Semantic Model-Driven Development of Web Service Architectures

Claus Pahl

Dublin City University
Dublin 9
Ireland

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 development is a recent approach to developing software systems based on the idea of making models the central artefacts for design representation, analysis, and code generation. We propose an ontology-based engineering methodology for semantic model-driven composition and transformation of Web service architectures. Ontology technology as a logic-based knowledge representation and reasoning framework can provide answers to the needs of sharable and reusable semantic models and descriptions needed for service engineering. Based on modelling, composition and code generation techniques for service architectures, our approach provides a methodological framework for ontology-based semantic service architecture.

Keywords. Service-oriented Architecture, Service Process Composition, Model-Driven Development, Service Ontology, Semantic Models, Web Services.

1 Introduction

Model-driven development (MDD) is an approach to the development of software systems that has gained wide support over the past years (OMG, 2003). MDD focuses on notions of abstraction and automation. MDD emphasises the importance of abstract modelling in the software development process. Detailed models serve as design specifications that support the maintainability of systems and can also provide the basis for automated code generation. Service-oriented architecture (SOA) (Bass et al., 2003) is a specific development and platform approach for service engineering that would benefit from a tailored MDD solution in order to realise the MDD objectives.

Our focus is here on the methodological and technical support of central development activities in service-oriented architecture (Alonso et al., 2004; W3C, 2006). Composition is a central activity in a paradigm that addresses architectures of orchestrated services (Bass et al., 2003). Service orchestration refers to the assembly or composition of services to service processes (Peltz, 2003). Within the Web Services platform (Alonso et al., 2004) - which is the concrete platform for service-oriented architecture that we target here - the business process execution language WS-BPEL (WS-BPEL Coalition, 2004) is the most widely used implementation language for service processes.

We present a methodology for architecting service-based software systems that embraces the MDD-philosophy. We propose to base this methodology on ontology technology. Supporting service engineering using MDD and ontology-based semantic modelling 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 models of services are a prerequisite for the reliable composition of services (Rao et al., 2004).
- Modelling becomes a central activity where services from possibly different providers are assembled to form complex, distributed architectures. Ontologies can define an advanced semantics-based conceptual modelling approach (McIlraith and Martin, 2003).
- Ontologies provide a formal, logic-based framework for reasoning about composition activities and the automation of code generation (Pahl, 2007).

Services description and composition has been combined with ontology technology (Mandell and McIlraith, 2003). Model-driven development has been enhanced to ontology-driven architecture (Gašević et al., 2006). However, the integrated application of both ontology technology and MDD to service architecture has so far not been adequately addressed. In particular, we propose a novel, process-centric composition approach as the core of our framework.

We introduce an ontology-based approach, i.e. a logic-based knowledge representation framework, to enable sharable representations of semantic service and process descriptions and models. Various attempts in this direction include service ontologies such as OWL-S (DAML-S Coalition, 2002), WSMO (Lara et al., 2005), or SWSF (SWSL Committee, 2006). Industrial bodies such as the OMG have also recognised the importance of semantic modelling using ontologies, which is reflected in the OMG's Ontology Definition Metamodel (ODM) initiative (OMG, 2006). The need

for service providers to publish their services in an accepted, standardised format is another argument in favour of ontologies.

UML-style semantic modelling is an important objective. A tailored UML profile based on activity diagrams provides a graphical modelling notation for service process composition, which is supported by a mapping from this profile into a service ontology. Composition occurs in two forms:

- Firstly, the orchestration of services to processes at the abstract level using process operators (Plasil and Visnovsky, 2004).
- Secondly, the association of concrete implemented services that match the requirements of abstract service process elements.

The ontology and associated techniques act as a service architecting engine for both forms of composition and also support code generation for process execution and service publication aspects.

We apply the MDD philosophy to service development. The overall aim is to demonstrate, firstly, the internal coherence of an overarching methodology that can integrate a number of specific techniques, secondly, and to illustrate the benefits of semantic modelling and MDD, and, finally, its implementability (feasibility) in the context of Web services, MDD, and modelling platforms.

We start with an overview of the service engineering process in our context in Section 2. We structure the presentation by the architecting activities. 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

A Web service is defined as a software system, whose public interfaces are defined and described using XML (W3C, 2006). Other systems can interact with the Web service through XML-based messages. The composition of services to orchestrated service processes is a major concern in current software architecture and service engineering research (Plasil and Visnovsky, 2002; Bass et al., 2003). These recent developments have strengthened the importance of these architectural questions. Behaviour and interaction processes are central modelling concerns for service-based software architectures (Allen and Garlan,

1997). Explicit semantic descriptions and exchangeable models enable developers and clients of services to create reliable service architectures. Three frameworks - methodology, ontology, and platform - provide the pillars for our semantic model-driven service engineering approach.

