sbpm06
Budva, Montenegro
Semantics for Business Process Management 2006
Workshop at the 3rd European Semantic Web Conference 2006
11-14 June 2006

Knut Hinkelmann
Dimitris Karagiannis
Nenad Stojanovic
Gerd Wagner
(Editors)

Proceeding of the Workshop on

Semantics for Business Process Management

Workshop at the 3rd European Semantic Web Conference
11 June 2006, Budva

Content

Automatic User Support for Business Process Modeling

Stefanie Betz, Stefan Klink, Agnes Koschmider, and Andreas Oberweis

Institute of Applied Informatics and Formal Description Methods
Universität Karlsruhe (TH), Germany
{betz,klink,koschmider,oberweis}@aifb.uni-karlsruhe.de

Abstract. The main purpose of business process modeling is the representation and analysis of alternative process designs by formal or semiformal process models. Manual modeling of business processes is a time-consuming task. Typos and structural modeling errors make it particularly error prone to model business processes manually. Users can be assisted in modeling business processes by providing an autocompletion mechanism during the modeling process. In this paper we will describe on-going work for autocompletion of business process models. This approach is based upon an OWL DL description of Petri nets. Our autocompletion mechanism requires validation methods to check process properties of the automatically completed business process, which we will introduce as well. Consequently, we aim to improve manual process modeling by automating process modeling to a significant extent.

1 Introduction

The main purpose of business process modeling is the representation and analysis of alternative process designs by formal or semiformal process models. Many modeling languages – most of them being based on textual programming languages or graphical notations such as Petri nets [20], EPCs [23] or BPMN [27] – have been proposed for process modeling. Novel orchestration- and choreography languages such as BPEL [1] focus on tracking and executing business processes by business applications. To enable verification of BPEL [8] proposes a Petri net semantics. Petri nets have been established as a suitable language for modeling business processes with intuitive graphical notation. Furthermore, Petri nets have a mathematical foundation, which enables simulation and analysis of system behavior.

In this paper we will describe on-going work for autocompletion of Petri net based business process models. Manual modeling of business processes is a time-consuming task. Typos and structural modeling errors make it particularly error prone to model business processes manually. Users can be assisted in modeling business processes by providing an autocompletion function during the modeling process. However, process element names might differ in syntax even when they have the same meaning (homonyms) or one process can be modeled in different ways even when utilizing the same modeling language. It is possible that these process elements will not be suggested as fitting elements to provide autocompletion.

To solve ambiguity issues caused by the use of different names for describing the same tasks a machine readable and interpretable format, which might be used for machine reasoning, is required for Petri nets. Business processes modeled with Petri nets can be translated into the Web Ontology Language OWL [17], an unambiguous format which allows ontological reasoning. So-called semantic business process models combine process modeling methods with semantic technologies to achieve automatic processing of business process models instead of manual processing. We will use a semantic description of Petri nets to make it easier to find appropriate process templates (reference processes), which can be proposed for autocompletion. During the modeling process, a recommendation mechanism determines possible subsequent fragments of all templates by computing similarities. If the system detects a high similarity between one element of a template and a modeling element, then subsequent elements of this element template are proposed for autocompletion. To ensure correct process flow behavior the system has additionally to check properties such as deadlock-freeness.

The structure of this paper is as follows. Firstly, we will recall the main notions of Petri nets and a semantic description of Petri nets with OWL DL. In Section 3 we will describe an approach for measuring similarity between semantic business process models by utilizing *syntactic-*, *linguistic-* and *structural* similarity measurements. To validate process behavior properties we will illustrate analysis methods in Section 4. Modeler's behavior can be observed and learned with machine learning techniques, which we will briefly survey in Section 5. Section 6 concludes the paper with an outlook on future research.

2 Foundations

Next subsection introduces Petri net notation and Petri net modeling of business processes.

2.1 Petri nets

Petri nets are a graphical language and a formalism used to model business processes and verify system behavior. Formally, a Petri net is a directed bipartite graph with nodes and arcs. It can be described by the triple $N = (P, T, F)$, where P is a set of places, T is a set of transitions (which is disjoint from P) and $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation. Elements of P are graphically represented as circles, elements of T as boxes and elements of F as directed arcs between places and transitions. A place p is an input place of a transition t, if there exists a directed arc from p to t. A place p is an output place of a transition t, if there exists a directed arc from t to p. The set of all input places of a transition t is denoted by $\bullet t$ and is called *preset*. The set of all output places is denoted by $t\bullet$ and is called *postset*.

Numerous Petri net variants have been proposed, which can be subsumed in elementary or high-level Petri nets. In elementary Petri nets places contain tokens, which represent anonymous objects. A transition t is enabled to change the marking m of the net, if each place p in the preset contains at least one token. If such a transition t occurs, then t consumes tokens from the preset and inserts tokens in the postset. The occurrence sequence, which leads from m to m' is defined by $m \overset{t}{\rightarrow} m'$. A marking m' is called *reachable* from m, if there exists an occurrence sequence from m to m' ($m \overset{\delta}{\rightarrow} m'$) with $\delta = t_1, t_2 \ldots t_n$.

For modeling system behavior or business processes with Petri nets we distinguish several process flow structures [26] as depicted in Figure 1. The flow structure OR-split allows to model alternative branching. The two alternative branches are again integrated by a so-called OR-join or a so-called AND-join (synchronization). By the use of AND-split (concurrency) tokens are distributed to two places.

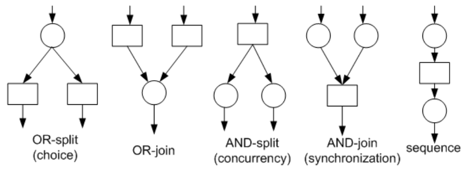


Fig. 1. Overview of Process Flow Structures

The formal foundation of Petri nets allows to verify whether a modeled business process meets certain properties such as deadlock-freeness. If a marking is reached, which is not an intended final process state and which doesn't enable any transition, then the process is in a deadlock, as depicted in Figure 2.

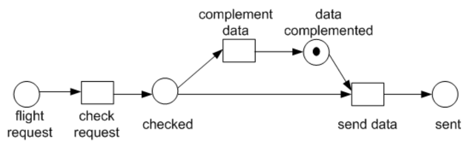


Fig. 2. Deadlock in a Petri net

To make tokens distinguishable, variants of high-level Petri nets have been proposed such as Coloured Petri nets [6] or Predicate/Transitions nets [10]. In this paper, we focus on Predicate/Transitions nets (Pr/T nets) where places are interpreted as predicates representing relation schemes. Transitions occur according to a logical expression which may be attached to them.

2.2 Semantic Business Process Models

To describe business processes modeled with Petri nets in an unambiguous format we have proposed a translation of Pr/T nets to OWL DL [13]. For our work we will refer to OWL DL (Description Logic) in order to be able to use available off-the-shelf reasoning technologies. OWL DL can be considered as a syntactic variant of the $\mathcal{SHOIN}(D)$ Description Logic which is known to be decidable [9]. In Table 1 the main constructs of $\mathcal{SHOIN}(D)$ are shown.

Table 1. Constructs of $\mathcal{SHOIN}(D)$ Syntax

subClass	$A_1 \sqsubseteq A_2$
intersection	$A_1 \sqcap ... \sqcap A_n$
union of	$A_1 \sqcup ... \sqcup A_n$
allValuesFrom	$\forall$ P.A
someValuesFrom	$\exists$ P.A
maxCardinality	$\leq$ nP
minCardinality	$\geq$ nP

In our Pr/T net ontology each Petri net element corresponds to an OWL concept. *Places* are described by the concept *Place*, *transitions* by *Transition* and *arcs* by $FromPlace$ $(P \times T)$ and $ToPlace(T \times P)$. For instance, the concept PetriNet is defined by at least one transition, place and arc.

Table 2. Pr/T net Ontology

$PetriNet \equiv$	$\geq 1 hasNode.(Transition \sqcap Place)$
	$\sqcap \geq 1 hasArc.(FromPlace \sqcup ToPlace)$
$Transition \equiv$	$placeRef.Place \sqcap = 1 haslogicalConcept.LogicalConcept$
$Place \equiv$	$transRef.Transition \sqcap = 1 hasMarking.IndividualDataItem$
$FromPlace \equiv$	$\geq 1 hasInscription.Delete \sqcap \exists hasNode.Place$
$ToPlace \equiv$	$\geq 1 hasInscription.Insert \sqcap \exists hasNode.Transition$
$LogicalConcept \equiv$	$= 1 hasConditon.Condition \sqcup = 1 has.Operation.Operation \sqcap$
	$\exists hasAttribute.IndividualDataItem$
$IndividualDataItem \equiv$	$\geq 1 hasAttribute.Attribute$
$Delete \equiv$	$\forall hasAttribute.IndividualDataItem$
$Insert \equiv$	$\exists hasAttribute.IndividualDataItem$
$Atrribute \equiv$	$\leq 1 hasValue.Value$
$Value \equiv$	$hasRef.Value$
$Condition \equiv$	$forall(string) \sqcup exists(string) \sqcup and(string)$
$Operation \equiv$	$function(string)$

To the best of our knowledge, there exists no other approach that transforms high-level Petri nets into OWL DL. In the next section we will explain how to measure similarity between a pair of semantic business process model elements.

3 Measuring Similarity between Process Elements

Before proposing appropriate process fragments, the recommendation mechanism has to compare modeling elements with process templates and has to find similar elements, which will be proposed as fitting subsequent elements. Thus, the mechanism has to compare process templates with process elements, which are currently modeled, by computing their similarities. The autocompletion system observes and learns from the modeler's behavior. Figure 3 gives an overview of our recommendation process[1]. Instead of representing OWL syntax, which is not readable for modelers, we have depicted business processes in graphical Petri net notation. However, the automatical similarity computation between process elements is based upon the OWL serialisation of Petri nets.

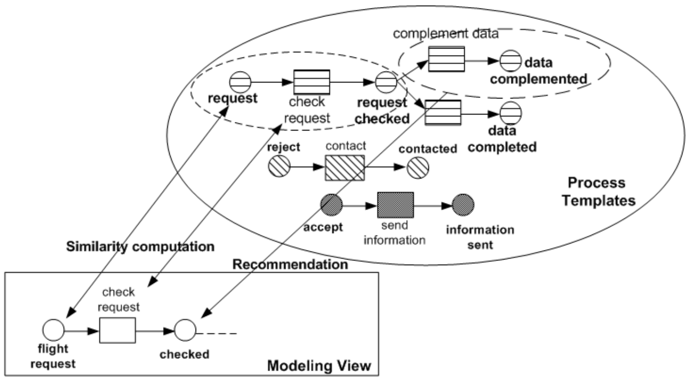


Fig. 3. Overview of the Auto Completion process

Our approach for automatic process element propositions is similar to the autocompletion function in mobile phones. The installed system observes what the user is currently typing, and tries to complete words automatically.

We propose to compute similarities between two process element names by utilizing the similarity measures *syntactical*, *linguistic* and *structural* measurements. Syntactical similarity measures take as input two character strings and compare them. A well-established string similarity measurement has been proposed by Levenshtein [14] which takes into account the amount of operations (insertion, deletion and substitution) needed to transform one string into another. For example, to turn *flight_request* to *request* requires seven deletions. Based on the Levenshtein method [16] has proposed a syntactic similarity measurement which returns for the similarity calculation similarity degrees between 0 and 1. But, syntactical similarity measurements alone are not sufficient since they do not regard the semantics of words. In addition to the syntactical similarity measurement we are utilizing background knowledge in terms of ontologies.

[1] different shading of process templates visualizes independent process models.

Petri nets obey an operational semantics that describes the control flow. However, a missing semantic description of Petri net components hampers the automated processing of process elements. To uncover synonyms or homonyms of process element names we need a description of Petri nets in an umambiguous format such as OWL DL. With the background ontology – numerous background ontologies are modeled with OWL – we compute linguistic similarity measures. Ontologies have paved the way for standardized formal conceptualizations of all kinds of knowledge. To compute linguistic similarity degrees we have worked with WordNet[2] via the JWNL API [5] and with a specific UML Profile [4]. The benefit of the UML-based background ontology is that a converting tool provides an automatic translation from the visual UML modeling to OWL DL syntax. WordNet is in contrast to the UML Profile predefined and fix.

However, to improve the aggregation of syntactical and linguistic similarity measurements we take into account the context of element names by considering structural information of names. For instance, the structure of place names is influenced by places attributes, values and the subsequent transitions as depicted in Figure 4 exemplarily for the place *flight request*. This place has a marking with the name *R_flight_request* and the corresponding attributes *Name*, *Destination*, *Date* and *Quantity* where *Destination* has the values *PAR* (corresponding to the destination Paris) and *FRA* (corresponding to the destination Frankfurt). The subsequent transition of *flight request* is *check request*.

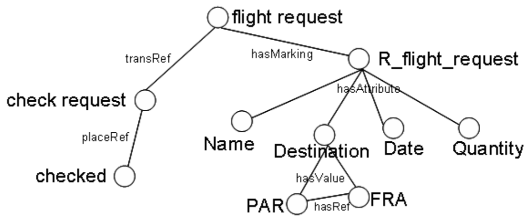


Fig. 4. Context of the Place *flight request*

However, each concept influences place names with different weights. Therefore, we have determined different weights for the concepts as shown in Table 3. In business process models with a lot of places, attributes and transitions weights play a less important role than in small processes with less process elements. The more instances are modeled in a SBPM the more extensive is the context of instances. The processes modeled in this paper have only few places and transitions[3]. Furthermore, depending on the features the similarity measure might differ. To determine structural similarity between place names we compute the syntactical similarity degree of names (measure for names is syn-

[2] an english online lexical reference system, which provides synonym and hypernym sets consisting of nouns, verbs, adjectives, and adverbs [18].

[3] To make the business processes readable we did not assign values for attributes

tactical similarity) and the linguistic similarity degrees of their attributes, values and subsequent transitions.

Table 3. Features and Similarity Measures for Petri nets

Comparing	Feature	Measure	Weight
Places	name	synt sim.	0.2
	Attribute/Value	str sim.	0.5
	successor	ling sim.	0.3
Attributes	name	synt sim.	0.2
	sibling Attribute	ling sim.	0.3
	Values	ling sim.	0.3
	Place	ling sim.	0.2
Values	name	synt sim.	0.2
	Attribute	ling sim.	0.5
	Value reference	ling sim.	0.3
Transitions	name	synt sim.	0.2
	ToPlace	sling sim.	0.4
	FromPlace	ling sim.	0.4

By aggregating these three similarity measures to a combined similarity measure we consider syntactical, linguistic, and structural properties of elements and can compute a more significant similarity between two elements. Table 4 shows some combined similarity degrees (sim_{com}) for process names ($name_{BP}$) and fragments ($name_{BF}$). If the algorithm computes a combined similarity degree > 0.5 between a template element and a process element, which is currently modeled, then the recommendation system proposes it for autocompletion. The user can then decide if (s)he accepts the proposition or rejects it. To learn a threshold θ instead of using a fix value we will present in Section 5 several machine learning techniques.

Table 4. Results of Combined Similarity

$name_{BP}$	$name_{BF}$	sim_{com}
flight request	request	0.8
check request	check request	1.0
request checked	checked	0.9
...	...	...

In the following section we will sketch a method for validating behavior properties of autocompleted business processes.

4 Analysis Methods for Petri nets

To check if a process model meets certain properties, several analyzing methods have been proposed [25]. In our approach we utilize such methods to validate that the insertion of the proposed process fragments does not cause deadlocks and synchronization errors. Thus, our automatic user support includes more than the recommendation of process fragments.

In the following we are focusing on validation of process properties by checking if the autocompleted business process is deadlock free and without lack of synchronization. While utilizing validation algorithms for Pr/T nets we are not considering any inscriptions of places, transitions, or arcs. Instead we are focusing on the net structure of the modeled process by identifying all possible flows based on instance subgraphs [22] (see Figure 5 where we have modeled the instance subgraph for Figure 2). We assume, that the analysis of a specific Pr/T net satisfies the following requirements: the Pr/T net has only one source place and one sink place, every place p and every transition t is on the path between the source and the sink [24]. Additionally, cycles in the net structure must be regarded as single execution units.

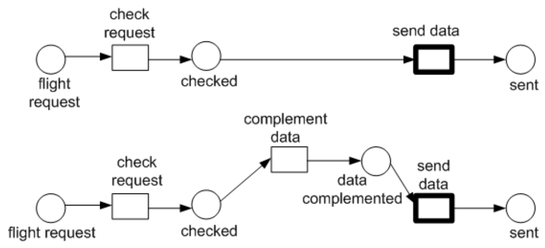


Fig. 5. Instance Subgraphs of Petri net in Figure 2

A deadlock can occur, if branches of an OR-split (such a split can be easier identified by an instance subgraph) are synchronized by an AND-join, as depicted in Figure 2. For instance, if the user inserts the transition *complement data* and its subsequent place *data complemented* and intends to synchronize these elements by inserting a connection from *checked* and *data complemented* to *send data*, then a deadlock occurs.

A lack of synchronization occurs, if an AND-split is synchronized by an OR-join, as depicted in Figure 6. For instance, if the user inserts the transition *travel request* and intends to synchronize the concurrent branches by inserting the place *transport request*, then a lack of synchronization occurs. To facilitate manual modeling we aim to develop a recommendation mechanism, which advices only validated fragments.

We have sketched how deadlocks and lack of synchronization can be discovered and now we will introduce reduction rules to validate both process properties [22], [15], [25]. According to the reduction rules a process is free of structural

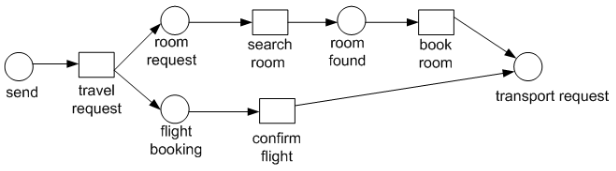


Fig. 6. Example for a Lack of Synchronization

errors if the reduction results in an empty graph. Because of the simplification of the presented processes, we applied only two of the rules, the terminal reduction rule (trr) and the sequential reduction rule (srr) as shown in Figure 7. At the end there are two nodes left over, so the modeled process is not free of deadlocks.

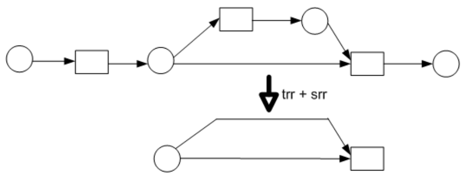


Fig. 7. Execution of the Reduction

5 Learning User Behavior

As described in Section 3 our system is capable of recommending similar elements and the user is asked to select appropriate templates. For further recommendations the users behavior could be observed, i.e., logging which decisions have been made in which situation/context and at which process position or template.

To recommend appropriate elements, a classification method must be utilized to classify the set of stored elements into partitions. These partitions can then be compared with the current user-behavior and the best one(s) concerning context and other criteria as described above will be recommended for selection. Several methods can be utilized to learn user's behavior. The following list gives an overview of established learning techniques.

Support Vector Machines The goal of a support vector machine (SVM) is a classifier in a high-dimensional feature space which can handle even not-separable problems. SVMs can be used to find hypotheses which guarantee a minimal error [11]. SVMs determine those coefficients which separate the training set with the shortest distance vectors. Only the so-called *support vectors* are of interest and are spanning the separating hyperplane. One advantage of SVMs is that they can learn non-linear hypotheses by using a mapping function which maps the original feature space into a (commonly) higher dimensional feature space or by using learning techniques like polynom classifiers, radial-basis classifiers [3] or two-hidden layer (2HL) networks which can form even more complex decision boundaries [2]. Another advantage of SVMs is that they can handle noise-(error-)reduction by deleting those examples – in our case the observed user-behavior – which cause the non-separability. SVMs are robust and do not need any parameter optimization, i.e., they can work on raw data.

Neural Nets Neural nets are a well-established and a successfully constituted technique in the area of soft computing to process information in a vague and tolerant manner. But the most important feature of neural nets is their learning capability which increases the adaptivity of the system and supports the complexity of heterogenous data. A variety of models are available to choose. For recommender systems in general backpropagation algorithms are especially suitable to handle the users feedback and selections to train the net. In our case, neural nets can be used for example to learn a threshold θ in Section 3 for the combined similarity – instead of using a threshold with a fixed value of 0.5.

Bayesian Networks Alternative names are *belief networks*, *probabilistic independence networks*, *influence diagrams*, or *causal nets* [19]. A Bayesian network is a directed acyclic graph (DAG) which represents probabilities of dependencies between a set of random features [7]. The nodes of the graph represent the set of random variables $\mathbb{X} = \{X_1, \ldots, X_n\}$ and the directed edges represent the dependency between these variables. The second part of a Bayesian network is a set of conditional probabilities $P(X_{ij}|X_i)$ according to the associated graph. One advantage of Bayesian networks is that the representation of a probability distribution as a directed graph enables the analysis of complex conditional user events with a graph theoretical approach and ensures the consistency of the system [21]. With Bayesian networks it is possible to calculate conditional probabilities, i.e., to estimate the users future behavior in case of observations made in the past.

Information Filtering Information Filtering is well known from shopping web sites. When the customer clicks on an item I_1, then the system proposes: "Customers who have bought this item I_1 also have bought another item I_2 too...". The advantage of this technique is its simplicity. It can be implemented very easily (only log files have to be parsed) and it can learn incrementally. In our case, Information Filtering can be used to learn behaviors like: "Customers who have inserted this process template at this position also have done this and that...". Furthermore, with Information Filtering techniques the system is able to generate and to compare user-profiles which help to categorize users, e.g., in beginners or specialists [12].

Content-based Information Filtering Content-based Information Filtering is a new extension which also takes the content of the underlying items into account, i.e. I_1 and I_2 is also compared and their similarity is regarded while ranking. With this method the combined similarity of process template elements described in section 3 can be used to calculate the similarity and to rank the recommendations retrieved by the Content-based Information Filtering technique.

Summarizing the upper list, especially each variant of Information Filtering is a promising technique for recommending process templates, which will be further considered.

6 Conclusion

In this paper we have presented on-going work for assisting users in business processes modeling. Compared to manual process modeling we aim to improve the reusability of business processes by an autocompletion mechanism. We propose to use a recommendation system, which observes the user's behavior and suggests possible subsequent elements. Furthermore, the system facilitates validated process properties of the automatically completed process to avoid deadlocks and lacks of synchronization.

