

# Ontology Design Patterns for the Semantic Business Processes

Violeta Damjanovic  
Salzburg Research  
Jakob Haringer Strasse 5/II  
5020 Salzburg, Austria  
+43 622 2288427

violeta.damjanovic@salzburgresearch.at

## ABSTRACT

This paper discusses two research paradigms: the first one is based on using Meta-Object Facility (MOF) to support any kind of metadata, whereas the second one emphasizes the role of Ontology Design Patterns (ODPs) to support knowledge transformation between the source and the target models. More precisely, we represent the Business Process ODP, which reflects both the syntax and semantics of business process specification and brings an abstract solution to the typical problem of designing and modeling semantic business processes.

## Categories and Subject Descriptors

D.2.11 [Software Engineering]: Software Architectures – domain specific architectures, patterns.

D.3.3 [Programming Languages]: Language Constructs and Features – classes and objects, control structures, patterns.

## General Terms

Management, Design, Languages.

## Keywords

Ontology Design Pattern, Ontology, Semantic Business Process.

## 1. INTRODUCTION

This paper addresses how business processes and their execution can be improved based on ontology-supported specification of business processes that is known as Ontology Design Patterns (ODPs). We discuss certain issues with experiences from a recent research project in which was dealing with ontology engineering for designing mechatronic processes. One lesson learned is that domain experts in a specific area, e.g. mechatronic engineers cannot be expected to be experts in the domain of business process management. As a consequence, we have a situation in which the knowledge specification requires participation of both a domain, mechatronic engineer and a business process expert to support the following:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*SBPM 2009*, June, 2009, Heraklion, Crete, Greece.

Copyright 2009 ACM 1-58113-000-0/00/0004...\$5.00.

- design an ontology that supports automatic business process extraction and execution by the workflow engine;
- specify a mapping between the ontology concepts and properties, on the one hand, and business process constructs, on the other hand.

Thus, the mapping between conceptual models (ontology) and business process models requires both an ontology expert to understand the meaning of the ontology elements to be mapped, and a business process expert to recognize how the business process elements should be mapped into the ontology.

Secondly, we have found Meta-object Facility (MOF) and ODP to be useful paradigms for consolidation of different knowledge models, especially for the automation of the mapping between conceptual model and business process model and facilitating communication between the domain expert and the business process expert.

The emphasis of this paper is on the illustration of the Business Process ODP (BP ODP) that specifies dependencies of business process constructs in a declarative way, but includes concepts from host languages such as Business Process Execution Language (BPEL) and Web Service Description Language (WSDL).

The paper is organized as follows. In Section 2, the related work is classified with respect to two aspects: first, we have reviewed the ODP initiative, and second, we stay focused on the ontological representation of business processes. Section 3 discusses the research paradigms within which the idea of ontologizing business processes in the form of ODP has been modeled. Section 4 describes the BP ODP. Section 5 concludes and presents our thinking on the way forward for BP ODP.

## 2. RELATED WORK

Analogous to software engineering design patterns, ODPs bring similar advantages to the ontology engineers. They describe proven and documented solutions to a known modeling problem that repeatedly appears when designing different software systems [1]. The notion of ODP was introduced in 1999 for a particular problem domain in biology [2]. Afterwards, ODPs appeared under different names such as semantic patterns, knowledge patterns. The semantic patterns that represent a language independent description of certain concepts, relations, and/or axioms are described in [3]. The knowledge patterns as conceptual patterns that are “morphed” into a given knowledge base by using a set of mapping axioms are represented in [4], whereas both [5] and [6]

are more focused on designing patterns for Semantic Web ontologies that are now called ODPs.

The main similarities between traditional design patterns in software engineering and ontologies are emphasized in [7].

Today, there are two research communities around ODPs:

- the one is located at the University of Manchester, around the Gene Ontology Next Generation (GONG) project<sup>1</sup> in which ODPs are promoted to enable migrating current bio-ontologies to a richer and more rigorous level;
- the other one is the ODP community<sup>2</sup>, also known as *ontologydesignpatterns.org*, started under the NeOn project<sup>3</sup> that investigates both development lifecycle and evolution lifecycle of networked ontologies.

Both communities collect ODPs through online and public catalogs, support their evaluation, improvements and reuse, and both are open for collaboration and are growing.

Additionally, we are interested to explore into the existing work that connects ontologies and business processes, such as:

- [8]: it describes an ontology for executable business processes using the formalism of the Web Service Modeling Language (WSML);
- [9]: it proposes a set of ontologies and formalisms and defines the scope of these ontologies by giving competency questions that is a common approach in the ontology engineering;
- [10]: it explores the behavioral aspects of Web services and proposes using Description Logics (DL) for their formalization, i.e. the constraints between Web service' operations that define the allowed order of execution.

