Information Relevant for Service Lifecycle Management

In the area of service lifecycle management, the following most obvious information should be covered as a starting point:

1. Service name
2. Service categories
3. Versioning information
4. Lifecycle status ("Planned", "Design", "Test", "Pilot", "Active – intensive maintenance", "Active – regular maintenance", "Sunsetting candidate", "Sunsetting in progress", "Sunsetted") (based on [2], extended)
5. Service provider information
6. Infrastructure the service runs on: server name, configuration management ID, etc.
7. Link to source code
8. Different responsibilities, roles, persons, e.g. for business aspect or maintenance
9. Link to further business description of the service
10. Pricing information (depending on QoS class)

For lifecycle management, the following obvious statement structures should be covered as a starting point:

1. Parameters with simple values, e.g. versioning information
2. Parameter names with RDF (Resource Description Framework) pointers to terms from predefined ontologies or resources (configuration database IDs for related infrastructure). Technically, this includes 1.
3. Tabular expressions, e.g. listing responsibilities for several areas
4. Free textual statements for a human reader

These statements are not very complex. As shown later, they can be realized relatively simply with OWL-S extensions. Free textual statements are introduced (also for the QoS) because we assume that in the first step it is not reasonable to put semantics behind every statement for automatic processing. Rare statements should be left for a human being to work with.

2.3 QoS Guarantees

Table 1 exemplarily describes QoS characteristics to be modeled in the service description.

Table 1. QoS information

QoS area	Explanation/example
General area	
QoS-level	Service level regarding performance and quality ("Gold", "Silver", and "Bronze" are defined in a separate SLA document)
Service category	Type of service/service domain (several categories per service possible)
Communication	Communication pattern (e.g. real time and batch)
Cost area	
Price	Specification of tariff models, e.g. per period of time, per service call, and volume-fixed
Performance area	
Time	Response time
Capacity	Data capacity of a database (normal/max after extension)
Accuracy	Accuracy of the result of a calculation
Arrival pattern	Jitter; arrival distribution
Ratios	Number of service requests per time period (throughput of data sets, calculations per time) – (normal/max after extension)
Quality area	
Functional correctness	Error rate
Reliability	Availability, business hours (weekdays/times), incident resolution time
End user usability	Rating with respect to ease of use/understanding
Security	Security level (high, medium, low – defined in separate document: encryption standard, access rights, and authenticity)
Other boundary conditions	
Organizational	Negative/positive list of partners
Cultural	Languages needed for end user communication
Normative	Compliance with laws/regulations, certification

The following structures of QoS statements should be supported as a basis to facilitate rich QoS specification in service description:

1. *Boolean statements*, e.g. "Component is Basel II certified – yes/no."
2. *Absolute requirements*, e.g. "Reliability should be at least 99.9%."
3. *Composed requirements*, e.g. "On weekdays, between 7am and 8pm, availability should be 99.9%; Otherwise, reliability should be 99%."
4. *Level statements*, e.g. "The QoS requirements as defined in level "Gold" should be complied with."
5. *Percentile statements*, e.g. "In 95% of the cases, response time should be below 10 ms."
6. *Free textual statements* for a human reader

In addition, it should be possible to specify several sets of QoS guarantees (QoS-level) with added price tags for one Web service that can be referred to during SLA (Service Level Agreement) negotiations.

3 Related Work

3.1 Standards for Service Description

A number of standards have evolved in the area of semantic service description. A quite mature one by now is OWL-S. OWL-S is an upper ontology language developed by the Semantic Web Services arm of the DAML (Darpa Agent Markup Language) program [7] [8]. Using OWL-S, it is possible to describe Web services, their properties, and capabilities in a semantically enriched form. Given this, 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 robust tools available for working with OWL ontologies as well as with OWL-S service descriptions. Both reasons support the intention of this article to show that, based on today's technology, standards, and tools, a reasonable basis for service management can be realized.

Other relevant semantic Web service description standards are WSMO (Web Services Modeling Ontology) and WSDL-S (WSDL with semantic extension). WSMO is a part of the WSMF (Web Services Modeling Framework) [9]. Its distinctiveness lies in its capability to import ontologies specified in other ontology languages, among others OWL, its usage of *mediators* bridging the gap between different Web services, as well as its *goal* concept describing functionality and interfaces from a user perspective.

