Ontology-Based Composition and Transformation for Model-Driven Service Architecture

Claus Pahl

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

Abstract. Building service-based architectures has become a major area of interest since the advent of Web services. Modelling these architectures is a central activity. Model-driven architecture is a recent approach to developing software systems based on the idea of making models the central artifacts for design representation, analysis, and code generation. We propose an ontology-based composition and transformation approach for model-driven service architecting. Ontology technology as a logic-based knowledge representation and reasoning framework can provide answers to the needs of sharable and reusable models and descriptions needed for service engineering. Based on UML-style visual modelling of service architectures and their mapping into an ontology representation, our approach enables ontology-based semantic modelling based on representation, analysis, and code-generation techniques for Web services.

Keywords: Service-oriented Architecture, Service Process Composition, Model-Driven Architecture, Service Ontology, Web Services.

1 Introduction

Model-driven architecture (MDA) is an approach to the development of software systems that has gained wide support over the past years [1]. MDA is supported by major standardisation bodies such as the Object Management Group (OMG). MDA emphasises the importance of modelling in the software development process. Detailed models in MDA serve as design specifications that support the maintainability of systems and can also provide the basis for automated code generation. Service-oriented architecture (SOA) [2] is a specific development and platform approach for service engineering that would benefit from a tailored MDA solution in order to realise the MDA objectives.

Our focus is here on central development activities in service-oriented architecture [3,4]. Composition is central in a paradigm that addresses architectures of orchestrated services [2]. Service orchestration refers to the assembly or composition of services to service processes [5]. Within the Web Services platform [3]– which is the concrete platform for service-oriented architecture that we

A. Rensink and J. Warmer (Eds.): ECMDA-FA 2006, LNCS 4066, pp. 198–212, 2006.

target here – the business process execution language WS-BPEL [6] is the most widely used implementation language for service processes.

Services description and composition has been combined with ontology technology [7,8]. Model-driven architecture has been enhanced to ontology-driven architecture [9]. However, the integrated application of both ontology technology and MDA to service architecture has so far not been adequately addressed. We propose an approach for architecting service-based software systems that embraces the MDA-philosophy. UML-based dynamic modelling is the starting point. A tailored UML profile based on activity diagrams provides the modelling notation for service process orchestration. The orchestration of services occurs in two forms. Firstly, the composition of services to processes at the abstract level using process operators. Secondly, the association of concrete provided services that match the requirements of abstract service process elements.

Supporting service engineering using MDA and ontologies is beneficial for the composition activity. For the SOA context, in particular Web services where compositions across organisations and network boundaries are the norm, explicit semantic descriptions of services are a prerequisite for the reliable composition of services [10]. We introduce an ontology framework, i.e. a logic-based knowledge representation framework, to enable sharable representations of semantic service and process descriptions. Various attempts in this direction include service ontologies such as OWL-S [11] and WSMO [12]. The OMG has also recognised the importance of logic-supported semantic modelling using ontologies, which is reflected in the OMG's Ontology Definition Metamodel (ODM) initiative [13]. The need for service providers to publish their services in an accepted, standardised format is another argument in favour of ontologies.

Our solution is UML-based service process modelling supported through a UML profile and a mapping from this profile into a semantic service ontology. The ontology acts as a service architecting engine for both forms of composition and also supports code generation for the process execution and service publication aspects. The aim is to apply the MDA philosophy to a specific software technology. Service-oriented architecture focusses on architectural problems such as process composition through service orchestration and the Web Services platform is characterised by specific languages such as WS-BPEL [6] for service process execution. Therefore, our solution will be dominated by semantic modelling techniques for these specific aspects.

We start with an overview of the service engineering process in our context in Section 2. In Section 3, we introduce UML-based modelling of service processes. Ontology-based composition is the topic of Section 4. We discuss the deployment of service processes in Section 5. We end with related work and some conclusions.