2.1 A Methodological Framework

We embed our service engineering methodology into an ontology-supported, MDD-based development context. The aim is to support platform-independent modelling. At the core is a service-specific software process model for ontology-driven semantic service architecture that is based on the following steps, summarised in Fig. 1:

- Service Process Modelling. This activity is about graphical UML modelling of process activities in terms of activity diagrams with service-oriented semantic extensions. Actions represent services.
- Service Process Composition. Two composition dimensions for service orchestration need to be supported:
  - Abstract Process Composition. The analysis activity, which is part of the process modelling, addresses the integrity of a process composition based on semantical model enhancements in an ontological representation; here we use the Web Service Process Ontology WSPO (Pahl, 2007).
  - Service Process Implementation. The focus of this activity is the discovery of individual services in repositories and directories that match the requirements of the service 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 (BPEL, 2004) based on the associated services. Deployment also includes the publication of the overall process as a service in terms of a service ontology; we suggest the Web Service Modelling Ontology WSMO (Lara et al, 2005).

An ontology-based service architecting engine (Pahl, 2005) supports the composition activities within the modelling layer and also guides the necessary transformations to the platform-specific deployment layer.

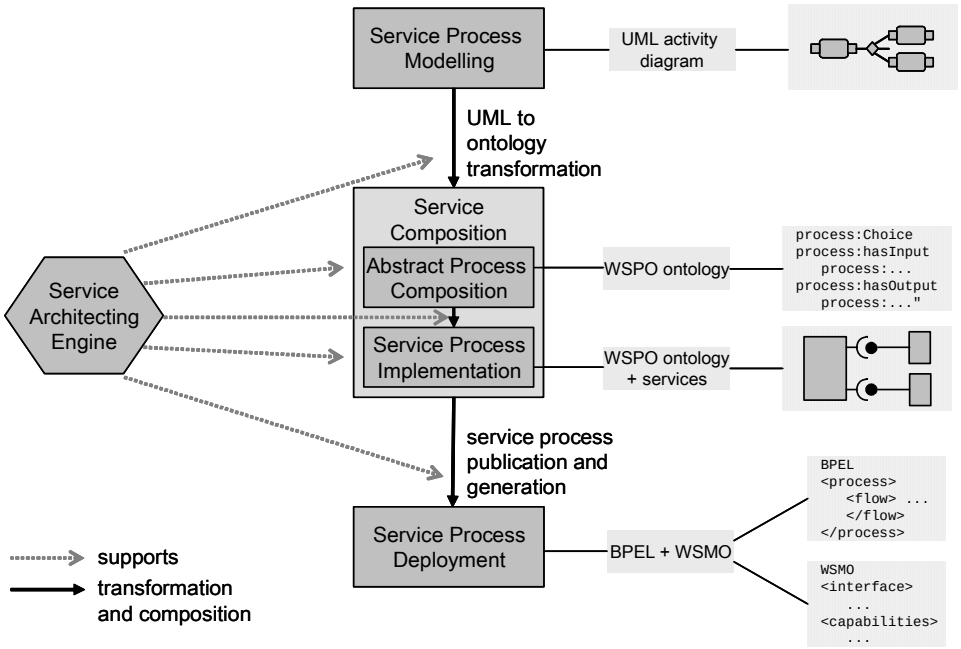

Figure 1. Overview of the Ontology-based Service Architecture Technique.

2.2 An Ontological Modelling Framework

Ontologies are beneficial for this context for two reasons: firstly, to provide semantic service descriptions in a sharable format, and, secondly, to allows reasoning about service properties. Both are essential to support discovery and composition in service-oriented architecture.

Logic-based knowledge representation can enable sharable representations of semantic service and process descriptions. The need for service providers to publish their services in a standardised, interoperable semantic format is an argument in favour of ontology-based knowledge representation for modelling, analysis, and publication (Daconta et al., 2003). Ontologies are knowledge representation frameworks formalised in an ontology language such as OWL (W3C, 2004), 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.