Modeling languages such as EPCs, BPMN or BPEL have no direct formal foundations and thus do not enable analysis methods. Hence, if the process is modeled with these languages the process to be automatically completed can not be directly validated regarding deadlocks or lacks of synchronization.

Currently, we are developing an algorithm to automatically analyze hierarchical specifications of process elements and to detect errors in process models. Processes with hierarchical specifications generally appear to be more intuitive and easier to understand.

References

1. A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C. K. Liu, S. Thatte, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. wsbpel-specification-draft-01, OASIS, September 2005.
2. R. Beale and T. Jackson. *Neural Computing: An Introduction.* Institute of Physics Publishing, Bristol, U.K. and Philadelphia, PA, 1990.
3. C. M. Bishop. Neural networks for pattern recognition. Technical report, Oxford, Clarendon, P., 1995.
4. S. Brockmans, M. Ehrig, A. Koschmider, A. Oberweis, and R. Studer. Semantic Alignment of Business Processes. In *Proceedings of the 8th International Conference on Enterprise Information Systems*, Paphos, Cyprus, May 2006. to appear.
5. J. Didion. *JWNL 1.3*, November 2003. http://www.codezoo.com/pub/component/196?/category=5.
6. H. J. Genrich and K. Lautenbach. System modelling with high level petri nets. *Theoretical Computer Science*, (13):109–136, 1981.
7. D. Heckerman. A Tutorial on Learning with Bayesian Networks. Technical report MSR-TR-95-06, Microsoft Research, March 1995.
8. S. Hinz, K. Schmidt, and C. Stahl. Transforming BPEL to Petri Nets. In *Business Process Management*, pages 220–235, 2005.

9. I. Horrocks. Applications of Description Logics: State of the Art and Research Challenges. In *Proceedings of the International Conference on Conceptual Structures*, Lecture Notes in Computer Science, pages 78–90. Springer, 2005.
10. K. Jensen. An Introduction to the Theoretical Aspects of Coloured Petri Nets. In J. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency – Reflections and Perspectives*, Lecture Notes in Computer Science, pages 230–272. Springer, 1994.
11. T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In C. Nedellec and C. Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning (ECML'98)*, Lecture Notes in Computer Science, pages 137–142, Chemnitz, Germany, 1998.
12. S. Klink. Query reformulation with collaborative concept-based expansion. In *Proceedings of the First International Workshop on Web Document Analysis (WDA 2001)*, pages 19–22, Seattle, Washington, USA, 2001.
13. A. Koschmider and A. Oberweis. Ontology based Business Process Description. In J. Castro and E. Teniente, editors, *Proceedings of the CAiSE-05 Workshops*, Lecture Notes in Computer Science, pages 321–333, Porto, Portugal, June 2005.
14. V. I. Levenshtein. Binary Code capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, (8):707–710, 1966.
15. H. Lin, Z. Zhao, H. Li, and Z. Chen. A Novel Graph Reduction Algorithm to Identify Structural Conflicts. In *Proceedings of the 35th Hawaii international Conference on System Sciences*, pages 536–550. IEEE Computer Society Press, 2002.
16. A. Maedche and S. Staab. Measuring similarity between ontologies. In *Proceedings of the European Conference on Knowledge Acquisition and Management*, Lecture Notes in Computer Science, 2002.
17. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. W3c recommendation, World Wide Web Consortium, 2004.
18. G. A. Miller, C. Fellbaum, and R. Tengi. WordNet – a lexical database for the English language, 2006. `http://wordnet.princeton.edu/`.
19. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.
20. W. Reisig and G. Rozenberg. *Lectures on Petri Nets: Basic Models*. Lecture Notes in Computer Science. Springer, 1 edition, 1998.
21. B. A. Ribeiro-Neto and R. Muntz. A belief network model for IR. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 252–260, Zurich, Switzerland, August 18-22 1996. ACM Press.
22. W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *Inf. Syst.*, (2):117–134, 2000.
23. A.-W. Scheer and M. Nüttgens. ARIS Architecture and Reference Models for Business Process Management. In *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806, pages 376–389. Springer, 2000.
24. W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In *Business Process Management*, pages 161–183, 2000.
25. W. M. P. van der Aalst, A. Hirnschall, and H. M. W. E. Verbeek. An Alternative Way to Analyze Workflow Graphs. In *CAiSE*, pages 535–552, 2002.
26. W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
27. S. A. White. Business Process Modeling Notation. Specification, BPMI.org, 2004.

Integrating Semantic Web Services and Business Process Management: A Real Use Case[⋆]

Christian Drumm, Jens Lemcke, and Kioumars Namiri

SAP Research Center CEC Karlsruhe
SAP AG
`firstname.lastname@sap.com`

Abstract. In this paper we aim to investigate how semantic Web services can improve standard business process management tools. Using a standard SAP process in the area of logistics as an example, we show the limitations of current approaches, both during design- and runtime. Based on this investigation we will present a new solution enabling the automation of certain task in the process of creating cross organizational business processes using ontologies and Semantic Web Services.

1 Introduction

In this paper we aim to investigate how semantic Web services can improve standard Business Process Management (BPM) software. Based on a standard SAP process in the area of logistics, we show the limitations of current approaches and present a solution based on semantic Web services. The remainder of this paper is organized as follows: After an brief introduction to the SAP Enterprise Services Architecture we will describe the process used throughout the paper as en example and highlight limitations of current implementations. Next we investigate advantages of BPM-based implementations and show where and how this tools can be improved using semantic Web services. We will close with a summary and an outlook on future work.

SAP Enterprise Service Architecture

Vision SAP is a provider of business software supporting enterprises to perform their more or less standardized business tasks as efficient as possible. The main target is to lower the Total Cost of Ownership (TCO) of their SAP business solutions needed to carry out these business activities. The main challenge with respect to business tasks in today's companies is to keep track with increasingly

[⋆] This material is based upon work partially supported by the EU funding under the projects DIP (FP6 - 507483) and ATHENA (FP6 - 507312). This paper reflects the author's views and the Communities are not liable for any use that may be made of the information contained therein.

dynamic markets. This changing business environment demands for more and more flexibility of the companies to adapt to changing market requirements. Therefore, SAP solutions aim at equipping its customers with solutions meeting this demand for flexibly performing business.

The demand for flexibility in markets results in the need for high adaptability of internal company IT structures, simplification of interoperation with business partners as well as strong support for integration and outsourcing of business units, respectively. At the same time, prior investments of SAP customers in their existing IT infrastructure should be leveraged.

Description In order to provide maximum support for the market requirements mentioned above, SAP aims at building an integrative platform for individual service providers and requesters. The facilitating SAP technology is called Business Process Platform. It provides a uniform means for companies to represent the interfaces of their business functionality by *Enterprise Services* which are fully based on the Web Service Description Language (WSDL) standard. However, Enterprise Services extend Web Services by providing business level functionality that is also concerned explicitly with aspects like scalability, security, and transactions etc., which are important in an industrial setting. For instance, the task of canceling an order in a real world scenario has usually subsequent consequences, which must be handled as well. An order can usually not be simply deleted in the database; rather it may involve a complex business process, where following questions might occur: Is roll-back of the order in the backend system possible at all and are manual steps in the roll-back process necessary? Which roles are involved and need notification through another Enterprise Services such as Logistic services, Loan services etc.? Must the approval process be stopped as well and is a cancellation fee applicable? With Enterprise Services this related business information is made explicit.

The facilitator of the Business Process Platform is the design of the SAP products in the so-called Enterprise Services Architecture (see Fig. 1). It defines a clear structuring of the SAP software architecture into separate layers. SAP NetWeaver basically acts as application server and technical integration platform for different business partners. The mySAP Business Suite provides the SAP business software in an Enterprise-Service-enabled way while xApps technology allows for simplified composition of new functionality re-using existing applications or parts of those. The Business Solutions layer provides a clear documentation on the different components of the SAP solutions from a manager's perspective.

2 Example Process

The process we want to investigate in this paper is the standard SAP order-to-cash process in the logistics domain. This process, depicted in Fig. 2, involves three parties: i) A customer, ii) a shipper, and iii) a carrier. As illustrated the customer first places a sales order with the shipper, which enters it in its local

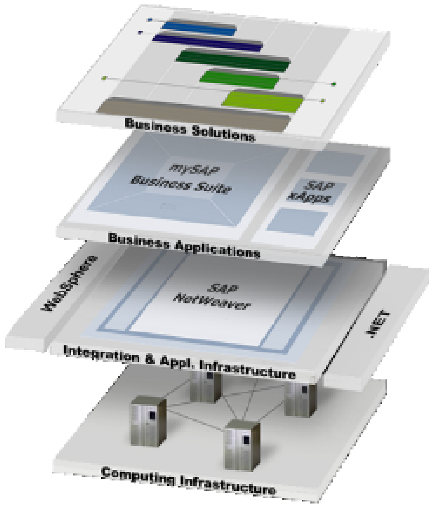


Fig. 1. The SAP Enterprise Service Architecture.

SAP system. After that, the appropriate steps for delivery and picking and packing are taken through. After sending the shipping information to the Express Shipping Interface (XSI), the goods are labeled and the manifest is sent to the carrier. This is the trigger for the actual shipping by the carrier which is tracked by the shipping process step in the SAP system of the shipper. The last activity is the execution of the billing process. Sometimes, carriers change their conditions of service. In this case, they need to notify the shipper of those updates. Also, for a new contractual cooperation between a shipper and a carrier, the new conditions need to be input into the shipper's system.

2.1 Architecture

In the order-to-cash process described in the previous section, different systems are involved. For this use case, we assume that the functionality of the carriers is provided via Web service interfaces. The shipper uses an SAP system. The SAP solution of the shipper is used to integrate the systems of the different parties. However, as connotated in Fig. 2, not all process steps are directly used as they are pre-existent in the SAP system.

Some process steps rather need some configuration adoption in order to fit the needs of the respective SAP customer (here: the shipper) for its specific business case. This is illustrated by the gold coloring in Fig. 2. Also, the process steps of labeling, preparing the manifest, and the shipping itself are highly influenced be the specific requirements of the respective carrier. Therefore, these steps are mainly implemented by the carries. This is illustrated by the blue coloring in Fig. 2.

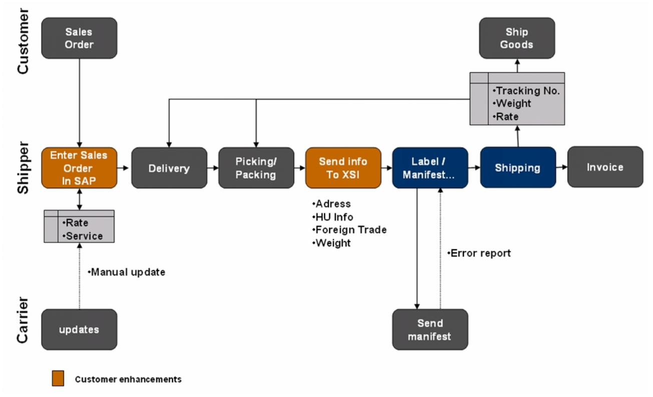


Fig. 2. Current Order-to-Cash Process.

To sum this up, the SAP system is a software that provides some main business functionality that is similar in most companies. Specific adoption to special requirements need to be done by extending the current SAP product via its interfaces in a static manner.

2.2 Pain Points

Due to the more or less static nature of the current solution, the different parties involved face some difficulties. These are explained in the following.

Carriers On the carrier side the following problems occur:

Difficulty to Publish New Services The publication of new services is difficult, because the process with an individual carrier is firmly implemented in the specific SAP system configuration of the shipper. Therefore manual development is necessary if a new service requires to enter this implementation.

Difficulty to Notify Customers of Service Update For this action, the carrier needs to interact with the shipper. If the change of service implies the adoption of the process between both partners, they face the same problem as in the former case.

Difficulty to Connect to SAP Systems In order to connect a third party system to a SAP system in general message mappings are necessary. In addition custom development might be necessary to adapt the business process of the carrier to fit the frame the SAP system sets.

Shippers

Difficulty to Add or Update Services Due to the firm implementation of the processes, this problem also appears to the shipper.

Difficulty to Change Service Provider Since processes are firmly coded into the SAP system, the change from an existing carrier with an already implemented process to a new business partner is difficult. The cost for changing carriers are quite high due to the implementation effort to take.

Different Service Interfaces Provided by Carriers The setup of a process with different carriers requires always new effort, because little knowledge can be reused due to their different interfaces.

Difficulty to Compare Services Provided by Carriers The diversity of interfaces and properties (like pricing model) of the services provided by the carriers makes their comparison difficult.

3 BPM-Based Implementation

In this section we will provide a high-level description of how current BPM tools could be used to implement the scenario described. A BPM-based implementation of the process described above would involve two main components: i) A process modeling tool, and ii) a process execution engine capable of executing the modeled processes. In our case, the process modeling tool is called *Maestro* [1] and the process execution engine is called *Nehemiah* (see Fig. 3).

Focusing on the shipper we will now describe the steps necessary to create an executable process using current BPM tools. Using the Maestro tool, a domain expert would create a graphical representation of the process executed at the shipper whenever a new sales order is processed. Note that this graphical representations is at first not linked to any of the shippers' business systems or any carrier services. Therefore in a second step, the domain expert manually connects the single process steps of the business process to services offered by either the internal or the partner business systems. Connecting different services and systems usually requires a mapping between different message formats. Consequently the domain expert also needs to create the necessary mappings converting between the input and output messages of the different business systems involved.

After the business process has been manually modeled and the involved business systems have been connected to the different process steps, the business process is stored into the process repository. During run-time the BPEL engine retrieves the process from this repository and executes an instance of the process for each incoming sales order.

The main advantage of BPM-based implementations of the scenario described above is the design-time flexibility. After the business process has been modeled using the graphical editor, services implementing different process steps can easily be exchanged during design-time. Integrating a new carrier into the systems

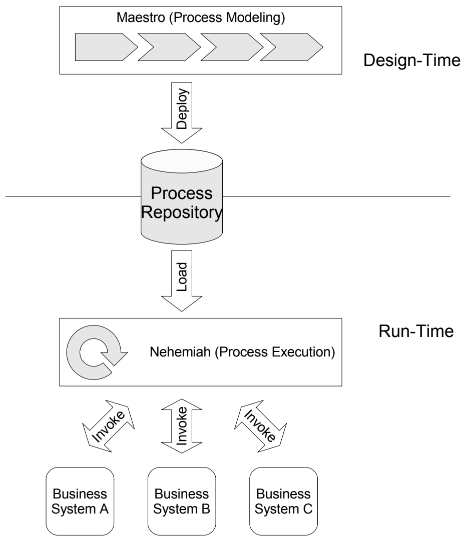


Fig. 3. Current solution using BPM tools.

for example only requires the connections of the carrier services to the appropriate process steps as well as the development of the necessary message transformations. However, BPM-based implementations do not have any additional flexibility during run-time, as the BPEL engine simply executes predefined processes. Therefore, dynamic exchange of carriers during run-time based on the availability of their services is not possible with current BPM-based solutions.

Public vs. Private Processes In the previous paragraph, we have described on a very high abstraction level how the given scenario would be implemented using current BPM tools. However we omitted an important detail during this description. Services available in the business systems of partners usually need to be invoked in a certain order as they are involved in the internal processes of the parter. In our scenario for example it does not make sense to invoke a "track and trace" service of a carrier before a shipping request has be send to that carrier. However, partners interacting during a business process only want to make parts of their internal processes visible to the outside. Therefore the notion of *private* and *public* processes becomes necessary. A private process is the detailed, internal process of a partner. Based on this process, a view—the public process—can be created. This public process hides all the confidential process details and only show the process steps that are required for an interaction. The public process is then used by business partners to create a so-called *Collaborative Business Process* (CPB) involving both internal and external business services (see Fig. 4).

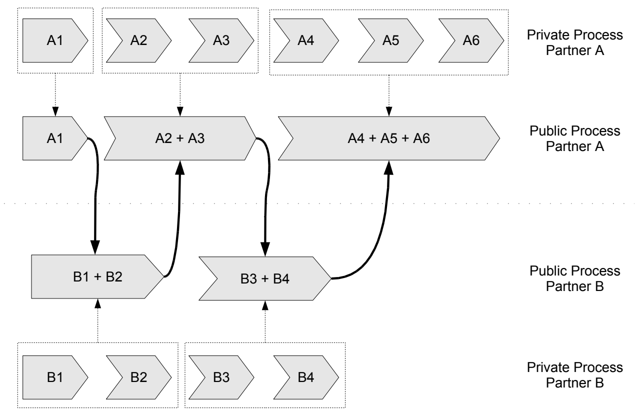


Fig. 4. Relation between public and private processes.

4 Added Value through Semantic Web Services

The previous high-level description of a BPM-based implementation of the order-to-cash process in the logistics domain shows several limitations of current solutions. The most prominent ones are:

- Necessity for manual development of message mappings
- Manual creation of the CPB
- Flexibility limited to design-time.

Using technologies developed in the semantic Web services area, these limitations of current BPM-based implementations can be overcome. In the subsequent sections, we will first describe our overall architecture for integration of semantic Web service technologies into current BPM tools. Following this, we will describe in detail how this architecture enables i) the automatic generation of necessary message mappings, ii) the automatic integration of the public processes of different partners into one Collaborative Business Process (CBP), and iii) the flexible service selection during run-time.

4.1 Definitions

Before we start with the description of our solution we first need to define some elementary terms which we will use during the later discussions.

Definition 1 (Ontology). *An ontology O is a structure*

$$O := (C, R, A, T, \leq_C, \leq_R, \leq_A, \leq_T, \sigma_R, \sigma_A)$$

where:

- *C is a set of concepts aligned in a hierarchy $\leq_C$*
- *R is a set of relations aligned according to $\leq_R$*
- *A is a set of attributes aligned according to $\leq_A$*
- *T is a set of data types aligned according to $\leq_T$*
- *$\sigma_R : R \rightarrow C \times C$ is the signature of R*
- *$\sigma_A : A \rightarrow C \times T$ is the signature of A.*

In addition to that we define the domain of $r \in R$ as $dom(r) := \pi_1(\sigma_R(r))$ and the range as $range(r) := \pi_2(\sigma_R(r))$. This definition of ontologies is based on [2].

Definition 2 (XML Schema). *An XML schema is a structure*

$$S := (E, A, CT, ST, \delta_e, \alpha_e, \delta_a, \Gamma)$$

where:

- *E is a set of element names*
- *A is a set of attribute names*
- *CT is a set of complex type names*
- *ST is a set of simple type name*
- *a function $\delta_e : E \rightarrow (CT \cup ST)$ defining the data type of an element $e \in E$*
- *a function $\delta_a : A \rightarrow ST$ defining the data type of an attribute $a \in A$*
- *a function $\alpha_e : E \rightarrow \mathfrak{P}(A)$ defining the set of attributes for each element $e \in E$*
- *a regular tree grammar Γ specifying the structure of a valid XML schema*

Definition 3 (Mapping). *A mapping map between two structures S and T is defined as set of mapping elements me. Each mapping element relates entities of the source structure $e_{S,i}, \ldots, e_{S,j}$ to entities of the target structure $e_{T,m}, \ldots, e_{T,n}$ using a mapping expression. The mapping expression specifies how entities of the source structure and the target structure are related.*

$$\begin{aligned} map_{S \rightarrow T} &= \{me\} \\ me &= (e_{S,i}, \ldots, e_{S,j}, e_{T,m}, \ldots, e_{T,n}, mapexp) \end{aligned}$$

This definition of mapping has a number of important implications. First a mapping is unidirectional as indicated by the arrow form S to T in our definition and cannot easily be inverted. Second the definition of a mapping given above does not restrict the relation between entities of the source and the target structure to be $1:1$, but rather allows for $m:n$ relations. Third the nature of the involved mapping expressions is not further restricted. This is due to the fact, that it depends strongly on the type of the source and target structure involved.

Definition 4 (Alignment). *An alignment $A_{S_S \rightarrow O_T}$ from an XML schema S_S to an ontology O_T is a mapping with:*

$$\begin{aligned} A_{S_S \rightarrow O_T} &= \{(e_{S_S,i}, \ldots, e_{S_S,j}, e_{O_T,m}, \ldots, e_{O_T,n}, mapexp)\} \\ e_{S_S,x} &\in E_{S_S} \cup A_{S_S} \\ e_{O_T,y} &\in C_{O_T} \cup R_{O_T} \cup A_{O_T} \end{aligned}$$

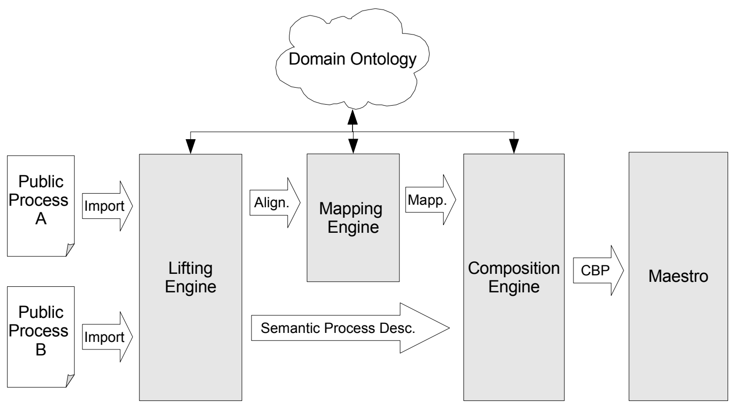


Fig. 5. Design-Time Architecture of the Enhanced Maestro Tool.

4.2 Solution Overview

Our overall architecture consists of two parts: A design-time, and a run-time component. During design-time, we want to simplify the creation of the CBP as much as possible. After loading two public processes, the Maestro tool should generate the CBP automatically (if possible) and present it to the user. Furthermore the tool should generate the message mappings necessary for invoking the involved Web service. Figure 5 shows the design-time architecture of our enhanced Maestro tool. We assume, that the representations of the two public processes not only contain the process flow but also the XSDs of the in- and output messages associated with each process step. After loading the two public processes specifications into our tool, the *lifting engine* generates two things: i) An alignment between the message elements of the XSDs and the domain ontology, and ii) a semantic description of the public processes. In the next step the *mapping engine* uses the alignments between the ontology and the XSDs to generate a list of possible mappings. Now the *composition engine* takes this list of possible mappings and the semantic process descriptions to generate the CPB which is finally presented to the user using the Maestro tool. After a check by the user and possible modifications of the CBP the result is stored into the central process repository.