WSDL-S heavily leverages the existing standard WSDL and is focused on compatibility [10]. It also is very flexible with respect to ontology languages (e.g. OWL) and mapping languages. However, being so flexible it is also more generic than WSMO and OWL-S.

3.2 QoS Specific Standards – UML Profile for QoS

Specification of QoS characteristics is an important topic in the area of IT systems. The existing standards can be grouped according to their main focus: software design/process description (e.g. UML Profile for QoS and QML – QoS Modeling Language [6]), service/component description (e.g. WS-Policy), and SLA-centric approaches (e.g. WSLA – Web Service Level Agreements [11] [12], WSOL – Web Service Offerings Language [13], SLAng – Service Level Agreement definition language [14], and WS-Agreement [15]). A good overview over most of them can be found in [4].

Several languages have been developed to support SLA negotiation and specification in a service provider/service requestor scenario. The SLA-centric approaches are very much linked to the problem of QoS characteristics specification. The difference

to other QoS specification languages is that they are more targeted towards SLA negotiation, specification, and SLA management.

UML Profile for QoS is a comprehensive framework for modeling QoS requirements and offerings in UML models. It extends the reference UML 2.0 meta-model mainly by using stereotypes. The current specification was published by OMG (Object Management Group) in May 2006 [16]. Originally it has been developed for software engineering of object-oriented systems. This article shows that it is also applicable to service description. UML Profile for QoS uses the following approach for QoS description. It describes a QoS model specific to the respective domain separately from the actual elements to be annotated. Then in the actual UML model the elements can be annotated using terms defined in the QoS model.

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 not domain- or project-specific. Secondly, it can be well integrated with business process modeling which is part of the Web services matching problem. Thirdly, compared to other specifications, UML Profile for QoS is quite mature and has been accepted by OMG as a standard. Its definition goes back to a thesis by J. Aagedal published in 2001 where a lot of other QoS-related work has been considered [17].

3.3 Approaches to Semantic Service Description, Discovery, and Selection

Roy Grønmo and Michael C. Jaeger propose a methodology for Web service composition using QoS optimization [18]. The main focus of their article is on a matchmaking algorithm that uses QoS requirements and offerings for achieving better results. For both, they use UML Profile for QoS. Other than in this article, they use a link from the WSDL operations to a document describing the QoS offerings.

Reference [4] proposes to have functional as well as non-functional specifications in separate repositories. By contrast, we recommend to use a single repository, since we do not see the necessity that a separate organization specifies the QoS characteristics. In fact, the functional and non-functional properties should be guaranteed together either by the organization itself or a third party. The third party could then be a trusted entity that is responsible for monitoring service levels or even for delivering the service levels itself.

Reference [5] describes a framework and ontology for dynamic Web services selection. It uses an agent-based system to support dynamic service selection and QoS ontologies for describing the non-functional characteristics. Although the approach covers QoS very extensively and comes with a realistic example, it has shortcomings. It uses its own service ontology which makes it proprietary. Also, semantic description of service lifecycle information and functional service description is not explicitly covered by the approach.

WS-QoS is a framework that allows the definition of QoS requirements as well as offerings for Web services and provides an infrastructure for managing those QoS-aware Web services. WS-QoS is based on a WS-QoS XML schema and can be extended. Although it is compatible with UDDI and WSDL by using their extension mechanisms, it is a proprietary approach when it comes to the QoS specification [19].

In [20], Klein and König-Ries present a process and a tool for describing services based on DAML-S. A layered set of ontologies is used and instantiated in a specific service description with the tool. The service description does not specifically deal with service management requirements. In [21], Klein, König-Ries, and Müssig develop an alternative service description language, called DIANE Service Description (DSD) that implements additional requirements that are not covered by OWL-S and WSMO. However, in this article we want to rely on current standards and existing tools as much as possible.

Matching, i.e. service searching, ranking, and selection, is an interesting application of semantically enriched service description. A lot of work is going on in this area. Apart from functional information also the non-functional information is important to be considered as the already mentioned sources [18] and [5] show. However, functional matching is usually the first step to find appropriate services. The recently published OWLS-MX matcher uses a hybrid approach, combining logic-based reasoning and approximate semantic matching, in particular content-based information retrieval techniques for the input and output parameters specified in the service profile of OWL-S [22].