A number of service ontologies have been proposed, with OWL-S (DAML-S Coalition, 2002) and WSMO (Lara et al., 2005) as prominent examples. The central goal of service ontologies is not only the semantic annotation of services, but also to support reasoning about semantics-based discovery in Web service registries and automated service composition. While OWL-S also supports service composition to a degree through process models, we use the Web Service Process Ontology WSPO - whose description logic foundations are presented in (Pahl, 2007) and which has been developed specifically to support service composition and architecture ontologically. An important current development is the Semantic Web Services Framework (SWSF), consisting of a language and an underlying ontology (SWSL Committee, 2006), which takes OWL-S work further and is also linked to convergence efforts in relation to WSMO. The FLOWS ontology in SWSF comprises process modelling and it equally suited as WSPO to support semantic modelling within the MDA context. It is, however, richer and as a consequence undecidable, which hampers the automation of reasoning. Some of the reasoning tasks we used ontologies for, could also have been addressed using simpler languages such as OCL (Warmer and Kleppe, 2003) and their reasoning support. However, ontologies provide a full-scale logic and additionally allow XML-based sharing and exchange in a Semantic Web framework.

WSPO is an OWL-DL (the Web Ontology Language - Description Logic variant) ontology. It uses description logic (Baader et al., 2003) to define composition techniques. 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. 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 as safety and liveness, making WSPO a highly suitable candidate for ontology that supports the logic-based platform-independent abstract modelling of service processes.

UML as a graphical modelling language is a particular target. Our modelling methodology can only be successful if existing UML models can be reused and integrated. Conforming to the UML notation and the Meta Object Facility (MOF) language definition framework, which defines the UML notation, is consequently central. It enables tool interoperability and application-specific notations to be supported by tools easily and it allows model and tool reuse.

2.3 An Implementation Platform Framework

Our efforts have to be seen in the context of MDD initiatives and modelling frameworks. The need for a specific MDD solution for the Web context is, due to the Web's ubiquity and the existence of standardised and accepted platform and modelling technology, a primary concern. Model Driven Architecture (MDA) defines a layered modelling approach for software systems with computation-independent, platform-independent, and platform-specific modelling, abbreviated CIM, PIM, and PSM, respectively (OMG, 2003). We refer here to the MDA platform layers occasionally (CIM, PIM, PSM); these references are meant to be indicative rather than prescriptive.

The current effort of defining and standardising an ontology definition metamodel (ODM) will allow the integration of our technique with OMG standards (OMG, 2006). ODM provides mappings to OWL-DL and also a UML profile for ontologies to make UML's graphical notation available. Meta-model compliancy for ODM is requested to facilitate tool support. Model representations can be exported into XML format, i.e. XML Schemas are generated from models. 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.

QVT (Query View Transformation) is a query, view, and transformation framework - also supported by the OMG - that allows transformations to be specified in both a declarative and operational format (OMG, 2005). A relations language, based on a pattern-matching technique, enables declarative specifications of any transformations. The ODM framework, for instance, refers to QVT as its notation for transformations.

While this implementation framework is central for the overall success of our proposed methodology, due to space constraints full specifications can not be given; we revert to schematic examples and motivation of principles.

3 Modelling of Service Processes

3.1 Service Modelling

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 - the other aspect of composition is the association of concrete services to abstract service placeholders in the composed process

description. Orchestration describes the control and data flow between services using basic flow operators. We deal with the correctness of service implementation in Section 4. In this section, we address the abstract modelling of service orchestrations.

Based on abstract service models, services can be composed to service architectures. This is done by defining the process of service invocations based on a set of process operators. The following textual representation summarises the syntactical aspects in a functional service interface format and an example of a service process. The process of using an online banking account is described in this service process definition. This application is based on four individual services, each described here in terms of input and output parameters.

```
application AccountProcess
   services
      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
```

The services are composed to a process that captures constraints for service invocations within this application, using the combinators sequence (;), iteration (!), and choice (+).

3.2 A Service Modelling and Composition Ontology

WSPO is a decidable encoding of a dynamic logic in terms of description logic, which enables reasoning about dynamic service process properties such as safety and liveness, making WSPO the most suitable service ontology for semantic model-driven service engineering. 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.

Core elements of an ontology are concepts and relationships. In WSPO, they are defined as follows. The central concepts in WSPO 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. We present WSPO here in a pseudo-OWL notation to avoid the full verbosity of XML-based descriptions. The @-construct used in some constraints refers to an attribute in the prestate, which we have borrowed from OCL (Warmer and Kleppe, 2003).

WSPO can be distinguished from other service ontologies by two specific properties. Firstly, although based on description logics, it adds an enriched relationship-based process sublanguage enabling process expressions based on iteration, choice, and sequential and parallel composition operators and adding data to processes in the form of input and output parameters - introduced as constant process elements into the process sublanguage (Pahl, 2007). Secondly, WSPO is based on a decidable description logic, which in comparison to other service ontologies, gives a crucial advantage in the context of automation.