Details on how the lifting, the mapping generation and the composition are performed will be given in subsequent sections.

During run-time we want to enable the dynamic selection of services based on different criteria. In our scenario we would for example like to select the carrier offering the cheapest price for a given shipment. Therefore we introduce a component called *semantic service selection*. Based on the concrete request, contractual information modeled in the domain ontology and a selection goal, the best process is selected from the process repository, instantiated and executed. Details on the semantic service discovery will be given in section 4.6.

4.3 Lifting Process Descriptions

The first step in our design-time architecture is the lifting of the public processes of the two partners. This lifting consists of two parts, the lifting of the input and output messages associated with the process steps and the lifting of the process descriptions itself.

Lifting Input and Output Messages In order to create an alignment between the domain ontology and the in- and output messages the lifting component executes a set of elementary matching algorithms. We are currently evaluating which elementary matching algorithms perform best in the given setting. An list of possible elementary algorithms can be found in [3] or [4]. These matching algorithms exploit the information available in in the XML schema and the ontology (like, e. g., element and concept names) to create a similarity matrix. This similarity matrix associates each pair of XML schema and ontology entities (e_S, e_O) with $e_S \in E \cup A$ and $e_O \in C \cup R \cup A$ with a similarity value. Based on this similarity matrix an alignment between the XML schema and the domain ontology can be calculated.

Creating Semantic Process Descriptions For the automatic process composition by the composition engine connotated in Fig. 5, the public process description of the shipper as well as the available WSDL descriptions of the carrier services need to be transformed to a format that the composer can work with. This step will be detailed in this section.

Format The composer technology we are going to use bases on the semantic Web services composition approach described in [5] and [6]. This approach will be detailed later on. For now, it is enough to know that for each partner which is to be integrated in the composed process, we need a semantic Web service interface description consisting of the following parts:

– The messages communicated by the semantic Web service given as ontology concepts, and
– Behavioral constraints between the single message exchanges of the semantic Web service.

In other words, the behavioral constraints can be understood as a workflow diagram, like an UML 2.0 Activity Diagram, containing control nodes, like decision, merge, fork and join. The activities in this diagram would be connected to input and output nodes representing the messages communicated. Here, each message is not understood as a technical XML schema description, but an ontology concept for that later on the corresponding XML schema can be nominated.

Shipper The information that is available for the shipper partner is its public process and the liks of the public process steps to specific WSDL operations. From this information, we build the semantic Web service description as follows:

1. Each WSDL operation becomes represented by an input node and an output node connected via a sequential control edge. We refer to this construct as a semantic Web service operation.
2. The data communicated by the input and output node are represented by ontology concepts that are obtained by the alignment step of the Lifting engine from the corresponding WSDL operation's XML schema (cmp. Fig. 5).
3. The semantic Web service operations are connected by semantic Web service behavioral constraints that resenble the workflow of the public process of the shipper.

Carrier For the carrier services, the only information we can get are the WSDL files of their Web servcies. From the lifting component, we again get the alignment of XML schema types communicated to ontology concepts. What is missing however, is a representation of causal interdependencies between the operations that we can use to create the semantic Web service behavioral constraints. Therefore, we build a trivial workflow that is required as input to the composer component. Details follow.

1. Each WSDL operation becomes represented by an input node and an output node connected via a sequential control edge. We refer to this construct as a semantic Web service operation.
2. The data communicated by the input and output node are represented by ontology concepts that are obtained by the alignment step of the Lifting engine from the corresponding WSDL operation's XML schema (cmp. Fig. 5).
3. The semantic Web service behavioral constraints describe a workflow that consists of a fork and a join node. Each branch of this fork-join construct contains exactly one semantic Web service operation.

4.4 Automatic Generation of Message Mappings

The automatic generation of message mappings is performed by the mapping engine. This component takes the alignments created by the lifting engine as input and generates executable mappings between XML schemas. In order to create a mapping between S_1 and S_2, the mapping engine takes the alignments $A_{S_1 \rightarrow O}$ and $A_{S_2 \rightarrow O}$ as input. For each mapping element in $A_{S_1 \rightarrow O}$ the mapping engine searches for a mapping element in $A_{S_2 \rightarrow O}$ that relates a schema entity of S_2 to the equivalent ontology entity. If such an entity is found, the mapping expression is used to determine how the schema entities of S_1 and S_2 are related. This in turn creates a new mapping expression that is added to the mapping $map_{S_1 \rightarrow S_2}$.

Note that mappings are not generated between each pair of schemas but only between input schemas of one public process and output schemas of the other and vice versa.

4.5 Automatic Integration of Partner Process Steps

As connotated in Fig. 5, the automated process composition by the composition engine is the final step before presenting the suggested CBP to the user via the Maestro tool.

Task For creating the CBP, we use the technology described in [5] and [6]. The main observation behind this technology is that business partners follow their own business processes. These business processes consist of several process steps that on the one hand exchange data amongst themselves, and on the other hand also collaborate with their partners. This however means, that, in order to integrate two business partners with each other, their business processes need to be interconnected. Here, it is important to understand that a business process can not be handled like an atomic transaction. In a process step, the company can have interactions with its partner before proceeding to the next step, which again results in some communication before it enters the next process step, and so forth.

The integration of two business partners in a message-based communication environment, like a service-oriented architecture, can however only work on the basis of atomic transactions. As an additional requirement, this integration must follow the sequential constraints of the message exchanges defined by both parties' business processes. We call these sequential constraints "behavioral constraints" or just the "behavior" of a business partner's systems. Please note, that such information cannot be represented by traditional Web service descriptions using WSDL. For an automatic creation of such an integration (the CBP), a partner's behavior needs to be explicitly specified in a machine-processable manner. This is why the observable part of a business partner's behavior appears in semantic Web service descriptions. The main task of a composer after all is to combine these sets of behavioral constraints to a combined business process of all parties involved.

Realization The inputs for the composer component in general are the semantic Web service descriptions of the participating business partners. In Sect. 4.3, we explained how to create these descriptions for the two parties shipper and carrier. The composition engine basically compares the inputs and outputs that are defined as ontology concepts in the two behavior descriptions and connects them where possible. For the decision whether a connection is possible, the composer relies on the results from the preceding alignment step. The alignment works in a way that it connects those XML schema elements that it later can generate a mapping for to the same ontology element. Therefore, the composer just needs to look for equivalent concepts in the two behavior descriptions that it can connect.

After identifying matching concepts, the composer connects fitting edges by a transformation activity node that defines a conversion which possibly needs to be performed in the real-time execution of the combined process. This conversion is given by the mapping function $map_{S_1 \rightarrow S_2}$. It defines how to transform actual data corresponding to the XML schema of the sender S_1 to a message corresponding to the receiver's format S_2. The result of the composition is a business process that contains the process steps of both parties, their interconnections via mapping activities, and those inputs and outputs that could not be interconnected. The composition therefore is successful, when there are no inputs and outputs left that could not be connected to corresponding communications of the other party.

For the next step of our use case, the composer result needs to be translated into the process representation format of the Maestro tool and will thus be presented to the user. Also in the case of an unsuccessful composer execution, the partly connected bunsiness process will be fed to the Maestro tool as a first suggestion for adaptation by the user.

4.6 Semantic Service Selection

After discussing how semantic Web service technologies can be used to improve the design-time of current business process management solutions, we will now investigate how they can be used to improve their run-time.

As stated in the solution overview, the semantic service selection is responsible for selecting the best fitting carrier for the current shipping request during runtime. The realization of this component is described in detail in [7]. It is based on an approach for semantic Web service discovery introduced by Li and Horrocks [8], and [9]. For applying this approach, an abstract service capability is described based on the domain ontology. The abstract service capability is carrier-independent and covers all possible Web service capabilities within the domain.

Additionally, a successful offline negotiation between a shipper and a carrier is required. The result of this negotiation phase is a contract between that carrier and shipper describing the provided service capabilities by that carrier. Each contract is modeled as a sub-concept of the service capability based on the domain ontology. These semantically described contracts are stored as OWL documents on a separate repository. The concrete shipping request created at run-time is then described either as an instance or as a most specific sub-concept of the abstract Web service capability according to the domain ontology. A shipping request can be fulfilled by a carrier Web service if the the concrete request subsumes a Web service capability. If more than one contract matches the concrete request an additional selection step is required in order to choose between the available carriers. This step usually requires run-time invocation of the carrier Web services in order to get information necessary for the selection according to the goals of the requester. A *selection goal* specifies which criterion defines the best suiting carrier, e. g., best price or shortest delivery. Since these two parameters are subject of frequent change due to the competition on the carrier market, we decided not to design these parameters in the semantically described contracts.

The important parts of the shipment request are: Ship from and ship to addresses, items to be shipped, and the selection goal. The received shipment request by the component is first semantically lifted according to the ontology in order to be processed by a DL-reasoner. In the next step, the repository containing all existing pre-negotiated contracts with carriers is queried. Additionally, the necessary URL to the WSDL of the Carrier Web service associated with the contract can also be obtained from the repository. Each available contract is consequently subsumed by a DL-reasoner to determine the contracts matching the (semantically lifted) shipment request. If the client application has specified, e. g., the best price as its selection goal in the shipment request, all matching carrier Web services are contacted and the carrier with the best price for this shipment is selected.

After the selection is done, the process associated with the selected carrier is loaded from the process repository, instantiated and executed.

5 Summary and Outlook

The proposed carrier/shipper-scenario trys to keep the described system simple in order to be able to concentrate on the important steps first. The important aspect is mainly to examine how semantic technologies can beneficially be applied to real-world business scenarios. We identified the automation of the so far manual business process integration as the main area of contribution for semantic Web technology. The solution extends and therefore bases on standard BPM modeling tools.

Furthermore, we abstain from the requirement of business partners to adhere to exactly the same software component interfaces. Thus, mediation comes into play and its interacting with the semantic Web service composition is a second aspect to focus on using this scenario.

After the successful implementation of this first scenario, the described setting can be extended to a more comprehensive application in a later version. In the current proposal, the composed business process is being created during design-time. When a carrier changes its conditions, the process of composition needs to be executed again in order to compute the potential adoptions to the process instance. In a more dynamic implementation, this step could be executed each time a customer requests a shipment. This way, the system would immediately and automatically incorporate changes to the carrier capabilities.

References

1. Greiner, U., Lippe, S., Kahl, T., Ziemann, J., Jkel, F.W.: Designing and implementing crossorganizational business processes - description and application of a modeling framework. In: Interoperability for Enterprise Software and Applications Conference I-ESA. (2006)
2. Stumme, G., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Sure, Y., Volz,

R., Zacharias, V.: The karlsruhe view on ontologies. Technical report, University of Karlsruhe, Institute AIFB (2004)

3. Do, H.H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: Proc. 28th VLDB Conference. (2002)
4. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Technical report, Informatica e Telecomunicazioni, University of Trento (2005)
5. Albert, P., Henocque, L., Kleiner, M.: Configuration-based workflow composition. In: ICWS, IEEE Computer Society (2005) 285–292
6. Albert, P., Henocque, L., Kleiner, M.: A constrained object model for configuration based workflow composition. In Bussler, C., Haller, A., eds.: Business Process Management Workshops. Volume 3812. (2005) 102–115
7. Friesen, A., Namiri, K.: Towards semantic service selection for B2B integration. In: submitted to Methods, Architectures & Technologies for e-Service Engineering (MATeS) at ICWE. (2006)
8. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proc. of the Twelfth World Wide Web Conference. (2003)
9. Preist, C., Cuadrado, J.E., Battle, S., Grimm, S., Williams, S.K.: Automated business-to-business integration of a logistics supply chain using semantic Web services technology. In: International Semantic Web Conference. (2005) 987–1001

Expressing Semantic Web Service Behavior using Description Logics[⋆]

Markus Fronk and Jens Lemcke

SAP Research, Karlsruhe, Germany
{markus.fronk, jens.lemcke}@sap.com

Abstract. For the *automation* of major tasks of the traditional Web service (WS) usage, semantic Web services research has identified the need for (1) the provision of additional aspects of Web services, as well as (2) the formalization of these descriptions. This work focuses on the behavioral aspects of Web services and proposes to use Description Logics (DL) for their formalization. We provide DL constructs for describing interactions in sequences and parallel splits. This yields several advantages for the tasks of WS retrieval and composition. (1) The development of the retrieval and composition software becomes *simplified* which therefore results in more *robust* code. (2) The Web service description can be *extended* by additional features without touching the evaluating code. (3) In combination with the ontology language OWL-DL, the Web service description can directly be *integrated* with existing semantic Web efforts, e. g. domain ontologies.

1 Introduction

In today's industries, coping with a growing number of software artifacts in a flexible and efficient manner becomes a more and more important issue. In addition, faster changes of market situations force companies to be able to quickly adopt their business processes and set up interoperations with other parties.

Service-oriented technology arose to address some of the challenges posed by this development. Using the Web Service Description Language (WSDL)[1] in conjunction with the Simple Object Access Protocol (SOAP)[2] and the "Universal Description, Discovery and Integration" (UDDI),[3] software components can be accessed through and communicate via standardized interfaces and protocols. This increases the flexibility of the companies, but every task of the Web Service (WS) usage process still remains a manual integration activity.

The aim of Semantic Web Services (SWS) research is to automate major tasks of this usage process. These are discovery, selection, composition, execution and monitoring. Major contributions in this area are OWL-S[4] and the Web Service Modeling

[⋆] This material is based upon work partially supported by the EU funding under the project DIP (FP6 - 507483). This paper reflects the author's views and the Community is not liable for any use that may be made of the information contained therein.

[1] http://www.w3.org/TR/wsdl, also: "Web Service Definition Language"

[2] http://www.w3.org/TR/soap12-part0/

[3] http://www.uddi.org/

[4] http://www.daml.org/services/owl-s/1.1/overview/

Ontology (WSMO [1]). In contrast to traditional Web service descriptions, Web service automation requires (1) some more information to be specified, and (2) a formal representation of all information given to facilitate their automatic processing.

In current approaches for Semantic Web service description, languages bearing a formal semantics are being used to add further information to "traditional" service descriptions. However, the capabilities of formalism are only used for defining the Web service description languages itself, rather than exploiting them to draw conclusions over the semantic of descriptions. For each task of the Web service usage process, specialized software is needed to work with the respective part of the formalized Web service description. This result in additional algorithms that have to be developed to interpretate the semantics described. In addition, different SWS approaches using different description formalisms can not easily be intergated. We therefore require a semantic Web service description whose formal capabilities can be further exploited. This means, that given Web service descriptions, standard software should be able to draw conclusions about which services are close to the evaluations needed to be performed during the tasks of the Web service usage process. This facilitates the separation of the model of aspects of the Web service from its interpreting application. Since we then can rely on standard reasoning software to realize standard parts, the *robustness* of software improves.

Further requirements for a general Semantic Web service description are its ability to be *extendable* for later enhancements and *integrable* with other aspects of Web service descriptions whose modelings were independently developed from each other. As an example, the "business semantics" aspect is one of these aspects which could be expressed in Description Logics (DL [2]) [3]. Other efforts are concerned with modeling Web service policies using DL [4]. Since these different aspects developed independently from each other adhere to the same formal semantics (the way inferences are drawn), they can potentially be integrated in a single analysis module considering all aspects as a whole.

In this paper, we propose to use Description Logics for the representation of Web service descriptions. A semantic Web service description consisting of a (1) technical, (2) behavioral and (3) contextual part provides enough information for the automation of the major WS usage tasks as named above. The common feature of all these tasks is finding matching service descriptions to a request description. We therefore design our Web service descriptions in such a way that through the standard subsumption reasoning this common task can be accomplished. For the implementation, we can therefore rely on complete and correct reasoner implementations.

Although all of the three aspects of semantic Web service descriptions should be expressed using DL, this paper focuses for demonstration on the behavioral aspect, i. e. the constraints between a service's operations that define the allowed order of execution. We choose DL, because it comes with a formal semantics, brings sufficient expressivity for our purposes, and is decidable. By using OWL-DL, we demonstrate its ability to serve as a general semantic Web service description language satisfying the previously described requirements of facilitating *robustness* of software, and ensuring *extendability* and *integrability* of WS descriptions.

2 Related Work

There are quite a few approaches that deal with the semantic annotation of Web Service technologies and standards to enable an automatic discovery and matchmaking process. The current UDDI discovery mechanism only insufficiently fosters the objective of automation. Most of the related work uses for the description of behavioral aspects of services either WSBPEL or language descriptions using ontologies. For matching of requests and services special algorithms have to be developed. The major difference of our approach is that we use DL to describe service behavior. Automated discovery and matchmaking can hence be realized using standard reasoner such as Racer[5] or Pellet[6]. Services matching a request can easily be determined by the subsumption mechanism when described with the DL expressions we suggest.

OWL-S Process Model The Web Ontology Language for Services (OWL-S) utilizes an ontology to describe Web services. Its concept is to provide markup language constructs to describe Web services in a semantic and thus computer-interpretable form. OWL-S builds an upper ontology for service description that consists of three main parts, namely the service profile, the service model and the service grounding.[7] The process model defines a subset of workflow features to describe a service as a process. In contrast to our solution special algorithms have to be developed to exploit the process descriptions characterized in OWL-S for matchmaking or similarity comparisons. Such algorithms based on OWL-S are described for example in [5] and [6]. The main difference of our solution is that with describing the service process flow using description logic, automated reasoning and matchmaking with standard reasoners become possible.

METEOR-S Process Designer The Managing End-To-End OpeRations for Semantic Web Services (METEOR-S) project at the Large Scale Distributed Information Systems (LSDIS) Lab at the University of Georgia annotates semantics to the complete Web service usage process. Its annotation framework is an approach to adding semantics to current industry standards such as WSDL. Finding an appropriate service for the composition is realized by a discovery engine querying an enhanced UDDI registry. The semantic descriptions published in this registry are annotated source code that is later transformed into either WSDL, WSDL-S or OWL-S.The OWL-S process model generally allows to semantically describe service behavior but the transformation made by the Semantic Description Generator however only considers the service profile and the service grounding. The process model is to the best of our knowledge not yet integrated in the transformation [7]. WSDL and WSDL-S anyway do not provide constructs to express service behavior. Behavioral aspects are hence not published in the registry. Therefore the METEOR-S approach in contrast to our solution does not consider the behavioral aspects of services in the discovery of adequate matches.

[5] http://www.sts.tu-harburg.de/ r.f.moeller/racer/
[6] http://www.mindswap.org/2003/pellet/
[7] http://www.daml.org/services/owl-s/1.0/owl-s.pdf

WSMO Choreography The Web Service Modeling Ontology (WSMO [1])[8] is a *conceptual* specification for describing ontologies, Web services, goals, and mediators—called WSMO entities. The behavioral aspects of a Web service are called its "choreography". The choreography in WSMO is described by an adoption of Abstract State Machine (ASM [8]) statements. Roughly spoken, these statements are of the form "$\mathbf{if}\ condition\ \mathbf{then}\ updates\ \mathbf{endif}$". In the $condition$, the existence of instances of an ontological concept can be queried, which may refer to a message that was just received. In the $updates$, instances of the internal ontology can be manipulated which may trigger the sending of respective messages. This description of causal dependencies between single communications is very similar to our approach. However, the ASMs are very expressive and do not define when, e. g., a service matches a certain request with respect to their behavioral constraints.

WSBPEL Abstract Processes The Web Services Business Process Execution Language (WSBPEL)[9] can be used to describe the implementation and the observable behavioral interface of Web services. This information could then be used for automating tasks of the Web service usage process—e. g. WS composition [9]. However, there is no decidable algorithm for the representation of service behavior in WSBPEL that can be used for reasoning about relevant properties. The conversion to a suitable representation is needed.

CoBPIA Particles The Collaborative Business Processes based on Intelligent Agent Technology (CoBPIA [10]) project uses Constraint Satisfaction Problem (CSP) techniques in order to describe process steps—called particles—that are to be composed to executable WSBPEL processes. To our knowledge, particles can only be atomic processes steps that are going to be composed into workflows. Web services could be interpreted as these particles. However, the interdependencies of Web service operations that we describe are not addressed in the CoBPIA particle representation.

3 Elements of behavioral aspect

In this section, we point out the behavioral patterns that can appear with regard to services and thus have to be represented in the DL. The elements in this section are besides being introduced described in terms of their meaning with regard to the service behavior. This means that each element has another sense in terms of the allowed order of messages. This is subsequently be described. The present work focuses on the basic control patterns. After having shown that these patterns can be represented in DL future work with regard to more complex patterns such as, e.g., loops can be motivated.
Messages exchanged by services can be distinguished in incoming and outgoing messages containing either parameters processed (inputs) or provided (outputs) by the service. The service behavior, as we understand it, is made up of three constitutive aspects that have to be considered. (1) The existence of inputs and outputs (interactions), (2) The sequence in which inputs and outputs occur and (3) Control constructs representing the allowed order of interactions.

[8] http://www.wsmo.org/
[9] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

Existence of interactions The fact that interactions occur in a service, we call existence. From a service requestor's point of view, describing a service environment[10], inputs *can* be provided and outputs *must* be received. From a provider's point of view, the described inputs are *required* and the outputs are *provided*. The semantics of these different interpretations of the existence of interactions is discussed in Sect. 4. For modeling purposes, the existence of inputs and outputs is in a first step considered individually before being consolidated. A service can consist of several inputs, outputs or most commonly a combination of both. The mere existence of interactions makes no statement about the ordering of interactions.

Sequence The order of interactions is specified defining a sequence of their appearance. Thinking of public and private processes, a special order of interactions needs for example to be defined when an internal state (that is not obvious for the external observer) triggered by an interaction has to be reached *before* another interaction can be processed. Although this implicit order of interactions is possibly not obvious in the first instant only seeing the external visible public process, it is however necessary for the successful execution of the operation. This requirement has to be met by services in order to match the request. Hence it has to be made explicit. A sequence can be defined either between inputs or outputs or between both interactions.

Control constructs The third behavioral aspect besides the mere existence of interactions and the sequence are control constructs describing special derivations from the basic sequence. In [11], 20 workflow patterns based on analysis of existing workflow management systems and workflow languages are identified and described. Our solution focuses on the basic control patterns to show how service behavior can be described with DL. These have been identified to model a common set of service behavior constructs. The basic patterns consist of *sequence* (described in Sect. 3), *parallel split* along with its *synchronization* and *exclusive choice* along with the *simple merge*.[11] Advanced patterns such as *multiple choice* and *cycles* are generally imaginable, they are however neglected in this initial approach. The *parallel split* expresses the concurrency of interactions. This means that the interactions described as concurrent can appear in any order. They only have to meet other sequential requirements that could be specified, e.g. when another interaction is defined to occur *before* the *parallel split*. An example use case from a service provider's point of view could be as follows: After entering necessary login information a user has to enter both credit card and address information. The sequence of entering the credit card and address information is however of no importance for the service functionality. Both only have to appear after the login. The *exclusive choice* describes the case when only one branch of interactions is executed. At present, the modeling of the *exclusive choice* in DL has not been realized. The potential combinations of the realized concepts as there are *existence and sequence*, *existence and parallel split*, *sequence and parallel split* and combinations of all three have been modeled and introduced in the design of the DL expressions. This is described in Sect. 5

[10] Environment in this regard describes the system or service landscape in which the searched or requested service is to be embedded with its functionality

[11] *Parallel split* and *exclusive choice* always include its *synchronization* or *simple merge* pattern. An explicit distinction between these patterns is waived.

4 Service matching

After having described in the previous section the elements that represent the behavioral aspects of services and their meaning to the order of interactions, this section focuses on the semantics of these elements with regard to the matching of services. We understand the matching of services as the major and common task in the SWS usage process. In this section we will outline the semantics due to which a service matches a request given the previously introduced elements. The decision whether Web services can be matched, depends on the previously described behavioral aspects. It can however not easily be claimed by just comparing the existence of these constructs as the semantic behind these is quite complex with regards to Web services. This will be described in more detail in the following sections. Each aspect described by a service request R has to be analyzed with regard to its effects on the fulfillment by a service S. We use DL in such a way that a service S fulfills the requirements of a request R when

$$R \sqsupseteq S \tag{1}$$

can be asserted. The semantic of the behavioral aspects has been modeled in DL to satisfy this equation. This means that the behavioral constructs are described in a way that matching services can be identified by satisfying (1).

Existence As indicated in Sect. 3, the existence of *inputs* described in a request has a different effect on a services' ability to assert $R \sqsupseteq S$ than the existence of *outputs*. For the DL expression representing inputs of a request R and a service S we write I_R and I_S, respectively. For outputs we write O_R and O_S. I_R describes the maximum set of inputs that *can* be provided by the environment. A service however does not have to use all the inputs provided. A service delivering the requested goal by using less inputs still fulfills the request. This is described by (2). The set of outputs specified by the environment defined as O_R is on the other hand mandatory. It specifies the minimum set that *must* at least be *provided* by the service's set of outputs O_S. This relationship is defined by (3).

$$\text{Inputs:} \quad I_R \sqsupseteq I_S \tag{2}$$

$$\text{Outputs:} \quad O_R \sqsubseteq O_S \tag{3}$$

Sequence The sequence describing the order of inputs and outputs specified by a service request is defined as Seq_R. It has to be met by the sequence of a matching service Seq_S in order to satisfy $R \sqsupseteq S$. Equations (2) and (3) must still hold. This means that a service does not necessarily have to consume an input, even if it is described in Seq_R. This is discovered by the standard reasoning mechanism. However, when a service consumes an input that is described in a special order in Seq_R, it has to appear with the same sequential constraints in Seq_S. Outputs, in contrast, always have to be provided. Additional outputs provided by the service S that are not necessarily required by R (i.e. they are not defined in O_R) can be part of the sequence Seq_S at any position. The same applies for interactions specified without any special order (cp. Sect. 3)in R.

$$\text{Sequence:} \quad Seq_R \sqsupseteq Seq_S \tag{4}$$

Control constructs The semantics related to a *parallel split* is as follows: A request that defines a parallel split between interactions (for example I_1 and O_1)[12] explicitly states, that the order of these interactions is not relevant. (cp. Sect. 3). This means for a service to fulfill the request that it can either specify the parallelism as well, or define an order of the interactions involved. A service stating the sequence "I_1 *and then* O_1" or vice versa also matches the request. The identification of possible matches has to consider this specialty since it can not just be derived from a one-to-one mapping of control constructs. A service does not necessarily has to have the same control construct as the request to match. The aspects of existence, sequence and the control constructs can be combined in any combinations. The previously described characteristics of each of the aspects is still considered for the matching when combined to a more complex behavior.

5 DL for behavioral aspects

In the previous sections we described the elements of service behavior and their semantics with regard to the matching of services. In this section we will introduce the DL-constructs that have been developed to represent the previously discussed elements and their semantics. This will outline the basis for automated reasoning and matching. The following proof of concept using the subsequently introduced DL-constructs will finally show the obtained advantages.

The Description Logics approach, and more precisely OWL-DL, was chosen because of its decidability and its ability to be integrable with other semantic web technologies facilitating the expandability of the WS descriptions. The matching of requests and services is represented by a subsumption relation of their DL constructs. (cp. (1)) Given several services annotated with the language constructs subsequently defined for describing the behavioral aspects, standard reasoners can be exploited to infer this classification. The following constructs are modeled and developed in order to be used with the conception of classification. Matching services in terms of the aspects previously described in Sect. 4 are *automatically* identified and classified as subordination of the service request. The required DL constructs are succedingly introduced. Requests, services and interactions are modeled as classes. The behavioral aspects are expressed by special properties representing the relationships between classes.

Existence For describing the semantics of the existence of interactions (specified in Sec. 4) we introduced the existence property "has". This can colloquially be interpreted as: "*A service has the following inputs and outputs!*" Due to the fact that inputs and outputs have different semantics for fulfillment, they are also modeled in a different way. The DL expression for a set of inputs $I_1, I_2, \ldots, I_n$ for a request R or service S is represented through the inputs combined by the *union of* specifier. Thereby, (2) is satisfied. The set for a service S and request R, respectively, is hence defined as follows.

$$I_{S/R} \equiv (I_1 \sqcup I_2 \sqcup \ldots \sqcup I_n) \tag{5}$$

[12] The parallel split can contain only inputs or only outputs or all kind of combinations

A request R *providing* several inputs is then described through defining a has-relationship with the relevant set:

$$R \equiv \exists\, \mathrm{has}.I_R \equiv \exists\, \mathrm{has}.(I_1 \sqcup I_2 \sqcup \ldots \sqcup I_n) \tag{6}$$

A service S described as $S \equiv \exists\, \mathrm{has}.(I_1 \sqcup I_2 \sqcup \ldots \sqcup I_m)$ where $m \leq n$ is hence identified as being adequate for R, because of satisfying $R \sqsupseteq S$. Services *requiring* more inputs ($m > n$) do not satisfy this condition. Outputs, in contrast, are enumerated using the *intersection of* operator for representing the semantics in (3). The DL expression for a set of outputs $O_1, O_2, \ldots, O_n$ is defined as follows.

$$O_{S/R} \equiv (O_1 \sqcap O_2 \sqcap \ldots \sqcap O_n) \tag{7}$$

A request *relying on* several outputs is then described, through defining a has-relationship with the relevant set:

$$R \equiv \exists\, \mathrm{has}.O_R \equiv \exists\, \mathrm{has}.(O_1 \sqcap O_2 \sqcap \ldots \sqcap O_n) \tag{8}$$

A service S described as $S \equiv \exists\, \mathrm{has}.(O_1 \sqcup O_2 \sqcap \ldots \sqcap O_m)$ with $m \geq n$ is identified as match, because $R \sqsupseteq S$. Services *providing* less outputs ($m < n$) are not considered to fulfill the request R. The more common case that services consist of both inputs and outputs is accommodated by the combination of (5) and (7). These concepts are combined using the *intersection of* operator. For the DL construct of the combined interactions we write $IO_{S/R}$. It is described with following equation.

$$IO_{S/R} \equiv I_{S/R} \sqcap O_{S/R} \equiv (I_1 \sqcup I_2 \sqcup \ldots \sqcup I_n) \sqcap (O_1 \sqcap O_2 \sqcap \ldots \sqcap O_m) \tag{9}$$

Requests *providing* several inputs and *relying on* several outputs are then described through defining a relationship similar to the cases (6) and (8):

$$R \equiv \exists\, \mathrm{has}.((I_1 \sqcup I_2 \sqcup \ldots \sqcup I_n) \sqcap (O_1 \sqcap O_2 \sqcap \ldots \sqcap O_m)) \tag{10}$$

This expression facilitates reasoning over both the constraints defined in (2) and (3).

Sequence The sequence of inputs and outputs is described through an ordering then-property. The succession of several interactions is represented by nesting the ordering property. Inputs and outputs are in a first step again treated independently. The nested expressions are combined similar the existence of inputs and outputs. Consecutive inputs are combined with the *union of*, consecutive outputs with the *intersection of* operator. Requests R and services S with a sequence of interactions are described as shown in (11) and (12).

$$Seq_R \equiv \exists\, \mathrm{then}.(I_1 \sqcup (\exists\, \mathrm{then}.(I_2 \sqcup (\exists\, \mathrm{then}.(\ldots \sqcup (\exists\, \mathrm{then}.(I_n))))))) \tag{11}$$

$$Seq_R \equiv \exists\, \mathrm{then}.(O_1 \sqcap (\exists\, \mathrm{then}.(O_2 \sqcap (\exists\, \mathrm{then}.(\ldots \sqcap (\exists\, \mathrm{then}.(O_n))))))) \tag{12}$$

The combination of inputs and outputs in a single sequence poses a complication that could not be completely handled yet. The difference to the combination of interactions with the DL-expression realized within the existence aspect is that within the sequence inputs and outputs have to be combined in the nested sequentiell expression. Combinations of the concepts such as inputs following other inputs, as well as outputs following other outputs can be matched correctly using the combined expression shown in (11) and (12). Combinations of the concepts such as inputs following outputs and outputs following inputs however are not correctly matched in every aspect. This means that some services are identified as match although they are not and vice versa. Goal is to have a DL representation, that expresses the combined sequence in a way, that all kinds of cases are correctly classified. In Sect. 6 we provide an example that show a case that correctly distinguishes appropriate and not appropriate services for a defined request, given the present DL-constructs.

Control constructs The semantics of a *parallel split*, as it is described in Sec. 4, is expressed as follows considering as example three existing interactions.

$$Seq_R \equiv \exists \text{then}.(I_1 \sqcup (\exists \text{then}.((I_2 \sqcup (\exists \text{then}.I_3)) \sqcup (I_3 \sqcup (\exists \text{then}.I_2))))) \quad (13)$$

$$Seq_R \equiv \exists \text{then}.(O_1 \sqcap (\exists \text{then}.((O_2 \sqcap (\exists \text{then}.O_3)) \sqcup (O_3 \sqcap (\exists \text{then}.O_2))))) \quad (14)$$

Equations (13) and (14) describe the concurrency of the second and the third interaction. The sequence of occurance is not relevant. Both a service S_1 that has the second *before* the third[13] and a service S_2 that has the third *before* the second[14] interaction are identified as match for the request R. ($R \sqsupseteq S$ is satisified)

Service behavior description A service behavior consists (as outlined in the previous sections) of the existence part (cp. Sec. 5) and the sequential ordering (cp. Sec 5 and Sec. 5). The DL expression for the service behavior description is hence the combination of both (represented by the *intersection of* operator). The behavioral aspects of requests R and services S are described as follows. Assuming a request that describes the three inputs (I_1, I_2, I_3) sequentially ordered (*first* I_1 *then* I_2 *then* I_3) is represented by the DL construct:

$$R \equiv \exists \text{has}.(I_1 \sqcup I_2 \sqcup I_3) \sqcap \exists \text{then}.(I_1 \sqcup (\exists \text{then}.(I_2 \sqcup (\exists \text{then}.(I_3))))) \quad (15)$$

Services described with the same DL expressions can be identified as a match by verifying the satisfaction of $R \sqsupseteq S$. This inference can be done using standard reasoners. Outputs are described accordingly, combining the existence and sequence constructs for outputs.

[13] $Seq(S_1) \equiv \exists \text{then}.(I_1 \sqcup (\exists \text{then}.(I_2 \sqcup (\exists \text{then}.I_3))))$
[14] $Seq(S_2) \equiv \exists \text{then}.(I_1 \sqcup (\exists \text{then}.(I_3 \sqcup (\exists \text{then}.I_2))))$

6 Proof of Concept

In the following we show a scenario applying the previously described DL-constructs. The behavioral aspects of a request and possible services are subsequently described with OWL-DL and existing inference mechanisms are used to draw conclusions about the match of the described services. This shows how code *robustness* can be improved exploiting existing standard reasoners, because the formal semantics of DL facilitates the separation of the model of aspects of the Web service from its interpreting application.
The first example only considering interactions is then extended to also consider behavior showing the simple *extendability* of our approach. Different aspects developed independently from each other that adhere to the same formal semantics can be integrated in a single analysis module considering all aspects as a whole. The use of OWL-DL constructs further allows for the later the non-intrusive integration with other ontological models using the same ontology language.
The scenario described in the subsequent section is as follows. A user requests an ordering service that requires a login, an order and user data as inputs and provides an order confirmation as output. Furthermore two services, (1) a store service, appropriate for our request, and (2) a fraud service spying user data are described. The corresponding DL-constructs are defined as follows.

$$\begin{aligned} R &\equiv \mathit{Ordering} \equiv \exists\,\mathrm{has}.((\mathit{login} \sqcup \mathit{order} \sqcup \mathit{userData}) \sqcap (\mathit{conf})) \\ S_1 &\equiv \quad \mathit{Store} \quad \equiv \exists\,\mathrm{has}.((\mathit{login} \sqcup \mathit{order}) \sqcap (\mathit{wMsg} \sqcap \mathit{conf})) \\ S_2 &\equiv \quad \mathit{Fraud} \quad \equiv \exists\,\mathrm{has}.((\mathit{login} \sqcup \mathit{order} \sqcup \mathit{userData}) \sqcap (\mathit{conf})) \end{aligned}$$

The store service requires only a login and an order and provides both a welcome message and a confirmation for the order. The fraud service in contrast has the same interactions as the ordering request. The reasoner given these service descriptions has as a result the inference shown in Fig. 1. Considering only the interactions existent within the services, both services are identified as match to the request R. ($R \sqsupseteq S$) Considering in addition the service behavior for the identification of appropriate services, we just extend the previous expressions with the introduced DL-constructs to define the allowed ordering of interactions. The most important constraint is that the order and user data is not provided until a login has occured. The store service satisfies this requirement, the fraud service however requires the user data before the login. The according DL-constructs for the request and the services are defined as follows.

$$\begin{aligned} &R \equiv \mathit{Ordering} \equiv \exists\,\mathrm{has}.((\mathit{login} \sqcup \mathit{order} \sqcup \mathit{userData}) \sqcap (\mathit{conf}) \\ &\quad \sqcap \exists\mathrm{then}.(\mathit{login} \sqcup (\exists\mathrm{then}.(\mathit{order} \sqcup (\exists\mathrm{then}.(\mathit{userData} \sqcap (\exists\mathrm{then}.(\mathit{conf}))))))))\end{aligned}$$

$$\begin{aligned} &S_1 \equiv \mathit{Store} \equiv \exists\,\mathrm{has}.((\mathit{login} \sqcup \mathit{order}) \sqcap (\mathit{wMsg} \sqcap \mathit{conf}) \sqcap \exists\mathrm{then}.(\mathit{login} \\ &\quad \sqcap (\exists\mathrm{then}.(\mathit{wMsg} \sqcup (\exists\mathrm{then}.(\mathit{order} \sqcup (\exists\mathrm{then}.(\mathit{userData} \sqcap (\exists\mathrm{then}.(\mathit{conf}))))))))))\end{aligned}$$

$$\begin{aligned} &S_2 \equiv \mathit{Fraud} \equiv \exists\,\mathrm{has}.((\mathit{login} \sqcup \mathit{order} \sqcup \mathit{userData}) \sqcap (\mathit{conf}) \\ &\quad \sqcap \exists\mathrm{then}.(\mathit{userData} \sqcup (\exists\mathrm{then}.(\mathit{login} \sqcup (\exists\mathrm{then}.(\mathit{order} \sqcap (\exists\mathrm{then}.(\mathit{conf}))))))))\end{aligned}$$

Given these service descriptions, the result inferenced by the reasoner is shown in Fig. 2. Considering as well the behavioral aspects only the store service is inferred as a match for the request. The fraud service does not follow the allowed order of interactions.

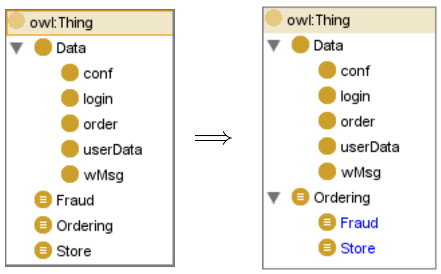


Fig. 1. Inference considering Interactions

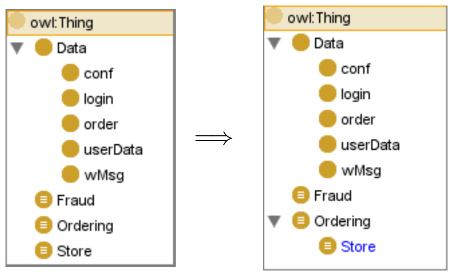


Fig. 2. Inference considering Behavior

7 Conclusions & Future Work

In this paper we have introduced a way to describe service behavior using OWL-DL. We described the different constructs for defining the mere existence of interactions, the sequence and the concurrency. With the use of DL as descriptive element our approach accomplishes the requirements of *robustness*, *extendability* and *integrability* often lacking in related work that describes service behavior aspects. On the basis of a simple example we finally showed the application of the constructs demonstrating the exploitation of standard reasoners. We see the solution described in this paper especially adaptable for the automation of the discovery and the composition of web services.

Based on the expressions given, and their evaluation by the subsumption-reasoning of a standard reasoner (also called "classification" [2, p. 48]), the task of finding Web services matching the technical (inputs and outputs) and behavioral aspects (causal constraints over inputs and outputs) of a Web service request can be automatically executed. Using the technique described for the matchmaking task, there is no additional application logics needed.

In addition, we understand our definition of a formal description of Web service behavior suitable for automatic matchmaking of requests and Web services as an important step towards the creation of an extensible, robust, automatic semantic Web service composer. The task of semantic Web service composition, amongst other jobs, mainly bases on well-known "syntactic" Web service composition as well as "semantic" discovery. Traditional Web service composition uses, e. g., planning [12] or configuration techniques [13] to come up with the composed workflow of Web services. Semantic composition mainly differs in the way the composer finds services being candidates for addition to the final workflow. For this step, it uses the additional information that is given in a semantic Web service description. This information may be the subsumption relations of different input and output elements, behavioral constraints, policy requirements and properties, or other arbitrary aspects of Web service descriptions. The described approach can therefore additionally be exploited for the tasks existent in the service composition as it supports the semantic description of services. Further research however has to occur in this area.

Future work will focus on three aspects of the service behavior description. (1) The combination of interactions in the sequence expression to cover all special cases in order to provide a set of DL constructs that allows to describe the main spectrum of service behavior. (2) An expression for modeling the *exclusive choice*. (3) The representation of cycles and advanced workflow patterns as extension to the basic expressions for describing even special service behavior. The future research, related to these aspects of describing the service behavior with the presented DL-constructs, will finally enable us to understand whether the expressibility of DL is enough for expressing the constraints that characterize service behavior.

References

1. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web Service Modeling Ontology. In: Applied Ontology 1. Volume 1. IOS Press (2005) 77–106
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: Description Logic Handbook, Cambridge University Press (2003)
3. Preist, C., Cuadrado, J.E., Battle, S., Grimm, S., Williams, S.K.: Automated business-to-business integration of a logistics supply chain using semantic Web services technology. In Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A., eds.: International Semantic Web Conference. Volume 3729 of Lecture Notes in Computer Science., Springer (2005) 987–1001
4. Grimm, S., Lamparter, S., Abecker, A., Agarwal, S., Eberhart, A.: Ontology based specification of Web service policies. In Dadam, P., Reichert, M., eds.: GI Jahrestagung (2). Volume 51 of LNI., GI (2004) 579–583
5. Bansal, S., Vidal, J.M.: Matchmaking of web services based on the daml-s service model (2003)
6. Ankolekar, A., Paolucci, M., Sycara, K.: Spinning the owl-s process model, toward the verification of owl-s process models (2004)
7. Rajasekaran, P., Miller, J., Verma, K., Sheth, A.: Enhancing web services description and discovery to facilitate composition (2004)
8. Börger, E., Stärk, R.F.: Abstract State Machines—A Method for High-Level System Design and Analysis. Springer-Verlag (2003)
9. Trainotti, M., Pistore, M., Calabrese, G., Zacco, G., Lucchese, G., Barbon, F., Bertoli, P., Traverso, P.: Astro: Supporting composition and execution of web services. In Benatallah, B., Casati, F., Traverso, P., eds.: ICSOC. Volume 3826 of Lecture Notes in Computer Science., Springer (2005) 495–501
10. Wahl, T.: Konzeption und realisierung einer ontologie zur modellierung und ableitung von geschaeftsprozessen. Diplomarbeit, Technische Universitaet Berlin, DEUTSCHLAND (2005)
11. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns (2002)
12. Pistore, M., Barbon, F., Bertoli, P., Shaparau, D., Traverso, P.: Planning and monitoring web service composition. In: AIMSA. (2004) 106–115
13. Stumptner, M.: Configuring web services. In: Proceedings of the Configuration Workshop at the 16th European Conference on Artificial Intelligence (ECAI). (2004) 10–1/10–6

POVOO – Process Oriented Views On Ontologies

Eva Gahleitner, Wolfram Wöß

Institute of Applied Knowledge Processing (FAW)
Johannes Kepler Universität Linz, Austria
A-4040 Linz, Altenbergerstr. 69
{egahleitner, wwoess}@faw.uni-linz.ac.at

Abstract. Ontologies still lack of including and considering the dynamic aspects of business processes. Therefore existing ontology-based information systems provide only static information which does not suit the actual working context of a user. In this project we extend information retrieval techniques with ontologies through a Process Oriented View On Ontologies (POVOO). The purpose is to satisfy a user with information that depends on the current process the user is working in. Due to a context aware approach it is possible to dynamically adapt the information to the user's current working situation. We introduce a methodology for generating views on ontologies and we illustrate how an application can use them to query highly specialized knowledge bases.

1. Introduction

Ontologies are widely used in the area of computer science, but did not really reach the step into the area of commercial business engineering. Whereas in the field of information retrieval (IR) ontologies emerged as a major support for improving the recall and precision of search mechanisms, they only play a subordinate role in process modeling[1]. Nevertheless, in daily work business processes are often the starting point for software development and define requirements for software systems. Research and industry have addressed the alignment of business processes and information technology (IT) only marginally. This leads to separate modeling areas: one for information management and retrieval and one for business engineering.

Ontology-based query techniques suffer from a number of disadvantages which have major impacts on their usage in business process modeling:

- Ontologies provide a single monolithic structure, splitting them up into small units is hardly possible.
- Ontologies do not consider dynamic aspects. A process is typically characterized by a dynamic sequence of events and operations. The need for knowledge may change according to these different process events and operations. For example, a technician who designs a new car engine needs information which is different from

[1] Within business process modeling ontologies are used to represent explicit formal specifications of the terms in the entire process management domain and relationships among them.

the information a worker at the assembly line or a car dealer requires for the customers.

- As the size of the ontology raises so does the complexity of its structure and therefore the complexity for a user to find the right concepts (cf. highly specialized ontologies in medicine like UMLS [24]).
- The context in which a user (an employee in a department, a user of a software application, etc.) works determines the user's view on the available knowledge. Much work has already been done in the field of context-based ontologies for certain users or user groups, but little for particular views on knowledge in the context of business processes (cf. [31]).
- One structure does not fit all: information can not easily be categorized within a single (tree) structure, so that users will always find what they are looking for. This is due to the multi-dimensional nature of the information. Any piece of information can be categorized according to one or more facets. Such a multi-facet categorization better reflects the different viewpoints one can have on a single piece of information.

In this work we introduce our approach to integrate views on ontologies in the information retrieval process with specific consideration of business processes. The acronym POVOO stands for *Process Oriented Views On Ontologies*. The purpose of POVOO is to satisfy a user with information that depends on the current process the user is working in. We propose a context aware solution which considers a user's working process and the corresponding information required by a user during certain tasks in this process. For example, when office workers are working on a specific task they are working in a certain context, thus only specific information is necessary to get the work done. Working contexts differ according to the required information and the involved people. This characteristic of work is exploited in the IR mechanism of our approach where views on ontologies represent the working contexts. In this way the system can search and present relevant information in the current context.

For highly specialized knowledge bases which can be found e.g. in medicine or biology, we assume that the information itself, which is relevant to execute a certain task (the documents in a knowledge base), stays the same, whereas the relationships, the various specialization and generalizations, the integration of various concepts in a new one, the ordering of the concepts etc. may differ depending on the user or the actual business process step. We therefore emphasize an approach which uses ontologies not only as simple vocabulary to define a lingua franca in business process engineering but rather as a way to structure the knowledge for particular processes. Within POVOO we develop a methodology for generating views on ontologies (which we call *ontology views*) and we demonstrate how applications can use them to query ontology-based knowledge bases.

The remainder of the paper is organized as follows: Section 2 overviews related work concerning ontologies in IR and process modeling, view based search and process oriented ontologies. Section 3 describes the characteristics of highly specialized knowledge bases, presents POVOO's three ontology levels and the concept of ontology views. Section 4 gives an explanation of our query mechanisms and how ontologies are connected with process modeling techniques. Finally, section 5 describes further research and concludes the paper.

2. Related Work

A business process defines the sequence of activities and the kind of resources (machine or human) which a process or an activity needs for its execution [3]. In recent years various (business) process modeling techniques have been introduced: Starting from the well known Petri nets [4] or high-level Petri nets [5], over UML activity diagrams [6] and object behavior diagrams [7] to more enterprise and business related techniques such as Event-Driven Process Chains (EPC) [8], UML Profile for Enterprise Distributed Object Computing (EDOC) [6] or the Business Process Modeling Language (BPML) [9].

All these techniques have in common that they describe the *behavior* of a system. By contrast, ontologies describe the *knowledge* of the system. Ontology is an explicit specification of a conceptualization [10], it captures the knowledge of a certain domain. But ontologies are not limited to the description of domain knowledge. They can also be used to define problem-solving knowledge (so-called task knowledge or task ontologies). In business engineering task ontologies create an ordering over sets of tasks and subtasks and are therefore defined as hierarchically ordered task ontologies [11] [12] [13].

Whereas in business engineering ontologies are simply used as a common vocabulary for processes and tasks, ontologies in information engineering are applied in various ways. E.g. in information retrieval (IR) ontologies have been commonly used to improve recall and precision [2]. Their main advantage relies in their ability to organize information into hierarchically ordered taxonomies of concepts, and to define attributes and relationships between these concepts. Two approaches are used in IR: *query expansion* and *conceptual distance measures*. The former expands the user query by adding terms semantically related to those used in the original user's query and therefore documents that do not necessarily contain the queried terms may be retrieved [14]. The latter uses a conceptual distance measure to calculate the similarity between terms in a query and terms in a document [15].

An extension to these IR methods is the concept of view-based or multi-faceted search methods [16] [17]. Here the idea is to organize the terminological keywords of the underlying knowledge base into various hierarchies which help the user to better formulate queries. For example, the keywords of a knowledge base can be ordered according to different aspects, e.g. "Time" or "Place". Such hierarchies are often called facet or views. The facets provide complementary views on the content along different dimensions.

However, existing multi-facet search tools use simple subclass-taxonomies [18]. They do not consider various relations between the concepts of an ontology (they are built for database querying). The Ontogator [19] approach combines the usage benefits of multi-facet search with the answer quality benefits of ontology-based search. But Ontogator does not support automatic querying; the users have to define the queries on their own.

3. Views on Ontologies

Ontologies describe those parts of knowledge, which are interesting for a certain domain. If a user likes to tailor ontologies to specific aspects of the phenomena of interest (e.g. to implement a certain application) he/she has to create different versions of the same ontology. Ontology versioning is a well known research area in the field of ontology engineering [1]. Unfortunately, these approaches only take care of the changes in the ontology *itself*, they do not deal with different *views* somebody may have when working with the ontology within a given process. In that case the ontology does *not* change, only the parts which are relevant to a certain user query change.

In general, views create virtual schemas and resource descriptions reflecting only the users' (applications') conception of a specific application domain. There is a large body of work on views for the relational data model. For example, the commonly used structured query language SQL [25] serves as a view definition language. By contrast, in the semantic web ontology views have been regarded only marginally until now. One known adoption is the use of scopes within topic maps [20] [21]. Currently, there are only two semantic web view languages for ontologies, both of them are built upon RQL query language and are aimed at RDF(S) data models: RDF View Language (RVL) [22] and the ontology view language proposed by Volz et al [23].

3.1 3-Level Architecture of Ontologies

Views (also called facets) provide complementary views on the content along different dimensions. They are widely used by database management systems (DBMS). A prominent role in DBMS plays the ANSI 3-schema architecture [26] that describes the different views on a database. In the center of the 3-schema architecture is the logical schema, which represents a complete business-oriented view on the information model. The underlying physical schema reflects the physical representation of data according to the requirements of the database. The external schema on top of the architecture represents specific views on the logical schema from the perspective of an individual application.

This ANSI reference model can be adapted to the area of ontologies (cf. Figure 1). The ontology concepts and their interrelationships, which are described according to the terms and principles of the domain (the semantic), represent the logical schema. We call this level the *semantic level.* The physical schema represents the syntactic specification of the ontology (e.g. built-in constructs given in RDF(S) or OWL) (*syntactic level*). The external schema is a mapping between the ontology schema and the schema the application is using. In the simplest case this is just a subset of the concepts, attributes, and relations of the ontology. For more complex applications, views are arranged in the way how the ontology concepts and relationships are viewed by an agent (human or software agent). More precisely, creating such a view over some data on the semantic web essentially consists of the creation of virtual metadata schemas and descriptions consistent with the agent's perception of those data. We call this level the *application level.*

Whereas the semantic and syntactic level is well discussed in the semantic web, views on ontologies are only marginally regarded. The application level has a major impact on the usability of existing ontology based knowledge systems. For example, ontology change management could be based on views where each view represents a major change in the knowledge model. Different versions of the same ontology could be specified in different views on that ontology. Additionally, working with views makes maintenance of ontologies easier.

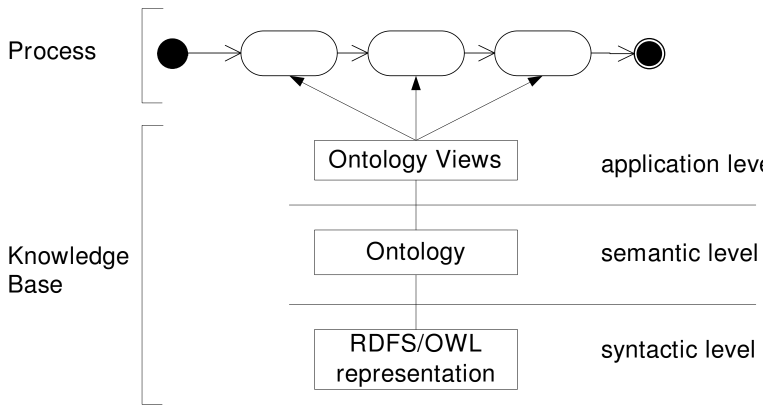


Fig. 1. Ontologies and ontology views analogous to the ANSI 3-level architecture

Another advantage of ontology views is that they describe information according to different contexts. This characteristic can be used to align ontology management systems with process oriented approaches. The kind of working process and its different conditions and dependencies between the single process steps have a major influence on the kind of information a worker requires. This information is described through various ontology views (compare Figure 1). The ontology views are based on the underlying ontology of the system (of the semantic level).

3.2 Specialized Domain Ontologies

Views on ontologies can only be built for relatively static and constant processes. Such processes can be found e.g. in medicine (e.g. diagnostic processes in medicine have a common structure). Ontologies belonging to specialized fields of long academic and professional tradition show a high degree of stability. Although disciplines such as medicine or biology have experimented drastic changes, this does not however mean that these novelties completely invalidate earlier conceptual organizations. An ontology about oncology may be affected by scientific advances but it is much less likely that it will be reformed in its totality. This relies on the high level of international consensus that some of these disciplines demonstrate.

Another feature of specialized field ontologies is the high granularity of their content. For example UMLS [24] provides a content base with highly specialized terms and documents. Additionally, the sources used for constructing specialized domain ontologies are well structured. Proof-reading and controlled communication leaded to a high formality of the sources. This high level of granularity makes it easier to split up knowledge in small, coherent knowledge pieces.

These properties (stability, high granularity and formality) make specialized domain ontologies a reliable resource for the retrieval of information, as well as a more effective one than its counterparts of non restricted fields and those used for common language. In other domains, which are not that structured and well defined the building of views on ontologies and the alignment of views to processes may lead to modeling problems.

3.3 Creating Ontology Views in POVOO

Existing approaches in view based ontology management [19] have a number of disadvantages:

- Views are only built on taxonomies: the multi-facet search just regards concept hierarchies (*subclass_of* or *part_of* relationships) not the entire semantic relationships of an ontology.
- The taxonomies are built on hierarchy rules which tell how to construct the taxonomies. In the mentioned approaches the taxonomies are built on simple Java applications and are therefore hardly to maintain.
- The GUIs are not suitable for large ontologies with various views on the ontology.

In POVOO we regard ontology views not only as a set of simple taxonomies. In our approach a view consists of various concepts, attributes and relationships which themselves build an ontology. The querying is therefore not restricted to a set of hierarchical ordered concepts but considers the entire semantic dependencies of concepts. Based on these modeling conventions, in POVOO views on ontologies are created in two ways:

- Manually by using an editor: this editor allows the integration of different classification schemas into the ontology as well as various relationships between the concepts. The editor will be integrated in the Protégé ontology editor framework[2].
- With the help of semantic web querying languages, e.g. RDQL (RDF Query Language) [27] and OWL-QL (OWL Query Language) [28].

The connection of process modeling and ontology views disburdens the user from choosing the right facets in the querying process. POVOO automatically identifies the necessary ontology views for the given process step, expands the user query and displays the result set according to this view. For example, when searching for medical reports one gets an anatomical ordering during the anamnesis process, whereas the same reports are relevant in a temporal ordering when preparing a surgery. Due to traceability reasons a user can always switch to a non-view-based search. The result set is then presented according to the underlying ontology of the information system.

[2] http://protege.semanticweb.org/

4. Integrating Process Models and Ontologies

When integrating structural and behavioral system aspects into an information system it is necessary to know a) which processes should be performed, b) who is responsible for certain tasks, c) which kind of information is needed and d) which resources are used. These different viewpoints are regarded in various business process engineering models. For example, ARIS (ARchitecture of integrated Information Systems [29]) a well known method in the German speaking part of Europe for analyzing processes distinguishes between a workflow model, functions, data and data flows as well as organizational units. In addition, INCOME/WF [30] follows a very similar approach. It supports four kinds of workflow views, an information object view, a view on existing resources, and a management view.

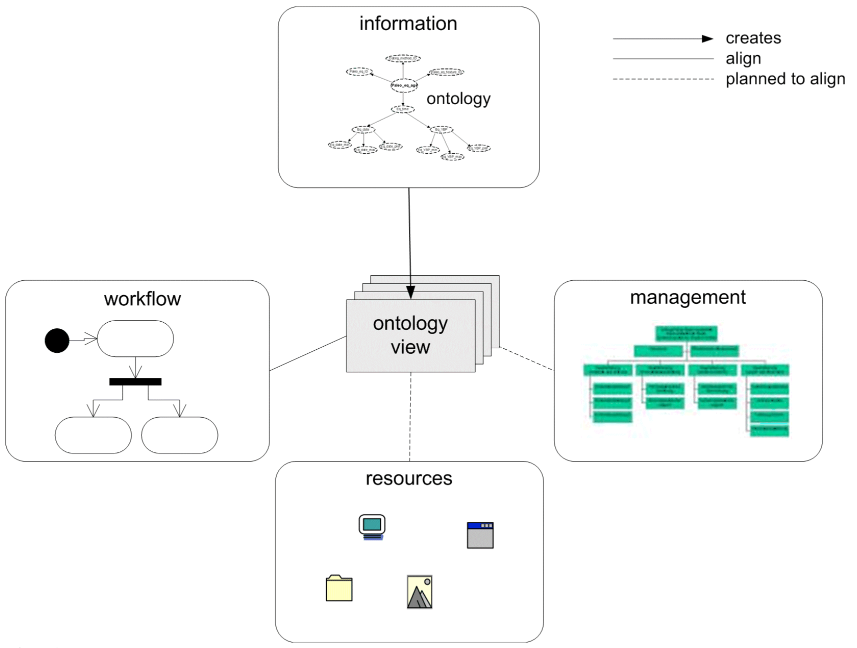


Fig. 2. Ontology views for integrating information, workflow, management and resource model

POVOO maintains the mentioned four viewpoints and connects them with the help of ontology views. In POVOO ontologies are used to represent the information model of a company's knowledge base. The information model (the ontology) is the central model for which various ontology views are built in order to connect all other models together (compare Figure 2). The workflow or process models are based on existing process modeling languages. Analogously this is true for the management and resource models.

In the first phase of our project we will place emphasis on the integration of ontologies and workflow models, in later stages we will also integrate of the used resources (e.g. lexicons) and the responsibilities of the users in the process (their position in the organization).

4.1 POVOO Querying Mechanisms

With POVOO a user has the opportunity to search for relevant information in two ways. Firstly, by using a simple keyword based search mechanism, and secondly, with the help of views on ontologies. The search mechanism then regards the certain role a user is playing when acting in a process. For example, in a medical environment a user may be a surgeon, an internist, a nursery, etc. who plays a certain role in the process. In the first phase of a process one may need more generic information including only generic knowledge bases, whereas in subsequent process steps one may need a more specialized view on the ontology, including more specialized knowledge bases.

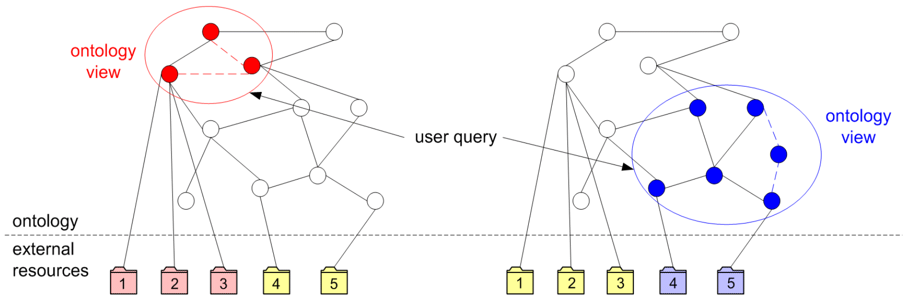


Fig. 3. View based search in POVOO

Figure 3 shows a possible scenario where different views provide different result sets for the same user query. The views are built on the same ontology. Within a view the structure of the ontology may be changed, e.g. a new or existing concept is added or deleted, or the relationship between concepts are changed, etc. Whereas on the left side documents 1, 2 and 3 are in the result set of the query, documents 4 and 5 are in the result set of the same query based on the view on the right side. Additionally, the result entries are organized according to the structure of the current ontology view. For traceability reasons the user can at any time switch to the underlying ontology.

4.2 Connecting Ontologies with Process Modeling Techniques

In order to connect ontologies with process modeling techniques we rely on existing process modeling languages described in section 2. It is more efficient to use established de facto standards instead of introducing another new process modeling language, which then may be perfectly suitable for aligning process models with ontologies, but which has no acceptance and no support in existing management tools.

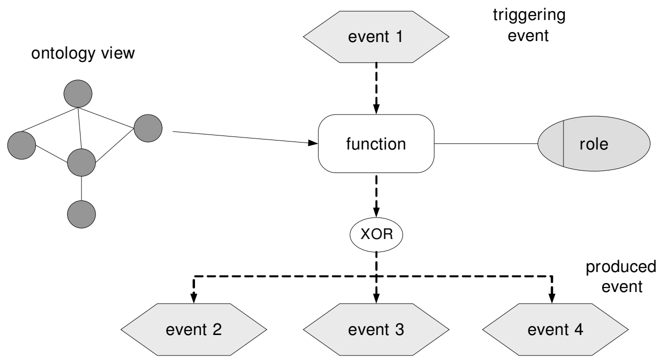


Fig. 4. Example connection EPC with Views

Figure 4 shows a connection between processes and ontology views. In this figure the modeling language EPC (Event-driven Process Chains) is connected with a certain ontology view for a certain process step (in EPC this is modeled by using "functions"). EPC is an important aspect of the ARIS model and connects all other views and describes the dynamics of the business process. Therefore it is possible to identify a user's role, the triggering event of the process step and the generated events. If the user acts in this specific role for this specific function the querying mechanisms are based on the given ontology view.

5. Conclusion and Further Research

In this paper a new approach to integrate ontology-based information systems with business process engineering is introduced. To connect and integrate both areas we use process oriented views on ontologies. Within our project we want to identify similarities and differences between ontology and process modeling techniques. We therefore analyze existing process modeling techniques such as EPC, UML 2.0, etc. and compare their possibilities to connect the various viewpoints on business models (information model, resources, management and workflow). Both techniques for process modeling and ontology modeling are then implemented in a prototype using existing tools (e.g. Protégé). In a later phase of this project we try to integrate the various used resources and the responsibilities a user has within a process.

One of our visions is to apply POVOO in a Grid Computing middleware layer, which integrates the underlying information resources and workflows based on grid computing technology and semantic mediation. GRID networks are characterized by a huge number of knowledge bases containing semantically related information that's DBMS make high demands on the used IR techniques. During the Austrian Grid Project [32] the G-SDAM (Grid Seamless Data Access Middleware) prototype is developed. G-SDAM enables electronic data interchange between various distributed and heavily heterogeneous information sources using semantic mediation techniques and allows (authorized parties) to seamlessly bind those information sources for querying and processing data in grid environments. G-SDAM processes queries over multiple data sources and translates data accordingly by applying domain ontologies.

References

1. M. Klein, D. Fensel, A. Kiryakov, D. Ognyanov, Ontology, Versioning and Change Detection on the Web, in 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), page 197 ff, Siguenza, Spain, October 1 - 4, 2002
2. E. Mönch, SemanticMinerTM: Ein integratives Ontologie-basiertes Knowledge Retrieval System, Workshop Ontologie-basiertes Wissensmanagement, 2. Konferenz Professionelles Wissensmanagement - Erfahrungen und Visionen, Schweiz, April 2003
3. Workflow Management Coalition, Terminology & Glossary, Document Number WFMC-TC-1011, Document Status - Issue 3.0, Feb 99, http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf
4. C.A. Petri, Kommunikation mit Automaten, Ph.D. thesis, Institut für Instrumentelle Mathematik, Bonn, 1962
5. W. Brauer, W. Reisig, Petri Nets: Central Models and Their Properties, Lecture Notes in Computer Science, Vol. 254: Advances in Petri Nets 1986, Part I, proceedings of an Advanced Course, Bad Honnef, Springer-Verlag, September 1986
6. Unified Modeling Language (UML), Object Management Group, http://www.uml.org/
7. P. Bichler, G. Preuner, and M. Schrefl, Workflow Transparency, in Advanced Information Systems Engineering, in Proceedings of the 9th International Conference (CAiSE '97), Barcelona, Spain, pp. 423 – 436, Springer Verlag, June 1997
8. J. Staud, Geschäftsprozessanalyse: Ereignisgesteuerte Prozessketten und objektorientierte Geschäftsprozessmodellierung für Betriebswirtschaftliche Standardsoftware, Berlin, Springer, 2001, ISBN 3-540-41461-4
9. Business Process Modeling Language (BPML), Business Process Management Initiative, http://www.bpmi.org/BPML.htm
10. T. R Gruber, A Translation Approach to Portable Ontology Specifications, in Knowledge Acquisition, Vol. 5(2), pp. 199-220, Academic Press, Stanford University, USA, 1993
11. V. R. Benjamins, L. Nunes de Barros, A. Valente, Constructing Planners through Problem-Solving Methods, in Proceedings of KAW'96 (Banff), pp. 14.1-14.20, 1996
12. A. Tate, Roots of SPAR - Shared Planning and Activity Representation, The Knowledge Engineering Review, Vol. 13(1), pp. 121-128, Special Issue on "Putting Ontologies to Use" (eds. Uschold, M. and Tate, A.), Cambridge University Press, 1998.
13. A. Pease, The Warplan: A Method Independent Plan Schema, in L. de Barros, R. Benjamins, Y. Shahar, A. Tate, A. Valente (edts), in Proceedings of the AIPS 1998, Workshop on Knowledge Engineering and Acquisition for Planning, AAAI Technical, Report WS-98-03, 1998
14. N. Guarino, C.Masolo, G. Vetere, Ontoseek: Content-based Access to the Web, IEEE Intelligent Systems, Vol. 14 (3), pp. 70 – 80, May 1999
15. L. Khan, Ontology-based Information Selection, PhD thesis, Department of Computer Science, University of Southern California, 2000
16. A.S. Pollit, The Key Role of Classification and Indexing in View-Based Searching, technical report, University of Huddersfield, UK 1998
17. E. Mäkelä, Hyvönen, T. Sidoroff, View-Based User Interfaces for Information Retrieval on the Semantic Web, in Proceedings of the ISWC-2005 Workshop End User Semantic Web Interaction, November, 2005
18. A. S. Pollitt, G. P. Ellis Martin P. Smith, HIBROWSE for Bibliographic Database, Journal of Information Science Vol. 20 (6), pp. 413 – 426, 1994, ISSN:0165-5515

19. E. Hyvönen, S. Saarela, K. Viljanen: Ontogator, Combining View- and Ontology Based Search with Semantic Browsing, in Proceedings of the International SEPIA Conference, Helsinki, September 18-20, 2003
20. XML Topic Maps (XTM) 1.0, TopicMaps.Org Specification, http://www.topicmaps.org
21. T. Luckeneder, K. Steiner, W. Wöß, Integration of Topic Maps and Databases: Towards Efficient Knowledge Representation and Directory Services, in lecture notes in Computer Science 2113, DEXA 2001 – 12th International Conference on Database and Expert Systems Applications, Springer-Verlag, pp. 744-753, München 2001
22. A. Magkanaraki, V. Tannen, V. Christophides, D. Plexousakis, Viewing the Semantic Web through RVL Lenses, in Proceedings of the 2nd International Semantic Web Conf., Sanibel Island, USA, pp. 96-112, 2003
23. R. Volz, D. Oberle, R. Studer, Views for Light-Weight Web Ontologies, in Proceedings of the ACM Symposium on Applied Computing, NY, USA, pp. 1168-1173, 2003
24. Unified Medical Language System, http://www.nlm.nih.gov/research/umls/
25. Structured Query Language, http://www.sql.org/
26. D. Tsichritzis, A. Klug, The ansi/x3/sparc/dbms Framework Report of the Study Group on Database Management Systems, in Information Systems Vol 3 (3), pp. 173 – 191, 1978
27. RDQL - A Query Language for RDF, W3C Member Submission 9 January 2004, http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
28. R. Fikes, P. Hayes, I. Horrocks, OWL-QL - A Language for Deductive Query Answering on the Semantic Web, Knowledge Systems Laboratory, Stanford University, CA, 2003
29. A. W. Scheer, Architektur integrierter Informationssysteme, Springer, Berlin/Heidelberg; 1992; ISBN 3540554017
30. A. Oberweis, R. Schätzle, W. Stucky, W. Weitz, G. Zimmermann, INCOME/WF – A Petri Net Based Approach to Workflow Management, in: H. Krallmann (Edt): Wirtschaftsinformatik '97, Springer-Verlag, pp. 557-580, 1997
31. A. Abecker, G. Metzas, M. Legal, S. Ntioudis, G. Papavassiliou, Business-Process Oriented Delivery of Knowledge through Domain Ontologies, in Proceedings of 2nd International Workshop on Theory and Applications of Knowledge Management, Munich, 3-7 September 2001
32. http://www.austriangrid.at/

Modeling Web Services with URML

Adrian Giurca[1], Sergey Lukichev[1], and Gerd Wagner[1]

Institute of Informatics, Brandenburg University of Technology at Cottbus
`{Giurca, Lukichev, G.Wagner}@tu-cottbus.de`

Abstract. A Web service can be specified on the basis of a business vocabulary, including a business event model, and of a rule-based behavior model. We use the general rule language R2ML and the UML-based rule modeling language URML for modeling the behavior of Semantic Web services.

1 Introduction

A Web Service is a software application identified by a URI, whose interfaces and bindings are capable of being described in XML, e.g. by means of the Web Service Description Language (WSDL)[9] and which can directly interact with other software agents using XML-based messages (represented in the SOAP format [4]) exchanged via Internet protocols (HTTP or SMTP).

In this paper we discuss the modeling of Web Services with the help of the UML-Based Rule Modeling Language (URML)[6] developed by the REWERSE Working Group I1[1] for the semantic business process management on the Web. In order to support rule interchange between different rule engines, we have developed a general rule language, called R2ML ([10], [11]), with an XML serialization format. The metamodels of URML and R2ML largely overlap. URML can be considered as a language that is derived from R2ML in order to provide UML-based rule modeling.

We envision the following business process modeling scenario:

– A business process modeler models Web services based on a vocabulary and rules with a rule modeling tool, f.e. Strelka ([3]);
– The entire Web service specification can be serialized using WSDL and the rule description language R2ML.

A rule-based Web service may be implemented using a reaction rule engine and a SOAP listener. The rule engine receives SOAP messages, executes triggered rules and performs actions. The SOAP listener is a Web application, which captures SOAP messages via HTTP(S) and passes them to the rule engine.

In Section 2 we describe R2ML reaction rules, in Section 3 we describe the R2ML event metamodel with the focus on atomic events, represented by SOAP messages, in Section 4 we describe R2ML actions, and in Section 5 we give an example of a reaction rule, modeled with URML and serialized into R2ML. In Section 6 we give an outline of the future work and conclusions.

[1] Working Group I1 `http://www.rewerse.net/I1`

2 Reaction Rules

The main goal of R2ML is to provide a representation of rules, targeted to the rule engine platform that is independent of a vendor specific engine (PIM Level). This allows R2ML to be a rule interchange language between different PSM level specific rule engines. This section presents R2ML *reaction rules* also known as *Event-Condition-Action rules* (*ECA rules*) and their usage in describing business process modeling.

There are several advantages of using REACTION rules for specifying business processes: business requirements are often captured in the form of rules in a natural language, formulated by business people; REACTION rules are easier to maintain and integrate with other kinds of rules, used in business applications (integrity rules, which specify constraints the data must fulfill, derivation rules, which explain how a model element can be derived); the topic of rules validation and verification is well-studied; REACTION rules emphasis on events gives a flexible way to specify control flow.

The R2ML metamodel for reaction rules is depicted in Figure 1.

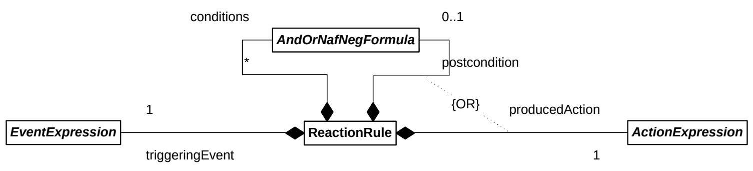


Fig. 1. Reaction rule metamodel

A *reaction rule* is a statement of programming logic that specifies the execution of one or more actions in the case of a *triggering event* occurrence and if rule *conditions* are satisfied. Optionally, after the action execution *post-conditions* may be made true.

Reaction rules therefore have an operational semantics (formalizing state changes, e.g., on the basis of a state transition system formalism).

A reaction rule has the following components:

– `triggeringEvent` is an R2ML EventExpression, which is either atomic or composite (Figure 2);
– `conditions` are represented as a collection of quantifier free logical formulas;
– `producedAction` is an R2ML action, which represents the state change of the system. The latest version of R2ML defines composite actions, which are, for instance, sequential actions and parallel actions.
– an optional `postcondition` specifies a state change in a declarative manner.

All components of a reaction rule contain expressions that refer to rule variables. The R2ML distinguishes between object variables, which are instantiated with objects, and data variables, which are instantiated with data values. The rule variables are bound to specified classes/datatypes. An object variable in R2ML can be bound to a specific class either using an `ObjectClassificationAtom` or the optional attribute `classID` of the variable. Similarly, a data variable can be bound using a `DataClassificationAtom` or the optional attribute `datatypeID` of the variable.

3 R2ML Events Metamodel

The R2ML Events metamodel (see Figure 2) specifies the core concepts, which are necessary for dynamic behavior of rules and provides the infrastructure for the support of more detailed behavior definition.

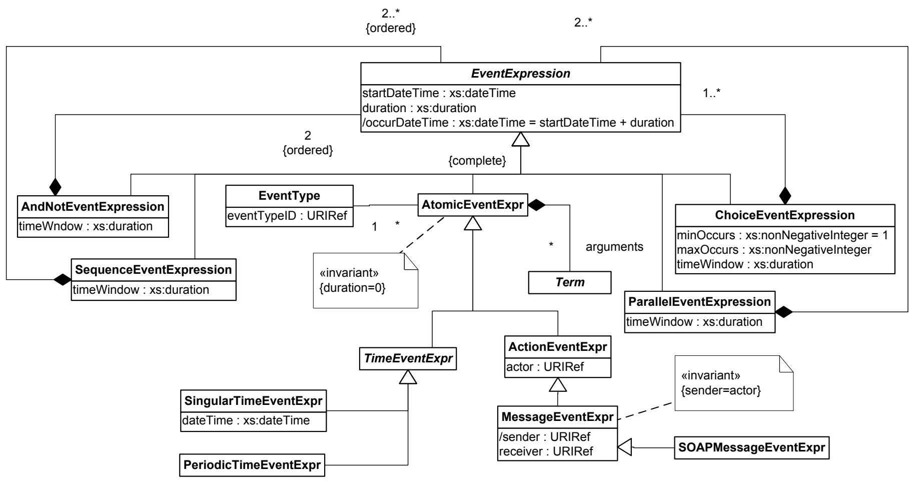


Fig. 2. R2ML Event Expressions

The basic properties of an *R2ML event expression* are:

– `startDateTime` is an event start date and time;
– `duration` is a value specification that specifies the temporal distance between two time expressions, which define time instants;
– `occurDateTime` is a derived property, which is given by the addition of duration to the existent start date time.

All R2ML Events are subclasses of `EventExpression`. `EventExpression` is either a composite event or an atomic event.

3.1 Composite event

Composite event in R2ML is either an AndNotEventExpression, a SequenceEventExpression, a ParallelEventExpression and a ChoiceEventExpression. Each event expression has a property `timeWindow`, which represents the duration of the corresponding event observation. The event expression metamodel is depicted in Figure 2.

AndNotEventExpression has two event expressions as arguments (`EvtExpr1` and `EvtExpr2`). It describes a complex event where an instance of `EvtExpr1` but no instance of `EvtExpr2` occurs.

SequenceEventExpression refers to an ordered list of event expressions, which are processed in a sequence of events, following the existent order and considering a finite value of `timeWindow` observation.

ParallelEventExpression refers to a collection of events that are concurrently processed inside of the corresponding `timeWindow`.

ChoiceEventExpression refers to a collection of events that requires processing of at least one event expression from the collection inside of the corresponding `timeWindow`.

3.2 Atomic event

Atomic event in R2ML is an *AtomicEventExpression*, which main characteristic is that it has no duration (`duration = 0`). As a consequence, the occurrence date time is the same as the start date time.

An atomic event expression:

– Refers to an EventType, which is its classifier;
– Is composed from an ordered, possible empty, list of terms as arguments.

The R2ML distinguishes between two main classes of atomic events: `MessageEventExpression` and `TimeEventExpression`.

Message event expression has a property `sender`. In the discussing approach for business process modeling in web services, a *sender* may be HTTP_REFERER. One category of message event is a SOAP message event.

SOAP Messages Events. SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics[4].

We use SOAP messages as transport containers for events, which are expressed in R2ML as `SOAPMessageEventExpr`'s (see Figure 3). SOAP is typically used for Remote Procedure Calls (RPC). The SOAP specification[4] defines two special message formats: a SOAP RPC Request Message, represented in R2ML by `SOAP-RPC-RequestMsgEvtExpr` and a SOAP RPC Response Message, represented in R2ML by `SOAP-RPC-ResponseMsgEvtExpr`.

The following example shows a sample SOAP message, which contains an R2ML SOAP RPC request. The message contains two pieces of application-defined data not defined by the SOAP specification: a SOAP header block and a body element with a local name of `ref`. In general, SOAP header blocks contain information which might be of use to SOAP intermediaries as well as the ultimate destination of the message.

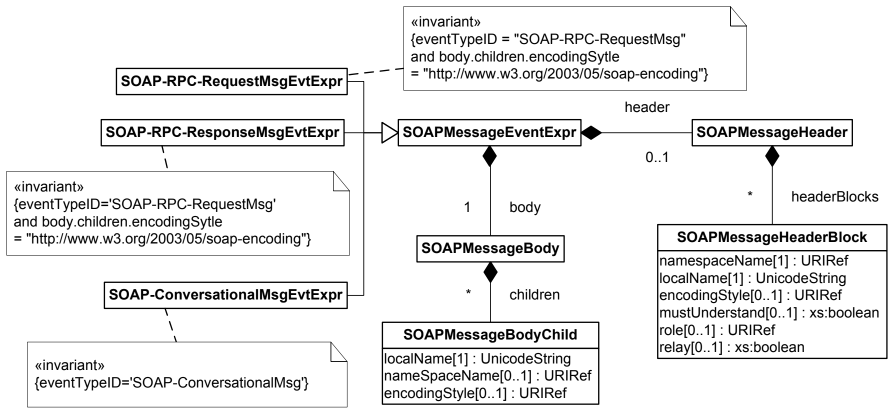

«invariant»
{eventTypeID = "SOAP-RPC-RequestMsg"
and body.children.encodingSytle
= "http://www.w3.org/2003/05/soap-encoding"}
SOAP-RPC-RequestMsgEvtExpr
SOAP-RPC-ResponseMsgEvtExpr
SOAPMessageEventExpr
header
SOAPMessageHeader
0..1
«invariant»
{eventTypeID='SOAP-RPC-RequestMsg'
and body.children.encodingSytle
= "http://www.w3.org/2003/05/soap-encoding"}
1
body
*
headerBlocks
SOAPMessageBody
SOAPMessageHeaderBlock
namespaceName[1] : URIRef
localName[1] : UnicodeString
encodingStyle[0..1] : URIRef
mustUnderstand[0..1] : xs:boolean
role[0..1] : URIRef
relay[0..1] : xs:boolean
SOAP-ConversationalMsgEvtExpr
*
children
SOAPMessageBodyChild
localName[1] : UnicodeString
nameSpaceName[0..1] : URIRef
encodingStyle[0..1] : URIRef
«invariant»
{eventTypeID='SOAP-ConversationalMsg'}

Fig. 3. SOAP message event expression in R2ML

In this example an intermediary might prioritize the delivery of the message based on the priority and expiration information in the SOAP header block. The body contains the actual event payload, in this case the customer's request for a car .

Example 1 (SOAP RPC Request).

```
 <?xml version='1.0' ?>
 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  <r2ml:SOAP-RPC-RequestMsgEvtExpr xmlns:rew="http://www.rewerse.net/I1/R2ML"
       r2ml:sender="eshop.com"
       r2ml:startTime="2006-03-21T09:00:00"
       r2ml:duration="P0Y0M0DT0H0M0S"
       r2ml:eventTypeID="productOrder">
1    <r2ml:arguments>
```

```
2     <r2ml:ObjectVariable r2ml:name="car" r2ml:classID="srv:Car"/>
3    </r2ml:arguments>
4    </r2ml:SOAP-RPC-RequestMsgEvtExpr>
  </env:Body>
 </env:Envelope>
```

The body of the SOAP is an R2ML SOAP-RPC-RequestMsgEvtExpr. Lines 1-3 define a list of arguments for the event: the ObjectVariable in line 2 is a variable `car` of type `srv:Car`, which is a particular car, requested by the customer.

4 Actions

The R2ML supports both production rules and reaction rules. With this respect it defines the concept of an *action*. Following the OMG Production Rule Representation submission[2], an action (Fig. 4) is either an *InvokeActionExpression* or an *AssignActionExpression* or a *CreateActionExpression* or a *DeleteAction-Expression*. The R2ML provides also message actions in the form of a concrete *SOAPMessageEventExpr*.

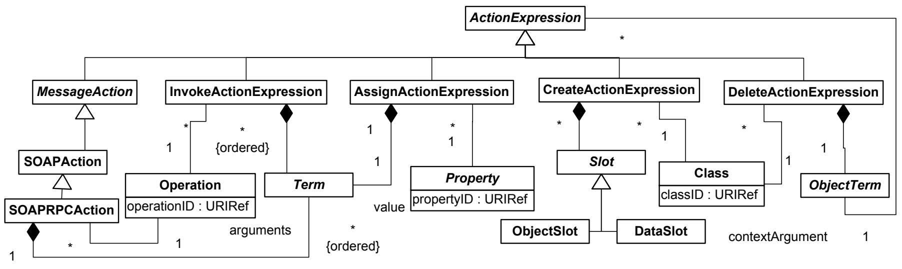


Fig. 4. Actions

All actions refer to a *context* which is an R2ML object term.

InvokeActionExpression models an object operation invocation. It refers to a UML *Operation* and contains an ordered, possible empty list of arguments represented as R2ML terms. The execution of this action is done by the corresponding operation-call.

Example 2 (InvokeActionExpression).

"Calculate the total payment of the purchase order."

```
<InvokeActionExpression r2ml:operationID="totalPayment">
 <contextArgument>
  <ObjectVariable r2ml:name="purchaseOrder" r2ml:classID="Order"/>
 </contextArgument>
</InvokeActionExpression>
```

In this example the operation `totalPayment` has no arguments.

AssignActionExpression refers to a UML *Property* and contains a DataTerm as a *value*. This action assigns a value to a property.

Example 3 (AssignActionExpression).

> "Set to `10` the property `discount` of the object variable `purchaseOrder` (`purchaseOrder.discount = 10`)."

```
<AssignActionExpression r2ml:propertyID="discount">
 <contextArgument>
   <ObjectVariable r2ml:name="purchaseOrder"
                   r2ml:classID="Order"/>
 </contextArgument>
 <value>
  <TypedLiteral r2ml:lexicalValue="10"
                r2ml:type="xs:positiveInteger"/>
 </value>
</AssignActionExpression>
```

CreateAction refers to a UML *Class* and contains a list of slots (object slots and/or data slots). The execution of this action consist in a constructor-call for creation of a new object in the system.

Example 4 (CreateActionExpression).

> "Create purchase order for one book named 'Harry Potter' with the price 11.25 and discount 10

```
<CreateActionExpression r2ml:classID="Order">
 <contextArgument>
   <ObjectVariable r2ml:name="purchaseOrder"/>
 </contextArgument>
 <DataSlot r2ml:attributeID="title">
  <TypedLiteral r2ml:lexicalValue="Harry Potter"
                r2ml:type="xs:string"/>
 </DataSlot>
 <DataSlot r2ml:attributeID="price">
   <TypedLiteral r2ml:lexicalValue="11.25"
                 r2ml:type="xs:float"/>
 </DataSlot>
 <DataSlot r2ml:attributeID="discount">
   <TypedLiteral r2ml:lexicalValue="10"
                 r2ml:type="xs:positiveInteger"/>
 </DataSlot>
</CreateActionExpression>
```

DeleteActionExpression refers to an UML *Class* and contains an *ObjectTerm*. This action removes an instance of the *Class*.

Example 5 (DeleteActionExpression).

Delete order `puchaseOrder`.

```
<DeleteActionExpression r2ml:classID="">
 <contextArgument>
   <ObjectVariable r2ml:name="purchaseOrder" r2ml:classID="Order"/>
 </contextArgument>
</DeleteActionExpression>
```

5 Business Process Modeling Example

Let's consider a part of a business process when a customer makes a request for a book from a web site. The customer fires an event, which is captured by the server. The server searches for an appropriate rule for this event and checks rule condition: whether the requested book is available or not. If the condition holds, i.e. the book is available, then it performs an action: approve order. The rule postcondition is that the amount of books in stock must be less by a requested quantity than before the rule execution. A part of the business vocabulary is depicted on Figure 5. The rule is modeled using a URML[6].

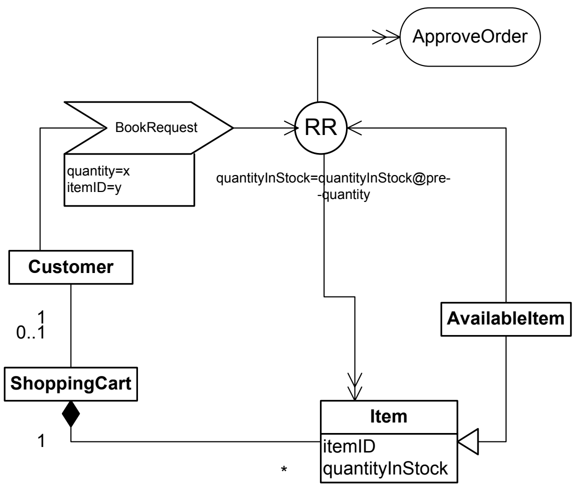


Fig. 5. On customer book request, if the book is available, then approve order and decrease amount of books in stock.

The business vocabulary consists of a customer, which may have a shopping cart. A shopping cart consists of items. An item has an *itemID* and *quantityIn-Stock*. There is an item category *AvailableItem*, which contains items, available for the order and delivery. In the URML a rule is represented as a circle with a label "RR". Incoming arrow from AvailableItem is a rule condition. SOAP RPC request message is represented with a UML signal sign and connects a customer, who fires the event and a rule circle. The event contains list of parameters: *itemID* and *quantity.* Outgoing double-head arrow to an activity *ApproveOrder* is an action to approve the order. This action is defined in the WSDL interface of the service. Outgoing double-head arrow to an *Item* class represents rule postcondition with an OCL expression `quantityInStock=quantityInStock@pre-quantity`, that states that the amount of items in stock must be less by 1.

In order to be processed by the ECA engine, this rule should be serialized into rule interchange format R2ML. Corresponding R2ML syntax is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<r2ml:ReactionRuleSet xmlns:dc="http://purl.org/dc/elements/1.1/"
                     xmlns:rew="http://www.rewerse.net/I1/R2ML"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://www.rewerse.net/I1/R2ML"
                     r2ml:id="ID000000">
    <r2ml:ReactionRule r2ml:id="rr1111" xmlns:srv="http://www.example.org/">
        <r2ml:RuleText r2ml:ruleDiagram="" r2ml:textFormat="text/xml"
                                         r2ml:ruleVocabularyDiagram=""/>
        <r2ml:SourceCode r2ml:language="R2ML"/>
        <dc:subject>Reaction rules, R2ML, Markup Languages</dc:subject>
        <r2ml:triggeringEvent>
            <r2ml:SOAPMessage r2ml:sender="" r2ml:startTime="2006-03-21T09:00:00"
                             r2ml:duration="P0Y0M0DT0H0M0S"
                             r2ml:eventTypeID="productOrder">
                <r2ml:arguments>
                    <r2ml:ObjectVariable r2ml:name="x"
                                        r2ml:classID="srv:Item"/>
                    <r2ml:DataVariable r2ml:name="quantity"
                                        r2ml:dataTypeID="xsd:integer"/>
                    <r2ml:ObjectVariable r2ml:name="customre1"
                                        r2ml:classID="srv:Customer"/>
                </r2ml:arguments>
            </r2ml:SOAPMessage>
        </r2ml:triggeringEvent>
        <r2ml:conditions>
            <r2ml:ObjectClassificationAtom r2ml:classID="srv:AvailableItem">
              <r2ml:ObjectVariable r2ml:name="x"/>
            </r2ml:ObjectClassificationAtom>
1           <r2ml:EqualityAtom>
              <r2ml:AttributeFunctionTerm r2ml:attributeID="srv:quantityInStock">
                <r2ml:contextArgument>
                  <r2ml:ObjectVariable r2ml:name="i" r2ml:classID="srv:Item"/>
                </r2ml:contextArgument>
              </r2ml:AttributeFunctionTerm>
              <r2ml:DataVariable r2ml:name="q" r2ml:dataTypeID="xsd:integer"/>
            </r2ml:EqualityAtom>
        </r2ml:conditions>
        <r2ml:producedAction>
            <r2ml:SOAPRPCAction r2ml:operationID="approveOrder">
                <r2ml:contextArgument>
                    <r2ml:ObjectVariable r2ml:name="x"/>
                </r2ml:contextArgument>
                <r2ml:arguments>
                    <r2ml:ObjectName r2ml:objectID="customer1"
                         r2ml:classID="srv:Customer"/>
                </r2ml:arguments>
            </r2ml:SOAPRPCAction>
        </r2ml:producedAction>
        <r2ml:postcondition>
2         <r2ml:EqualityAtom>
3           <r2ml:AttributeFunctionTerm r2ml:attributeID="srv:quantityInStock">
```

```
              <r2ml:contextArgument>
                <r2ml:ObjectVariable r2ml:name="i" r2ml:classID="srv:Item"/>
              </r2ml:contextArgument>
            </r2ml:AttributeFunctionTerm>
4           <r2ml:DataOperationTerm r2ml:operationID="minus">
              <r2ml:contextArgument>
                <r2ml:ObjectVariable r2ml:name="i" r2ml:classID="srv:Item"/>
              </r2ml:contextArgument>
              <r2ml:arguments>
                <r2ml:DataVariable r2ml:name="q"/>
                <r2ml:DataVariable r2ml:name="quantity"/>
              <r2ml:arguments>
            </r2ml:DataOperationTerm>
          </r2ml:EqualityAtom>
       </r2ml:postcondition>
   </r2ml:ReactionRule>
</r2ml:ReactionRuleSet>
```

It is important to note, that the postcondition expression `quantityInStock=quantityInStock@pre-quantity` is represented as combination of 2 atoms: equality atom in the condition part of a rule (line 1), which is considered as a variable `q` initialization with initial value of the attribute `quantityInStock` and equality atom in the postcondition part of the rule (line 2), which is considered as an assignment of a new value for the attribute `quantityInStock`. The new value is specified by the DataOperationTerm (line 4) with "minus" operation on old attribute value `q` and quantity `quantity`. For more rule examples in URML and R2ML we refer to the Working Group I1 web site and, in particular, to the EU-Rent case study[2], which contains a domain model and rules, modeled using URML and to the R2ML example rule set[3].

The transformation of URML model into R2ML is implemented in the Strelka tool. Metamodels of R2ML and URML largely overlap and URML concepts like rule, condition and conclusion can be directly mapped into R2ML. So called OCL filter expressions, used in URML conditions and postconditions to filter instances of a conditioned classifier (f.e. class or association), can be represented in the R2ML since it has corresponding functional atoms for representing OCL expressions. Since R2ML does not support collections yet, not all OCL expressions can be serialized into R2ML.

6 Conclusion

In this paper we have shown how a UML-Based Rule Modeling Language can be used for the modeling of Web Services, based on reaction rules. We have also presented a part of the R2ML language, related to reaction rules and gave examples of rule modeling in XML syntax of R2ML.

Concerning the future work on this topic we consider the following issues:

[2] EU-Rent Case Study in URML, using Strelka tool: `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/12`

[3] R2ML project page: `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/6`

– Give a focus on rule sets and respective control flow modeling in URML. Control flow modeling by means of reaction rules has been already introduced in [5] and we are going to adopt it in URML;
– Analyze the suitability of R2ML for expressing control flow patters, identified, for instance, by Van der Aslst et al. [7]. Control flow patterns representation by means of reaction rules has been specified in [8] and we have to investigate how they can be captured in reactive rules part of R2ML;
– R2ML needs a mechanism to specify exceptions.
– The issue of web service composition is currently under consideration.

References

1. Gelfond, M., Lifschitz, V., The stable model semantics for logic programming, In Proc. of ICLP-88, pp. 1070-1080.
2. W3C Workgroup on RIF Charter, `http://www.w3.org/2005/rules/wg/charter`
3. Strelka - A UML-Based Visual Rule Modeling Tool. `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/10`
4. SOAP Version 1.2 Part 1: Messaging Framework W3C Recommendation 24 June 2003, `http://www.w3.org/TR/soap12-part1/`
5. Wagner G., The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. Information Systems 28:5 (2003), pp. 475-504.
6. A UML-Based Rule Modeling Language (URML) on REWERSE Working Group I1 website: `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/7`
7. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1) (2003)
8. Taveter, K.: A multi-perspective methodology for agent-oriented business modelling and simulation. PhD thesis, Tallinn University of Technology, Estonia, 2004 (ISBN 9985-59-439-8)
9. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Candidate Recommendation 27 March 2006 `http://www.w3.org/TR/wsdl20`
10. G. Wagner, A.Giurca, S. Lukichev (2005). R2ML: A General Approach for Marking up Rules, Dagstuhl Seminar Proceedings 05371, in F. Bry, F. Fages, M. Marchiori, H. Ohlbach (Eds.) Principles and Practices of Semantic Web Reasoning, `http://drops.dagstuhl.de/opus/volltexte/2006/479/`
11. Wagner, G., Giurca, A., Lukichev, S. (2006). A Usable Interchange Format for Rich Syntax Rules. Integrating OCL, RuleML and SWRL. Will appear in proceedings of Reasoning on the Web Workshop at WWW2006, May 2006

m3pl: A Work-FLOWS ontology extension to extract choreography interfaces

Armin Haller and Eyal Oren

Digital Enterprise Research Institute (DERI)
Galway, Ireland
firstname.lastname@deri.org

Abstract. Cross-organisational interoperability is a key issue for success in B2B e-commerce applications. To achieve this interoperability, choreography descriptions are necessary that describe how the business partners can cooperate. In existing approaches, these choreography descriptions are independent of the internal workflows of the partners.
We present a framework for extracting choreography interface descriptions from internal workflow models. Our approach comprises two steps: first we map internal workflow models into a intermediary formal model, then we generate choreography interfaces from it. In this paper we present m3pl, an ontology extension based upon the First Order Ontology for Web Services (FLOWS) [2]. The extensions provide relations to model workflow views and choreography interfaces.

1 Introduction

Organisations offer business functionalities to their customers, and implement these functionalities in their business processes. For years, organisations have used Workflow Management Systems (WfMSs) to describe and execute their business processes [6]. Underlying these WfMS are different workflow languages with many different metamodels. These workflow languages vary in the available modelling constructs and in the semantics of their constructs. To capture these semantics and to allow interoperability of WfMS the Process Specification Language (PSL) [16] was developed. PSL is an ontology that defines workflow concepts and their semantics. Various extensions have been developed (as part of the PSL standard), including the First Order Logic for Web Services (FLOWS) [2] ontology for modelling (compositions of) Web Services.

With the advent of Service Oriented Computing [13] organisations started to expose their business functionality explicitly as reusable and composable services using standardised protocols such as WSDL and SOAP. Web Services abstract the access to the business functionality from the specifics of programming languages. For using these services organisations provide choreography descriptions written in languages such as WS-CDL [11], Abstract BPEL [20] or ebXML CPP [10]. A choreography describes the message exchange patterns employed by a Web Service interface. These patterns describe how consumers should interact with the Web Service; they can be described from a global (collaboration) viewpoint or from a local (participant) viewpoint. We will use the term *choreography* for the global viewpoint, and *choreography interface* for the participant's viewpoint[1].

A fundamental limitation in current approaches to model choreographies is its independence to the underlying workflow definitions. Although a few recently published work address the correlation between a choreography interface and its underlying workflow, current approaches do not focus on an automated mapping between them. This independence leads to two problems: (1) if any change occurs in the internal workflow model, choreography descriptions have to be manually synchronised with the workflow definition, and (2) it is not possible to automatically verify consistency of internal workflow descriptions and external choreography interfaces.

This paper presents a framework for combining internal workflow definitions and external choreography descriptions; an overview is shown in Fig. 1. With the framework one can semi-automatically generate choreography interfaces in various languages from workflow models in various languages. The framework is based on PSL [16], an ontology for capturing business processes and FLOWS [2], an extension to PSL for Web Service interactions.

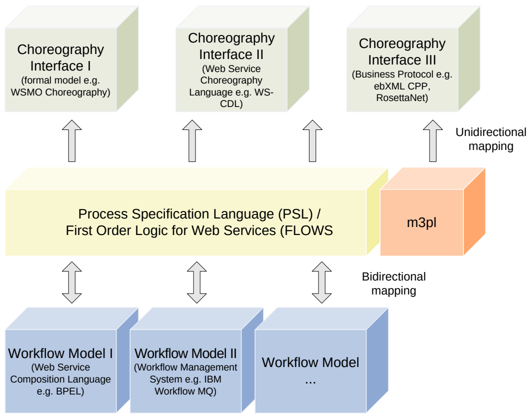

Choreography Interface I
(formal model e.g. WSMO Choreography)
Choreography Interface II
(Web Service Choreography Language e.g. WS-CDL)
Choreography Interface III
(Business Protocol e.g. ebXML CPP, RosettaNet)
Unidirectional mapping
Process Specification Language (PSL) / First Order Logic for Web Services (FLOWS
m3pl
Bidirectional mapping
Workflow Model I
(Web Service Composition Language e.g. BPEL)
Workflow Model II
(Workflow Management System e.g. IBM Workflow MQ)
Workflow Model
...

Fig. 1. Relating workflow models to choreography interfaces

The paper is structured as follows: based on a motivating RosettaNet collaboration example described in section 2, we analyse the requirements for our framework in section 3. We present our ontology in section 4. In section 5 we outline the methodology to follow to map from the internal workflow model to m3pl and to extract different choreography interfaces. Finally we discuss related work in section 6 and conclude in section 7.

[1] The choreography interface is also called behavioural interface by Dijkman and Dumas [5] or abstract process in BPEL4WS [20].

2 Motivating Example

In this section we present an example cross-organisational collaboration. We will illustrate the problems that companies face when designing collaborative business processes with a request-for-quote (RFQ) process.

2.1 Current situation

An automotive parts vendor implements and executes his internal processes with IBM Websphere MQ Workflow[2]. One of the vendor's processes concerns the processing of requests for quotes. Figure 2 shows a simplified view of this modelled in MQ Workflow. The symbols on the left of the picture denote a source and sink node and represent the start and end of the MQ Workflow process model. Dashed arrows show data transferred between activities and solid arrows denote the control flow.

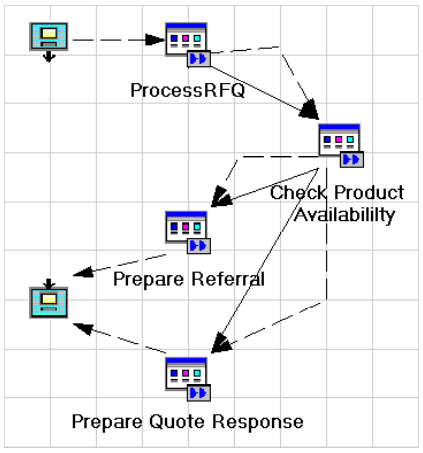


Fig. 2. IBM MQ Workflow RFQ

The process starts with an RFQ from a customer. The vendor checks whether the requested part, say an electric generator, is available in stock and can be delivered within the time specified. If the product is available the vendor prepares a quote, otherwise he returns a referral including the reason for non-delivery.

2.2 Preferred situation

The vendor wants to automate the collaboration with his partners. This would minimise the manual labour by enforcing partners to directly invoke interfaces to

[2] for our analysis we have used v3.4 of the product.

its internal WfMS. An example for such an automation is the initial data input. Currently this data is manually entered into the system; the goal of the vendor is for this input to come directly from the external business partner. To enable automatic collaboration the vendor needs to describe the public view on his business processes. To comply with industry standards this public process should conform to the standardised RosettaNet choreography interface PIP 3A1[3]; which describes a request for quotation.

Figure 3 shows a RosettaNet collaboration and the internal process model described above in a UML activity diagram. Public activities (the RosettaNet PIP 3A1) are displayed in black and private activities in white. The seller's choreography is formed by the black activities in the right swimlane and the buyer's choreography by the black activities in the left swimlane respectively.

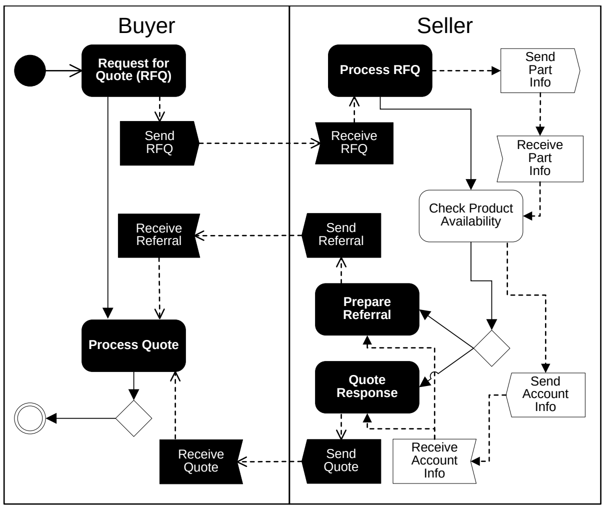


Fig. 3. External Process (RosettaNet PIP)

In this example the internal workflow is straightforward and for the purpose of simplification it is already aligned to an external standard process in terms of a RosettaNet PIP 3A1. Thus it is not difficult to model the external part of the process in any choreography description language. However, in reality the processes can be significantly more complex, and automatic extraction of choreography interfaces is desired.

In order to automatically extract the choreography interface, the internal business process has to be extended by information specific to external processes identified in the following section. Subsequently the model should be extracted to a choreography descriptions language. These features are currently not offered by MQ Workflow or any other WfMS.

[3] http://www.rosettanet.org/PIP3A1.

3 Requirements Analysis

We can derive four basic requirements for the above collaboration scenario. They also reflect requirements on a choreography language identified in [1].

1. **Model internal workflow:** we need to model the internal workflow (shown in figure 2) of the business partner whose choreography interface we want to generate (the supplier in this example).
   The internal workflow has to be formally specified to describe the business processes unambiguously. A mere syntactic model could lead to inconsistent interpretations; e.g. a *split* can have different meanings in different models.
2. **Model choreography-related concepts in the workflow:** to generate the choreography interface from the internal workflow we need to add additional annotations. These annotations (such as visibility of activities or role of partners) are not part of the internal workflow, because they are of no significance for workflow enactment, but only for a cross-organisational choreography. It is necessary to annotate:
   (a) the **choreography interface** as a partial view on the internal workflow of the service provider. Choreography interfaces are currently modelled in a multitude of languages. These languages are either **task-flow based** (i.e. WS-CDL, BPEL4WS) or **dependency based** (i.e. WSMO Choreography, OWL-S Process Model). Thus the choreography interface model has to be capable of capturing both modeling alternatives.
   (b) the **visibility** of tasks: some tasks in a collaboration are **private**, other are **public**. Also, some tasks might be publicly visible to one participant, but private to another. The generated choreography interface for one partner should only include the tasks marked as visible to him.
   (c) the **role** a party can play when engaging in collaborations. A role defines the observable behaviour a party exhibits in order to collaborate with other parties. A "buyer" role for example is associated with the purchase of goods or services and the "seller" role is associated with providing those goods or services.
   (d) the **direction of the communication** represents the communication route in a specific interaction and represents constraints on what roles have to be adopted by the participants. A wholesaler for example might play the "seller" role in one interaction and the "buyer" role in some other interaction. A direction relation requires a sending and a receiving **participant**.
   (e) **messages**. As it can be seen in figure 3 messages are used to transfer data between activities. The explicit representation of messages is commonly not part of workflow models. Even if this fundamental approach to model data flow is possible in the underlying workflow model, it is only used to transfer data internally between activities. In the case of a collaboration these messages are sent between the participants and have usually a message type and some payload associated with it.

(f) the **transactional boundaries** of activities to facilitate recovery in the event of a participant failure. The model should allow to define transactional blocks that contain one or many activities that are followed when the effects of a service need to be reversed.

3. **Construct choreography interface from internal workflow:** this is the requirement that drives the framework: the internal workflow model of a particular business process should be reused when constructing a choreography interface, and this process should be automated. Automation requires that mediators are available to different choreography specification representations.
4. **Validate compatibility of choreography interface to internal workflow:** there are several cases where a pre-existing choreography interface has to be validated against an existing workflow model. For example, when partners use a standardised choreography, and extract the choreography interfaces of the participants from this agreement. But a participant might very well already have a workflow model implemented for his business functionality. It is then necessary to verify whether the existent workflow model is compatible to the choreography interface (behavioural equivalence).

4 Ontology for Choreography Interfaces

In what follows we describe the relations in m3pl capturing the requirements identified in the previous section. Our ontology is an extension to PSL [16] modelled in a first-order language. Due to space limitations we do not include the axioms constraining the relations described below. However, where possible we explain how a relation is constrained by the primitive lexical relations axiomatised in PSL-Core.

4.1 Introducing m3pl

To **model the internal workflow** we base our model on PSL [16]. PSL follows a layered approach in the language design, which gives us the resources to express information involving concepts that are not part of the PSL-Core. Thus we can represent arbitrary any workflow model in PSL by introducing extensions which are either defined by relations in the PSL-Core or by axioms that are constraining the interpretation of each new language construct.

The first requirement on the relations associated with the **choreography model** in m3pl is to encompass the two prominent modelling primitives. First we have to be able to extract to different task-flow based choreography description languages, i.e. to Abstract BPEL [20], WS-CDL [11] and ebXML CPP [10] and second to dependency-based ontology-based choreography descriptions, i.e. WSMO Choreography [18], OWL-S Process Model [12]. PSL provides relations to incorporate both workflow modelling primitives.

The m3pl extension offers a model to describe the choreography interface of some internal workflow model, whereas the choreography interface represents a model of some functionality (i.e. services). The functional entity in m3pl is a member of the set of such services in the universe of discourse of the interpretation. Services are reusable behaviours within the domain and relate to activities in PSL. A service occurrence models an occurrence of a PSL complex activity that is associated with the service.

```
service(?functional_entity)
service_activity(?functional_entity,?activity)
service_occurrence(?functional_entity,?occurence)
```

Listing 1.1. Service Relations

The views extension defines a relation to give one the possibility to restrict the **visibility** of specific activity occurrences to a certain role and thus create different views [4, 17] on a workflow model. The `visible(?occurrence,?role)` relation associates an activity occurrence to a role. By relating a participant to a specific role the visibility of activity occurrences is guaranteed to be constrained to the defined business partner only.

```
visible(?occurrence,?role)
```

Listing 1.2. Visibility Relation

Roles define the conversational relationship between two or more partners by defining the part played by each of them in the conversation. The `partner_link(?role,?functional_entity)` relation models such conversational relationships. The `participate(?agent,?role)` relation is used to relate an organisation to a role that it is playing in a specific collaboration.

```
partner_link(?role,?functional_entity)
participate(?agent,?role)
```

Listing 1.3. Conversational Role Relations

A key element in choreography description languages as identified above is the notion of **messages**. Since there exist different strategies of data passing in commercial workflow systems and workflow models, we offer relations which allow to model all three strategies as identified in [14].

Data is modelled with predicates and terms in first-order language. They act as fluents whose values may change as the result of service occurrences. Similar to FLOWS we use the `described_by` relation to associate a message_type to a fluent. Multiple fluents might be associated with one message_type, which should be interpreted as a conjunction of them.

Further we allow to associate the fluent to a channel. The `send` and `receive` relations are used to "transfer" fluents from one activity occurrence to the next. Both relations are associated with the `participates_in` relation of PSL, which is used to constrain which objects are involved in a particular occurrence of an activity. Thus in this data passing modelling approach no occurrence of an activity can begin without first receiving, and cannot send before it ends. The read relation is similar to receive, but with a weaker ontological commitment on the occurrence. It is not required that a send occurrence preceded the occurrence associated with the read relation.