2 Engineering of Service-Based Software Architectures

In [4], a Web service is defined as a software system, whose public interfaces are defined and described using XML. Other systems can interact with the Web

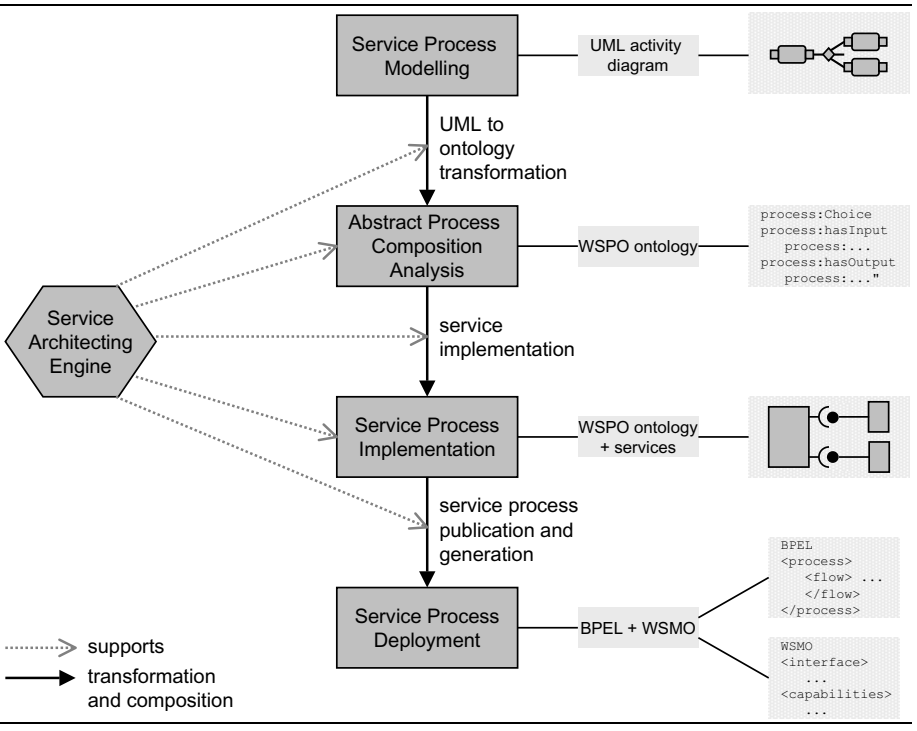

Fig. 1. Overview of the Ontology-based Service Architecture Technique

service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols. The composition of services to orchestrated processes is a major concern in current Web service research [14,15,2]. These recent developments have strengthened the importance of architectural questions such as service composition. Behaviour and interaction processes are central modelling concerns for service-based software architectures. Explicit semantic descriptions and exchangable models enable developers and clients of services to create reliable service architectures using tool support.

We embed our proposed service composition technique into an ontology-supported, MDA-based development approach for the platform-independent layer (PIM). Our service-specific software process model for ontology-driven semantic service architecture is based on the following steps, see Fig. 1:

- Service Process Modelling. This activity is about visual UML modelling of process activities. Activity diagrams with service-oriented semantic extensions form the notation. Individual actions represent services.
- Abstract Process Composition Analysis. The analysis activity part of the process modelling addresses the integrity of a process composition based on semantical model enhancements in an ontological representation; here we use

