

Ontology Transformation and Reasoning for Model-Driven Architecture

Claus Pahl

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

Abstract. Model-driven Architecture (MDA) is a software architecture framework proposed by the Object Management Group OMG. MDA emphasises the importance of modelling in the architectural design of service-based software systems. Ontologies can enhance the modelling aspects here. We present ontology-based transformation and reasoning techniques for a layered, MDA-based modelling approach. Different ontological frameworks shall support domain modelling, architectural modelling and interoperability. Ontologies are beneficial due to their potential to formally define and automate transformations and to allow reasoning about models at all stages. Ontologies are suitable in particular for the Web Services platform due to their ubiquity within the Semantic Web and their application to support semantic Web services.

1 Introduction

The recognition of the importance of modelling in the context of software architecture has over the past years led to model-driven architecture (MDA) – a software engineering approach promoted by the Object Management Group (OMG) [1]. MDA combines service-oriented architecture (SOA) with modelling techniques based on notations such as the Unified Modelling Language (UML). Recently, ontologies and ontology-based modelling have been investigated as modelling frameworks that enhance the classical UML-based approaches. While formal modelling and reasoning is, to some extent, available in the UML context in form of OCL, ontologies offer typically full reasoning support, for instance based on description logic. A second benefit of ontologies is the potential to easily reuse and share models.

Modelling and developing software systems as service-based architectures is gaining increasing momentum. Modelling and describing services is central for both providers and clients of services. Providers need to provide an accurate description or model for a service that can be inspected by potential clients. In particular the attention that the Web Services Framework (WSF) [2, 3] has received recently emphasises the importance of service-orientation as the architectural paradigm. Service-oriented architecture (SOA) [4, 5] is becoming a central software engineering discipline that aims at supporting the development of distributed systems based on reusable services.

MDA has been developed within the context supported and standardised by the OMG, i.e. UML as the modelling notation and CORBA as the implementation platform. It is based on a layered architecture of models at different abstraction levels. Our focus is the Web Services platform based on techniques supported by the World-Wide Web Committee (W3C) [2], such as WSDL and SOAP, but also extensions like WS-BPEL for service processes [6, 7, 8]. This specific area is particularly suitable to demonstrate the benefits of ontology-based modelling due the distributed nature of service-based software development with its emphasis on provision and discovery of descriptions and on sharing and reuse of models and services.

While the OMG has started a process towards the development of an Ontology Definition Metamodel (ODM) [9] that can support ontological modelling for particular MDA model layers, we take a more comprehensive approach here with ontologies for all layers. Our approach will also be more concrete, i.e. based on concrete ontologies. Other authors, e.g. [10], have already explored OWL – the Web Ontology Language – for MDA frameworks. We will extend these approaches by presenting here a layered, ontology transformation-based modelling approach for software services. We will introduce ontology-based modelling approaches for three MDA layers – computation-independent, platform-independent, and platform-specific. Ontologies will turn out to support a number of modelling tasks – from domain modelling to architectural configuration and also service and process interoperability. We will put an emphasis on processes, which will play an important role in modelling domain activities but also in modelling interaction processes in software architecture configurations.

Our contribution is a layered ontological transformation framework with different ontologies focussing on the needs of modelling at particular abstraction layers. We will indicate how such a framework can be formally defined – a full formalisation is beyond the scope of this paper. We will present our approach here in terms of an abstract notation, avoiding the verbosity of XML-based representations for services and ontologies.

We will start with an overview of service-oriented architecture and ontology-based modelling in Section 2. Service modelling with ontologies is introduced in Section 3. We will address the transformations between ontological layers in Section 4. We discuss our efforts in the context of OMG adoption and standardisation in Section 5. Related Work is discussed in Section 6. We end with some conclusions in Section 7.

2 Service Architectures and Models

The context of our work is model-driven architecture (MDA). Our platform is service-based; our modelling approach is ontology-based. These aspects shall now be introduced.

2.1 Services and the Web Services Framework

A service provides a coherent set of operations at a certain location [3]. The service provider makes an abstract service interface description available that can

be used by potential service users to locate and invoke the service. Services have so far usually been used 'as is' in single request-response interactions [11]. However, the configuration and coordination of services in service-based architectures and the composition of services to processes is becoming equally important in the second generation of service technology. Existing services can be reused and composed to form business or workflow processes. The principle of architectural composition is process assembly.

The discovery and invocation infrastructure of the Web Services Framework (WSF) [2] – a registry or marketplace where potential users can search for suitable services and an invocation protocol – with the services and their clients form our platform, i.e. a service-oriented architecture. Languages for description are central elements of a service-oriented architecture. With the second generation of service technology and efforts such as MDA, the emphasis has shifted from description to the wider and more comprehensive activity of modelling.