Individual service descriptions form the basis of the modelling activity. An example that enhances the previous syntactic description of an account service by semantic conditions is

```
process:Service rdfID="money transfer"
   service:hasInput
      service:Input rdfID="account"
      service:Input rdfID="destination"
      service:Input rdfID="amount"
      service:precondition
            rdfConstr="valid(sessionID) and balance(account)>0"
   service:hasOutput
      service:Output rdfID="void"
      service:postCondition
            rdfConstr="balance = balance@pre - amount"
```

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 process models in WSPO. Syntactical parameter information in relation to the individual activities and also semantical information such as preconditions and postconditions (see example above) are attached to each activity defined in the template.

The pre- and poststates will remain implicit in the notation. Service processes are defined in terms of operators such as choice, iteration, or sequence:

```
process:Choice
   process:hasInput
      process:Input rdfID="login"
   process:hasOutput
      process:Output rdfID="balance enquiry"
      process:Output rdfID="money transfer"
```

The three services on the right-hand side of Fig. 2 (money transfer has already been presented) are part of a composed process, shown on the left-hand side and already explained above. The process is based on a choice construction (based on the decision control flow operator of the UML activity diagrams). 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. This choice process can form the first step in a sequence with a logout service to form the overall process that we have presented in Section 3.1 in abstract textual form.

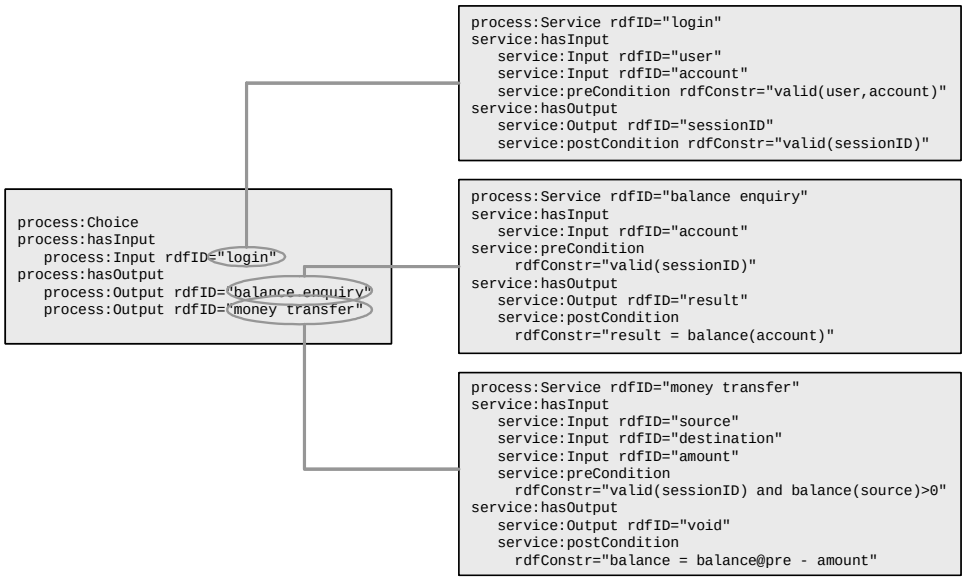

Figure 2. WSPO Process and Service Model.

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.

3.3 UML-based Service Process Modelling

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

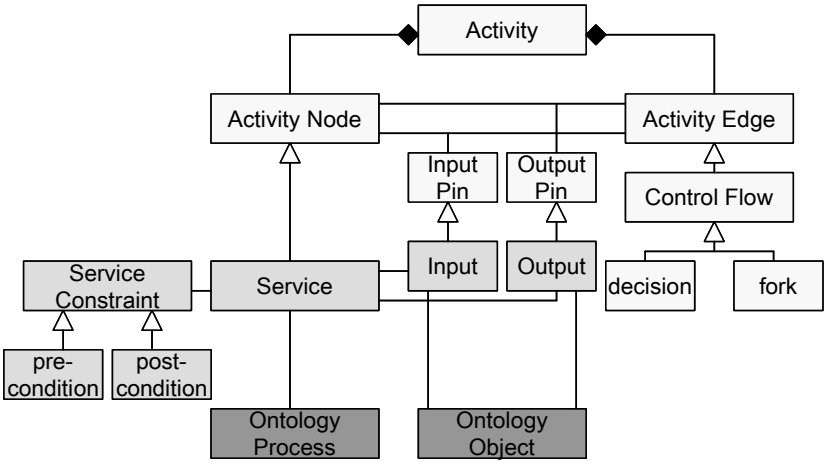

Figure 3. UML Profile (Metamodel) for Semantic Service Process Modelling.