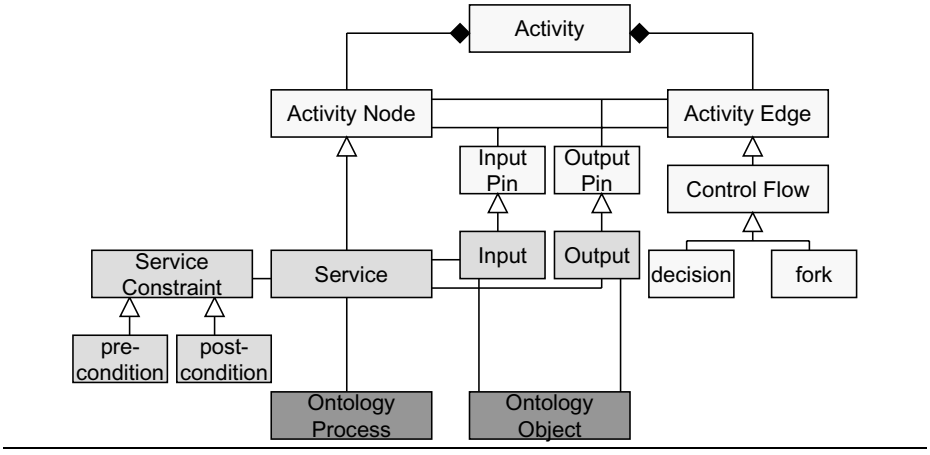

Fig. 2. UML Profile (Metamodel) for Semantic Service Process Modelling

the Web Service Process Ontology WSPO [16,17] – which we will motivate later on – to analyse for instance the integrity of service process definitions.
- Service Process Implementation. The focus of this activity is the discovery of individual services in repositories and directories that match the requirements of the actions specified in the process model. These concrete services can then be associated to the abstract services from the process model.
- Service Process Deployment. The deployment activity enables the implementation of the process as an executable WS-BPEL process [6] based on the associated services. Deployment also includes the publication of the overall process as a service in a service ontology; here we use the Web Service Modelling Ontology WSMO [12].

An ontology-based service architecting engine supports the composition activities within the platform-independent (PIM) layer and also guides the necessary transformations to the Service Deployment, i.e. platform-specific (PSM) layer.

3 Modelling of Service Processes

Service processes are assemblies of individual services or other service processes. This form of service composition is part of what is often called service orchestration [5] – the other aspect of orchestration is the association of concrete services to abstract service placeholders in the composed process description. It describes the control and data flow between services using basic flow operators.

UML activity diagrams capture activities that are to be performed as executable activity nodes in a graph-like structure. The overall system flow based on the activities is modelled. The basic diagram elements are executable activity nodes, called actions, and edges between these activity nodes that represent

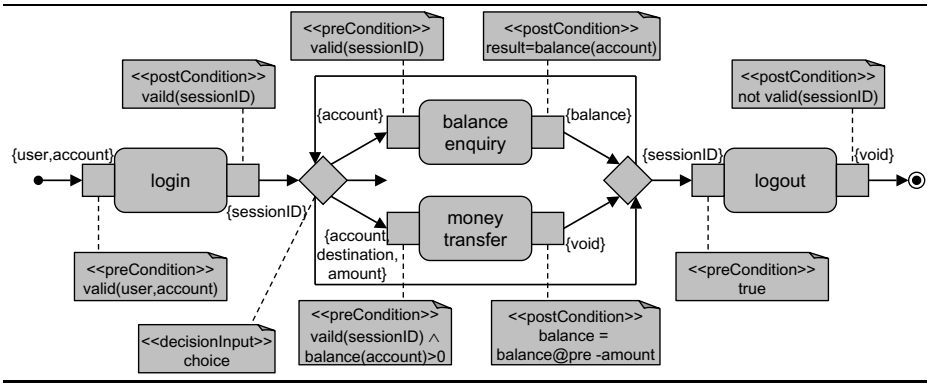

Fig. 3. Semantic Service Process Model based on UML Activity Diagrams

flow. Control flow nodes allow the description of choice (decision) or concurrency (fork) with their joining counterparts. The control flow can be enhanced by explicit objects and input and output pins that represent input and output elements at activities.

We require some extensions to activity diagrams, which we will capture in form of a UML profile, to address the needs of semantic service process description. This metamodel is defined in Fig. 2. White rectangles denote the standard UML activity diagram elements; medium grey ones denote our service-specific extensions; dark grey ones associated elements of a possible domain ontology[1]. A service is an activity. A service's input and output objects are linked to the input and output pins of activities. In addition to input and output elements, we need to add semantic service descriptions, here in the form of pre- and postconditions.

The application of the profile is presented in Fig. 3. It represents an online bank account application. The process of using such an account is described. This application is based on four individual services, each described in terms of input, output, precondition, and postcondition.

The following textual representation summarises the syntactical aspects in an IDL-style service interface format.

application AccountProcess
  service login (user:string, account:int) : ID
  balance enquiry (account:int) : real
  money transfer (account:int, destination:int, amount:real) : void
  logout (sessionID:ID) : void
  process login; !(balance enquiry + money transfer); logout

[1] Often, a domain ontology or model captures central concepts, i.e. key objects and processes, of an application context [16]. The architecture model here is linked to the domain model. Although we do not discuss this aspect further, the integration with a domain model is central for a coherent ontology-based modelling approach.

The services are composed to a process, here using the combinators sequence (;), iteration (!), and choice (+) in the textual representation above, which is also captured in UML notation in Fig. 3[2].