4 Extension of OWL-S

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

4.1 Extension for Service Lifecycle Management

Extension of OWL-S happens in the *ServiceProfile*, one of the four classes OWL-S uses. It is targeted at describing functional and non-functional aspects for service discovery. 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*, *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 – North American Industrial Classification System), *serviceProduct* (mapping to an OWL ontology of products, e.g. UNSPSC – United Nations Standard Product and Services Classification), as well as *ServiceCategory* (mapping to taxonomies potentially outside of OWL or OWL-S). A semantic name can be given to a service using *serviceName*. Free text descriptions can be represented with *textDescription*.

Especially important for the extension is *ServiceParameter*. With this element the remaining additional service lifecycle characteristics are defined (Table 2). Future extensions also can be realized using *ServiceParameter*.

Table 2. Defined elements for service lifecycle management

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
ServiceLifecycle-Status	Lifecycle status of the service component	
LifecycleStatus (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
Service Infrastructure	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
Service Responsibility	Responsibility for service from business and technical perspective	
BizResponsibility	anyURI	Link to organization/person with business responsibility
TechResponsibility	anyURI	Link to organization/person with technical responsibility
BusinessDescription	Information about business background	
BizDescription	String	Textual description of business background
BizInfLink	anyURI	Link to further information resources
ServicePricing	Pricing information	
PricingModelQ1	anyURI	Link to pricing model for QoS level 1, e.g. “Gold”
…	…	…
PricingModelQ5	anyURI	Link to pricing model for QoS level 5

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 1 shows the definition of *ServiceVersion* in OWL-S as an example. *VersionName* and *VersionNumber* are defined as datatype properties (type *xsd:string* and *xsd:float*) of the class *ServiceVersionInfo* (subclass of *owl:Thing*). Figure 2 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”.

```
<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:Class>
<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. 1. Definition of *ServiceVersion* 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. 2. Instance of a service description for *CalculateRoute* with details for *ServiceVersion*

4.2 Extension for QoS with UML Profile for QoS Description

Section 2.3 gives a flavor of what the level of complexity needed 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 the requirements. Hence we propose to use UML Profile for QoS together with OWL-S to bring QoS functionality to Web services description.

The QoS model does not have to be defined in OWL-S. Its definition remains in UML and can be reused for other services and systems. 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. The stereotypes *QoS Characteristic* and *QoS Dimension* are used in the QoS model to specify respectively quantify aspects of QoS. It is possible to use statistical values (maximum value, minimum value, range, mean, variance, standard deviation, percentile, frequency, moment, and distribution) as well as to express preferences about the direction when comparing or optimizing parameters (increasing or decreasing).

For annotating the elements with QoS requirements and offerings UML Profile for QoS uses three types of constraints: *QoS Required*, *QoS Offered*, and *QoS Contract*. *QoS Required* and *QoS Offered* describe required and offered limitations of *QoS Dimensions* for annotated elements, either by listing the allowed elements or by stating the limits. *QoS Contract* can be used for agreed limitations. Different QoS levels supported by a system, which can be used in SLAs, can be defined with *QoS Level*.

OCL (Object Constraint Language) expressions are used in the QoS statements. This enables rich expressions as those mentioned in 2.3. The respective *QoS Characteristic* is indicated in the annotation statement via *context*. An example *QoS Offered* statement in OCL is shown below: "From Monday to Friday 8:00am to 8:00pm, the response time can be guaranteed to be below 10 ms."

```
<<QoSOffered>>
{context Time_Performance inv:
(Set{'Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday'} ->includes(getToday()) and getCurrentTime() >
'8:00' and getCurrentTime() < '20:00') implies respon-
seTime < 10}
```

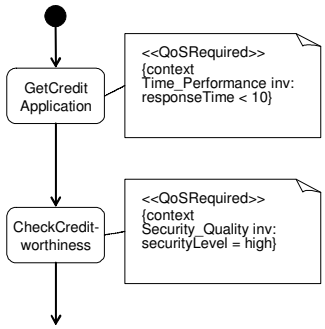

Fig. 3. Example QoS requirements in a UML Activity diagram

Introducing such QoS annotations into the OWL-S service descriptions can simply be done by adding *QoSCharacteristics* as a new *ServiceParameter* in *ServiceProfile* and *QoSStatement* as a subclass of *owl:Thing*. *QoSStatement* has the datatype property *statement* of the type string. This field contains the QoS constraints in OCL of the element to be annotated. Figure 3 shows example QoS requirements on the service requestor side in a UML Activity diagram. *responseTime* of *GetCreditService* is required to be lower than 10 ms. Figure 4 shows the corresponding QoS offering in the

service description of *GetCreditService* that would be a match during service matching.

```
<profile:Profile
  rdf:ID="GetCreditService_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 Time_Performance
            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. 4. QoS offering for *GetCreditService* in the service description

5 Service Management Prototype

5.1 Overview – Architecture and Functionality

The first version of the prototype is a combination of self-developed systems and available open source tools. It is realized as a web application and contains a web browser-driven user interface and two service repositories, one for the standard UDDI publishing and discovery, and one for the semantic search. Two repositories are necessary because the OWL-S-based repository is not UDDI standard compliant, while UDDI as the current standard for service repositories does not offer semantic support. The UDDI registry can be filled automatically with the information from the OWL-S repository. In order to make that possible, a mapping for many of the repositories' elements has been defined.

Figure 5 gives an overview of the prototype's architecture which is structured in 3 layers. The first layer is the web client and client application. It contains the user interface as a web browser application. Via this web-based front-end the user has access to the functionality described in the next section. User authentication functionality as well as storing the account information in a database is implemented here. The client accesses the UDDI and OWL-S repository on a web application server via SOAP, the standardized XML-based message exchange format for Web services. The UDDI repository is based on jUDDI as persistence layer. The OWL-S repository 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 (Re-

source Description Framework Query Language) [23]. 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. The prototype itself is written in Java. It uses RMI (Remote Method Invocation) for communication between the Java components.

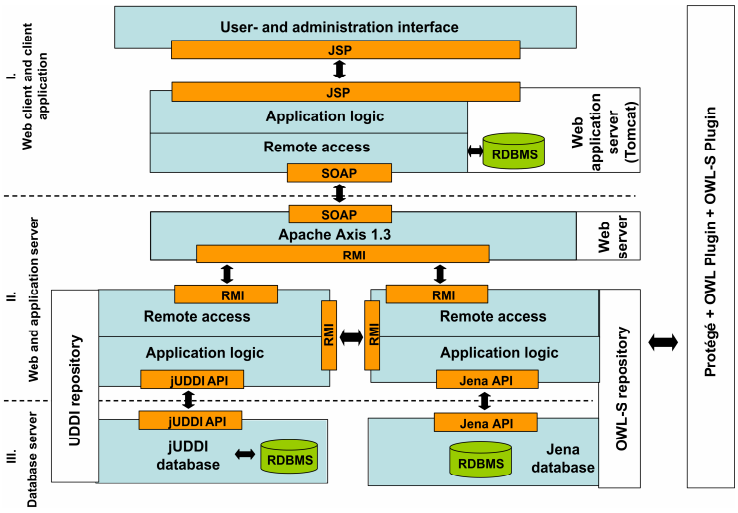

Fig. 5. Overview of service management prototype

Apart from the self-written parts, the prototype uses the readily available packages Protégé, Protégé-OWL, and OWL-S Editor. Protégé is a free, open source ontology editor from Stanford University [24]. Protégé with Protégé-OWL, a plug-in for defining ontologies in OWL, also from Stanford University (available at [25]), is used for the taxonomy definition. OWL-S Editor is a Protégé plug-in developed at SRI International (available at [26]). It helps to define 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.

5.2 Functions and Methodology of the Prototype

The first version of the prototype supports the following tasks as a basis for the mentioned service management responsibilities: taxonomy/ontology definition, service description, semantic annotation, service registration, service discovery, service review, and user access control.

5.2.1 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. The generic way of defining/redefining the service taxonomy is an important feature. It is a matter of fact that there is no stable service description in complex environments.

5.2.2. Service Description and Semantic Annotation

Service description and semantic 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. 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.

5.2.3. Service Registration

Service registration is done by importing the OWL-S service description into the prototype and its database. This is necessary after each change to it. The prototype can then perform the search activities laid out in the next section.

5.2.4. 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 additional parameters such as *ServiceVersion*, *ServiceResponsibility*, and *ServiceLifecycleStatus*
6. Taxonomy tree search – services that belong to one taxonomy can be found by navigating through a simple taxonomy tree (uses Tigra Tree Menu [27]) or a hyperbolic graph (uses HyperGraph [28])

Number 3 refers to a simple matching functionality that can be used for service orchestration and will be extended in the future. To increase the flexibility of the search, it is possible to use the outcome of one search run as the basis for another search.

5.2.5 User Access Control

For service management in complex environments it is absolutely necessary to support role-specific views combined with access rights management. The numerous services are the core of an IT system of an enterprise. Therefore they need to be pro-

tected against malicious attacks as well as erroneous and uncoordinated activities of careless or unaware users. Hiding unnecessary information improves usability, reduces the number of errors, and is sometimes a must when it comes to confidentiality. The prototype's user authentication module controls the activities of individual users according to the rights associated with their roles. An "Administrator" can add new accounts and associate them to a role. "Users" are only allowed to search and browse through the service repository. "Developers" can in addition perform detailed search operations. The "Architect" is also allowed to register and delete services in the repository.

6 Importance for Matching, SLA Negotiation and Enforcement

Currently, the search needs to be done manually. Having visibility about all services implemented and the possibility of managing meta-information of the services centrally and thus in a consistent way is a big advantage and a precondition for the success of an SOA. However, if the IT systems based on the services get bigger and bigger and the number of services is expanding, a process that includes more automated support is necessary. The semantic description of input and output parameters and non-functional characteristics is a prerequisite for that. Only if service requestor and service provider refer to the same ontological concepts, the service matching module can "understand" them. That is why the additional effort of managing the semantic metadata is justified. A common way of performing the matching or SLA negotiation is a two-step approach as proposed by METEOR-S, Grønmo/Jaeger [18], or in "Semantic WS-Agreement Partner Selection" [29]. The first step performs functional matching. We suggest a hybrid semantic matching based on input and output parameters, e.g. by using OWLS-MX. In addition we propose to use the service category. Due to the semantic information not only exact matches of parameters and taxonomies are found but also parameters that stand in a class-sub-class relationship, e.g. car – convertible. The second matching step performs the non-functional matching using particularly the QoS-related information. Constraints about the QoS-characteristics on the service consumer side (*QoS Required*) are compared with the QoS-offerings specified in the service description (*QoS Offered*). The outcome is a ranking of the existing services that perform the desired functionality according to how well they meet the QoS requirements. Once a service is chosen, an SLA, a formal specification of the agreement between service consumer and service requestor (inter- or intra-organizational) can be specified.

It is planned to extend the prototype's service matching functionality and also to introduce an SLA specification, and SLA management module. According to a service request with a set of semantically enriched functional and non-functional information this module will discover existing services in the repository, provide their WSDLs and specify the SLA in a nearly fully automated way. The format for the SLA will be WSLA or WS-Agreement. The machine-readable SLA is a good basis for automated SLA-enforcement and monitoring during run-time. In case of problems, the person responsible can find the respective service in the registry and has access to information, e.g. contact details, infrastructure the service runs on. Matching

and SLA specification functionality will ease the life of system developers as well as SLA authors/enforcers. It will also foster reuse, one of the goals of SOAs.

7 Conclusion and Outlook

As SOAs will be very complex from an IT service management point of view, in order to deliver their full value automated tool support is necessary. Semantic description of non-functional service characteristics is one important prerequisite for that.

The contribution of the presented work is a practical approach to service description and discovery 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. With the approach, it is possible to describe – along with the functional characteristics – the non-functional characteristics with respect to service management (service lifecycle management and QoS) in a single OWL-S-based repository. The approach is extendable with respect to changes of the used taxonomy as well as the elements used for service description. At the same time it is a compatible upgrade of the existing Web services description standards. Besides the presented approach, the article also gave an overview over relevant standards and related work in the area of non-functional service description.

The prototype will be extended to support better integrated service description functionality. Extensions for automated service discovery, SLA specification, and SLA management are planned.

References

1. Sallé, M.: IT service management and IT governance: review, comparative analysis and their impact on utility computing (2004), http://www.hpl.hp.com/techreports/2004/HPL-2004-98.pdf
2. Woolf, B.: Introduction to SOA governance - Governance: The official IBM definition, and why you need it. IBM (2006), http://www-128.ibm.com/developerworks/webservices/library/ar-servgov/index.html
3. O'Sullivan, J., Edmond, D., ter Hofstede, A.: What's in a service? Towards accurate description of non-functional service properties. Kluwer Academic Publishers (2002), http://www.infosys.tuwien.ac.at/Teaching/Courses/IntAppl/Papers/WhatsInAService.pdf
4. Dobson, G.: Quality of Service in Service-Oriented Architectures (2004), http://digs.sourceforge.net/papers/qos.html
5. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic Web services selection. IEEE Internet Computing 08 (2004) 84-93
6. Frolund, S., Koistinen, J.: Quality of Service specification in distributed object systems design (1998), https://www.usenix.org/publications/library/proceedings/coots98/full_papers/frolund/frolund.pdf
7. DAML: DAML Services (2006), http://www.daml.org/services/owl-s/
8. 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: Se-

mantic markup for Web services (2006), http://www.ai.sri.com/daml/services/owl-s/1.2/overview/
9. Web Service Modeling Ontology - ESSI WSMO working group (2006), http://wsmo.org
10. 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/
11. Emerging Technologies Toolkit. IBM (2006), http://www.alphaworks.ibm.com/tech/ettk
12. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web Services Level Agreement (WSLA) Language Specification (2003), http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf
13. Tosic, V., Patel, K., Pagurek, B.: WSOL - Web Service Offerings Language. In: Bussler, C., et al. (ed.): CAiSE'02 (2002) 57-67
14. Lamanna, D.D., Skene, J., Emmerich, W.: SLAng: A Language for Defining Service Level Agreements (2003), http://www.cs.ucl.ac.uk/staff/w.emmerich/publications/FTDCS03/slang.pdf
15. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement) (2005),
16. 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
17. Aagedal, J.Ø.: Quality of Service support in development of distributed systems. Department of Informatics, Faculty of Mathematics and Natural Sciences, Vol. Doctor Scientiarium. University of Oslo (2001)
18. Grønmo, R., Jaeger, M.C.: Model-driven methodology for building QoS-optimised Web service compositions. The 5th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS)
19. Tian, M.: QoS integration in Web services with the WS-QoS framework. Department of Mathematics and Computer Science. Freie Universität Berlin, Berlin (2005)
20. Klein, M., König-Ries, B.: A process and a tool for creating service sescriptions based on DAML-S (2003), http://hnsp.inf-bb.uni-jena.de/DIANE/docs/TES2003.pdf
21. Klein, M., König-Ries, B., Müssig, M.: What is needed for semantic service descriptions - a proposal for suitable language constructs. International Jounal on Web and Grid Services (2005)
22. Klusch, M., Fries, B., Sycara, K.: Automated Semantic Web Service Discovery with OWLS-MX. AAMAS 2006. ACM, Hakodate, Hokkaido, Japan (2006)
23. Jena – A Semantic Web Framework for Java. sourceforge.net, http://jena.sourceforge.net/
24. Welcome to Protégé. Stanford Medical Informatics (2006), http://protege.stanford.edu/
25. What is Protégé-OWL? Stanford Medical Informatics (2006), http://protege.stanford.edu/overview/protege-owl.html
26. The OWL-S Editor (2004), http://owlseditor.semwebcentral.org/
27. SoftComplex: Tigra Tree Menu. SoftComplex, http://www.softcomplex.com/products/tigra_tree_menu/
28. HyperGraph, http://hypergraph.sourceforge.net/
29. Oldham, N., Verma, K., Sheth, A., Hakimpour, F.: Semantic WS-Agreement Partner Selection. International World Wide Web Conerence Committee (IW3C2). ACM, Edinburgh, Scotland (2006)
30. Schmietendorf, A., Dimitrov, E.: Management serviceorientierter Architekturen auf der Grundlage von ITIL, http://www.cecmg.de/doc/tagung_2006/fileadmin/trilog/download/cecmg_2006/Referenten/2/2C1_Langf_Schmietendorf.pdf