These properties make UML activity diagrams an ideal starting point to develop a standards-compliant and interoperable grahical modelling interface for WSPO. We require some extensions to activity diagrams, which we will capture in form of a UML profile, to address the specific needs of semantic service process description. This metamodel for service

architectures is defined in Fig. 3. White rectangles denote the standard UML activity diagram elements; medium grey ones denote service-specific extensions; dark grey ones associate elements of a possible domain ontology. Note that often a domain ontology captures central concepts, i.e. key objects and processes, of an application context (Pahl, 2005). The architecture model here is linked to the domain model. Although we do not discuss this aspect in detail, the integration with a domain model is central for a coherent ontology-based modelling approach.

A WSPO service is an activity in UML terms. A service's input and output objects are linked to the input and output pins of activities. The lowest layer in the metamodel diagram addresses an ontology grounding, i.e. the connection of the model elements to concepts of an underlying domain ontology. This ontology - we distinguish passive objects and active processes - introduces central, accepted domain concepts and their semantics. 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. 4. This model represents the online bank account process that we have used earlier on to illustrate the OWL-based textual representation in WSPO. It visualises the process login; !(balance enquiry + money transfer); logout together with in- and out-parameters and semantic conditions.

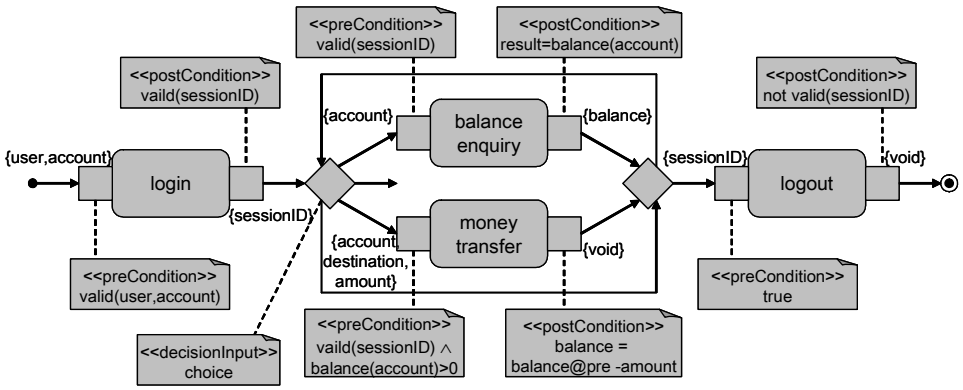

Figure 4. Semantic Service Process Model based on UML Activity Diagrams.

We work with a subset of activity diagram features as shown in Fig. 4, 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 (swim lanes), could be used in extensions of this approach to consider non-functional aspects such as service distribution (Barrett et al., 2006).

3.4 Mapping Activity Diagrams to Service Ontologies

UML activity diagrams and our extension to model service processes based on the UML profile serve mainly as a tool for visual modelling, but also enable interoperability, which allows existing UML models to be reused and integrated into our proposed framework. The underlying ontology framework provides the service architecting engine for platform-independent and platform-specific modelling. It performs composition checks and creates executable process implementations. The ontology acts as a formally defined internal representation that enables transformation, composition, and reasoning activities.

The mapping from activity diagrams into WSPO is straightforward. The ontology representation in Fig. 2 is the result of the transformation of (parts of) the UML model in Fig. 4. WSPO is based on a standard template that matches the structure of the given activity diagram:

- A state-based process specification forms the core.
- Individual descriptions of services that participate in the process are associated.

The activity nodes and edges from UML activity diagrams are mapped onto the process template:

- Service connectors: The activity nodes of services connected by the activity edges to processes 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.
- Services: For each service (an 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.

This illustrates the ideas behind the transformation. These principles can be implemented in terms of the ODM framework. ODM provides metamodels for UML and OWL and transformations between them, which

make the formulation of the UML-to-WSPO transformation straightforward.

4 Composition of Service Processes

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

- Process orchestration. 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 the semantic level.
- Refinement. The composition of the abstract service process as the client and individual implemented 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.

We introduce in this section an ontology-based engine to support the architecting of service-based systems based on these two composition forms. Ontology support aids to achieve the consistency of compositions. 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 process composition analysis before looking at the refinement dimension.

4.1 Abstract 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 in Fig. 4 exemplifies this aspect.

The consistency of a process composition, e.g. for a sequential composition, needs to be checked - both in terms of input elements in and output elements out and also the semantic matching of postconditions of the predecessor $post_P$ and precondition pre_S of the successor service.

- A required input element in of a 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 condition can be verified within the WSPO encoding of dynamic logic.

This applies to all composition operators such as sequence, choice, or concurrency.