4 Composition of Service Processes

Composition occurs, as already mentioned, in two forms in service architectures:

- Process composition. The assembly of services to processes is the first form of composition. The visual modelling of this composition form is supported by the UML profile. The semantic consistency of the process composition needs to be addressed at a different level.
- Refinement. The composition of the abstract service process as the client and individual concrete services as providers of functionality is the second form where the provider properties refine the required properties. The problem is the discovery of services in service repositories or directories that match the semantic requirements.

Both forms together are usually refered to as orchestration.

We introduce in this section an ontology-based engine to support the architecting of service-based systems based on these two composition forms. The notion of a service architecting engine – emphasising the focus on architecture development – captures semantic properties of services and processes, and supports process- and refinement-style composition. An operator calculus for process composition and inference rules to support matching are integral elements of this engine. As we will see later on, a service composition ontology can also provide the foundations for the generation of deployment code. We start with the composition ontology itself (Section 4.1), before looking at mappings between UML and this ontology (4.2) and process composition (4.3) and implementation (4.4).

4.1 A Service Composition Ontology

A number of service ontologies have been proposed, with OWL-S [11] and WSMO [12] as two prominent examples. The central aim of service ontologies is the semantic annotation of services. While OWL-S also supports service composition to a degree through its process model, we use the Web Service Process Ontology WSPO – whose foundations are presented in [17,18] and which we developed specifically to support service composition and architecture ontologically. WSPO is an OWL-DL (the Web Ontology Language – Description Logic variant) ontology. It uses description logic [19], which also provides the foundation of OWL, to capture composition techniques. WSPO is actually an encoding of a dynamic logic (a modal logic of programs) in a description logic format, which enables reasoning about dynamic service process properties such

[2] We have used iteration as an explicit control flow abstraction here, even though it is not part of the UML notation, since its simplifies expressions on the textual level.

as safety and liveness, making WSPO the most suitable candidate for ontology that supports the MDA PIM layer. The ontology can be used to check the integrity of service process definitions (a safety condition), e.g. determine if the output of a service satisfies the semantic requirements of the next service in the process.

Ontologies are knowledge representation frameworks formalised in an ontology language (such as OWL) [20,21], which is usually based on a terminological logic (such as description logic). Knowledge is represented in form of concepts and (quantified) relationships between these concepts to characterise them semantically. Services (and processes) in WSPO are not represented as concepts, as one might intuitively assume, but as relationships denoting accessibility relations between states of the system. The states are represented as concepts.

- The central concepts in this approach are states (pre- and poststates) for each service. Other concepts are parameters (input- and output-parameters) and constraints (pre- and postconditions).
- Two forms of relationships are provided. The services themselves or their composition to processes are called *transitional relationships*. These processes are based on operators such as sequence, choice (decision), and concurrency (fork) – other operators not present in activity diagrams, such as the iterator, could also be added as control flow abstractions. Essentially, the transitional relationships define a (labelled) transition system. Syntactical and semantical descriptions – here input and output parameter objects (syntax) and constraints (semantics) – are associated to individual services through *descriptional relationships*.

The benefit of this non-standard approach are improved reasoning capabilities for dynamic properties such as lifeness and safety. WSPO can be distinguished from other service ontologies by two specific properties.

- Firstly, although based on description logics, it adds a relationship-based process sublanguage enabling process expressions based on iteration, choice, and sequential and parallel composition operators.
- Secondly, it adds data to processes in form of in- and output parameters – introduced as constant process elements into the process sublanguage.

We will present WSPO here in a pseudo-OWL notation to avoid the full verbosity of XML-based descriptions, see e.g. Fig. 4. The @-construct used in some constraints refers to the attribute in the prestate, cf. [22].

A service process template with a central process element (the transitional relationship) and associated services (descriptional relationship) defines the basic structure of states and service processes. Syntactical parameter information in relation to the individual activities and also semantical information such as preconditions are attached to each activity as defined in the template. The pre- and poststates will remain implicit in the notation.

The three services on the right-hand side of Fig. 4 are part of a composed process, shown on the left-hand side. The process is based on a choice construction (based on the decision control flow operator of the UML activity diagrams).

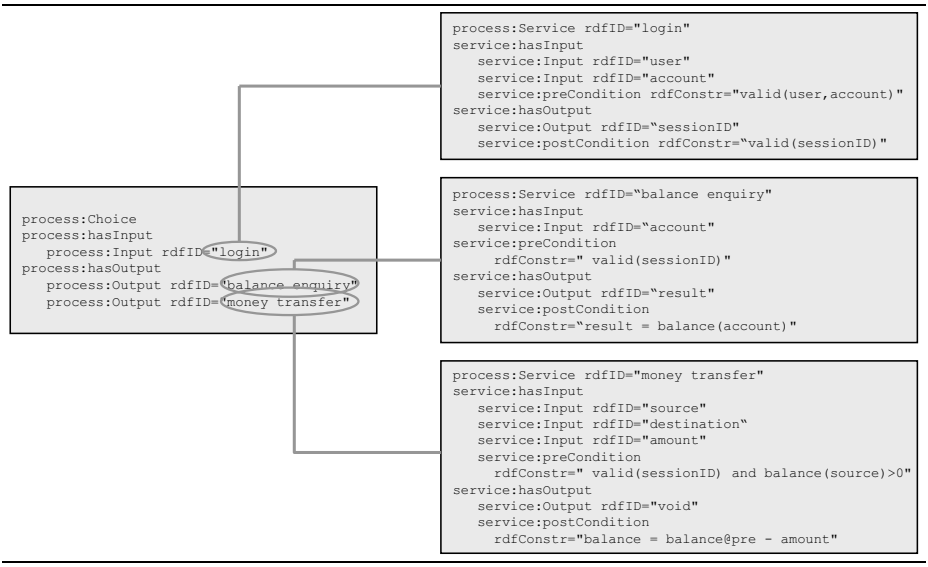

Fig. 4. WSPO Process and Service Model

The left-hand side is a transitional relationship expressing the composed process itself. The three services `login` (as input) and `balance enquiry` and `money transfer` (both as output of the control flow operator) are combined. Input and precondition are (implicitly) associated to the prestate and output and postcondition are (implicitly) associated to the poststate. Although the pre- and poststates are not explicit in the WSPO notation, their presence is necessary as the overall process specification is interpreted by labelled transition systems. The transitional relationships, i.e. the process specification itself, defines the accessibility relationship between pre- and poststates.

Pre- and postconditions for the composed process can be derived from the individual service specifications – once the overall consistency of the abstract composed process definition is established – in order to represent the process as a single service to potential users.

4.2 Mapping Activity Diagrams to Service Ontology

In our framework, UML activity diagrams and our extension to model service processes based on the UML profile serve mainly as a tool for visual modelling. The ontology framework provides the service architecting engine for the platform-independent and platform-specific model layers. It performs composition checks and creates executable process implementations.

The mapping from the activity diagrams based on the profile into the WSPO is straightforward. The ontology representation in Fig. 4 is the result of the transformation of the UML model in Fig. 3. WSPO is based on a standard template. A process specification forms the core, to which individual service specifications

of services that participate in the process are associated. The standard activity nodes and edges from UML activity diagrams are mapped onto the process template:

- The activity nodes of services connected by the activity edges to processes (see Fig. 3) with their input and output elements are mapped onto the process part of the template. The UML control flow operators, such as decision and fork, are represented by the WSPO process combinators, such as choice and concurrency.
- For each service (activity node), a separate service part with input and output, precondition and postcondition information is generated, where each of the individual information elements is considered as attached through a descriptional relationship. The UML input and output pins are mapped to WSPO service input and output concepts. Pre- and postconditions of the UML extension are equally mapped to WSPO concepts.

We currently work with a subset of activity diagram features as shown in Fig. 3, which is sufficient to express abstract functional service and process properties. The transformation between this subset and WSPO is straightforward. Other diagram elements, such as activity partitioning mechanisms (swimlanes), could be used in extensions of this approach to consider non-functional aspects such as service distribution. We have investigated modelling of service distribution in [23].

The ontology acts as a formally defined internal representation that enables transformation, composition, and reasoning activities. UML provides an interface for visual modelling, but also interoperability, which allows existing UML models to be reused and integrated into our proposed framework.

4.3 Process Composition

Services that are visually composed do not necessarily match semantically. A semantical analysis of the composition between these abstract specifications is required. The excerpt of the bank account model.

