A Native Ontology Approach for Semantic Service Descriptions

Rajesh Thiagarajan and Markus Stumptner

Advanced Computing Research Centre
University of South Australia
Email: {cisrkt,mst}@cs.unisa.edu.au

Abstract

A number of ontology-based approaches have been suggested for the description of service behaviors to be used in service composition and matching in service oriented architectures. We examine an approach based on classical software engineering notation and compare it to other approaches.

1 Introduction

Semantic Web technology has been a significant driver of recent research in a number of related areas such as data and application integration, distributed business process management, and service oriented architectures, all areas where declaratively describing the behavior of pieces of software can be used for the management, combination, and sometimes execution control of these pieces. These approaches typically build on approaches used in particular sub-communities, such as the use of OWL-S 1 as a language for specifying various ontologies used within Web Service based applications. Past work has examined the similarities and analogies holding between these domains and languages and classical OO principles used in software engineering (Koide, Aasman & Haflich 2005, Djuric, Gasevic & Devedzic 2005). In this paper we try to build the bridge back into Model Driven Architecture (MDA) territory by using the types of descriptive semantics employed in Semantic Web service technology, in the guise of classical OO design notations. We describe services in the style of Semantic web specifications, but use classical software engineering notations and sublanguages, in the shape of UML and OCL, and we reason directly on these service specifications.

Web Services are programs that can be remotely accessed using the protocols of the World Wide Web, with communications based on standards such as Simple Object Access Protocol(SOAP)\(^2\). The Web Service Description Language (WSDL)\(^3\) is the current standard to describe Web Services. WSDL describes a service in terms of multiple ports, each of which defines sets of ingoing and outgoing messages that can be used to communicate with the service through that port. A message is an XML structure that contains the parameters (called parts) necessary for execution of the service, or passes back the result. WSDL defines four operation modes, notifications, one-way, solicit-response, and request-response, each of which corresponds to a particular combination of output and input messages (e.g., solicit-response generates an output message and receives a return message). In addition, a WSDL description gives a binding for a service, specifying the actual communication protocol (e.g., SOAP, HTTP, or MIME) and its settings.

The service matching counterpart to WSDL’s service descriptions is UDDI (Universal Description, Discovery and Integration). UDDI describes businesses in terms of physical attributes (name, address, and lists of services), plus extended attributes called TModels that describe services by reference to standard taxonomies such as NAICS (North American Industry Classification System). However, UDDI has been recognized to be severely limited in practice due to the fact that it solely supports keyword searches (in effect, string matching) on its attributes and does not permit any description of actual service semantics.

On the other side, WSDL has also been found too weak to fulfill the higher level tasks that were envisioned for it (Petrie, Genesereth, Bjornsson, Chirkova, Ekstrom, Gomi, Hinrichs, Hoskins, Kassoff, Kato, Kawazoe, Min & Molsin 2003). It does not provide semantics for its operations in any machine interpretable way, merely specifying their names, nor does it specify the relationship (sequences) between different operations.

From this standpoint, WSDL and UDDI are therefore perfectly matched; both rely on correct naming and interpretation of names. As a result, UDDI and WSDL-related technologies descriptions are considered insufficiently powerful to capture the information required for effective tool support for combining services, and extensive research on other methods is being conducted.

Semantic Web (Berners-Lee, Hendler & Lassila 2001) technology offers the vision of Web of resources where each resource is annotated with machine interpretable descriptions. This capability would enable development of intelligent applications to perform a variety of different tasks on the resources automatically. Like any other resource that are part of the Semantic Web, Web services should also be annotated with machine interpretable descriptions - service descriptions that provide insight into the semantics of the service rather than just the standard port/parameter information described above, are referred to as Semantic Web Service. Figure 1 shows the descriptive associations of such a service. Representing the semantics of Web Services and utilizing the described semantic knowledge to develop and use within intelligent applications is an active research domain. Automatic composition of Web Ser-
The primary aims of semantic Web service descriptions are to facilitate automated discovery by semantic matchmaking, as well as to provide automated support for Web service composition (Albert, Henocq & Kleiner 2005, Medjahed, Bougouttaya & Elmargamid 2003, Wu, Parsia, Sirin, Hendler & Nau 2003). Matchmaking is the process of selecting a set of candidate services that matches a given request. Figure 2 shows a simple example of a syntactic and semantic matchmaking process. In this paper we mostly concentrate on the matchmaking process for ease of exposition.