For example, 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. The postcondition satisfies the guarding condition of the subsequent service. This case, even though a simple one, illustrates the need to check the consistency of the composition.

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.

4.2 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 model. Concrete services need to be found that match the requirements expressed in the abstract models of service orchestrations. Matching is often based on the so-called IOPE (Input Output Precondition Effect) characteristics. A refinement relation can define the matching notion.

Ontologies enable reasoning about models and their properties. In (Pahl, 2007), a refinement notion is integrated into an ontological framework, based on the ontological subsumption (subclass) relationship. This matching notion can be applied to determine whether a service provider can be connected to a service user based on their individual service and process requirements. Weakening the precondition and strengthening the postcondition or effect, which we use here, is a refinement condition that can be applied to individual services. (Pahl and Casey, 2003). This can be verified using the ontology.

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 for 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 weakens the required precondition assuming that the transfer amount is always positive and strengthens the required postcondition as an additional result is delivered by the provided service. This is a safety condition. Lifeness conditions, i.e. conditions that will eventually become true, such as termination, can be dealt with in the same way. WSPO focuses on functional properties of services, such as safety and lifeness, only. Note that we have used a pseudo-RDF notation here to simplify the example. RDF is at the core of OWL.

5 Deployment of Service Processes

The deployment of services within specific platforms such as the Web services platform involves two perspectives - clients invoking and executing service processes (Section 5.1) and providers publishing abstract descriptions and making the process services available (Section 5.2).

5.1 Code Generation for Service Process Invocation and Execution

Automated code generation is one of the central objectives of MDD. In the context of SOA, code generation essentially means the generation of exectuable service processes. WS-BPEL (BPEL, 2004), which has already been looked at from a semantic, ontology-based perspective (Mandell and McIlraith, 2003), has emerged as the most widely accepted process execution language for orchestrated Web services.

A summary of the transformation rules from WSPO to WS-BPEL is presented below. WSPO defines a simple process 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 (the WSPO process combinators) can also be mapped directly. The principles of the transformation are the following:

- WS-BPEL process: The complex WSPO process relationships are mapped to BPEL processes.
- WS-BPEL process partners: For each process, a BPEL partner process (client and server) is created.
- WS-BPEL orchestration: Each process expression is converted into BPEL-invoke activities at the client side and BPEL-receive and -reply activities at the server side.
- WS-BPEL process activities: The process combinators ';', '+', '!', and '||' are converted to BPEL flow combinators sequence, pick, while, and flow, respectively.

These principles can be formulated in terms of QVT (OMG, 2005). We only present a schematic example here to illustrate the concept of declarative transformation specification:

```
transformation WSPO_to_WS-BPEL (wspo : WSPO, bpel: WS-BPEL)
{

   top relation TransRelationship_to_Process
                                    /* maps transitional relationships to processes */
   {
         checkonly domain wspo w:TransRel {name = pn}
         enforce     domain bpel b:Process  {name = pn}
   }

   relation RelExpr_to_ProcessExpr
                                      /* map each process expression recursively */
   {
         domain wspo r:Relationship {
            e1 = e:LeftExpr {},
            e2 = e:RightExpr {},
            op = c:Combinator {}
         }
         domain bpel p:Process {
            process = p:Process {left=e1, pr=op, right=e2},
            client =  pe:ProcessExpr {invoke=op(e1,e2)},
            server =  pe:ProcessExpr {receive=(op,e1,e2), reply=op(e1,e2)}
         }
         when {
            CombinatorMapping(op) and ProcessPartnerCreation(p);
         }
         where {
            RelExpr_to_ProcessExpr(e1) and RelExpr_to_ProcessExpr(e2);
         }
   }
```

```
    relation ProcessPartnerCreation  /* creates server and client process partner */
    {...}

    relation CombinatorMapping       /* converts individual process combinators */
    {...}
}
```

A variety of languages for transformation exist. Among these, QVT is poised to become the lingua franca for model-driven transformations due to its OMG support. Its declarative nature allows specifying transformations at a high level of abstraction. Although its standardisation is not completed and only limited tool support is currently available for the declarative specifications, QVT is an essential tool in the implementation of our methodology in the near future. Therefore, we have presented the transformations in terms of QVT, even though we have used a Java-based implementation in our prototype.

5.2 Description and Publication of Services and Service Processes

The Web Services platform proposes a specific architecture 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 – ideally in semantical format 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. OWL-S, WSMO, or SWSF would be suitable here. This transformation would only be a mapping into a subset of these ontologies, since they 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 outlined below. 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.