### 3. THE CENTRAL PARADIGMS AND MOTIVATION

This section attempts to explain the main research paradigms within which the importance of ODPs in designing semantic business processes is considered.

Our initial paradigms is based on using Meta-Object Facility (MOF) to define a metadata architecture for the Model-Driven Architecture (MDA) [11] [12]. The key purpose of using MOF is supporting any kind of metadata and allowing new kinds to be added as required.

The MOF consists of three layers (shown in Figure 1):

- M1: the model layer contains the definition of the required structures,
- M2: the metamodel layer defines the terms in which the model is expressed, and
- M3: the meta-metamodel layer that is provided by the programming environment.

With respect to the MOF, we have defined the transformation scenario shown in Figure 1. In this scenario, different technical spaces (TS) and three different MOF layers (M1, M2, M3) are connected (e.g. OWL TS, MOF TS and XML TS are connected at

the model layer (M1)). According to Object Management Group (OMG) "a TS is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities." [18]

Before defining the transformation models, we have identified the source and the target metamodels. Thus, on the left side of Figure 1, the OWL TS box contains OWL DL source model corresponding to the OMG Ontology Definition Metamodel (ODM) approach. The OWL DL source model uses an RDF/XML presentation syntax that has to be serialized into OWL RDF/XML exchange syntax. Because the process of deploying BPEL process requires both WSDL description of all business operations and BPEL description of business processes that orchestrate Web services, in the right side of Figure 1, the XML TS box is shown that contains two target models such as WSDL and BPEL models that correspond to WSDL and BPEL metamodels, respectively.

The MOF TS is located between OWL TS and XML TS. In the MOF TS, the source and the target models have to be transformed between OWL RDF/XML exchange syntax (*ecore* XML XMI format), on the one hand, and WSDL and BPEL, on the other hand. At the end, WSDL and BPEL *ecore* models have to be transformed and serialized into WSDL and BPEL presentation syntax that is based on XML. Transforming source (meta)models into target (meta)models requires the target knowledge capacity to be fully covered by the source knowledge models. Hence, our second research paradigm describes the way of supporting knowledge transformation between the source and the target models that is shown in Figure 1. Here we have noticed an intrinsic role of the design patterns in articulating knowledge that is expected to be exchanged between different data models. In our case, a source model is represented as an ontology model, whereas a target model is a non-ontology model (business process model). The process of articulating different data models is nontrivial in sense that requires both the ontology developers and the business process experts to collaborate in defining formal meaning of transformation. In case of transforming ontological models into non-ontological models, the process of articulating knowledge is known as - reengineering domains.

According to the classification of ODP given in [13], Reengineering Ontology Design Patterns (ReODPs) often describe the process of reengineering domains between ontologies and non-ontologies. Hence, in [14] two ReODPs called WSDL-DDPO and BPEL-DDPO are declaratively described. These patterns provide knowledge specification for business processes (designed by using WSDL description and BPEL specification of business processes) that are grounded on the DDPO module (DOLCE Descriptions and Situations (D&S) Plan and Tasks Ontology) that has been served as a formal framework for ontological description of business processes.

In this paper, for the sake of simplification of the implementation effort, we stay focused on designing BP ODP by considering only WSDL and BPEL language constructs for specifying business process behavior defined in [15]. The main role of BP ODP is facilitating the ontological representation of BPEL business processes, as well as making a shift toward automatic processing of such knowledge and its automatic execution on the workflow engine.

The BP ODP is illustrated in the next Section.

1 [http://www.gong.manchester.ac.uk/?page\\_id=7](http://www.gong.manchester.ac.uk/?page_id=7)