Behaviour and interaction processes are essential parts of modelling and understanding software system architectures [12, 13, 14, 15]. However, the support that architectural description languages offer with regard to behavioural processes in architectures is sometimes limited. MDA focuses on UML modelling to support architectural designs. In [16], scenarios, i.e. descriptions of interactions of a user with a system, are used to operationalise requirements and map these to a system architecture.

2.2 Ontologies and the Semantic Web

Making the Web more meaningful and open to manipulation by software applications is the objective of the Semantic Web initiative. Knowledge representation and logical inference techniques form its backbone [17, 18]. Ontologies – the key to a semantic Web – express terminologies and precisely defined semantical properties and create shared understanding of annotations of Web resources such as Web pages or services. Ontologies usually consist of hierarchical definitions of important concepts in a domain and descriptions of the properties of each concept, supported by logics for knowledge representation and reasoning.

Ontologies are, however, important beyond sharable and processable annotations of Web resources. Some effort has already been made to exploit Semantic Web and ontology technology for the software engineering domain in general and modelling in particular [19]. OWL-S [20] for instance is a service ontology, i.e. it is a language that provides a specific vocabulary for describing properties and capabilities of Web services, which shows the potential of this technology for software engineering.

An ontology is defined in terms of concepts and relationships. An ontology is a model of a domain made available through the vocabulary of concepts and relationships. Concepts capture the entities of the domain under consideration. Instances are concrete objects of a particular concept. Relationships capture the relationships between concepts and their properties. In this respect, ontologies are similar to modelling notations such as UML. Ontologies, however, combine modelling with logic-based reasoning. Properties of concepts are specified

in terms of (universal or existential) quantifications over relationships to other concepts.

Formality in the Semantic Web framework facilitates machine understanding and automated reasoning. OWL (the Web Ontology Language), in particular OWL-DL, is equivalent to a very expressive description logic [21]. This fruitful connection provides well-defined semantics and reasoning systems. Description logic is particularly interesting for the software engineering context due to a correspondence between description logic and dynamic logic (a modal logic of programs).

2.3 Model-Driven Architecture

Model-driven architecture (MDA) is a software architecture approach emphasising the importance of modelling for the architectural design of software systems [1]. The platform targeted by MDA are service-based architectures. MDA suggests a three-layered approach:

- The Computation Independent Model (CIM) describes a system from the computation-independent viewpoint, addressing structural aspects of the system. A CIM is often called a domain model.
- The Platform Independent Model (PIM) can be seen as defining a system in terms of a technology-neutral virtual machine or a computational abstraction.
- The Platform Specific Model (PSM) usually consists of a platform model that captures the technical concepts and services that make up the platform and an implementation-specific model geared towards the concrete implementation technique.

The archetypical OMG MDA is based on UML for platform independent modelling and CORBA as the platform with its languages such as IDL as the platform-specific notation. In our context, the platform is a service-based infrastructure. Different platform types can be distinguished. The generic platform is SOA here, the technology-specific platform is the WSF, and vendor-specific platform technologies are for instance the Apache Axis or Collaxa service engines.

3 Modelling with Ontologies

MDA proposes three modelling layers – each with a distinct focus that, as we aim to demonstrate, can be supported ontologically.

- The computation-independent layer focuses on domain capture.
- The platform-independent layer focuses on architecture configuration and service process composition.
- The platform-specific layer focuses on interoperability and discovery support.

A case study from the banking domain will accompany our investigations.

3.1 CIM – Computation Independent Model

The purpose of the Computation Independent Model (CIM) is to capture a domain with its concepts and properties. Typically, two viewpoints of domain modelling can be distinguished. Concepts are represented in form of hierarchies – called the information viewpoint in MDA. Behaviour is represented in a process-based form – called the enterprise viewpoint in MDA, based on open distributed processing (ODP) concepts. Our aim is to provide a single ontological notation that can capture both viewpoints. A process-oriented ontology shall capture both types of domain entities, see Fig. 1.

- Two types of concepts shall be distinguished: objects, which are static entities, and processes, which are dynamic entities.

Domain ontology (OWL-style) – CIM layer

```

model          ::= relationship* | constraint*
concept        ::= object | process
relationship_type ::= is_a | has_part | depends
relationship    ::= concept relationship_type concept
constraint     ::= conceptConstraint | relationshipConstraint

```

Service process ontology (WSPO) – PIM layer

```

model          ::= pre process post
process        ::= preState procExpr postState
pre            ::= preState preCond condition |
                 preState inObj syntax
post           ::= postState postCond condition |
                 postState outObj syntax
procExpr       ::= process | ! procExpr | procExpr ; procExpr |
                 procExpr + procExpr | procExpr || procExpr

```

Service ontology (WSMO) – PSM layer

```

model          ::= service
service        ::= interface capabilities
interface      ::= messageExchange nonFuncProp
capabilities   ::= preCond postCond assumption effect nonFuncProp

```

Fig. 1. Abstract Syntax of Ontologies

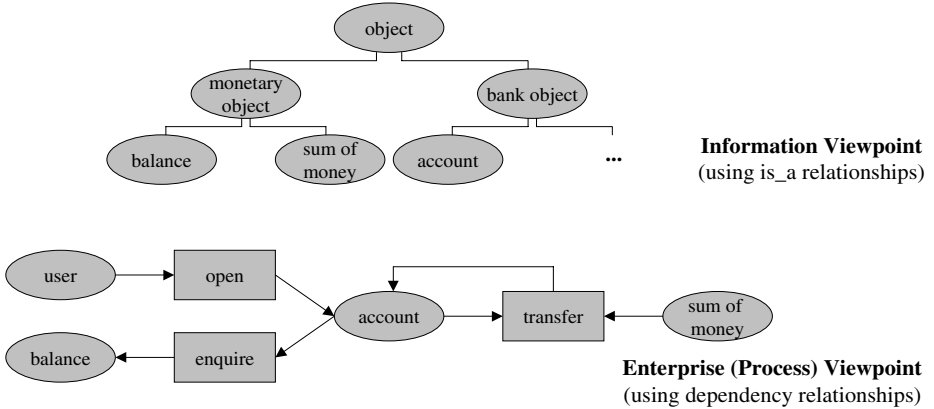


Fig. 2. CIM-level Excerpts from a Banking Domain Ontology

- Three relationship types shall be distinguished: is_a (the subclass relationship), has_part (the component relationship), and depends (the dependency relationship).
- Constraints, or properties, on concepts and relationships can be expressed.

The subclass relationship is the classical form of relating concepts in ontologies. For domain-specific software systems, the composition of objects and processes from a component perspective is an additional, but also essential information. Dependencies are useful to describe input-output relationships between objects and activities that process them. Specific ordering requirements on composed processes can be expressed through constraints. The abstract syntax of this ontology language is summarised in Fig. 1, upper part. We will discuss the semantics at the end of this section.

We need to define or identify an ontology language that can provide the necessary notational framework. An OWL-based ontology with support for the component and dependency relationships can form the notational framework here.

Example 1. The example that we will use to illustrate the modelling and transformation techniques throughout the paper is taken from the banking domain. We can identify:

- objects such as *account* and *sum* (of money),
- activities such as *account create*, *close*, *lodge*, *transfer*, and *balance* and processes such as for instance *create*; *!(balance + lodge + transfer)*; *close* which describes sequencing, iteration, and choice of activities¹,
- constraints such as a precondition $balance \geq sum$ on the transfer activity.

¹ The process combinators are ';' (sequential composition), '!' (iteration), '+' (choice), and '||' (parallel composition).

The example in Fig. 2 shows a simplified domain ontology for the bank account example.

Reasoning facilities of an ontological framework can be deployed to check the consistency of ontologically defined domain models.

Example 2. With instances attached to the entities, an inference engine can, for example, determine all bank account instances with a negative account balance. Another example is the satisfaction of a precondition for a money transfer on a particular account.

3.2 PIM – Platform Independent Model

The Platform Independent Model (PIM) changes the focus from the computation-independent capture of the domain to a focus on constraints imposed on the knowledge representation by the computational environment. Architectures and processes are here the key aspects at this modelling level. The architectural focus is on services, their architectural configuration, and interaction processes [13, 22]. Architectural configuration addresses the interaction processes (remote invocation and service activation) between different agents in a software system. Again, we will use an ontology to express these aspects.

Services are the components of the system architecture. They form the starting point of architecture modelling. Different approaches for service ontologies have been proposed. These differ in the way service and processes are represented in the ontologies – see Section 6 for a more detailed review. Since representing not only services, but also their configuration and assembly into processes is important here, we use the Web Service Process Ontology (WSPO), whose foundations were developed in [23, 24]. This ontology will bring us closer to the architectural perspective than more abstract service ontologies such as OWL-S [20], which however also provides support for service composition. Services (and processes) in WSPO are not represented as concepts, but as relationships denoting accessibility relations between states of the system. A PIM service process template, see Fig. 3, defines the basic structure of states and service processes.

The abstract syntax of this ontology is presented in Fig. 1, middle part.

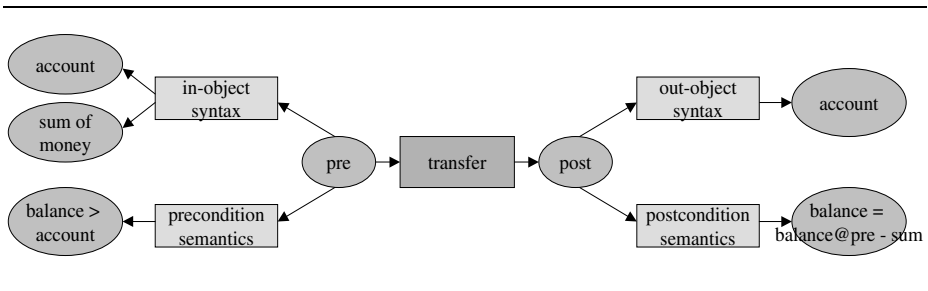


Fig. 3. Ontological Service Process Template (WSPO)

- Concepts in this approach are states (pre- and poststates), parameters (in- and outparameters), and conditions (pre- and postconditions).
- Two forms of relationships are provided. The processes themselves are called transitional relationships. Syntactical and semantical descriptions – here parameter objects (syntax) and conditions (semantics) – are associated through descriptonal relationships.

This ontological representation in WSPO is actually an encoding of a simple dynamic logic (a logic of programs) in a description logic format [23, 24], allowing us to avail of modal logic reasoning in this framework.

WSPO provides a standard template for service or service process description. Syntactical parameter information in relation to the individual activities – to be implemented through service operations – and also semantical information such as pre-conditions like are attached to each activity as defined in the PIM template. Example 4 will illustrate this. 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, sequential and parallel composition, and choice operators. Secondly, it adds data to processes in form of parameters that are introduced as constant process elements into the process sublanguage.

Example 3. The architecture- and process-oriented PIM model of the bank account focuses on the activities and how they are combined to processes. The process $create;!(balance + lodge + transfer);close$, which describes a sequence of account creation, an iteration of a choice of balance enquiry, lodgment, and transfer activities, and a final account closing activity, can be represented in WSPO as a composed relationship expression:

$$\begin{aligned} & create \circ acc; \\ & ! (balance \circ acc; lodge \circ (acc, sum); transfer \circ (from, to, sum)); \\ & close \circ acc \end{aligned}$$

Ontologies enable reasoning about specifications. WSPO enables reasoning about the composition of services in architectures. In [23], 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.

Example 4. Assume that in order to implement an account process, a transfer service needs to be integrated. For any given state, the process developer might require²

$$\begin{aligned} & \forall preCond . (balance > amount) \\ & \forall transfer . \forall postCond . \\ & \quad (balance() = balance()@pre - amount) \end{aligned}$$

² The @-construct refers to the attribute in the prestate.

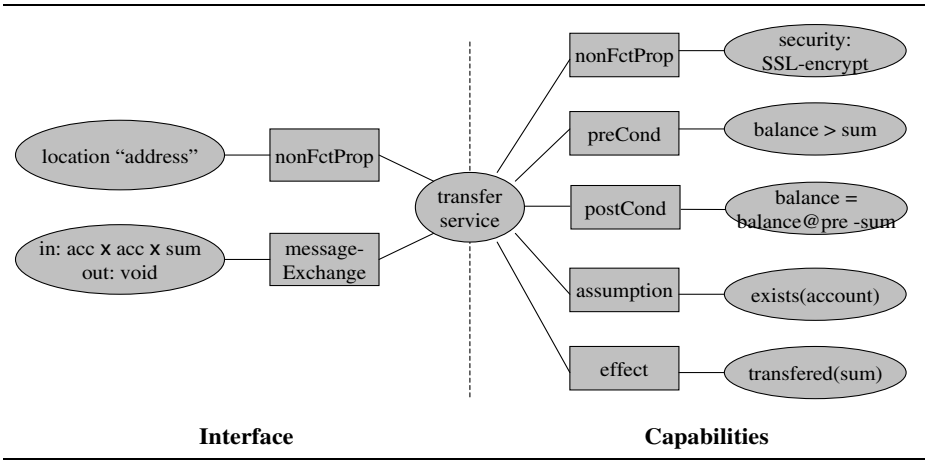


Fig. 4. Ontological Service Template (WSMO)

which would be satisfied by a provided service

$$\begin{aligned}
 &\forall preCond . true \\
 &\forall transfer . \forall postCond . \\
 &\quad (balance() = balance()@pre - amount) \wedge \\
 &\quad (lastActivity = 'transfer')
 \end{aligned}$$

based on a refinement condition (weakening the precondition and strengthening the postcondition).

The refinement notion used in the example above is based on the consequence inference rule from dynamic logic integrated into WSPO.

While architecture is the focus of this model layer, the approach we discussed does not qualify as an architecture description language (ADL) [26], 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.

3.3 PSM – Platform Specific Model

Our platform is the Web Service Framework (WSF) – consisting of languages, protocols, and software tools. Models for the platform specific layer (PSM) need to address two aspects: a platform model and implementation specific models. The platform model is here defined by the Web Services Framework and its service-oriented architecture principles. The implementation specific models characterise the underlying models of the predominant languages of the platform.

Interoperability of services is the key objective of the WSF. Two concerns determine the techniques used at this layer: the abstract description of services to support their discovery and the standardised assembly of services to processes. Two different models capturing executable and tool-supported languages are therefore relevant here:

- Description and Discovery. Abstract syntactical and semantical service interfaces shall be supported. The Web Services Description Language (WSDL) supports syntactical information. WSDL specifications can be created by converting syntactical information from the WSPO into abstract WSDL elements. We will, however, focus here on semantically enhanced descriptions enabled through service ontologies specific to the platform. Services as the basic components of processes can be represented as concepts in ontologies [27]. This approach is followed by widely used service ontologies such as OWL-S [20] and WSMO [25]. WSMO defines a template for the representation of service-related knowledge, see Figs. 1 and 4. The WSMO concepts are the central services concept and auxiliary domains for descriptive entities, i.e. expressions of different kinds. Relationships in the template represent service properties of two kinds. Properties such as *preCond*, *postCond*, *assumption*, and *effects* are called capabilities relating to the service semantics. Properties such as *messageExchange* are syntactically oriented interface aspects. In addition to these functional aspects, a range of non-functional aspects is supported.
- Processes and Composition. The Business Process Execution Language for Web Services (WS-BPEL) is one of the proposed service coordination languages [6]. WS-BPEL specifications can be created by converting process expressions from WSPO. We do not discuss this further here as the semantical support required here is already available at the PIM-level.

The benefit of using an ontology for description and discovery can easily be seen when the discovery and matching of OWL-S or WSMO-based service descriptions is compared with syntax-oriented WSDL descriptions.

Example 5. WSMO descriptions capture syntactical and semantical descriptions as WSPO does, see Examples 3 and 4. It adds, however, various non-functional aspects that can be included into the discovery and matching task. Standardised description and invocation formats enable interoperability. Required functionality can be retrieved from other locations. An example are authentication features for an online banking system.

3.4 Semantics of the Ontology Layers

The abstract syntax of the ontologies we discussed has already been presented in Fig. 1. Now, we focus on the semantics of the individual ontology layers. Ontologies are based on ontology languages, which in turn are defined in terms of logics. Here, we can exploit the description logic foundation of ontology languages such as OWL [21]. While a full treatment is beyond the scope of this paper,

we address the central ideas due to the definition of ontology transformations requires underlying formal semantical models.

Ontology languages are logics defined by interpretations and satisfaction relations on semantical structures such as algebras (sets and relations) and state-based labelled transitions systems (e.g. Kripke transition systems). A semantical metamodel for each of the layers can be formulated based on standard approaches in this context [21]:

- A domain ontology can be defined in terms of sets (for concepts) and relations (for relationships).
- The architectural and process aspects can be defined in terms of labelled transition systems, such as Kripke transition systems (KTS), where sets represent states and relations represent transitions between states.
- The interoperability aspects can be split into interface (defined in terms of sets and relations) and configuration and process behaviour (defined in terms of state transition mechanisms).

This shall form the basis for the approach presented here. In the future, we plan to map our semantics to the OMG-supported Ontology Definition Metamodel (ODM). This can be expected to be straightforward due to an ODM-OWL mapping as part of ODM. We will address this aspect in Section 5.

4 Ontology-Based Model Transformations

Without explicitly defined transformations, a layered modelling approach will not be feasible. The transformations play consequently a central role. Transformations between the model layers need to be automated to provide required tool support and to enable the success of the approach. Following the OMG-style of defining transformation, we define transformation rules based on patterns and templates.

4.1 Transformation Principles

While it is evident that the transformations we require here are about adding new structures, for instance notions of state and state transition for the architectural PIM layer, the original model should be recoverable and additional information on that layer should not be added. What we aim at is therefore not a refinement or simulation notion in the classical sense – although these notions will help us to define the transformations.

Refinements where additional application-specific model specifications are added can occur within a given layer. Refinement within a model layer can be based on subsumption, the central reasoning construct of ontology languages that is based on the subclass relation between concepts or relationship classes, respectively. In [23], we have developed a constructive refinement and simulation framework for service processes based on refinement and simulation relations as special forms of subsumption.

Rule Aspect	Description
CP0 template	For each process element in the CIM, create a PIM template.
CP1 process element	The PIM process element is the process element of CIM.
CP2 states	Create default concepts for pre- and post-states.
CP3 syntax	For each in- and out-parameter of processes, create a separate syntax (object) element.
CP4 semantics	Create pre- and postconditions depending on availability of external additional information in form of constraints.
CP5 process expressions	If process expressions available in form of constraints, then create complex process using relationship expressions in WSPO.

Fig. 5. Transformation Rules for the CIM-to-PIM Mapping

Our focus in this paper is the illustration of the different modelling capabilities of ontology languages and ontologies on the different model layers. Our objective is to motivate the need for and the benefits of a layered ontological modelling and transformation approach. A formal model of transformations is beyond the scope of this paper. Graph transformation and graph grammars provide suitable frameworks for this task [28, 29].

4.2 CIM-to-PIM Mapping

The CIM-layer supports abstract, computation-independent domain modelling. This model is mapped to a computation-oriented, but still platform-independent service-based model. The PIM-layer supports analysis and reasoning for architecture and process aspects, such as configuration and composition, on an abstract level. Consequently, information needs to be added to a CIM to provide a sufficient level of structure for the PIM-level. A process-specific PIM template, see Fig. 3 for a template application to the banking context, guides the transformation process. We have defined the rules for the CIM-to-PIM transformation in Fig. 5.

In MDA, the transformation steps are defined in terms of model markings and applications of templates. Marks are annotations (or metadata) of entities in the original model to support the mapping that indicates how these entities are used in the target model. Marks can support the determination of the mapping template to be deployed. The CIM-to-PIM transformation rule, that defines the creation of a PIM-template for CIM-concepts marked as 'process', is an example of this.

Example 6. Fig. 3 represents the result of the transformation of the 'transfer' process from Fig. 2 using the rules defined in Fig. 5. The 'transfer' concept in Fig. 2 is marked as a process, which based on rule CP0 creates a PIM process template with explicit states (rule CP2). The CIM concept 'transfer' becomes the transitional relationship element at the centre of the PIM template (rule

CP1). The input and output elements, associated to 'transfer' using dependencies (see Fig. 2), are mapped to syntax descriptions (rule CP2). Equally, additional constraints in the CIM are mapped to the PIM semantical descriptions (rule CP4).

Proposals for a mapping language are, similar to the ontology metamodel proposals, currently being requested by the OMG. Graph transformations and graph grammars [28, 29] would suit the need here to formalise the transformation rules. We have used graphs as the visualisation mechanism for ontological models. Graph-based models and CIM-to-PIM transformation semantics are therefore a natural combination. The semantics of a CIM can be seen as a directed labelled graph with nodes (objects and processes) and edges (relationships). The semantics of a PIM can be seen directed labelled graph, where descriptive and transitional roles are distinguished. This is equivalent to a KTS, see Section 3.4. This can be implemented as a graph expansion, where essentially state concepts are introduced. The original CIM can be retrieved by projecting on individual PIMs and then merging all process PIMs into one CIM.

4.3 PIM-to-PSM Mapping

The platform specific model (PSM) is defined in our approach by two separate models: service ontology descriptions to address service discovery and process orchestration and choreography descriptions to address service composition. The corresponding transformation rules for these two aspects – we chose WSMO for

Rule	Aspect	Description
PP1	WSMO	From the WSPO-based PIM, map process relationships to WSMO service concept and fill messageExchange and pre/postCond properties accordingly, see WSMO-template in Fig. 4.
PP1.1	WSMO messageExchange	Map the WSPO in and out objects onto WSMO message-Exchange descriptions.
PP1.2	WSMO pre-/postconditions	Map the WSPO pre- and postconditions onto WSMO pre-/postconditions and postconditions.
PP2	WS-BPEL	The complex WSPO process relationships can be mapped to BPEL processes.
PP2.1	WS-BPEL process partners	For each process create a BPEL partner process.
PP2.2	WS-BPEL orchestration	Convert each process expression into BPEL-invoke activities and the client side BPEL-receive and -reply activities at the server side.
PP2.3	WS-BPEL process activities	Convert the process combinators ';', '+', '!', and ' ' to the BPEL combinators sequence, pick, while, and flow, resp.

Fig. 6. Transformation Rules for the PIM-to-PSM Mapping

ontology-based description and WS-BPEL for service orchestration to illustrate this mapping – are presented in Fig. 6.

The WSPO-to-WSMO mapping copies functional properties – both syntax and semantics – to the PSM. Similar to states that are added to CIMs to provide the structure to express process behaviour, here we add structure in form of non-functional aspects to PIMs to add further descriptions for service discovery.

Example 7. The WSMO example in Fig. 4 is the result of mapping the PIM, presented in Fig. 3, to the WSF platform layer according to rule PP1 defined in Fig. 6. Syntactical elements for the interface and semantical capabilities such as pre- and postconditions are directly mapped from the corresponding WSPO elements according to the transformation rules PP1.1 and PP1.2.

The WSPO-to-WS-BPEL mapping converts process expressions into a BPEL business process skeleton, see Fig. 6. WS-BPEL is an implementation language. Process specifications in form of process orchestrations is supported by service engines available from various providers.

5 OMG-Adopted Technologies

Our efforts have to be seen in the context of the OMG approach to MDA. The OMG supports selected modelling notations and platforms through an adoption process. Example of OMG-adopted techniques are UML as the modelling notation and CORBA as the platform [1]. While Web technologies are not (yet) adopted, the need for a specific MDA solution for the Web context is, due to

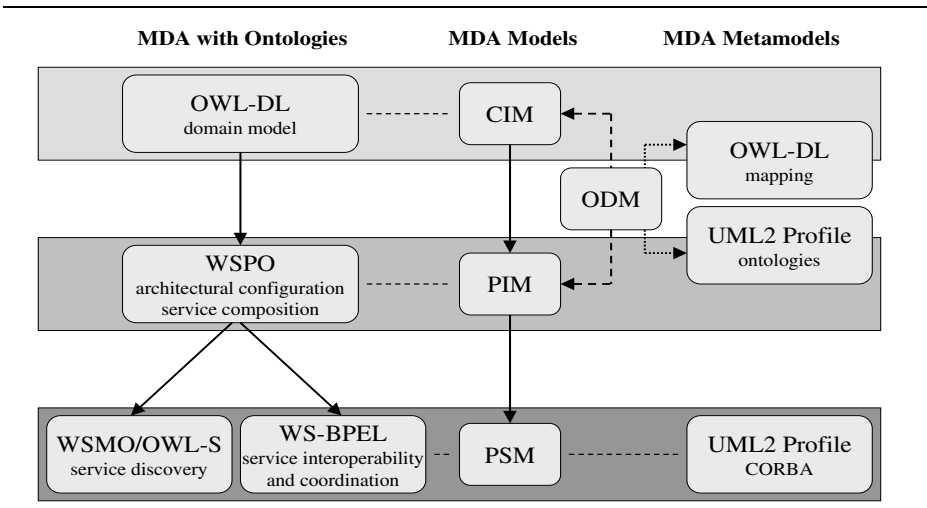


Fig. 7. Overview of MDA and Ontologies – with transformations between the layers and the influence of ODM for the ontology layers

the Web's ubiquity and the existence of standardised and accepted platform and modelling technology, a primary concern.

The current effort of defining and standardising an ontology metamodel (ODM) will allow us to integrate our technique further with OMG standards [9]. ODM will provide mappings to OWL-DL and also a UML profile for ontologies to make UML's graphical notation available. This might lead to a 'Unified Ontology Language' in the future, i.e. OWL in a UML-style notation [30]. A UML profile is about the use of the UML notation, which would allow ontologies to be transformed into UML notation. MOF compliancy for ODM is requested to facilitate tool support. XML, 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. We have summarised the MDA framework and compared it with our proposed extension in Fig. 7.

6 Related Work

Service ontologies are ontologies to describe Web services, essentially to support their semantics-based discovery in Web service registries. WSMO [25] and OWL-S [20] are the two predominant examples. 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 [23, 24] is also a service ontology, but the focus has shifted here to 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 already looked at the technical aspects of WSMO descriptions. WSMO supports the description of services in terms more abstract assumptions and goals and more concrete pre- and postconditions.

We have already discussed the OMG efforts to develop an ontology definition metamodel (ODM) in the previous section, which due to its support of OWL would allow an integration with UML-style modelling. ODM, however, is a standard addressing ontology description, but not reasoning. The reasoning component, which is important here, would need to be addressed in addition to the standard.

Some developments have started exploiting the connection between OWL and MDA. In [31], OWL and MDA are integrated. An MDA-based ontology architecture is defined, which includes aspects of an ontology metamodel and a UML profile for ontologies – corresponding to OMG's ODM. A transformation of the UML ontology to OWL is implemented. The work by [10, 31] and the OMG [1, 9], 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.

7 Conclusions

We have presented a layered ontological modelling and transformation framework for model-driven architecture (MDA). The effort leading towards model-

driven architecture acknowledges the importance of modelling for the architectural design of software systems. We have focused on two aspects:

- Firstly, ontologies are a natural choice to enhance modelling capabilities. While this is recognised in the community, we have exploited the new degree of sharing and ubiquity enabled through Web ontology languages and the reasoning capabilities of logic-based ontology languages.
- Secondly, ontology-based transformations allow the seamless transition from one development focus to another. These ontology transformations allow the integration of domain modelling, architectural design, the description and discovery of services.

Our approach addresses a Web-specific solution, reflecting the current development of the Web Services Framework and the Semantic Web. The primary platform we aim to support is the Web platform with the second Web services generation focusing on processes, utilising the Semantic Web with its ontology technology support. A platform of the expected importance in the future, such as the Web, requires an adequate and platform-specific MDA solution.

A critical problem that has emerged from this investigation is the need for conformity and interoperability. As MDA and the Web as a platform are developed and standardised by two different organisations (the OMG and the W3C, respectively), this can potentially cause problems. The current OMG developments, such as the Ontology Definition Metamodel (ODM), however, aim to reconcile some of these problems. With ODM soon to be available, our proposed ontologies can, due to their grounding in OWL, be expected to fit into the ODM.

Our aims here were to demonstrate the benefits and the feasibility of layered ontological modelling and transformation for service-oriented architecture, but a number of issues have remained open. We have developed a conceptual modelling and transformation framework. The automation of the transformation processes – central for the success of the technology – needs to be fully implemented. While we have developed some basic reasoning support specific to the architectural modelling activities, more techniques are also possible that exploit the full range of modal reasoning for service description, discovery, and composition and architectural configuration.

References

1. Object Management Group. *MDA Guide V1.0.1*. OMG, 2003.
2. World Wide Web Consortium. *Web Services Framework*. <http://www.w3.org/2002/ws>, 2004. (visited 08/07/2005).
3. World Wide Web Consortium. *Web Services Architecture Definition Document*. <http://www.w3.org/2002/ws/arch>, 2003.
4. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services – Concepts, Architectures and Applications*. Springer-Verlag, 2004.
5. E. Newcomer and G. Lomow. *Understanding SOA with Web Services*. Addison-Wesley, 2005.

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 08/07/2005).
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. McIlraith. 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. Object Management Group. *Ontology Definition Metamodel - Request For Proposal (OMG Document: as/2003-03-40)*. OMG, 2003.
10. D. Gašević, V. Devedžić, and D. Djurić. MDA Standards for Ontology Development – Tutorial. In *International Conference on Web Engineering ICWE2004*, 2004.
11. J. Williams and J. Baty. Building a Loosely Coupled Infrastructure for Web Services. In *Proc. International Conference on Web Services ICWS'2003*. 2003.
12. R. Allen and D. Garlan. A Formal Basis for Architectural Connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, 1997.
13. F. Plasil and S. Visnovsky. Behavior Protocols for Software Components. *ACM Transactions on Software Engineering*, 28(11):1056–1075, 2002.
14. L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice (2nd Edition)*. SEI Series in Software Engineering. Addison-Wesley, 2003.
15. 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.
16. R. Kazman, S.J. Carriere, and S.G. Woods. Toward a Discipline of Scenario-based Architectural Evolution. *Annals of Software Engineering*, 9(1-4):5–33, 2000.
17. W3C Semantic Web Activity. Semantic Web Activity Statement, 2004. <http://www.w3.org/2001/sw>. (visited 06/07/2005).
18. M.C. Daconta, L.J. Obrst, and K.T. Klein. *The Semantic Web*. Wiley, 2003.
19. 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.
20. 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.
21. F. Baader, D. McGuinness, D. Nardi, and P.P. Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
22. 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.
23. 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.
24. 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.
25. 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.

26. 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.
27. T. Payne and O. Lassila. Semantic Web Services. *IEEE Intelligent Systems*, 19(4), 2004.
28. J. Kong, K. Zhang, J. Dong, and G. Song. A Graph Grammar Approach to Software Architecture Verification and Transformation. In *27th Annual International Computer Software and Applications Conference COMPSAC'03*. 2003.
29. L. Baresi and R. Heckel. Tutorial Introduction of Graph Transformation: A Software Engineering Perspective. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. 1st Int. Conference on Graph Transformation ICGT 02*. Springer-Verlag, LNCS 2505, 2002.
30. 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.
31. D. Djurić. MDA-based Ontology Infrastructure. *Computer Science and Information Systems (ComSIS)*, 1(1):91–116, 2004.