The syntactic candidate selection is the process of finding services with input and output data types matching the request. The syntactic matching process only takes into account the data types of inputs and outputs while making the selection. It may not always be the case that services exist that exactly match the request. Under these circumstances, the matchmaker can be allowed to make non-exact matches such as ‘Plug-In’ and ‘Relaxed’ (Kawamura, Blasio, Hasegawa, Paolucci & Sycara 2004). As a consequence, the matching process may result in a number of false positives. For example in Figure 2, the ‘bookInfo’ service is selected as a match for a request which intended to find a dictionary. The undesired result was returned because the input and output data types of the service matched with the request.

Semantic matchmaking processes take into consideration the functionality of the service while selecting candidates. The semantic matchmaking process shown in Figure 2 works under the assumption that all services in the directory are semantically described. For example, even though the input and output data types of the service ‘bookInfo’ matched with the request, the service is not selected because the functionality of the request did not match with the functionality of ‘bookInfo’.

1.2 Motivation

Most of the approaches for semantic service definition and composition use OWL-S to define the service description and to define the ontologies within the service description. However, when applied to practical application scenarios, direct use of OWL-S is still subject to a number of restrictions due to the in-progress nature of the approach (Balzer, Liebig & Wagner 2004), with issues arising during profile matchmaking and process execution. Rather than suggesting local workarounds to counter these issues we approach this problem from a model based software engineering perspective. In this paper, we examine an approach based on classical software engineering notation and justify the aspects of our approach that addresses the issues raised in (Balzer et al. 2004).

Albert et al. (Albert et al, 2005) used a constrained object model, which was based on the configuration framework by Mailharro (Mailharro 1998), to devise work flow composition as a configuration problem. The constrained object model had a meta model of the components in the workflow, an ontology mapping of data types and a set of composition constraints such as ‘at least one of the inputs of the choice node should be active to pass control to the next node’. Their work demonstrated the effectiveness of using configuration based approaches for composition of services whereas our approach is towards modeling the service descriptions using standard software engineering practices.

1.3 Example Scenario

A fictitious HotelKroneBookingProcess was defined in (Balzer et al. 2004). The booking process in Figure 3 was used to investigate the pitfalls of OWL-S in a practical situation. In this paper we use the same application scenario to examine the aspects of the languages in our approach that addresses the issues. We assume objects, both parameters and intermediate objects, involved in the process can have relationship between them. For example the objects CreditCard and Customer may be linked with belongsTo relationship. We also assume inheritance hierarchies between the objects in the process. For example ShippingAddress inherits from Address.

The rest of the paper is organized as follows. Section 2 is an account of the relevant related work about the role of UML and OCL on the Semantic Web. Section 3 outlines our examination of the aspects of our UML/OCL based approach that addresses the drawbacks of OWL-S identified in (Balzer et al. 2004). In Section 6 we summarize our observations.
2 UML, OCL and the Semantic Web

The Object Constraint Language (OCL) is a part of the UML specification provided by the Object Management Group (OMG). OCL was introduced in UML to define constraints on the objects in a model based architecture and plays a crucial role in expanding the scope of UML models, by expressing complex relationships that are not easily captured through the various UML diagram notations. In some cases, OCL is also used to define business rules within the model. UML and OCL are widely accepted as a part of the object oriented software engineering approach by both the academic and industrial community.

The feasibility of using UML as a graphical definition language in a Semantic Web context has been investigated in (Falkovych, Sabou & Stuckenschmidt 2003), where the focus is placed mainly on solving the modeling problem and the transformation of UML diagrams to Web ontology languages. Drawbacks such as the absence of variables to represent procedural knowledge and the additional type mapping requirements discovered in (Balzer et al. 2004) were not addressed. Composition of services has been discussed in (Skogan, Grønmo & Solheim 2004) where the UML Activity diagrams with minor modifications are applied. UML diagrams were used as a graphical paradigm to assist human comprehension but the service semantics were defined in OWL-S.

2.1 Ontology Representation

Djuric developed a meta model to represent systems centered around ontologies in a standard UML format (Djuric 2004). A UML profile for ontologies enables the modeling of semantic services within the MDA framework. A feasibility study and an approach for translating UML profiles defined in MDA to OWL-S is detailed in (Gasevic, Djuric & Devedzic 2005). A composition approach was proposed in (Gronmo, Jaeger & Hoff 2005) based on the UML profiles defined in (Djuric 2004). The main focus of the work mentioned here is to examine transformation between UML models and languages such as OWL-S. We investigate an approach based on UML and OCL in a practical application scenario, and consider how the approach addresses some of the issues raised in (Balzer et al. 2004).

3 OWL-S vs. UML/OCL

OWL-S does not yet provide means for expressing semantic properties in a standardized way. In particular, the description of effects and pre and post conditions rely on external languages such as Knowledge Interchange Format (KIF) and Semantic Web Rule Language (SWRL). This diversity adversely affects the integration and reuse of formal descriptions, as translation between formalisms in general implies loss of information. In this paper, however, our focus is only to examine a parallel approach for describing Semantic Web Services. In this section, we examine how the use of UML/OCL would address these issues raised in (Balzer et al. 2004).

3.1 Conditions in the Process Model

The absence of variables in OWL-S is a disadvantage (Balzer et al. 2004). Critical procedural knowledge such as the conditions and parameter instance bindings cannot be expressed in OWL-S. Integrating additional concepts to the OWL-S process ontology would bring in to OWL-S the ability to express conditions (Balzer et al. 2004). Figure 4 shows a part of the modified process model proposed in (Balzer et al. 2004). The modified process ontology has a reified concept in the HotelKroneBookingProcess (see Figure 3) can be expressed in the OCL grammar and process instance template. As counterparts to the hasPrecondition and hasEffect properties in OWL-S, OCL has the pre and post aspects that enable qualifying a process described in OCL with Boolean constraints. Generally conditions are strictly bound to a specific instance of parameters. Variables in OCL can be used to identify specific parameter instances. For example a specific instance of the parameter Customer in the HotelKroneBookingProcess can be mapped to a variable currentCustomer and conditions such as currentCustomer.isMember() can be evaluated within condition expressions. The target of the process effect can also be effectively expressed using OCL variables. For example assume accountCharged to be an effect of the HotelKroneBookingProcess (see Figure 3). The target of the effect can be expressed
as a variable account where account is a specific instance of CreditCard which is in turn linked to a specific instance of Customer class. OCL variables can hold specific parameter instances which does away with the limitation that preconditions should implicitly be modeled with the inputs. If the preconditions are modeled with the inputs then conditions involving multiple input objects cannot be expressed. Unlike (Balzer et al. 2004) where the preconditions are modeled with inputs, OCL expressions give the freedom to express conditions such as ‘(Customer credit card is valid) or (Customer is a frequent visitor and Customer has positive credit history)’. Unless refined concepts proposed in the modified process ontology (Balzer et al. 2004) are introduced, OWL-S will not have the expressiveness to represent critical procedural knowledge. On the other hand, OCL has sufficient expressiveness to represent procedural formalisms such as conditions and parameter instances (if desired).

3.2 Grounding

Balzer et al. noted that a different XSLT sheet would have to be developed for every serialization of an OWL type into XML Schema, making serialization of the OWL types into XML Schema using XSLT problematic (Balzer et al. 2004). They developed a semantic RDF mapping ontology to overcome this. In the mapping ontology an RDF class XSDType is used to map the corresponding OWL type. ComplexType, SimpleType and ArrayType are the three sub classes of XSDType class. The attributes of the OWL types are mapped using the mapsTo property of SimpleType class. UML can be used as an intermediate transformation language while transforming OWL to WSDL (Ha & Lee 2006). However, the transformation of OWL types into UML also suffers the earlier mentioned XSLT serialization problem.

Figure 5 shows a snippet of the OWL types Customer and ShippingAddress. The figure also shows, in our approach, the UML equivalents of these OWL type definitions. XML Metadata Interchange (XMI) is the standard XML-based format to interchange metadata such as class schema information. Strict serialization of UML classes can be configured in XMI. Once configured, for a given UML type there could possibly only be one serialization. This serialized structure can be mapped to either its equivalent RDF class or it can be mapped to a complexType in XML schema. Figure 5 also shows our mapping of UML type Customer into a complexType Customer. This type definition in XML schema can be incorporated into WSDL types and eventually into WSDL messages. The message buyingProcessInput in Figure 5 consists of a part whose data type is complexType Customer. Djuric developed a profile at the UML meta layer to encompass various ontology related operators, demonstrating the capabilities of the UML language to represent ontologies defined using OWL (Djuric 2004). This work provides evidence that OWL data type definitions with complex characteristics can be expressed using UML.

4 System Architecture

Existing processes can be adopted to synthesize knowledge base for configuration from UML and OCL specifications (Felfernig, Friedrich, Janmach & Zanker 2002). Information derived from UML diagrams can be used to create knowledge bases for configuration on the Semantic Web (Felfernig, Friedrich, Janmach, Stumptner & Zanker 2002a, Felfernig, Friedrich, Janmach, Stumptner & Zanker 2002b). Figure 6 shows our system architecture based on afore mentioned findings. The ‘Meta layer’ in the diagram shows the key components of the system. The ‘Instance layer’ shows some examples of the instances that fall under specific category. The ArgoUML® open source UML modeling tool is used to model the service descriptions. Figure 6 shows the complex constraint (introduced in Section 3.1) modeled in OCL. The constraint instance in KB, shown in the figure, is modeled using ILOG JConfigurator syntax (as specified in (Albert et al. 2005), (Felfernig, Friedrich, Janmach & Zanker 2002)). The translation from UML/OCL models to XMI can be achieved using the ‘Export’ feature of the ArgoUML tool. The OCL expressions are contained in the XMI file in OCL syntax, which can be easily converted to the input language of a constraint reasoner such as ILOG JConfigurator (Albert et al. 2005). The actual composition is then found by the configurator as demonstrated in (Albert et al. 2005).
In this paper, we have examined the use of UML and OCL for the semantic description of service specifications. We compare the approach to the widely suggested use of Semantic Web technology to support or automate composition or matching tasks in a service-oriented environment. Our approach is based on existing proposals to use UML as an ontology language (Djuric 2004, Gasevic et al. 2005) but the extension to include OCL specifications results in higher expressiveness than traditional Semantic Web formalisms that have shown to exhibit a number of drawbacks (Balzer et al. 2004). The resulting specifications also lend themselves directly to the composition task with constraint-based reasoning engines (Albert et al. 2005, Felfernig, Friedrich, Jannach & Zanker 2002). They therefore combine ease of use, familiarity with traditional notation, and expressive power in one approach.

References


5 UML/OCL and Semantic Web Protocol Stack

Earlier work has shown the possibility of transforming UML models into particular styles of Semantic Web specifications. For example, Djuric et al. have investigated the vision of Modeling Spaces (MS) (Djuric et al. 2005) and subsequently established the overlap of Technological Spaces (TS) (Gasevic, Djuric, Devedzic & Damjanovic 2004) between the Model Driven Architecture (MDA) and the Semantic Web. A transformation map between the UML Meta Object Facility (MOF) and RDF(S) within their respective MS was provided in (Djuric et al. 2005). This transformation can be used to map UML models into RDF. RDF Schema can then be used to integrate component specifications (Korthaus, Schwind & Seedorf 2005) defined using UML. However, RDF(S) is too restricted to represent the semantics such as constraints. OCL of course, along with UML are the OMG’s intended standard for component specification and can do this with clarity and consistency.

In our approach, the UML Profile defined in (Djuric 2004) is used to define the structural and non-functional properties of a Web Service, while OCL is used for developing semantic descriptions of the functionality, pre conditions, effects of Web Services. The position of UML/OCL in the Semantic Web protocol stack is shown in Figure 7. In Section 3 we examined the aspects of UML and OCL that exhibits better expressiveness, in comparison with OWL-S under certain circumstances, of procedural knowledge and datatype mapping. UML and OCL have sufficient expressiveness to represent critical knowledge that cannot be expressed in OWL-S unless significant changes are adapted in OWL-S ontologies. However, in terms of ontology representations and reasoning capabilities OWL-S clearly shows potential. We acknowledge that OWL-S with suitable modifications will be a front runner for semantic web service descriptions. On the other hand, UML and OCL are widely accepted as a part of the object oriented software engineering approach by both the academic and industrial community. We expect that using standard practices, which have been actively used in software development, will ease the effort for development of ontologies and semantic descriptions for Web Services thereby encouraging development of more ontologies for the Semantic Web by businesses.

Figure 7: Semantic Web Protocol Stack: UML/OCL


