Layered Ontological Modelling for Web Service-oriented Model-Driven Architecture

Claus Pahl

Dublin City University
School of Computing
Dublin 9, Ireland
cpahl@computing.dcu.ie

Abstract. Modelling is recognised as an essential activity in the architectural design of software systems. Model-driven architecture (MDA) is a framework implementing this idea. Ontologies are knowledge representation frameworks that are ideally suited to support modelling in this endeavour. We propose here a layered ontological framework that addresses domain modelling, architectural modelling, and interoperability aspects in the development of service-based software systems. We illustrate the benefits of ontological modelling and reasoning for service-oriented software architectures within the context of the Web Services.
Keywords: Model-Driven Architecture, Service-Oriented Architecture, Web Services, Modelling, Ontologies.

1 Introduction

Modelling has been recognised as an important aspect in the development of software architectures. Model-driven architecture (MDA) – a development framework proposed by the Object Management Group (OMG) – reflects this view [1]. MDA supports the development of component- and service-based software systems through modelling techniques based on notations such as UML. In particular service-oriented architecture (SOA) with its focus on the distributed development and deployment based on Internet and Web technologies requires an explicit representation of models [2, 3]. Global software development (GSD) is another approach gaining importance recently that requires explicit, sharable models and descriptions in order to facilitate collaboration between developers on an abstract level as well as services on an implementation platform. Service-based platforms combined with the wide acceptance of Web technologies provide here suitable support. These contexts also necessitate a higher degree of reliability and dependability, which requires more rigour in the development activities. Formal reasoning is often required to automate development processes.

We will look here at the Web Services Framework (WSF) in particular as our platform [4, 5]. The WSF is a service-based platform based on Internet- and Web-specific description languages, protocols, and core services. Modelling and describing services is essential for both providers and clients of services due to the distributed, inter-organisational nature of service-based development and deployment [6–8]. Sharing and reuse of models is a prerequisite for globally distributed software development.

Ontologies and ontology-based modelling have been proposed to enhance classical UML-based modelling. The essential benefits of ontologies are, firstly, interoperability and sharing and, secondly, advanced reasoning. Ontologies are sharable, extensible knowledge representation frameworks. They can also provide a further improvement on reasoning capabilities available in UML-extensions such as the Object Constraint Language (OCL) [9]. They can capture a wider range of functional and non-functional properties. Ontologies are consequently an ideal technology to support modelling for SOA and GSD. The potential of ontologies has already been recognised in MDA. The OMG has started the development of an Ontology Definition Metamodel (ODM) that can support ontological modelling for the MDA model layers [10]. The combination of ontologies and MDA has also been investigated in academic research, where the Web Ontology Language (OWL) has been integrated into MDA [11, 12].

We propose an extension to these approaches. A layered, ontology-based modelling framework for software services can address the requirements of service-based and globally distributed software development. Similar to UML, which consists of different diagrammatic notations that address different modelling aspects, our framework will combine diferent ontology-based modelling techniques. The three different layers of the MDA framework – computation-independent, platform-independent, and platform-specific – shall be supported by three specific ontology techniques addressing the central concerns of these layers. These concerns are computation-independent domain modelling, platform-independent architectural configuration, and platform-specific service modelling. Architectural configuration is a central activity in software architecture. Service configurations and processes shall play a central role in our architecture framework.

We start with a short introduction to MDA, Web services, and ontologies in Section 2. In Section 3, we outline our ontology-based MDA framework for Web services. In the subsequent sections we address each layer separately – domain modelling in Section 4, architectural configuration in Section 5, and service implementation in Section 6. We end with a discussion of related work and some conclusions. We will use a Web-based e-learning system that we have developed and used over a couple of years as our case study. This system is currently being re-engineered based on a Web service-based architecture.

2 Services and Ontologies

Our objective is to open MDA to the Web Services Framework as the platform. Ontologies and Semantic Web technologies shall serve as the modelling approach.

2.1 Model-Driven Architecture

Model-driven architecture (MDA) is a software architecture framework that emphasises modelling as a central task in the development process of software systems [1]. MDA distinguishes three model layers:

– The computation-independent model layer (CIM) focusses on computation-independent aspects, i.e. modelling the domain-specific context of a system.
– The platform-independent model layer (PIM) aims to define a system in terms of computational abstractions. Typically, a computational paradigm or an abstract machine can form the basis of this modelling approach.
– The platform-specific model layer (PSM) consists of a platform model addressing the concepts and core services of the platform and an implementation-specific model addressing the concrete service architecture implementation.