2 <http://ontologydesignpatterns.org/wiki/Odp>About>

3 <http://www.neon-project.org/web-content/>

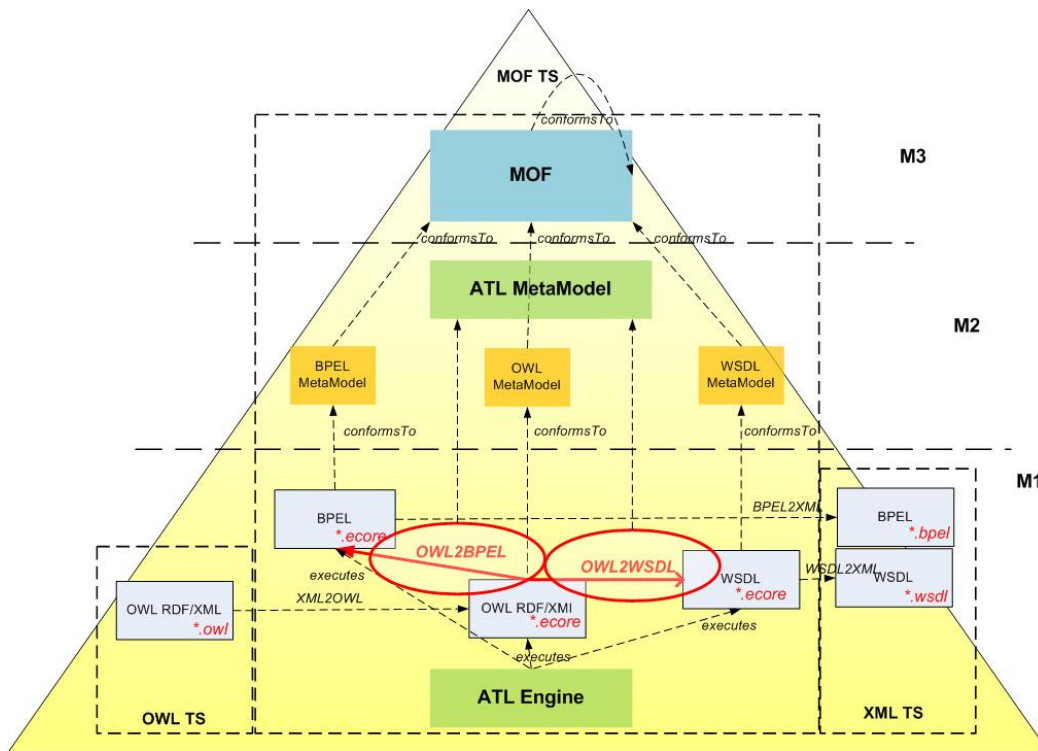


Figure 1. Transformation Scenario.

#### 4. THE BUSINESS PROCESS ONTOLOGY DESIGN PATTERN

Designing a BPEL process is not enough to make this process executable. Certain rules and mapping formalisms between ontologically described knowledge about business process and BPEL and WSDL definitions of business process must be defined to enable semantically meaningful execution. However, both BPEL and WSDL are not equipped with the formal semantics and cannot provide formally reasoning about process behaviour [16]. They cannot express semantic description of business process neither provides well-defined semantics for automated composition and execution of business process, what has motivated us to continue by looking further into the ODP as mechanisms able to semantically articulate knowledge about the execution of business process.

The BP ODP is based on the conceptualization of both BPEL and WSDL specification given in [16]. This means that each element of BPEL and WSDL languages has a conceptual equivalent in the BP ODP. For example, BPEL model the behaviour of both executable business processes and abstract business processes that are not intended to be executed and serve a descriptive role. At the same time, BPEL process represents all partners and interactions with these partners in terms of abstract WSDL interfaces (port types and operations).

The class hierarchy of BP ODP is shown in Figure 2. Furthermore, the BPEL executable processes consist of two classes of activities: basic and structured. Basic activities describe basic steps of the process behavior, whereas structured activities encode control-

flow logic and can contain other basic and/or structured activities recursively [16]. The class hierarchy of BPEL activities is shown in Figure 3.

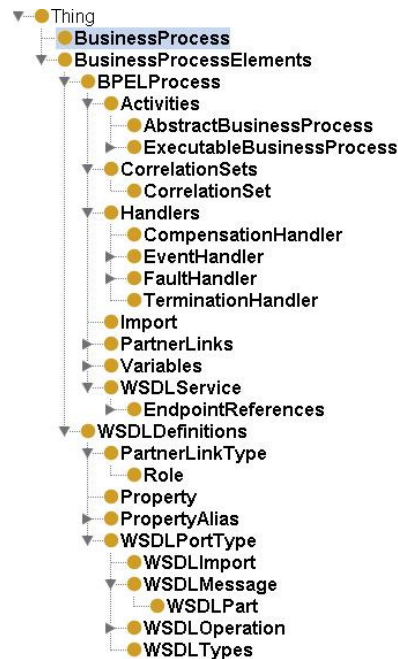


Figure 2. Business Process ODP Class Hierarchy.

## 4.1 Designing attributes and elements of BPEL activity in the Business Process ODP

Each BPEL activity has optional *standard attributes* that contains the name of the BPEL activity and the attribute `suppressJoinFailure` indicating whether a join fault should be suppressed if it occurs [16]. In the proposed BP ODP, we represent those attributes as datatype properties of `StandardAttribute` class. At the same time, each BPEL activity has optional containers (`Sources` and `Targets`) that contain optional *standard elements* represented by `Source` and `Target` classes of the BP ODP, respectively.

## 5. CONCLUSION AND FUTURE WORK

ODPs are usually small ontologies that solve complex modeling issues to semantic interoperability of different knowledge components. ODPs have their lifecycle which comprises the following three phases: (a) identifying ODP, (b) testing ODP, and (c) becoming a part of the system/language. In this paper, we have firstly identified the BP ODP that serve for (semi)automatic transformation of ontological knowledge into non-ontological knowledge about business processes. Then, we have presented some of the design decisions on BP ODP.

In our future work, we are particularly interested in improving BPEL opaque expressions by employing DL for formalizing behavioral aspects of semantic business processes, which causes the BP ODP to be semi-automatically applicable to business process execution. Some representatives of opaque expressions are as follows: a *transition condition* of `Source`; a *join condition* of `Targets`; a *condition* element of `While`, `RepeatUntil`, `If`, `ElseIf`; a *duration expression* or *deadline expression* in `Wait` activity; an *unsigned integer value* in `StartCounterValue`, `FinalCounterValue` of `ForEach` activity. Despite the fact that DLs for reasoning about Web services (business processes) can lead to semantic and computational problems, DLs plays an important role in the Semantic Web since they are the basis of the OWL and forms the core of OWL-DL. Hence, we are interested to explore DL reasoning in  $\mathcal{L}$  extended by nominals as a formalism describing the functionality of Web services that is proposed in [17].

## 6. ACKNOWLEDGMENTS

Some of the material in this paper is based upon work supported by the ImportNET project, within the Sixth Framework Programme and by the Interactive Knowledge Stack (IKS) project, within the Seventh Framework Programme.

## 7. REFERENCES

- [1] Aranguren, M. E. 2005. Ontology Design Patterns for the Formalization of Biological Ontologies. M.Sc. thesis at the University of Manchester, Department of Computer Science.
- [2] Reich, J. R. 1999. Ontological Design Patterns for the Integration of Molecular Biological Information. In Proceedings of the GCB'99. Germany.
- [3] Staab, S., Erdmann, M., and Maedche, A. 2001. Engineering Ontologies Using Semantic Patterns. In Proceeding of the IJCAI'01. USA
- [4] Clark, P., Thompson, J., and Porter, B. 2003. Knowledge Patterns. International Handbooks on Inf. Systems. Springer.
- [5] Svatek, V. 2004. Design Patterns for Semantic Web Ontologies: Motivation and Discussion. In Proceedings of the 7th Conf. on Business Information Systems. Poland.
- [6] Gangemi, A. 2005. Ontology Design Patterns for Semantic Web Content. In Proc. of the ISWC 2005. LNCS 1729.
- [7] Devedzic, V. 2002. Understanding Ontological Engineering. Communications of the ACM 45(4): 136-144.
- [8] Nitzsche, J., Wutke, D., and van Lessen, T. 2007. An Ontology for Executable Business Processes. In: M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, N. Stojanovic (eds.): Semantic Business Process and Product Lifecycle Management. Proceedings of the Workshop SBPM 2007, Innsbruck, CEUR Workshop Proceedings, ISSN 1613-0073.
- [9] Hepp, M. and Roman, D. 2007. An Ontology Framework for Semantic Business Process Management. In Proceeding of the 8<sup>th</sup> International Conf. Wirtschaftsinformatik. Germany.
- [10] Fronk, M. and Lemcke, J. 2006. Expressing Semantic Web Service Behavior using Description Logics. In Proceeding of the ESWC 2006 Workshop on SBPM 2006, Montenegro.
- [11] Brockmans, S. and Haase, P. 2006. A Metamodel and UML Profile for Rule-extended OWL DL Ontologies – A Complete Reference. Universität Karlsruhe (TH), Technical Report. Online available: <http://www.aifb.uni-karlsruhe.de/WBS/sbr/publications/owl-metamodeling.pdf>
- [12] ATL 2006. ATL User Manual, ver. 0.7. Online available at: <http://www.eclipse.org/m2m/atl/doc/>
- [13] Presutti, V. 2008. NeOn D2.5.1: A Library of Ontology Design Patterns: Reusable Solutions for Collaborative Design of Networked Ontologies. NeOn Project.
- [14] Damjanovic, V. 2009. Reengineering Patterns for Ontologizing Business Processes. Submitted to the I-Semantics 2009 on April 06, 2009 (under review).
- [15] WS-BPEL 2007. WS-BPEL Ver2.0. Online available at: [http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html#\\_Toc164738488](http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html#_Toc164738488)
- [16] Lucchi, R. and Mazzara, M. 2006. A Pi-Calculus Based Semantics for WS-BPEL. Journal of Logic and Algebraic Programming (JLAP), 70: 96-118.
- [17] Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F. 2005. A Description Logic Based Approach to Reasoning about Web Services. In Proceedings of WWW 2005 Workshop on Web Service Semantics (WSS2005)
- [18] Bezivin, J. et al., 2003. First Experiments with the ATL Model Transformation Language: Transformation XSLT into XQuery. In Proc. of the OOPSLA 2003 Workshop on Generative Techniques in the Context of Model-Driven Architecture.