- WSMO service: Based on the WSPO model, process relationships are mapped to WSMO service concept and fill messageExchange and pre- and postCondition properties accordingly.

- WSMO messageExchange: The WSPO in and out objects are mapped onto WSMO messageExchange descriptions.
- WSMO pre-/postconditions: The WSPO pre- and postconditions are mapped onto WSMO pre - and postConditions.

A formulation in QVT would follow the style to the WSPO-to-WS-BPEL transformation presented in Section 5.1.

6 Evaluation

The evaluation of the proposed techniques and the methodology is essential to demonstrate their effectiveness. Our main aim was to demonstrate the benefits of an ontology-based approach. We have applied our framework in two settings:

- Firstly, we have used the banking scenario, from which the previous examples were drawn, as a conceptual case study. This acts a comparison instrument as the example is widely used in the literature.
- Secondly, we have used the model-driven development of a learning technology system as an empirical setting of a knowledge-driven application.

Effectiveness of techniques and methodology is the central quality that we aim to demonstrate. Tractability and implementability of the techniques are additional requirements. As far as the latter are concerned, tractability is given through the decidability of the approach. While visual modelling in the UML/MOF context and ontology processing are well supported by tools, the MDD and transformation context requires more platform technologies, before the proposed techniques can be fully implemented using non-proprietary solutions.

The methodology as such is effective. In our empirical case study, we have gained semantical models, which serves us now as system documentation and which also have eased the creation of code through partial automated code generation. This results in better quality and some cost savings. Although currently the higher time investment for the semantic models is not fully compensated by the semi-automated code generation (which is partly due to a lack of tool support), we believe in a return of investment when change and evolution has to be dealt with. In particular, the learning technology context with the availability of subject and instruction domain ontologies, has proved suitable for the application of our approach.

7 Related Work

Some developments have started exploiting the connection between ontologies - in particular OWL - and MDA. In (Djurić, 2004), 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. It clarifies the role of what we have called the implementation framework (Section 2.3). The work by (Gašević et al., 2006) and (Djurić, 2004) and also the OMG (OMG, 2003; OMG, 2006), however, needs to be carried further

- to address the ontology-based modelling and reasoning of specifically service-based architectures using dedicated service ontologies instead of OWL in general - in particular, the Web Services architecture needs to be addressed in the context of Web-based ontology technology,
- to provide a process-centric solution that allows the semantic representation of service compositions in terms of an ontology and that allows reasoning about service process compositions.

Grønmo et al. (2005) introduce - based on ideas from (Djurić, 2004) - a model-driven service development approach similar to ours that is based on service ontologies and that addresses the first shortcoming identified above. 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 (Grønmo et al., 2005) in this respect by considering process compositions in the UML profile and by mapping into a service ontology that focuses on providing explicit support for service processes. Other authors (Mantell, 2005) 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. Our approach goes beyond these approaches in that it is based on a service process ontology, WSPO in this case, rather than a service ontology. We can therefore provide improved composition support.

8 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:

- MDD 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 development and deployment environment.

Our main contribution is a methodology for service engineering based on an ontology 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 rigorous, logic-based semantic modelling and reasoning framework that support architectural design activities for services such as composition. Secondly, ontologies provide a knowledge integration and interoperability platform for multi-source semantic service-based software systems. Thirdly, service ontologies can be integrated with domain ontologies to integrate different software development activities - for instance at the computation-independent layer of MDA.

We set out to achieve a number of objectives with this service engineering methodology.

- We have demonstrated the suitability of ontologies for this service engineering environment through examples and technology discussion - for both WSPO to support architectural issues but also for WSMO here to support service discovery.
- We have embedded this service composition ontology into a coherent architecture modelling technique, integrating visual UML-based modelling, transformation, ontology-based reasoning, and code generation.
- We have outlined that with an implementation context consisting of ODM as the metamodel framework for ontological modelling, MDA as a standardised framework for MDD, and QVT as a central transformation technique, the implementation and application of our methodology becomes feasible.

Compared to the current state of the art in MDD, in our approach ontologies replace the classical UML models, except that we keep the graphical UML notation, but give semantics to a UML profile for service architecture by mapping UML models to service 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 methodology, a number of aspects can extend the proposed methodology.

- Practical considerations: The integration of a wider range of UML models can 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.
- Standards and Trends: The Ontology Definition Model ODM has recently been standardised and can be expected to be widely adopted if adequate tool support is available. The full integration of our approach with this standard is necessary for interoperability reasons and will facilitate model reuse, and should turn out to be feasible due to OWL-DL as the common underlying ontology language.

References

Allen, R. and Garlan, D. (1997). A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology, 6(3), pp. 213–249.

Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004). Web Services – Concepts, Architectures and Applications. Springer-Verlag.

Baader, F., McGuiness, D., Nardi, D. and Schneider, P.P. editors (2003). The Description Logic Handbook. Cambridge University Press, 2003.

Barrett, R., Patcas, L. M., Murphy, J. and Pahl, C. (2006). Model Driven Distribution Pattern Design for Dynamic Web Service Compositions. In International Conference on Web Engineering ICW'E06. ACM Press.

Bass, L., Clements, P. and Kazman. R. (2003). Software Architecture in Practice (2nd Edition). SEI Series in Software Engineering. Addison-Wesley.

Daconta, M.C., Obrst, L.J. and Smith, K.T. (2003). The Semantic Web. Wiley.

DAML-S Coalition (2002). 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, pp. 279–291. Springer-Verlag.

Djurić, D. (2004). MDA-based Ontology Infrastructure. Computer Science and Information Systems, 1(1), pp. 91–116.

Gašević, D., Djurić, D., Devedžić, V. (2006). Model Driven Architecture and Ontology Development. Springer-Verlag.

Grønmo, R., Jaeger, M.C. and Hoff, H. (2005). Transformations between UML and OWLS. In A. Hartman and D. Kreische, editors, Proc. Model-Driven Architecture – Foundations and Applications, pp. 269–283. Springer-Verlag, LNCS 3748.

Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. (2005). Web Service Modeling Ontology. Applied Ontology, 1(1), pp. 77–106.

Mandell, D.J. and McIlraith, S.A. (2003). 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, pp. 227–241. Springer-Verlag, LNCS 2870.

Mantell, K. (2005). From UML to BPEL – Model Driven Architecture in a Web services world. IBM. http://www28.ibm.com/developerworks/webservices/library/wsuml2bpel/.

McIlraith, S. and Martin, D. (2003). Bringing Semantics to Web Services. IEEE Intelligent Systems, 18(1), pp. 90–93.

Object Management Group (2003). MDA Model-Driven Architecture Guide V1.0.1. (OMG Document omg/03-06-01). OMG.

Object Management Group (2005). Query View Transformation - MOF QVT Final Adopted Specification (OMG Document ptc/05-11-01). OMG.

Object Management Group (2006). Ontology Definition Metamodel - Submission (OMG Document: ad/2006-05-01). OMG.

Pahl, Claus (2002). A formal composition and interaction model for a web component platform. In Proc. ICALP'2002 Workshop on Formal Methods and Component Interaction, 8-13 Jul 2002, Malaga, Spain.

Pahl, C. (2005). Layered Ontological Modelling for Web Service-oriented Model-Driven Architecture. In European Conference on Model-Driven Architecture ECMDA'2005. Springer LNCS Series.

Pahl, C. (2007). An Ontology for Software Component Matching. International Journal on Software Tools for Technology Transfer (STTT), Special Edition on Component-based Systems Engineering, 7.

Pahl, C. and Casey, M. (2003). Ontology Support for Web Service Processes. In Proc. European Software Engineering Conference and Foundations of Software Engineering ESEC/FSE'03. ACM Press.

Pahl, C. and Zhu, Y. (2005). A semantical framework for the orchestration and choreography of web services. In Proc. International Workshop on Web Languages and Formal Methods WLFM'05, 19 Jul 2005, Newcastle, UK.

Payne, T. and Lassila, O. (2004). Semantic Web Services. IEEE Intelligent Systems, 19(4).

Peltz, C. (2003). Web Service orchestration and choreography: a look at WSCI and BPEL4WS. Web Services Journal, 3(7).

Plasil, F. and Visnovsky, S. (2002). Behavior Protocols for Software Components. ACM Transactions on Software Engineering, 28(11), pp. 1056–1075.

Rao, J., Küngas, P. and Matskin, M. (2004). Logic-Based Web Services Composition: From Service Description to Process Model. In International Conference on Web Services ICWS 2004, pp. 446–453. IEEE Press.

Semantic Web Services Language (SWSL) Committee (2006). Semantic Web Services Framework (SWSF). http://www.daml.org/services/swsf/1.0/.

Warmer, J.B. and Kleppe, A.G. (2003). The Object Constraint Language – Precise Modeling With UML. Addison-Wesley. (2nd Edition).

World Wide Web Consortium (2004). Semantic Web Activity Statement. http://www.w3.org/2001/sw.

World Wide Web Consortium (2006). Web Services Architecture. http://www.w3.org/TR/ws-arch.

WS-BPEL Coalition (2004). WS-BPEL Business Process Execution Language for Web Services – Specification Version 1.1. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel.