Semantic Web Services

Final Report

Jan Schulte

Course Advanced Topics in Software Engineering (LP 02/08)

BlekingeTekniskaHökskola

18 January 2009

Table of Contents

Introduction

In the last years the acronym SOA has become one of the most emerging trends within the IT industry. SOA stands for service-oriented architecture and came-up from the need to better integrate IT departments within a company and IT organizations amongst each other.

Traditionally the software used within a certain department of a company solves only the local department problems and is mostly UI-based (s. [1], p. 55). As can be seen in Figure 1, these systems are also usually highly heterogeneous, written in different languages or being COTS products.

Since the need for automation is increasing and applications need to access other applications' functionality (s. [**1**], p.55), integration becomes a major problem. This is especially true when new applications need to be added or other companies need to be connected.

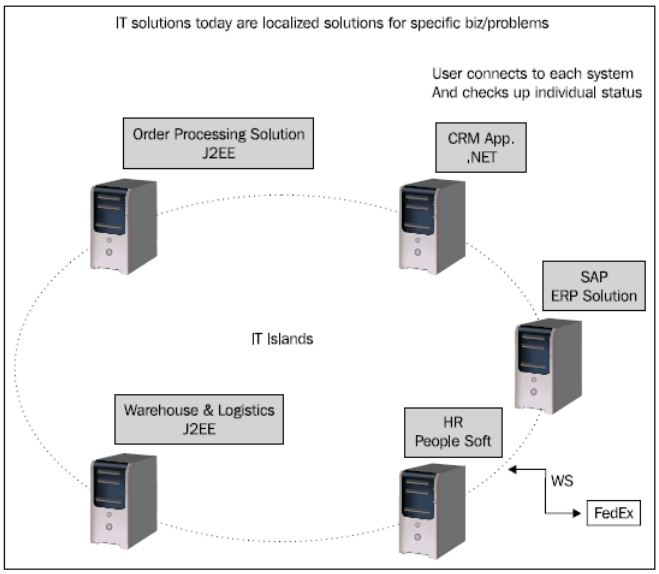

Figure 1: IT-Islands within an organization ([1],p. 61)

To overcome the integration gap and maintain flexibility and scalability, the idea of service-oriented architecture arouse. The software is hereby designed up-front as loosely-coupled services, which reflect the business processes of the company (s. [**2**]). These services have to "provide business value, hide implementation details, and be autonomous. Service consumers are software entities, which call the service and use its functionality." (s. [**3**], p. 13).

The most-common approach for implementing a service-oriented architecture is using web-services, which use "the Internet as the communication medium and open Internet-based standards" (s. [**4**], p. 64). The most important standards hereby are WSDL for the service interface description, UDDI for service registries, SOAP for transmitting data and BPEL for service composition.

To be interchangeable among different businesses and allow the automatic use of web services, they should contain semantics. That is, a web service description should not only contain the syntactical information of the method signature, but also describe the meaning, i.e. what the service does, under which circumstances, etc. Current standards offer little support for semantics; hence there is a need for further research in the field of semantic web services.

Report Structure

This report is split into two parts. The first part gives an overview into semantic web services. It begins with an introduction into web services and the current standards. Semantics and why it is needed will be explained in the subsequent chapter. The following chapter gives a short overview on the combination of semantics and web services and a chapter about current research in this topic, amongst them service composition, closes the first part. Part two will then give an insight into one approach for semantic service composition using model-driven architecture (MDA). It starts with a brief introduction to MDA and explains the composition process in the next chapter. An outlook on future work ends this part.

Semantic Web Services

Web Services

Nowadays the term web services is used throughout the literature to refer to a whole set of standards for providing services based on internet technology. Web services are today supported by all major vendors (s. [**3**], p. 9) and are widely regarded as the way a service-oriented architecture should be implemented (s. [**5**], p. 209).

So what are web services? Basically, web services are a set of open and XML-based standards. Since XML technology is "the de-facto standard for data-level integration" and web-services are built upon them, they have become the "technological foundation" for achieving the required interoperability among businesses (or departments) (s. [**3**], p. 8). Applications can now use "different software platforms, operating systems, and programming languages" and can still remain interoperable.