MDA is typically applied useing UML for platform-independent modelling and considers CORBA as the platform with IDL for the platform-specific description.

2.2 Web Services

The Web Services Architecture WSA defines a Web service as a software system identified by a URI, whose public interfaces are defined and described using XML. Other systems can interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols. MDA is targeted towards service-based software systems, but the Web services platform was originally not considered. We will make the Web services platform our focus here, with service-oriented architecture (SOA) as the generic platform, the Web Services Framework as the technology-specific platform, and vendor-specific technologies such as service engines for SOAP-based service invocation forming the concrete platform infrastructure [5].

While first-generation Web service technology focussed on the use of services 'as-is' in single request-response interactions, the next generation of the Web services platform is more development-oriented [13]. The composition of services to processes is a major concern in current Web service research [14–17]. Service description and service discovery in repositories are essential elements of service development. These recent developments in the context of Web services have strengthened the importance of architectural questions. Behaviour and interaction processes are central modelling concerns for service-based software architectures. Ontology-based MDA can, as we will see, provide an ideal framework for this type of development support.

2.3 Ontologies

The Semantic Web is an initiative that aims to bring meaning to the Web [18, 19]. Ontologies plays the central role in this endeavour. Ontologies are knowledge representation frameworks that allow knowledge to be shared. They combine terminological aspects with a formal logic. Ontologies usually consist of hierarchical definitions of important concepts of a domain and the description of properties of these concepts in terms of other concepts. An ontology is a model of a domain made available through the vocabulary of concepts and relationships.

Ontologies have already been used to support software engineering activities [20]. They have been exploited to support the annotation of Web services within

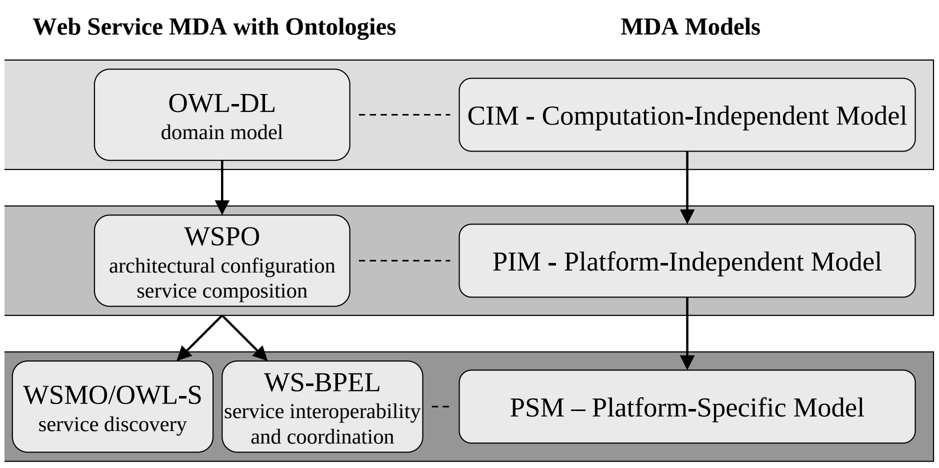
Web Service MDA with Ontologies
MDA Models
OWL-DL
domain model
CIM - Computation-Independent Model
WSPO
architectural configuration
service composition
PIM - Platform-Independent Model
WSMO/OWL-S
service discovery
WS-BPEL
service interoperability
and coordination
PSM – Platform-Specific Model

Fig. 1. Overview of the Ontology-based MDA Framework for Web Services.

the context of semantic Web services [21]. Ontologies are used to capture a variety of functional and non-functional properties of services (the terminological aspect of ontologies) and to retrieve matching provided services from repositories based on a client's requirements specification (the logical aspect of ontologies).

Ontology languages such as OWL are defined based on description logic, which allows the integration of formal reasoning with ontology-based modelling [22]. Description logics are particularly interesting for the software development context due to a correspondence between description logic and modal logics. Modal logics such as dynamic and temporal logics have been used extensively in the behavioural specification of software systems [23]. Dynamic logic forms a framework that captures the pre- and postcondition technique used in design-by-contract approaches [24] and specification languages such as OCL [9].

3 A Layered Ontological Modelling Framework

3.1 Modelling

MDA supports the architectural design of software systems. It integrates domain engineering with software architecture. MDA proposes a three-layered modelling framework addressing computation-independent, platform-independent, and platform-specific aspects. In our Web Services context, we have identified three central concerns that we can map to the MDA layers:

– Domain modelling is a concern that is independent of a concrete computational paradigm. Capturing the domain context of a service is essential for SOA as providers need to document the features of a service.

– Architectural configuration on a platform-independent level is important since service integration and composition are the central SOA activities.
– Modelling services within the given platform technology is important since service models have to be provided for discovery and service deployment in architectures and processes.

Our modelling framework – see Fig. 1 – consists of ontologies for all three layers. We will demonstrate that ontologies can address the concerns that have led to the definition of different model layers. For the Web services platform, an explicit sharable representation of these models for all layers is a requirement.

3.2 Logic and Semantics

Ontologies can address a variety of problems ranging from domain modelling to architectural configuration and semantic service description. The richness of an ontology language such as OWL-DL allows these different ontologies for different purposes to be developed [18].

In addition to providing notational modelling solutions for each of the concerns through different ontologies, ontology technology provides also the added benefit of enabling a uniform semantic framework for all three layers. Description logic is the formal foundation of many ontology languages, including the Web Ontology Language OWL.

The current work towards an Ontology Definition Metamodel (ODM), that has already been started by the OMG, will, once finished, provide an MOF-based semantic framework. We will discuss this aspect later on.

4 Computation-Independent Domain Modelling

4.1 Modelling Concern

The focus of the computation-independent modelling layer is the capture of domain properties. Here, often two viewpoints are distinguished. The information viewpoint captures structural aspects of information in form of concept hierarchies. The enterprise viewpoint looks at the behaviour and processes in a system. We add a third viewpoint addressing the structural aspects through compositional relationships.

4.2 Ontological Modelling

Ontologies consist of two basic entities – concepts of a domain and relationships between these concepts that express properties of one concept in terms of another concept. Classical ontologies relate concepts in a subclass hierarchy, which creates a taxonomy for a particular domain.

A single ontological notation, for example based on the Web Ontology Language OWL, can support the three viewpoints of the CIM layer.

– Two kinds of concepts – objects representing static information and processes presenting dynamic behaviour – can be distinguished in an ontology.
– The set of relationships shall comprise a subclass relationship for concept hierarchies (information viewpoint), a dependency relationship (enterprise viewpoint), and a component relationship (the structure viewpoint for both objects and processes).

The choice of relationship types here is critical to address the needs of process-centric domain modelling. Although, the concern here is domain modelling, domain activities and processes are central as they often form the starting point for further detailled models. Dependency relationships express how information objects are processed by process entities. Composition is important for both objects and processes.

We have illustrated computation-independent modelling in Fig. 2. In addition to the classical information viewpoint based on classification hierarchies and the enterprise viewpoint focusing on processes, we have also included a structural viewpoint addressing the compositional structure of objects and processes. Objects are elliptic entities such as learning object or assessment object. Processes are rectangular entities such as learning activity or evaluation. These entities – concepts in an ontology – are represented from the three viewpoints.

Other properties, such as sequencing constraints on processes can also be expressed in addition to concepts and relationships. Iteration, choice, concurrency, or sequence are process combinators that are often better expressed in a separate sublanguage. For instance, individual activity steps of a learning activity could be sequenced using additional constraints.

4.3 Ontological Reasoning

The reasoning capabilities of an ontological framework can be utilised in different ways for this form of domain modelling:

– The internal consistency of a model can be checked. For instance, cyclical definitions in concept taxonomies can be recognised.
– Inference rules can also be used to query an ontology. For instance, rules about transitional process behaviour (based on dependency relationships) can be used to determine the input/output behaviour of composite processes.

The description logic on which an ontology language like OWL-DL is based provides the formal framework here [22].

In the context of the e-learning example, an inference engine can be used to compile all prerequisites of a sequence of learning activities. It could also be used to check whether the learning outcomes of the first activity in a sequence satisfy the prerequisites of the next activity.

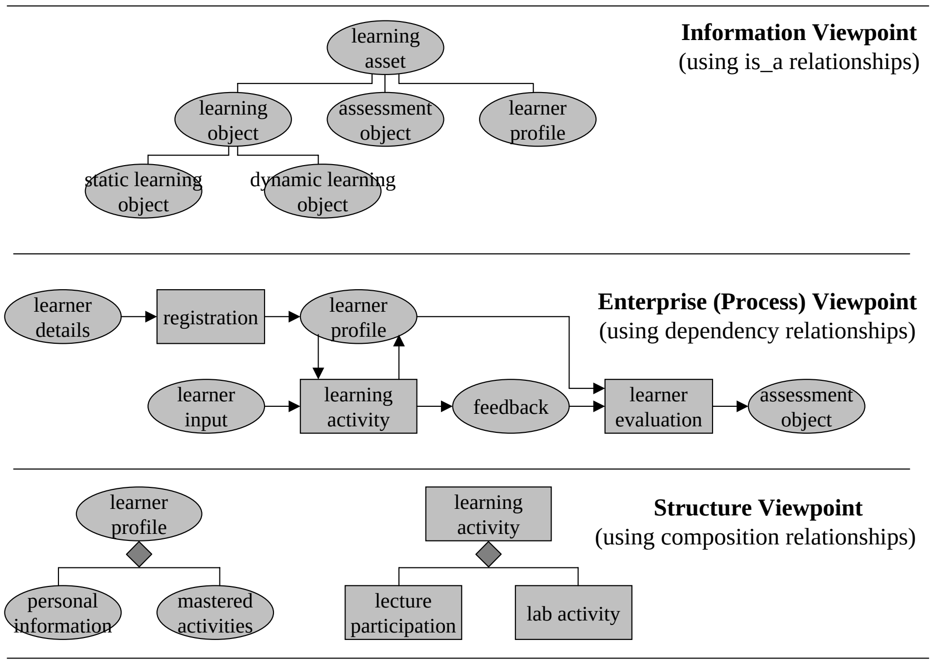

Fig. 2. CIM-level Excerpts from an E-Learning Domain Ontology.

5 Platform-Independent Architecture Modelling

5.1 Modelling Concern

Platform-independent modelling introduces a computational paradigm into the modelling process. Service-oriented architecture is this paradigm for the Web Services Framework. In the context of service-based software, the architectural design of a software system is of central importance. Behaviour and service processes are part of the architectural configuration of a system [15]. Architectural configuration addresses the interaction processes (remote invocation and service activation) between different services in a software system.

5.2 Ontological Modelling

Various service ontologies exist [25]. WSPO – the Web Service Process Ontology – can be distinguished from other service ontologies such as OWL-S [21] and WSMO [26] through its process-orientation [27–29]. In WSPO, the focus is on the behaviour of software systems. Relationships of the ontology are interpreted as accessibility relations between system states. This is in fact an encoding of a (modal) dynamic logic in a description logic format [30]. WSPO ontologies are based on a common template, see Fig. 3:

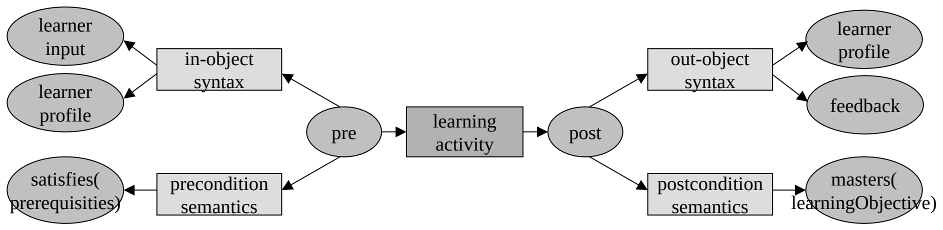

Fig. 3. Ontological Service Process Template (WSPO) – applied to E-Learning.

– The central concepts are the system states – pre- and poststates of state transition-based services. Other concepts capture service parameters (in- and out parameters) and conditions (such as pre- and postconditions).
– Two forms of relationships characterise this ontology. The central relationship type is the service or service process itself. These so-called transitional relationships are enhanced by a process combinator sublanguage comprising the operators sequence, choice, iteration, and parallel composition. This relationship sublanguage allows process expressions to be formulated. Auxiliary relationship types are so-called descriptional relationships, which associate the auxiliary concepts to the states.

The architecture- and process-oriented PIM model of the e-learning example focuses on the activities and how they are combined to processes, see Fig. 3. The process combinators that we used to model the e-learning activity service are ';' (sequential composition), '!' (iteration), '+' (choice), and '||' (parallel composition). The operators are interpreted by their usual set-theoretic semantics, e.g. iteration is defined as the transitive closure of the relation that interprets the argument and the non-deterministic choice is interpreted by the union of both argument interpretations. The symbol $\circ$ is used to denote the application of a process to a given list of parameters. The process

$$lecture;\ !(\ labExercise1 + labExercise2\ );\ selfAssessment$$

describes a sequence of lecture participation, an iteration of a choice of two lab exercises, and a final self-assessment. This can be represented in WSPO as a composed relationship expression:

$$\begin{array}{l} lecture \circ profile; \\ !\ (\ labExercise1 \circ (profile, input1);\ labExercise2 \circ (profile, input2)\ ); \\ selfAssessment \circ profile \end{array}$$

While architecture is the focus of this model layer, the approach we discussed does not qualify as an architecture description language (ADL) [31], although the aim is also the separation of computation (within services) and communication (interaction processes between services). ADLs usually provide notational means to describe components (here services), connectors (channels between services), and configurations (the assembly of instantiations of components and connectors). Our approach comes close to this aim by allowing services as components and process expressions as configurations to be represented.

5.3 Ontological Reasoning

The close link to modal logic allows modal reasoning about reactive systems to be incorporated.

- Dynamic logic, for instance, incorporates pre- and postcondition-based reasoning. Matching between client requirements and service properties in terms of abstract functional behaviour can be decided using this technique. This link also allows the integration of a refinement technique.
- The process expression language in WSPO is enhanced by supporting behavioural theory from temporal logics and process calculi. A notion of simulation allows process expressions to be compared. This can be used in matching.

Matching is a common problem that needs to be addressed if a new component, such as a service or process here, has to be embedded into a given context. For any given state, the process developer might require

$$\begin{array}{l} \forall preCond\ .\ (profile.masteredActivities \in activity.prerequisite) \\ \forall \textit{learning activity}\ .\ \forall postCond\ . \\ \qquad\qquad\qquad (activity.objective \in profile.masteredActvities) \end{array}$$

which would be satisfied by a provided service

$$\begin{array}{l} \forall preCond\ .\ true \\ \forall \textit{learning activity}\ .\ \forall postCond\ . \\ \qquad\qquad\qquad (activity.objective \in profile.masteredActvities) \wedge \\ \qquad\qquad\qquad (typeof(lastActivity) =\ \textit{'labExercise'}) \end{array}$$

based on a refinement condition (weakening the precondition and strengthening the postcondition). The dot-notation used in the conditions refers to a component of the object. Quantified expressions are used to express these safety conditions. The postcondition in this example states that by carrying out the activity, the intended activity objective is accomplished and can therefore be added to the mastered activities of the learner in her/his profile.

5.4 Transformation

Transformations between the layers are crucial. A high degree of tool support and automation is necessary for an MDA framework in general. Although a detailed discussion of transformations in our framework is beyond the scope of this paper, we will outline the principles here. We have investigated transformations for this framework in more detail in [29].

The CIM-to-PIM transformation involves the mapping of a domain ontology into a process-centric service ontology. The following steps need to be considered:

– Service identification. In our case, the domain model is already service-oriented and services are clearly marked.
– Model mapping. We have defined a WSPO ontology template applicable to a single or a composed service. For the composed service processes, the actual process description comes from a separate constraint. The instantiation of this template leads to a service-specific ontology.

6 Platform-Specific Service Modelling

6.1 Modelling Concern

Platform-specific modelling (PSM) relates the previous layers to the concrete constraints of the chosen platform. It usually consists of two models – the platform model that describes the specifics of the platform and the implementation specific model that captures the essentials of the implementation languages.

6.2 Ontological Modelling

The platform here is the Web Services platform, on which a number of different languages are used. We will look at two of these languages here:

– Development support: We have chosen WSMO as one of the widely discussed and used service ontologies that aims at describing a service on an abstract level. WSMO incorporates some of the functional behaviour specification of WSPO, but also provides support for a wide range of non-functional properties, see Fig. 4. The aim of WSMO is to provide an interoperable form for the semantic description of services to support their discovery in repositories.
– Deployment support: Another aspect of service-oriented architecture is service composition – often the term service collaboration is used to indicate the distributed nature of service architectures. WS-BPEL is a business process description language that supports Web service orchestration (collaboration described from a local services' point of view). WSPO already captures the essentials of service and process interaction. This can easily be translated into WS-BPEL specifications.

Both are service-centric implementation languages. Interoperability is a central issue in both cases.

The learning activity service that we have focussed on in our case study could both be published (using WSMO) and integrated in a service process orchestration (using WS-BPEL):

– WSMO descriptions capture syntactical and semantic service descriptions. In this way it is similar to WSPO. It adds, however, various non-functional aspects that can be included into the discovery and matching task. We have added two non-functional properties to the learning activity descriptions – the location as an interface-related aspect and the security infrastructure as a capability issue, see Fig. 4.

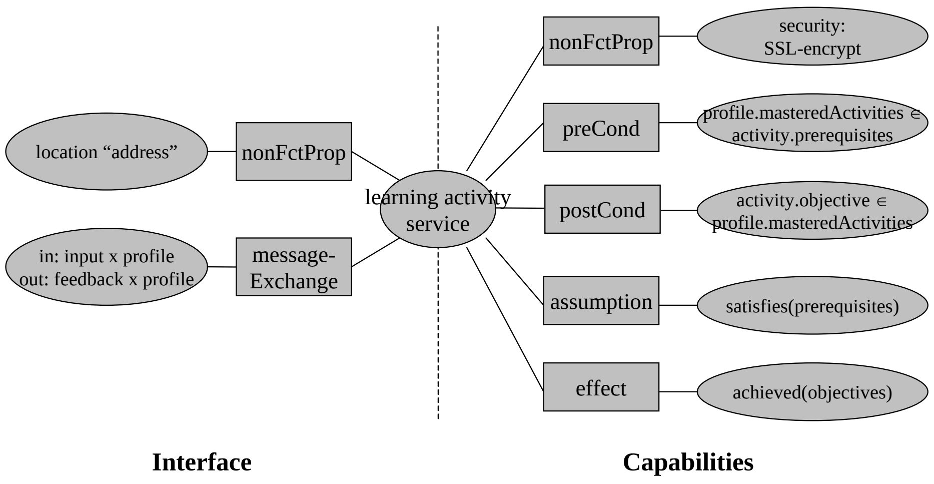

Fig. 4. Ontological Service Template (WSMO) – applied to an E-Learning Context.

Standardised description and invocation formats enable interoperability. A key objective is the provision of services. For instance, the learning activity could be advertised as a reusable content service in a learning technology repository. Required functionality can be retrieved from other locations. An example are registration and authentication features.

– WS-BPEL is an XML-based notation, based on an operator calculus similar to ours. Based on simple actions <`action`>, which describe simple request or response interactions, combinators such as <`sequence`> or <`flow`> can be used to define orchestrated service processes. Orchestration is an internal perspective on process assembly and interaction.

6.3 Ontological Reasoning

Again, a formal ontology basis enables further forms of reasoning. For the WSMO context, the matching notion can be extended to comprise a variety of functional and non-functional properties. The main aim of OWL-S and WSMO is the improved support of semantic Web service description and discovery compared to syntactical formats such as the Web Services Description Language (WSDL).

For WS-BPEL, classical process calculus-based analyses, e.g. a deadlock analysis, can be addressed. Although not part of WS-BPEL itself, a formulation of WS-BPEL using a process calculus would enable these analyses.

6.4 Transformation

The PIM-to-PSM transformation encompasses two mappings for the two different platform languages we support:

– The WSPO-to-WSMO mapping extracts information specific to individual services from a WSPO model. This comprises the syntactical elements (in- and out-objects) and the semantic information (pre- and postconditions).
– The WSPO-to-WS-BPEL extracts the process definitions and converts them into BPEL process expressions. It also uses the syntactical information to define the individual services.

Both mappings can create skeletons with partial information that would need to be completed by a software developer. While transformations are essential, we will not discuss them here – see [29] for a more detailed investigation.

7 Related Work

WSMO [26] and OWL-S [21] are the two predominant examples of service ontologies. Service ontologies are ontologies to describe Web services, aiming to support their semantics-based discovery in Web service registries. WSMO is not an ontology, as OWL-S is, but rather a framework in which ontologies can be created. We have used WSMO here to illustrate issues for a Web service platform-specific modelling approach. The Web Service Process Ontology WSPO [27, 28], which we have used for platform-independent modelling, is also a service ontology, but its focus is the support of description and reasoning about service composition and service-based architectural configuration. Both OWL-S and WSPO are or can be written in OWL-DL. WSMO is similar to our endeavour here, since it is a framework of what can be seen as layered ontology descriptions. We have introduced technical aspects of WSMO descriptions in Section 6. WSMO supports the description of services in terms more abstract assumptions and goals and more concrete pre- and postconditions.

Some developments have started exploiting the connection between OWL and MDA. In [32], OWL and MDA are integrated, i.e. an MDA-based ontology architecture is defined. This architecture includes aspects of an ontology metamodel and a UML profile for ontologies. A transformation of the UML ontology to OWL is implemented. The work by [11, 32] and the OMG [1, 10], however, needs to be carried further to address the ontology-based modelling and reasoning of service-based architectures. In particular, the Web Services Framework needs to be addressed in the context of Web-based ontology technology.

Our framework has to be looked at in the context of the MDA initiatives by the OMG. The OMG supports selected modelling notations and platforms through an adoption process. Examples of OMG-adopted techniques are UML as the modelling notation and CORBA as the platform [1]. While Web technologies have not adopted so far, the need for a specific MDA solution for the Web context is a primary concern. The ubiquity of the Web and the existence of standardised and accepted platform and modelling technology justify this requirement. The current OMG initiative to define and standardise an ontology metamodel (ODM) will allow the integration of our framework with OMG standards [10]. ODM will provide mappings to OWL-DL and also a UML2 profile for ontologies to make the graphical UML notation available. This might lead to a 'Unified Ontology Language' in the future, i.e. OWL in a UML-style notation [12]. A UML2 profile is about the use of the UML notation, which would allow ontologies to be transformed into UML notation. MOF2 compliancy for ODM is requested to facilitate tool support. XMI, i.e. production rules using XSLT, can be used to export model representations to XML, e.g. to generate XML Schemas from models using the production rules. The ontology definition metamodel (ODM) would allow an integration with UML-style modelling due to its support of OWL. ODM, however, is a standard addressing ontology description, but not reasoning. The reasoning component, which is important in our framework, would need to be addressed in addition to the standard.

8 Conclusions

Ontology technology offers a range of benefits for modelling activities in the MDA context.

- The formal definition based on description logics allows extensive reasoning to be used.
- Ontologies are sharable knowledge representation formats. Ontologies can easily be modified and extended.

Ontologies combine a terminological framework with a logical framework. It is this combination that we have used to enhance classical modelling techniques.

The benefits match in particular the requirements of a platform such as the Web Services Framework, where often globally distributed software development is the main style of development that relies on interoperable data formats and dependable service architectures. In heterogeneous environments and cross-organisational development, information about a variety of service aspects – as it can be represented in ontologies – is vital.

MDA defines a development process, addressing different concerns at each stage. We have identified process-oriented domain modelling, architectural configuration, and service implementation modelling as the three central concerns for the development with the Web Service Framework as the platform. We have demonstrated that ontology technology can provide an integrated, coherent solution for these concerns at all three modelling layers.

CORBA and UML are OMG-adopted technologies. The adoption process provides OMG support for a particular technology, either a platform or language. Web technologies have not been adopted so far. However, the ubiquity of the Web will require a solution in the future. The current OMG attempt to define an ontology definition metamodel ODM that includes mappings to OWL-DL and also a UML profile for ontologies is a first step integrating OMG with Web technologies.

We have neglected one central problem of a Web ontology-based MDA approach. Transformations between the individual layers need to be defined and, to a high degree, automated in order to make MDA feasible. The definition of a transformation framework is beyond the scope of this paper. Two transformation steps have to be addressed. Both are transformations between different ontologies. We have devised a graph-based transformation centred around the service and process elements; see [29] for more details.

References

1. Object Management Group. *MDA Guide V1.0.1.* OMG, 2003.
2. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services – Concepts, Architectures and Applications.* Springer-Verlag, 2004.
3. E. Newcomer and G. Lomow. *Understanding SOA with Web Services.* Addison-Wesley, 2005.
4. World Wide Web Consortium. *Web Services Framework.* http://www.w3.org/2002/ws, 2004. (visited 08/04/2005).
5. World Wide Web Consortium. *Web Services Architecture Definition Document.* http://www.w3.org/2002/ws/arch, 2003.
6. The WS-BPEL Coalition. *WS-BPEL Business Process Execution Language for Web Services – Specification Version 1.1.* http://www-106.ibm.com/developerworks/webservices/library/ws-bpel, 2004. (visited 08/04/2005).
7. C. Peltz. Web Service orchestration and choreography: a look at WSCI and BPEL4WS. *Web Services Journal*, 3(7), 2003.
8. D.J. Mandell and S.A. McIllraith. Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In D. Fensel, K.P. Sycara, and J. Mylopoulos, editors, *Proc. International Semantic Web Conference ISWC'2003*, pages 227–226. Springer-Verlag, LNCS 2870, 2003.
9. J.B. Warmer and A.G. Kleppe. *The Object Constraint Language – Precise Modeling With UML.* Addison-Wesley, 1998.
10. Object Management Group. *Ontology Definition Metamodel - Request For Proposal (OMG Document: as/2003-03-40).* OMG, 2003.
11. D. Gašević, V. Devedžić, and D. Djurić. MDA Standards for Ontology Development – Tutorial. In *International Conference on Web Engineering ICWE2004*, 2004.
12. D. Gašević, V. Devedžić, and V. Damjanović. Analysis of MDA Support for Ontological Engineering. In *Proceedings of the 4th International Workshop on Computational Intelligence and Information Technologies*, pages 55–58, 2003.
13. J. Williams and J. Baty. Building a Loosely Coupled Infrastructure for Web Services. In *Proc. International Conference on Web Services ICWS'2003.* 2003.
14. R. Allen and D. Garlan. A Formal Basis for Architectural Connection. *ACM Transacions on Software Engineering and Methodology*, 6(3):213–249, 1997.
15. F. Plasil and S. Visnovsky. Behavior Protocols for Software Components. *ACM Transactions on Software Engineering*, 28(11):1056–1075, 2002.
16. L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice (2nd Edition).* SEI Series in Software Engineering. Addison-Wesley, 2003.
17. N. Desai and M. Singh. Protocol-Based Business Process Modeling and Enactment. In *International Conference on Web Services ICWS 2004*, pages 124–133. IEEE Press, 2004.
18. W3C Semantic Web Activity. Semantic Web Activity Statement, 2004. http://www.w3.org/2001/sw. (visited 06/12/2004).
19. M.C. Daconta, L.J. Obrst, and K.T. Klein. *The Semantic Web.* Wiley, 2003.

20. M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In I. Horrocks and J. Hendler, editors, *Proc. First International Semantic Web Conference ISWC 2002*, LNCS 2342, pages 279–291. Springer-Verlag, 2002.
21. DAML-S Coalition. DAML-S: Web Services Description for the Semantic Web. In I. Horrocks and J. Hendler, editors, *Proc. First International Semantic Web Conference ISWC 2002*, LNCS 2342, pages 279–291. Springer-Verlag, 2002.
22. F. Baader, D. McGuiness, D. Nardi, and P.P. Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
23. D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 789–840. Elsevier, 1990.
24. Bertrand Meyer. Applying Design by Contract. *Computer*, pages 40–51, October 1992.
25. T. Payne and O. Lassila. Semantic Web Services. *IEEE Intelligent Systems*, 19(4), 2004.
26. R. Lara, D. Roman, A. Polleres, and D. Fensel. A Conceptual Comparison of WSMO and OWL-S. In L.-J. Zhang and M. Jeckle, editors, *European Conference on Web Services ECOWS 2004*, pages 254–269. Springer-Verlag. LNCS 3250, 2004.
27. C. Pahl. An Ontology for Software Component Matching. In M. Pezzè, editor, *Proc. Fundamental Approaches to Software Engineering FASE'2003*, pages 6–21. Springer-Verlag, LNCS 2621, 2003.
28. C. Pahl and M. Casey. Ontology Support for Web Service Processes. In *Proc. European Software Engineering Conference and Foundations of Software Engineering ESEC/FSE'03*. ACM Press, 2003.
29. C. Pahl. Ontology Transformation and Reasoning for Model-Driven Architecture. In *International Conference on Ontologies, Databases and Applications of Semantics ODBASE'05*. Springer LNCS Series, 2005. (submitted).
30. K. Schild. A Correspondence Theory for Terminological Logics: Preliminary Report. In *Proc. 12th Int. Joint Conference on Artificial Intelligence, Sydney, Australia*. 1991.
31. N. Medvidovic and R.N. Taylor. A Classification and Comparison framework for Software Architecture Description Languages. In *Proceedings European Conference on Software Engineering / International Symposium on Foundations of Software Engineering ESEC/FSE'97*, pages 60–76. Springer-Verlag, 1997.
32. D. Djurić. MDA-based Ontology Infrastructure. *Computer Science and Information Systems (ComSIS)*, 1(1):91–116, 2004.