The login service produces an output object `sessionID` that satisfies the postcondition `valid(sessionID)`. Although the `sessionID` is not required as an input element for the subsequent `balance enquiry` service, the validity of the `sessionID` is still required and guards this service. This case, even though a simple one, illustrates the need to check the consistency of the composition – both in terms of input elements *in* and output elements *out* and also the semantic matching of postcondition of the predecessor $post_P$ and precondition pre_S of the successor service. A required input element *in* of an output service of a composition must be provided as an output element of the preceding service in the process composition (cf. pipes) or must be supplied by the overall process instance (cf. calls). An implication $post_P \rightarrow pre_S$ is the semantic consistency constraint for the composition. This applies to all composition operators (sequence, choice, concurrency).

This type of composition can be characterised as horizontal, whereas in the following section, we will address the vertical dimension of composition by associating concrete provided services to abstract process models.

4.4 Composing Service Providers and Clients

The service process defined by modelling the control and data flow characteristics visually and by checking its consistency using the ontology engine is still an abstract description. Concrete services need to be found that match the requirements expressed in the abstract models, called service orchestration. Matching is often based on the so-called IOPE (Input Output Precondition Effect) characteristics. A refinement relation (e.g. weakening the precondition and strengthening the postcondition or effect, which we use here) defines the matching notion.

Ontologies enable reasoning about models and their properties. In [17], a refinement notion is integrated into an ontological framework, based on the ontological subsumption (subclass) relationship. There, we have presented an ontological matching notion that can be applied to determine whether a service provider can be connected to a service user based on their individual service and process requirements.

Assume that in order to implement an account process, an implementation for the `money transfer` service with input parameter `amount` needs to be integrated. For any given state, the process developer might require (using the `balance` enquiry)

```
service:preCondition  rdfConstr="balance() > amount"
service:postCondition rdfConstr="balance()= balance()@pre-amount"
```

which would be satisfied by a provided service

```
service:preCondition  rdfConstr="balance() > 0"
service:postCondition rdfConstr="balance()= balance()@pre- amount
    and lastActivity = 'transfer'"
```

The provided service would weaken the required precondition assuming that the transfer `amount` is always positive and strengthen the required postcondition as an additional result is delivered by the provided service. Note, that we have used a pseudo-RDF notation here to simplify the example.

5 Deployment of Service Processes

The deployment of services at the platform-specific layer involves two perspectives – clients invoking and executing service processes (5.1) and providers publishing abstract descriptions and making the process services available (5.2).

Rule	Aspect	Description
P1	WS-BPEL	The complex WSPO process relationships can be mapped to BPEL processes.
P1.1	WS-BPEL process partners	For each process create a BPEL partner process.
P1.2	WS-BPEL orchestration	Convert each process expression into BPEL-invoke activities and the client side BPEL-receive and -reply activities at the server side.
P1.3	WS-BPEL process activities	Convert the process combinators ';', '+', '!', and '||' to the BPEL combinators sequence, pick, while, and flow, resp.

Fig. 5. Transformation Rules – Executable Processes

5.1 Code Generation for Service Process Invocation and Execution

Automated code generation is one of the central objectives of MDA. In the context of SOA, code generation essentially means the generation of executable service processes. WS-BPEL [6], which has been looked at from a semantic perspective [24], has emerged as the most widely accepted process execution language for Web services.

A summary of the transformation rules from WSPO to WS-BPEL is presented in Fig. 5. WSPO defines a simple language that can be fully translated into WS-BPEL. BPEL process partners are the client and the different service providers. The WSPO specification is already partitioned accordingly. Flow combinators can also be mapped directly.

5.2 Description and Publication of Services and Service Processes

The Web Services architecture proposes a specific platform based on services provided at certain locations, which can be located using directory information provided in service registries. The description of services – or service processes made available as a single service – is therefore of central importance. Information represented in the process model and formalised in the service process ontology can be mapped to a service ontology. Both OWL-S and WSMO would be suitable here. This transformation would only be a mapping into a subset, since these ontologies capture a wide range of functional and non-functional properties, whereas we have focussed on architecture-specific properties in WSPO.

We have chosen WSMO here to illustrate this type of code generation. A summary of the transformation rules from WSPO to WSMO is presented in Fig. 6. Some correspondences guide this transformation. WSPO input and output elements correspond to WSMO `messageExchange` patterns, which are used in WSMO to express stimuli-response patterns of direct service invocations, and WSPO pre- and postconditions correspond to their WSMO counterparts.

Rule	Aspect	Description
D1	WSMO	Based on the WSPO model, map process relationships to WSMO service concept and fill `messageExchange` and `pre/postCond` properties accordingly.
D1.1	WSMO messageExchange	Map the WSPO in and out objects onto WSMO `messageExchange` descriptions.
D1.2	WSMO pre-/postconditions	Map the WSPO pre- and postconditions onto WSMO `pre` and `postconditions`.

Fig. 6. Transformation Rules – Semantic Service Descriptions

6 Related Work

Some developments have started exploiting the connection between ontologies – in particular OWL – and MDA. In [25], 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 [9,25] and the OMG [1,13], however, needs to be carried further to address the ontology-based modelling and reasoning of service-based architectures. In particular, the Web Services architecture needs to be addressed in the context of Web-based ontology technology. Some of the reasoning tasks we used ontologies for, could have also been addressed using OCL [22]. However, ontologies provide a full-scale logic and additionally allow XML-based sharing and exchange in a Semantic Web framework.

Grønmo et.al. [26] introduce – based on ideas from [25] – an approach similar to ours. Starting with a UML profile based on activity diagrams, services are modelled. These models are then translated into OWL-S. Although the paper discusses process composition, this aspect is not detailed. We have built on [26] in this respect by considering process compositions in the UML profile and by mapping into a service ontology that focusses on providing explicit support for service processes. Other authors [27,28] have directly connected UML modelling with WS-BPEL code generation, without the explicit ontology framework. Integrating ontologies, however, enhances the semantic modelling and reasoning capabilities in the context of service architectures.

WSMO [12] and OWL-S [11] 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. The Web Service Process Ontology WSPO [17,18] is also a service ontology, but its focus is the support of description and reasoning about service composition and service-based architectural configuration. An important current development is the Semantic Web Services Framework (SWSF), consisting of a language and an underlying ontology [29], which takes OWL-S work further and

is also linked to convergence efforts in relation to WSMO. The FLOWS ontology in SWSF comprise process modelling and it equally suited to support semantic modelling within the MDA context.

Our framework has to be seen in the context of MDA initiatives. The OMG supports selected modelling notations and platforms through an adoption process. While Web technologies have not been adopted so far, the need for a specific MDA solution for the Web context is a 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 [13]. ODM will provide mappings to OWL-DL and also a UML2 profile for ontologies. 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.

7 Conclusions

Service-oriented architecture is developing into a service engineering paradigm with its own specific techniques. The development of a service engineering methodology should – similar to other approaches – adopt accepted technologies:

- MDA provides, based on UML, a modelling approach that can satisfy the modelling requirements necessary to develop service architectures and that emphasises tool support and automation.
- Ontology and Semantic Web technologies provide semantic strength for the modelling framework necessary for a distributed and inter-organisational environment.

Our main contribution is an ontology-based engine that supports the process of service architecting. The central element is a service ontology tailored to support service composition and transformation. An ontology-based technique is here beneficial for the following reasons. Firstly, ontologies define a rigourous, logic-based semantics modelling and reasoning framework thats support architectural design activities for services. Secondly, ontologies provides a knowledge integration and interoperability platform for multi-source semantic service-based software systems. Thirdly, service ontologies can also be integrated with domain ontologies to integrate different software development activities – for instance at the computation-independent layer of MDA. Our aim here was to demonstrate the suitability of ontologies for this environment – for both WSPO to support architectural issues but also for WSMO here to support service discovery. We have embedded this service composition ontology into an architecture modelling technique integrating visual UML-based modelling, transformation, ontology-based reasoning, and code generation.

In this approach ontologies replace the classical UML models, except that we keep the visual UML notation, but give semantics to a UML profile for

service architecture by mapping UML models to ontologies. This approach has in addition to the visualisation of models also the benefit of allowing the reuse of existing models. Ontologies add rigorous semantic modelling and reasoning.