As mentioned previously, the term web services is used to refer to a set of XML-based standards managed by different organizations (e.g. the W3C, OASIS, etc.) and also commonly known as WS-Standards (s. [**5**], p. 210). Up until now, there exist over 50 standards that address all areas of interest, ranging from the most basic functionality to advanced topics like QoS or security, where some standards are still work-in-progress.

Out of these standards, the four most important will be discussed in the following section, which are WSDL for the service definition, UDDI for service registries, SOAP for service invocation and BPEL for service composition.

Standards

WSDL

Probably the most important standard is the web service description language (WSDL), which currently exist in its version 2.0 (s. [**6**]). Similar to the IDL in CORBA, WSDL defines the interface of the service by describing its signature, but also, and that is different to the IDL, it's binding, i.e. where the service is actually deployed and which protocol is used to invoke the service (s. [**5**], p. 211).

Therefore a WSDL file contains the service functions, the data types that are used, the transport protocol information and the address of the location in a platform-independent manner (s. [**7**], p. 119). Using this information, the client can invoke the service directly if it has the WSDL file.

UDDI

Universal Description, Discovery, and Integration is a protocol for "building a distributed directory of businesses and web services" (s. [**7**], p. 157). Hence the UDDI allows businesses to register itself and its services and also allows others to search for them (s. [**1**], p. 83).

Since any client needs to discover a service before it can use it, having a central registry seems a logical approach and in 2001, Microsoft and IBM started a centralized UDDI registry for businesses (s. [**7**], p. 157). However, history showed that it didn't work out and the central registry was turned down in 2006 (s. [**5**], p. 219). Nonetheless, UDDI is still useful and can be used as a registry within a company.

SOAP

SOAP is a protocol for invoking a web service and exchanging web service data (s. [**5**], p. 211). It is therefore similar to other RPC frameworks like e.g. Java RMI, but due to the use of XML it is platform-independent (s. [**7**], p. 49).

BPEL

The last important standard presented in this report is the business process execution language (BPEL). Apart from providing interoperability with other systems, the idea of SOA also is to improve reuse of software systems and therefore reduce costs (s. [**8**], p. 5). Hence, reusing existing services and composing them to new services is a key asset of SOA. Several languages for composition exist, but BPEL has become the most-popular language for composing web services (s. [**3**], p. 7).

Semantics

To better understand semantics and why it's needed let us first look at a simple example. Two fictitious airlines named SaveLine, a low-cost carrier, and LuxusAir, a luxury airline, provide a web service for travel agencies to ask for a simple price (see Figure 2). Both services have the same name, expect the same parameters and return a *Price*, i.e. they have exactly the same method signature.

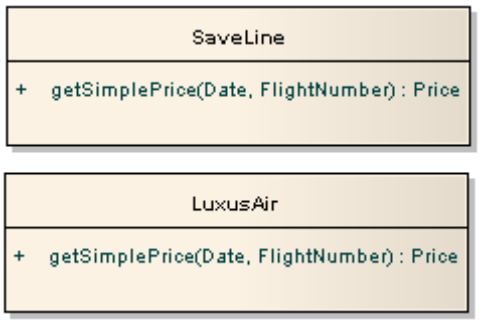

Figure 2: Web services of two imaginary airlines

On the first look, these two services appear to be the same. However, the problem lies in the meaning of simple price. For SaveLine, the simple price might mean the price without taxes, whilst for LuxusAir, the simple price might be the price in economy class (i.e. not business class).

Therefore, an application needs to know about the meaning, i.e. the semantic, of the web services to avoid using them mistakenly in the same manner.

On the other side, two services can have exactly the same meaning, but be syntactically different. Suppose the two logistic companies in Figure 3 offer the same web services with the same meaning, but with different names. An application will not be able to use these services in the same way without knowing about the meaning of them.

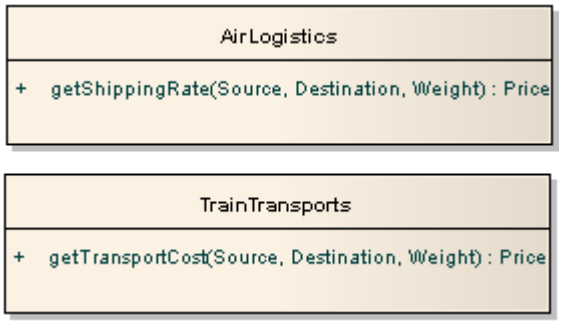

Figure 3: Web services of two imaginary logistic companies

As one can see from the examples above, there is a need that the computer can understand the meaning (semantics) of the data and the services (s. [**9**], p. 3). Without these semantics, the machine will not be able to "make any intelligent decisions" about the data or services (s. [**10**], p. 9).

The approach to integrate semantics into documents, data or web services is by using metadata. Metadata are information that capture the meaning of the data, and are essentially data about data (s. [**11**], p.9). The most common way to do this is by using an ontology, which will be explained in the next chapter.

Ontologies

"An *ontology* is an explicit specification of a conceptualization" (s. [**12**]). Conceptualization hereby refers to an „abstract and simplified view on the world". Therefore, an ontology describes formally a domain (s. [**11**], p. 10), and consists of concepts, which are classes of objects, and relationships amongst them. Besides that, formal axioms are used to verify the consistency of the ontology and to infer new knowledge from the ontology (s. [**13**], p. 46). Therefore we get much stronger semantics than with simple metadata.

An example of an ontology can be seen in Figure 4, which shows a simple newspaper ontology, taken from the Protégé tool, an ontology editor.

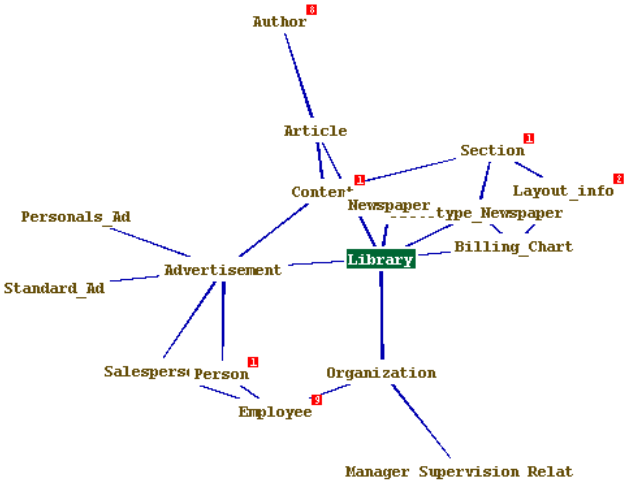

Figure 4: Visualization of the newspaper ontology from the Protégè tool

Another important aspect of ontologies is, that they "should provide a shared understanding of a domain" to "overcome differences in terminology (s. [**11**], p. 10). That means, that ideally an ontology is a shared work from the interest groups of a particular domain.

Probably the most important language for describing these ontologies is the Web Ontology Language (OWL), which is a W3C recommendation since 2004 (s. [**14**]). OWL is an XML language based on the popular RDF schema (s. [**10**], p. 95). In the following, OWL won't be further discussed in this report as extensive literature about this topic is available.

The question now arises, where these ontologies come from.

Basically, there are two ways of creating ontologies. The first is to do it manually, by using an ontology editor like Protégé. The second, that is also commonly used, is to create the ontology from a database, either from the database schema itself or from the data that it contains. Large ontologies exist especially in the field of biology.

Usually the ontologies are available from the web page of its creator, but there exist also public ontology registries (e.g. [**15**]), where the developer can upload and the user download ontologies.

Use of standards

Through standardization of the web services for each industry, the need for having semantics could be avoided (s. [**16**], p. 191).

However, since web services describe the business processes of a company, they are very much related to the company and creating a standard for each domain wouldn't be feasible in practise. Ontologies on the other hand describe domain knowledge. Therefore, two ontologies describing the same domain should be the same or very similar. Hence, standardizing ontologies or shared usage of the same ontology among different organisations should be much easier in theory. On the other side, reality shows that several different ontologies exist for the same domain and much research is done on mapping ontologies. A standardization organisation for ontologies could be an option.

Web Services & Semantics

As we have seen in the previous chapter, there is a need for knowing the semantics of the web services. The current standards like WSDL, UDDI and SOAP however are very limited in describing the semantic capabilities of the web services (s. [**16**], p. 191).

Up until now, there is no well-established standard for describing semantic web services, but several promising approaches exist. Among them are OWL-S, WSMO and WSDL-S.

OWL-S was submitted in November 2004 to the W3C (s. [**17**]) and is currently the standard getting most attention in the research community. OWL-S is based on OWL as the name already reveals and defines an upper ontology for web services. The idea of OWL-S was to provide a standard for web services that helps to answer the following questions (s. [**17**]):

- *What does the service provide for prospective clients?*
- *How is it used?*
- *How does one interact with it?*

Therefore, an OWL-S document consist of a *Service Profile*, describing what the service does, a *Service Grounding*, detailing how an agent can access the service (e.g. communication protocol, message format, etc.) and a *Service Model* providing the answer to the question of *how to use the service*, by describing "what happens if the service is carried out". More information on OWL-S can be found on the website of the W3C (s. [**17**]).

The web service modelling ontology (WSMO) provides a conceptual framework for web services in which an ontology has to be defined to add semantics to the service (s. [**16**], p. 200).

WSDL is another W3C submission (s. [**18**]) and extends the well established WSDL standard to include semantic annotations.

Current Research

The four most important areas of research in the field of semantic web services are automatic annotation, service discovery, service invocation and service composition. They will be discussed in the following.

Automatic Annotation

As seen in the previous chapter, semantic is added to the web services by annotating the service description with the concepts/classes of an ontology. Since ontologies can have and often do have more than thousands of concepts, it is clear that annotating a web service with an ontology is a work-intensive and error-prone task, if done by a human. Especially, if the organization offers hundreds of services. Current research leads into the direction of doing the process of annotation automatically by the computer.

Service Discovery

Another important research area is the automatic discovery of services through the use of semantics. In the context of the example from the previous chapter, this would mean, that an application or human could search for a logistics price information service and gets, as result from a search engine or registry, the available services. Automatic service discovery is especially useful in the area of software reuse, where as much functionality as possible is being reused from existing software assets (s. [**16**], p. 193).

Service Invocation

If a user or an application has found a web service, it is probably going to execute/invoke it. Therefore, a more detailed level of matching is necessary to automatically invoke the respective service (s. [**16**], p. 196). Let's take e.g. the FedEx rate service, which calculates the shipping fee for a certain package from a location A to a location B. The WSDL service description of this, rather simple, service has already 3200 lines of XML code, and requires the information in a deeply nested XML tree. It is clear, that even if we have found semantically similar services (e.g. the FedEx, UPS and DHL rate service), it is not possible to easily select one of them and invoke it. The information that are available must be transformed to fit the service signature of the respective service. Hence, an automatic invocation engine is needed.

Service Composition

One key aspect of service-oriented architecture is that services are reused to reduce cost in the software development. Hence, the idea is to use existing services and *compose* new services out of them. Ideally, service composition should be fulfilled automatically by a machine, which takes an abstract task description and composes a new service out of the repository of available services(s. [**16**], p. 196). This composition then can directly lead to an executable BPEL file.

Several approaches for automatic web service composition exist (see [**19**] for an overview), but they still face the problem that the matchmaking is still not solved. Therefore in the second part of this report, a semi-automatically approach is described, which uses the techniques from the model-driven architecture to compose web services, but with user-intervention.

Service Composition with MDA

This part will give an overview over semi-automatic service composition using model-driven architecture (MDA) proposed by Grønmo and Jaeger in 2005 (s. [**20**]). A short overview over MDA is given as an introduction.

Model-Driven Architecture

MDA is a software development approach that that facilitates portability, interoperability and reusability (s. [**21**], p. 202). The basic idea behind MDA is that developers extensively use models to reason about the problem and solution domain (s. [**20**], p. 2) and create sufficiently detailed models so that the system can be automatically generated from them (s. [**20**], p. 5).

Therefore, the software is developed by creating models that are transformed to the final code in a series of model-refinements (s. [**20**], p.7). This stepwise approach can be seen in Figure 5. The problem that the software is going to address is first analysed in the domain model, i.e. the Computation Independent Model (CIM), and focuses on the requirements and the environment (s. [**21**], p.199). This model is then transformed into a Platform Independent Model (PIM), that contains computational but no platform specific information. Eventually the PIM is further transformed into a Platform Specific Model (PSM) which contains details for the implementation platform.

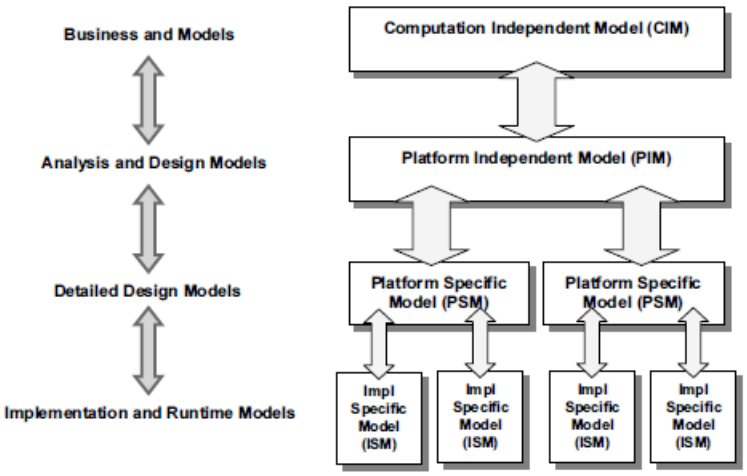

Figure 5: Stages in MDA ([24], p. 7)

The advantages of this systematic process of automatic model transformation are that it improves the quality and efficiency of validating the architecture against the requirements and that it provides abstractions that allow various stakeholders to reason about the system from various view points (s. [**21**], p. 202).

Composition Process

In [**20**] Grønmo and Jaeger use the model-driven approach for composing semantic web services. It targets mainly developers who have to compose web services and helps them doing this in an easy manner. Therefore, they have developed a methodology of four steps that helps to compose a web service by using semantics and Quality-of-Service (QoS) attributes (s. [**20**], p. 2). The service is decomposed by the developer into abstract sub tasks, which are then filled with concrete service implementations by the composition engine.

First of all, the methodology requires three registries, an ontology registry where the ontologies are stored, a QoS registry containing available QoS characteristics and a general web service registry with the available web services (s. [**20**], p. 2).

The following will explain the steps of the methodology in detail:

1. Modelling

The first step in this approach is that the developer starts to model an abstract web service. This can be done in a UML modeller. The abstract service is then annotated with appropriate concepts from the ontology registry to specify the semantics of the web service. This is the first appearance of model-driven development in this approach, since the ontology, usually in OWL format, needs to be transformed to be understood by the UML editor, which uses the XMI format to represent UML models. An XSLT based transformation could be an option. Afterwards the developer annotates the abstract services with desired QoS characteristics, which are once again imported to the UML modeller via model transformation. Figure 6 shows an example of an abstract composition with annotated semantics and QoS characteristics.

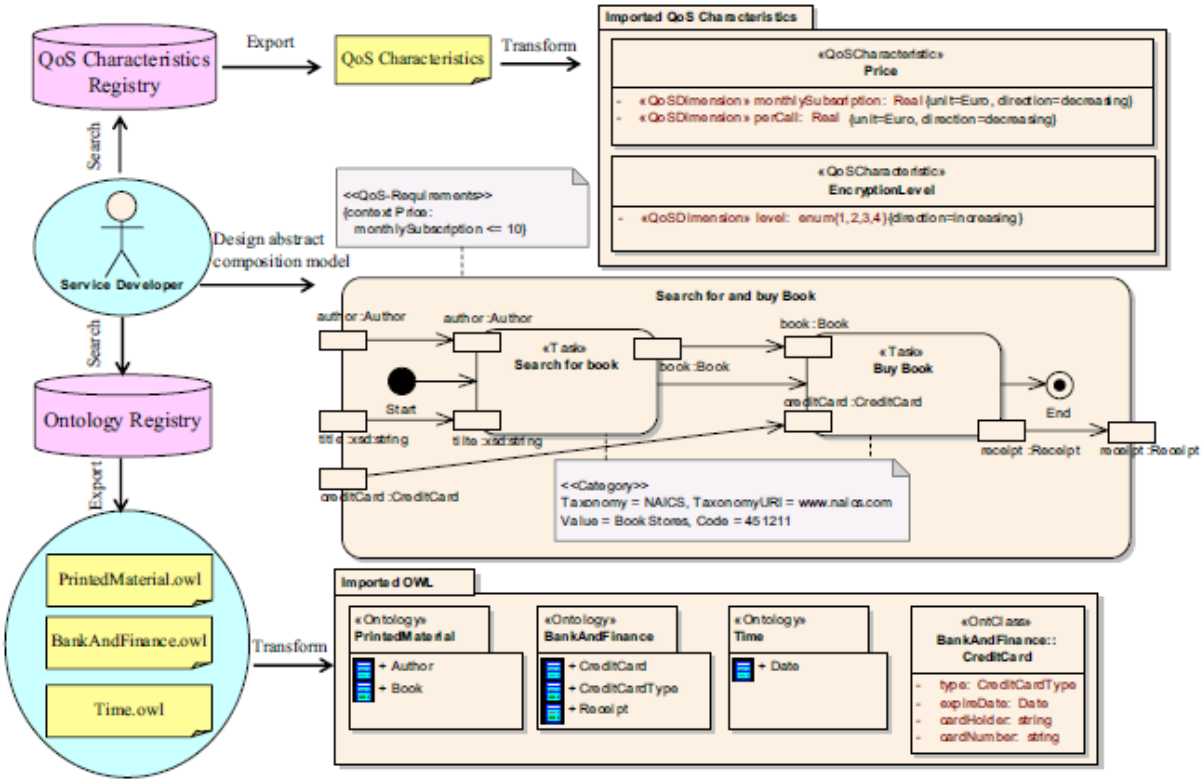

Figure 6: Abstract Composition with Tasks (s. [20], p. 4)

2. Discovery

The next step in the methodology is the automatic discovery of appropriate web services that fill the place-holders in the abstract web service. This is done by using semantic matchmaking to find services that fit the requirements. To make it possible for the application to do the matchmaking, it first needs a semantic representation of the web service. This is once again done by transforming the UML based XMI representation to a semantically annotated web service representation, for instance OWL-S.

3. Selection

After finding the appropriate service candidates, the services are ranked according to the QoS criteria. The developer will now check whether the selected services are really appropriate or if he

has to adjust or change something. This step is based on the assumption that the semantic matchmakers don't work 100%.

4. Deployment

Upon completion, the developer can now execute the composed service and publish it as a new service to the web service registry, as depicted in Figure 7. By transforming the newly created web service to an executable composition format like e.g. BPEL, he is able to run it in an execution environment. The web service is added to the registry by transforming it to a WSDL file.

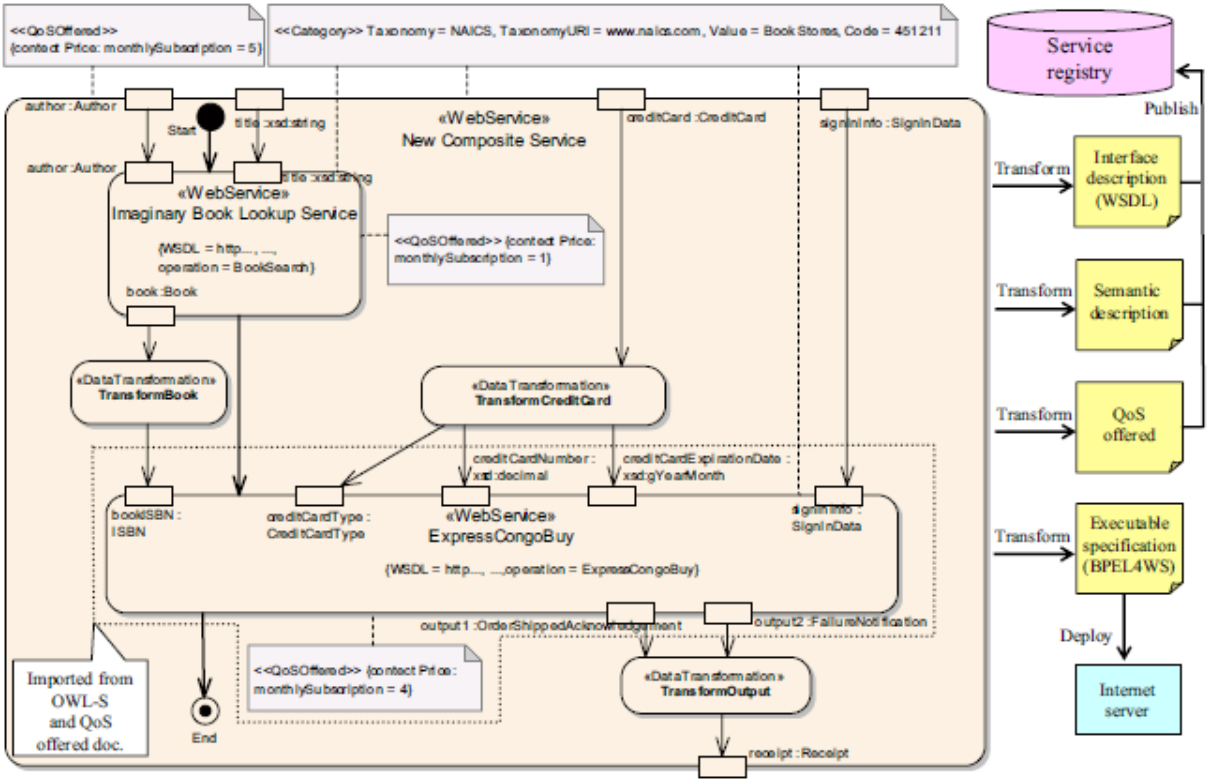

Figure 7: Concrete Composition Model with chosen web services (s. [20], p. 5)

Current and Future Work

This approach requires several model transformations ([**20**], p.7 provides a detailed list of all the necessary transformations). Not all of them are already available and some of them just exist as a proof-of-concept (s. [**20**], p.6). Therefore there is still work to do to make this approach useful in industry.

Another important research area is the one of semantic matchmaking. This is not only important for finding suitable services in this approach, but also for automatically annotating the abstract web service with concepts of an ontology, since an ontology can contain more than 1000 concepts, and therefore the process of annotating the service will be a very work intensive task.

Ontology Alignment Evaluation Initiative (OAEI) annually benchmarks current matchmaking algorithms and provides a detailed list about the results on their web page (s. [**22**]).

Comparison of Service Composition with a standard Search

Web services and composite web services can also be used to create a search for a subject, but is slightly different than standard search approaches like e.g. Google.

Google returns a list of web pages that are related to the search string. E.g. a search string „weather in Karlskrona“ will return a list of web pages that can provide this information, namely weather forecast providers. Whereas the idea of web services, respectively the composition of them is to return the concrete answer to a specified question. In the sense of the MDA approach of web service composition, this would mean that the developer or user describes an abstract web service, containing e.g. a web service that looks up a postal code for a specific city and calls then a second web service that provides the weather forecast for the previously obtained zip code. The matchmaking engine would than fill these abstract services with concrete services, e.g. with the WS of the Swedish post and the national weather forecast. If the user than invokes the composition with the input parameter Karlskrona, he would obtain the concrete answer, e.g. „“sunny, 20°“.

However, Google is already going into that direction. E.g. a search for *“ what is Barack Obama's birth date”* not only returns a list of web sites, but also the correct answer *“Barack Obama — Date of Birth: 4 August 1961”* (s. [**23**]).

Summary

Service-oriented architecture is increasingly becoming the standard way to integrate IT-departments and/or IT-organizations with each other. The software is hereby constructed as services that reflect the business processes of the company. Other departments/companies can then consume these services. This approach provides increased flexibility and scalability and is nowadays mostly realized through the use of web services.

The term web services is used in literature to refer to a whole set of XML-based standards to allow the use of services by internet technology. This makes them, at least in theory, platform-independent and interoperable. The most important standards are WSDL, an interface description language, SOAP, a communication protocol, UDDI, a service registry, and BPEL, a language for composing web services.

One of the visions in the field of web services and service-oriented architecture is to automate the use of services. For instance an application should be able to automatically discover and use a specific web service. The application therefore needs to know about the meaning, i.e. the semantics, of the web service it should use. Adding semantics to data or services is mainly done by describing it with an ontology, i.e. a shared conceptualization of domain knowledge. The web ontology language (OWL) is a widely accepted language to specify this ontology.

The current standards for web services provide little support for adding semantics, and some research has been done to cope with that issue, which resulted e.g. in OWL-S, a language to describe the semantics of a web service.

Current Research in the field of semantic web services focus mainly on four areas:

- Automatic semantic annotation of web services
- Semantic service discovery
- Semantic invocation of web service, by adapting the parameters to the signature of the target service
- Automatic service composition

One approach to service composition is using techniques from the field of model-driven architecture (MDA) to create compositions with user-intervention. Therefore it is not a fully automatic but a semi-automatic approach.

The principal idea behind MDA is to develop the software through the intensive use of models. The problem is first modelled in the problem-domain, and then, through the use of model-transformations, in several steps converted to a platform-specific model, namely the code.

The MDA-based approach defined by Grønmo and Jaeger In [**20**] is a step-wise approach and consist of the following four phases:

- Modelling: Model an abstract service and annotate it with an ontology
- Discovery: Transform the service to an OWL-S file and fill it with concrete services due to semantic matchmaking
- Selection: The user accepts the choice or adjust it accordingly
- Deployment: Service is transformed to a BPEL file and executed and deployed in the service registry as a WSDL file

Bibliography

[1] P. Sarang, *SOA Approach to Integration*. Packt Publishing, 2007.

[2] B. Borges, K. Holley, and A. Arsanjani. (2004, Sep.) Delving into Service-Oriented Architecture. [Online]. http://www.developer.com/design/print.php/10925_3409221_1

[3] M. B. Juric, *Business Process Execution Language for Web Services BPEL and BPEL4WS*. Packt Publishing, 2006.

[4] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *IEEE Computer*, vol. 40, pp. 38-45, 2007.

[5] N. M. Josuttis, *SOA in practice*, 1st ed. O'Reilly, 2007.

[6] W3C. (2007, Jun.) Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. [Online]. http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/

[7] E. Cerami, *Web Services Essentiials*. Sebastopol: O'Reilly, 2002.

[8] J. P. Lawler and H. Howell-Barber, *Service-Oriented Architecture*. Boca-Raton: Auerbach Publications, 2008.

[9] T. B. Passin, *Explorer's guide to the Semantic Web*. Greenwich: Manning Publications, 2004.

[10] L. Yu, *Semantic Web and Semantic Web Services*. Boca Raton: Chapman & Hall/CRC, 2007.

[11] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*. Cambridge: The MIT Press, 2004.

[12] T. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993. [Online]. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[13] O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "Ontological Engineering:What are Ontologies and How Can We Build Them?," in *Semantic Web Services*. IGI Global, 2007, ch. 3, pp. 192-216.

[14] W3C. (2004, Feb.) OWL Web Ontology Language Overview. [Online]. http://www.w3.org/TR/owl-features/

[15] DAML . (2004, Apr.) DAML Ontology Library. [Online]. http://www.daml.org/ontologies/

[16] R. Akkiraju, "Semantic Web Services," in *Semantic Web Services*. IGI Global, 2007, ch. 9, pp. 192-216.

[17] D. Martin, et al. (2004, Nov.) OWL-S: Semantic Markup for Web Services. [Online].

http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

[18] R. Akkiraju, et al. (2005, Nov.) Web Service Semantics - WSDL-S. [Online]. http://www.w3.org/Submission/2005/SUBM-WSDL-S-20051107/

[19] J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods," in *Semantic Web Services and Web Process Composition*. Berlin: Springer, 2005, pp. 43-54.

[20] R. Grønmo and M. C. Jaeger, "Model-Driven Semantic Web Service Composition," in *Software Engineering Conferencesia-Pacific*, 2005, pp. 8-16.

[21] I. Gorton, *Essential Software Architecture*. Berlin: Springer, 2006.

[22] OAEI. (2008) Ontology Alignment Evaluation Initiative. [Online]. http://oaei.ontologymatching.org/2008/

[23] Google. (2009, Jan.) Google search for Barack Obama's birth date. [Online]. http://www.google.com/search?hl=en&q=what+is+barack+obama%27s+birth+date&btnG=Search

[24] A. W. Brown, J. Conallen, and D. Tropeano, "Introduction: Models, Modeling, and Model-Driven," in *Model-Driven Software-Development*. Springer, 2005, pp. 1-16.