While we have outlined the core of an ontology-driven service architecture framework, a number of aspects have remained unaddressed. The integration of a wider range of UML models could be discussed in order to improve the reusability of UML models. For instance, interaction and sequence diagrams express aspects of relevance to service composition and interaction. Composition aspects such as time or error handling could be considered. A reversed mapping from ontologies into UML models could also be considered. A standardised ontology definition model (ODM) can be expected in the near future. The integration of our approach with this standard is necessary for interoperability reasons and will facilitate model reuse, but should turn out to be feasible due to OWL-DL as the common underlying ontology language.

References

1. Object Management Group. *MDA Model-Driven Architecture Guide V1.0.1.* OMG, 2003.
2. L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice (2nd Edition).* SEI Series in Software Engineering. Addison-Wesley, 2003.
3. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services – Concepts, Architectures and Applications.* Springer-Verlag, 2004.
4. World Wide Web Consortium. *Web Services Architecture.* http://www.w3.org/TR/ws-arch, 2006. (visited 28/02/2006).
5. C. Peltz. Web Service orchestration and choreography: a look at WSCI and BPEL4WS. *Web Services Journal*, 3(7), 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. S. McIlraith and D. Martin. Bringing Semantics to Web Services. *IEEE Intelligent Systems*, 18(1):90–93, 2003.
8. T. Payne and O. Lassila. Semantic Web Services. *IEEE Intelligent Systems*, 19(4), 2004.
9. D. Gašević, V. Devedžić, and D. Djurić. MDA Standards for Ontology Development – Tutorial. In *International Conference on Web Engineering ICWE2004*, 2004.
10. J. Rao, P. Küngas, and M. Matskin. Logic-Based Web Services Composition: From Service Description to Process Model. In *International Conference on Web Services ICWS 2004*, pages 446–453. IEEE Press, 2004.
11. 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.
12. R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.
13. Object Management Group. *Ontology Definition Metamodel - Request For Proposal (OMG Document: as/2003-03-40).* OMG, 2003.
14. R. Allen and D. Garlan. A Formal Basis for Architectural Connection. *ACM Transactions 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. C. Pahl. Layered Ontological Modelling for Web Service-oriented Model-Driven Architecture. In *European Conference on Model-Driven Architecture ECMDA2005*. Springer LNCS Series, 2005.
17. C. Pahl. An Ontology for Software Component Matching. *International Journal on Software Tools for Technology Transfer (STTT), Special Edition on Component-based Systems Engineering*, 7, 2006. (in press).
18. 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.
19. F. Baader, D. McGuiness, D. Nardi, and P.P. Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
20. M.C. Daconta, L.J. Obrst, and K.T. Klein. *The Semantic Web*. Wiley, 2003.
21. W3C Semantic Web Activity. Semantic Web Activity Statement, 2004. http://www.w3.org/2001/sw. (visited 06/11/2005).
22. J.B. Warmer and A.G. Kleppe. *The Object Constraint Language – Precise Modeling With UML*. Addison-Wesley, 2003. (2nd Edition).
23. R. Barrett, L. M. Patcas, J. Murphy, and C. Pahl. Model Driven Distribution Pattern Design for Dynamic Web Service Compositions. In *International Conference on Web Engineering ICWE06. Palo Alto, US*. ACM Press, 2006.
24. 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.
25. D. Djurić. MDA-based Ontology Infrastructure. *Computer Science and Information Systems (ComSIS)*, 1(1):91–116, 2004.
26. R. Grønmo, M.C. Jaeger, and H. Hoff. Transformations between UML and OWL-S. In A. Hartman and D. Kreische, editors, *Proc. Model-Driven Architecture – Foundations and Applications*, pages 269–283. Springer-Verlag, LNCS 3748, 2005.
27. K. Mantell. *From UML to BPEL –Model Driven Architecture in a Web services world*. IBM, http://www-128.ibm.com/developerworks/webservices/library/ws-uml2bpel/, 2005.
28. T. Ambühler. *UML 2.0 Profile for WS-BPEL with Mapping to WS-BPEL*. University of Stuttgart, 2005. Diploma Thesis.
29. Semantic Web Services Language (SWSL) Committee. *Semantic Web Services Framework (SWSF)*. http://www.daml.org/services/swsf/1.0/, 2006.