```
described_by(?message_type, ?fluent)
send(?fluent, ?channel, ?occurrence)
receive(?fluent, ?channel, ?occurrence)
read(?fluent, ?occurrence)
input_port(?channel,?occurrence)
output_port(?channel,?occurrence)
```

Listing 1.4. Data Modelling Relations

Channels are used to model message-based communication as used in Web Services. We adopt the relations offered in FLOWS [2]. However, we do not relate channels to service occurrences, but to activity occurrences. Channels are a way to model explicit data passing, but are not required to exist since fluents can be related to activity occurrences via the read relation.

In order to capture **dependency based** workflow models every atomic activity occurrence can be associated with preconditions and effects. The occurrence of an atomic activity therefore transforms an initial state of the world (preconditions) into a final state that represents the world (effects) after the execution. Essentially the two relation are similar to the send and receive relation described earlier, since preconditions and effects are also represented by fluents in the ontology. The only difference being that they are not associated with a channel. However, they are a different concept in the real world, since preconditions and effects are not necessarily constraints on data, but might be constraints on the existence of objects.

```
precondition(?conditional_fluent,?occurrence)
effect(?conditional_fluent,?occurrence)
```

Listing 1.5. Dependency relations

All **task-flow based** choreography languages use control constraints to model the control flow of Web Services. However they are not natively included in PSL. Thus we reuse the definitions in FLOWS with minor extensions, which are together sufficient to model the majority of constructs available in choreography description languages.

```
sequence(activity)
split(activity)
IfThenElse(activity)
LoopUntil(activity)
wait(activity)
```

Listing 1.6. Control Constraint Relations

A `sequence` relation specifies that all subactivity occurrences of a complex activity are totally ordered. It corresponds to a `soo_precedes` relation in the Duration and Ordering Theory of PSL.

The subactivity occurrences of a split (corresponds to a flow construct in BPEL) activity are constrained by two relations from the PSL ontology. One subactivity occurrence `soo_precedes` any number of subactivity occurrences while they are `strong_parallel` to each other.

The `IfThenElse` activity is a nondeterministic activity such that the subactivity occurrences are constrained on the state conditions that hold prior to the activity occurrence. The use of `IfThenElse` is equivalent to a conditional activity in PSL.

The subactivity occurrences of a `LoopUntil` activity occur multiple times until the state condition is satisfied. It is equivalent to a conditional activity in the PSL ontology whose occurrences are repetitive, whereas the occurrence trees have different structure, depending on the cardinality.

The subactivity occurrences of a wait activity delays the process for a certain timeperiod or until a timepoint has passed.

Error handling in collaborative interactions is as important as transactional support in local application environments. The use of ACID transactions [7] is not feasible in collaborations, because locks on some activities cannot be maintained for periods of an asynchronous interaction. Error handling therefore relies heavily on the well-known concept of compensation. That is, if some state occurs alternate activities are performed which reverse the effects of a previous activity that was carried out and caused the error. To model such situations, we add failure handling activities, which are conditioned over an exception state raised by an earlier activity occurrence.

5 A methodology to extract choreographies

In the following section we show the applicability of the m3pl ontology extensions by outlining the methodology to follow when extracting choreography descriptions from internal workflow definitions. We apply the methodology to our example introduced in section 2, whereas we will focus on the supplier.

1. First the syntactic model (c.f. figure 2) underlying most Workflow Management Systems has to be lifted to the PSL/FLOWS ontology. In order to generate it automatically, mapping rules are required. This is not a trivial task since the generic mapping rules have to capture the operational semantics of the underlying WfMS.
   In our scenario the supplier models and enacts its business processes with IBM MQ Workflow. The workflow model is serialised in a proprietary description file called *.ftl.* We have identified the mapping rules necessary to translate our example workflow. However, it is not in the scope of this paper to define a generic mapping framework for arbitrary any workflow in IBM MQ Workflow. Listing 2.1. shows a snippet of the model from figure 2, i.e. the *Check Product Availability* activity followed by a decision point and either the *Prepare Referral* or the *Prepare Quote Response* activity. The full listing can be found at `http://m3pe.org/ontologies/PSLRFQ.kif`.

$\mathit{state}(\mathit{productListedOK})$
$\mathit{state}(\mathit{productListedFailed})$

$\forall(?\mathit{occRFQWorkflow})$
$\quad \mathit{occurence\_of}(?\mathit{occRFQWorkflow},\ \mathit{RFQWorkflow})$

$\quad\quad \Rightarrow \exists(?\mathit{occProcessRFQ},\ ?\mathit{occCheckProductAvailability})$
$\quad\quad\quad (\mathit{occurrence\_of}(?\mathit{occProcessRFQ},\ \mathit{ProcessRFQ}) \wedge$
$\quad\quad\quad \mathit{occurrence\_of}(?\mathit{occCheckProductAvailability},\ \mathit{CheckProductAvailability}) \wedge$
$\quad\quad\quad \mathit{subactivity\_occurrence}(?\mathit{occProcessRFQ},\ ?\mathit{occRFQWorkflow}) \wedge$
$\quad\quad\quad \mathit{subactivity\_occurrence}(?\mathit{occCheckProductAvailability},\ ?\mathit{occRFQWorkflow}) \wedge$
$\quad\quad\quad \mathit{root\_occ}(?\mathit{occProcessRFQ}) \wedge$
$\quad\quad\quad \mathit{soo\_precedes}(?\mathit{occProcessRFQ},\ ?\mathit{occCheckProductAvailability},\ ?\mathit{occRFQWorkflow})) \wedge$

$\quad\quad\quad (\mathit{holds}(\mathit{productListedFailed},\ ?\mathit{occCheckProductAvailability}) \wedge$
$\quad\quad\quad \mathit{not}(\mathit{productListedOK},\ ?\mathit{occCheckProductAvailability}))$
$\quad\quad\quad\quad \Rightarrow \exists(?\mathit{occPrepareReferral})$
$\quad\quad\quad\quad\quad (\mathit{occurrence\_of}(?\mathit{occPrepareReferral},\ \mathit{PrepareReferral}) \wedge$
$\quad\quad\quad\quad\quad \mathit{subactivity\_occurrence}(?\mathit{occPrepareReferral},\ ?\mathit{occRFQWorkflow}) \wedge$
$\quad\quad\quad\quad\quad \mathit{leaf\_occ}(?\mathit{occPrepareReferral},\ ?\mathit{occRFQWorkflow})) \wedge$

$\quad\quad\quad (\mathit{holds}(\mathit{productListedOK},\ ?\mathit{occCheckProductAvailability}) \wedge$
$\quad\quad\quad \mathit{not}(\mathit{productListedFailed},\ ?\mathit{occCheckProductAvailability}))$
$\quad\quad\quad\quad \Rightarrow \exists(?\mathit{occPrepareQuoteResponse})$
$\quad\quad\quad\quad\quad (\mathit{occurrence\_of}(?\mathit{occPrepareQuoteResponse},\ \mathit{PrepareQuoteResponse}) \wedge$
$\quad\quad\quad\quad\quad \mathit{subactivity\_occurrence}(?\mathit{occPrepareQuoteResponse},\ ?\mathit{occRFQWorkflow}) \wedge$
$\quad\quad\quad\quad\quad \mathit{leaf\_occ}(?\mathit{occPrepareQuoteResponse},\ ?\mathit{occRFQWorkflow})) \wedge$

Listing 2.1. Snippet of internal workflow in PSL/FLOWS

2. Next, the ontology instance representing a semantically equivalent model to the underlying workflow definition has to be annotated with choreography specific constructs from m3pl. Since domain experts knowledgeable of what parts of the workflow model are required to be published to partners and technology experts competent in defining the message exchange are not necessarily familiar with formal frameworks (i.e. First Order Logic), editor support is required to ease the annotation task. We are currently building a domain specific GUI-based tool for annotating the extracted model with concepts defined in our ontology.
   However, in the context of this paper we have manually annotated the generated ontology instance without tool support. The complete annotated model can be found at `http://m3pe.org/ontologies/RFQm3pl.kif`. This annotation is comprised of relations defined in section 4 capturing the collaborative role model, the visibility constraints, the message descriptions and its passing directions.
   Listing 2.2. shows the m3pl annotations added to the snippet of our internal workflow from Listing 2.1.

$service(RFQProcessing)$
$service\_activity(RFQProcessing,\ RFQWorkflow)$
$service\_occurrence(service,\ activity\_occurrence)$
$role(Customer,\ RFQProcessing)$
$role(Supplier,\ RFQProcessing)$
$participant(Bosch,\ Supplier)$
...
$\forall(?occRFQWorkflow)$
  $occurence\_of(?occRFQWorkflow,\ occRFQWorkflow)$

$$
\begin{array}{l}
\quad\Rightarrow \exists(?occProcessRFQ,\ ?occCheckProductAvailability)\\
\quad ...\\
\qquad\quad visibility(?occProcessRFQ,\ Customer) \wedge\\
\qquad\quad visibility(?occCheckProductAvailability,\ Supplier) \wedge\\
\qquad\quad input\_port(transmitRFQ,\ ?occProcessRFQ) \wedge\\
\qquad\quad ...\\
\qquad\qquad \Rightarrow \exists(?occPrepareReferral)\\
\qquad\qquad ...\\
\qquad\qquad\quad visibility(?occPrepareReferral,\ Customer) \wedge\\
\qquad\qquad\quad output\_port(transmitReferral,\ ?occPrepareReferral)) \wedge\\
\qquad\quad ...\\
\qquad\qquad \Rightarrow \exists(?occPrepareQuoteResponse)\\
\qquad\qquad ...\\
\qquad\qquad\quad visibility(?occPrepareQuoteResponse,\ Customer) \wedge\\
\qquad\qquad\quad output\_port(transmitQuoteResponse,\ ?occPrepareQuoteResponse) \wedge
\end{array}
$$

Listing 2.2. Annotation added to the extracted PSL/FLOWS model

3. Based on this ontological model choreography interfaces for each partner in the collaboration can be generated. Most importantly all activities marked in the previous step as private to the supplier will be omitted in the choreography interface. The split modelled in the choreography interface published to the customer is abstracted in a way that the evaluation of the condition is non-deterministic to the buyer and is modelled in the choreography interface as follows: $(holds(True,\ ?occCheckProductAvailability) \wedge not(False,\ ?occCheckProductAvailability))$
4. If required a multiparty choreography can be assembled. Since our model is based on a formal ontology this matching process can be on different levels of abstraction. The simplest matching algorithm compares the message type and the counterparting message passing direction. More complex matching can include full reasoning over the first-order models proving the equivalence of two choreography interface models.
5. Finally, choreography descriptions in existing languages such as WS-CDL, Abstract BPEL or ebXML CPP can be generated. Similar to step one, mapping rules have to be defined for each choreography description language. Different to above though the mapping is unidirectional, since the choreography descriptions represent an abstraction omitting information necessary in the internal workflow definition

An example extracted choreography interface from our model in a BPEL description is shown in listing 2.3. The interface starts with the definition of the partners, generated from the manually added annotations. The actual process starts at line 8 and contains the three workflow activities as *invoke* and *receive* operations. The *check-product-availability* activity, the split conditions and the internal data transfer are omitted from the choreography interface since they were marked as private information.

```
 1  <wsdl>
 2   <plnk:partnerLinkType name="buyerSellerRelation">
 3    <plnk:role name="seller"><plnk:portType name="rfqpw"/></plnk:role>
 4    <plnk:role name="buyer"><plnk:portType name="rfqpwCallback"/></plnk:role>
 5   </plnk:partnerLinkType>
 6  </wsdl>
 7
 8  <process name="RFQProcessing">
 9   <partnerLink name="buyerSellerRelation" partnerLinkType="lns:buyerSellerRelation"
10     myRole="seller" partnerRole="buyer"/></partnerLinks>
11   <variables>
12    <variable name="rfqMessage" messageType="lns:rfq"/>
13    <variable name="quoteMessage" messageType="lns:quote"/>
14    <variable name="referralMessage" messageType="lns:referral"/>
15   </variables>
16   <sequence name="main">
17    <receive name="processRFQ" partnerLink="buyerSellerRelation"
18       portType="lns:rfqpw" operation="initiate" variable="rfqMessage"/>
19     <assign><copy>
20      <from opaque="yes"/><to variable="condition" property="xsd:boolean"/>
21     </copy></assign>
22     <switch name="quoteDecision">
23      <case condition="if bpws:getVariableData('condition') = true">
24       <invoke name="prepareReferral" partnerLink="buyerSellerRelation"
25        portType="lns:rfqpwCallback" operation="onResult"
26        inputVariable="quoteMessage"/>
27      </case>
28      <otherwise>
29       <invoke name="processQuote" partnerLink="buyerSellerRelation"
30        portType="lns:rfqpwCallback" operation="onResult"
31        inputVariable="referralMessage"/>
32      </otherwise>
33     </switch>
34   </sequence>
35  </process>
```

Listing 2.3. Abstract BPEL description

6 Related Work

Our work is most closely related to several approaches to views on process models, i.e. [3, 4, 17, 15, 21].

Chebbi et al. [3] propose a view model based on Petri Nets. They introduce cooperative activities, which can be partially visible for different partners. The approach is validated on mappings from two different WfMSs. However, the model requires n2 mappings and does not explain how to model the data aspect, i.e. the message transfer between partners.

Chiu et al. [4] present a cross-organisational meta model which is implemented in XML. Similar to the cooperative activities in [3] the model provides so called cross-organisational communications, which allow to define message transfer and its direction. The model is implemented in an extension to the ADOME-WfMS, called E-ADOME. The model deals only with sequential activities in the abstracted view and does not tackle an integrated approach in choreography extraction and requires the specific model to be used in the E-ADOME tool.

Schulz and Orlowska [17] introduce a Petri-Net based state transition approach that binds states of private workflow tasks to their adjacent workflow view-task, where existing workflows are augmented by means of one or more activities or sub-workflows of an external workflow. The model is conceptualised in a supporting architecture. The approach identifies mappings in its conceptual architecture, but it does not describe how to integrate different workflow models. Further the approach abstracts from the data aspect.

Sayal et al. [15] introduce service activities (that represent trade partner interaction) as workflow primitives, but their approach is specific to one workflow modelling tool and addresses neither workflow integration nor choreography interface extraction.

Zhao et al. [21] define a relative workflow model representing the view of one partner on local workflows of another partner. They present composition rules how to generate the relative workflow and a simple matching algorithm to connect two local workflow process. Similar to the other approaches it is meta model independent.

Several approaches address interoperability issues between Workflow Management Systems (WfMS), such as Mobile [9], Meteor [19] and CrossFlow [8]. However, all of these approaches require a pre-established partner agreement on the semantics of the process models. Further they were all proposed before the advent of Service Oriented Architectures and therefore do no deal with standard choreography description languages.

7 Conclusion

In existing approaches, choreography descriptions are independent of the internal workflows of the partners and have to be manually mapped.

We presented m3pl, an ontology extension to PSL and FLOWS to formally capture choreography-specific information. The ontology extension together with PSL can act as a connecting ontology to integrate different workflow models and subsequently extract external process models.

We have shown how the framework can be used to extract a choreography interface of an example workflow in a RosettaNet collaboration. This initial validation is based on the translation of an example workflow represented in IBM Websphere MQ Workflow to PSL, which is then manually annotated with relations offered in the m3pl extension to further extract a BPEL process.

One direction of our future work is to check the equivalence of to choreography models. Given a choreography interfaces it is desirable to verify whether it is compatible with the choreography interface of a partner and -if they are indeed compatible- to construct a multiparty choreography.

Acknowledgment

This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/04/BR/CS0694.

References

1. D. Austin, A. Barbir, E. Peters, and S. Ross-Talbot. Web Services Choreography Requirements. Working draft, W3C, Mar. 2004.
2. S. Battle, *et al.* Semantic Web Services Ontology (SWSO). Member submission, W3C, Sep. 2005.
3. I. Chebbi, S. Dustdar, and S. Tata. The view-based approach to dynamic inter-organizational workflow cooperation. *Data Knowl. Eng.*, 56(2):139–173, 2006.
4. D. K. W. Chiu, *et al.* Workflow view driven cross-organizational interoperability in a web service environment. *Inf. Tech. and Management*, 5(3-4):221–250, 2004.
5. R. Dijkman and M. Dumas. Service-oriented design: A multi-viewpoint approach. *Int. Journal of Cooperative Information Systems*, 13(4):337–368, Dec. 2004.
6. D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.
7. J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann, San Francisco, 1993.
8. P. Grefen, K. Aberer, H. Ludwig, and Y. Hoffner. CrossFlow: Cross-organizational workflow management for service outsourcing in dynamic virtual enterprises. *IEEE Data Engineering Bulletin*, 24(1):52–57, 2001.
9. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture and Implementation.* Int. Thomson Computer Press, 1996.
10. N. Kartha *et al.* Collaboration-protocol profile and agreement specification, v2.1, Apr. 2005.
11. N. Kavantzas *et al.* Web services choreography description language, Nov. 2005.
12. D. Martin, *et al.* Owl-s: Semantic markup for web services. Member submission, W3C, 2004. Available from: `http://www.w3.org/Submission/OWL-S/`.
13. M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10):25–28, 2003.
14. N. Russell, A. H. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow data patterns. FIT-TR-2004-01, Queensland University of Technology, 2004.
15. M. Sayal, F. Casati, U. Dayal, and S. Ming-Chien. Integrating workflow management systems with business-to-business interaction standards. In *Proc. of the 18th Int. Conf. on Data Engineering*, pp. 287–296. 2002.
16. C. Schlenoff, *et al.* Process specification language (PSL): Overview and version 1.0 specification. Tech. Rep. NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD, 2000.
17. K. A. Schulz and M. E. Orlowska. Facilitating cross-organisational workflows with a workflow view approach. *Data Knowl. Eng.*, 51(1):109–147, 2004.
18. J. Scicluna, A. Polleres, and D. Roman. Ontology-based choreography and orchestration of wsmo services. Wsmo working draft v0.2, DERI, 2005. Available from: `http://www.wsmo.org/TR/d14/v0.2/`.
19. A. Sheth, *et al.* The METEOR workflow management system and its use in prototyping significant healthcare applications. In *Proc. of the Toward an Electronic Patient Record Conf. (TEPR'97)*, pp. 267–278. Nashville, TN, USA, 1997.
20. S. Thatte *et al.* Business process execution language for web services, v1.1, May 2003.
21. X. Zhao, C. Liu, and Y. Yang. An organisational perspective on collaborative business processes. In *Proc. of 3rd Intl. Conf. on Business Process Management*, pp. 17–31. Sep. 2005.

Process Ontologies Facilitating Interoperability in eGovernment A Methodological Framework

Timo Herborn and Maria A. Wimmer

Institute for IS Research, Research Group eGovernment
Universitaetsstr. 1, 56070 Koblenz, Germany
{therborn|wimmer}@uni-koblenz.de

Abstract. eGovernment is characterized by the usage of multiple applications and heterogeneous data environments. Although the outcomes of administrative processes are defined quite good, the processes themselves are different depending on geographical, political or systemic factors. A vivid example for such a diversity are the business registers (BR) of the member states of the European Union (EU). They all have in common that they are gathering and providing information about companies and related data. But not only their data processed is totally different - also the underlying processes are. The project BRITE (Business Register Interoperability Throughout Europe) aims to build interoperability between the BRs in order to facilitate EU-wide transactional services for companies to e.g. register their branch in another country. These goals will be achieved by the application of ontology-driven semantics. This paper introduces the approach to develop a common BRITE domain ontology which links up national domain ontologies and BR processes. In this way, interoperability is reached among various national business registers and their respective services. The BRITE platform thereby serves as an intermediary to link up the diverging ontologies of national business registers.

Keywords. Interoperability, eGovernment Domain Ontology, High Level Domain Ontology (HLDO), Process Ontology, Business Registers, BRITE

1 Motivation

Governments are more and more using ICT to facilitate their tasks and responsibilities as well as to collaborate among public organizations more efficiently. In order to guarantee the free movement of European citizens and companies, the European Community perceives the imperative necessity to establish an appropriate and interoperable ICT basis [3]. To pave the way for interoperability by large, the EC co-funds a series of research and technology developments under IST[1], MODINIS [1] or IDABC [4].

[1] Examples of such projects from the 6th Framework Program IST of the EC are: GUIDE, INTELCITIES, ONTOGOV, QUALEG, Semantic-

In its visionary documents, the European Community (EC) attempts to become the world leading knowledge society [2] and [5]. Not only does this mean to enlarge the Community with new member states. Likewise importantly envisaged is a liberal market with companies being able to establish their branches in other EC Member State countries as quickly and simply as possible. A major aspect within this topic, which is, by the way one of the most important rights written down in the Roman Treaty, is the interoperability of registration systems. Therefore, the EU undertakes extensive homogeneisation efforts on political, economical and organizational levels. These efforts result in multiple bills and directives[2] which are demanding cooperating efforts of the Member State's governments.
This paper gives a short introduction to the scope of the EC co-funded project BRITE. Then, a methodological framework for developing a domain ontology for European Business Registers is presented. Thereby, different aspects of ontologies which are considered as key success factor in guaranteeing interoperation among Business Registers throughout Europe are discussed.

2 Context of Business Registers (BR) in Europe

In Europe, in order to run a business or company, the organization has to be registered at a business register office in the state, where the company or its seat is officially established. Legal forms of companies strongly depend on particular legislation of the country. The number of registration offices varies from country to country (depending on the state's constitutional distribution of responsibilities) as well as on the organization of the registration offices themselves. In some countries, the responsibilities for the company registration process are delegated to federal state or even district administration level[3]. With the vision of the EC becoming a world leading knowledge society with liberal market opportunities and simple registration procedures, the business registration processes need to be kept very simple, effective and without barriers. Consequently, the BRs need to smoothly interact. However in Europe, Business Register interoperability is currently not achieved. Especially in countries with BRs located at regional level, massive problems of interoperability may occur when interacting with BRs of other countries. In some cases, even within a country, interoperability among regional BRs is not reached yet. The barriers Business Registers are facing may start with language barriers, passing on to data and system incompatibility and ending in inconsistent registration processes as well as distinct strategic policies. Providing interoperability among BRs for the respective registration processes of a business are the topic of investigation in the Integrated Project BRITE,

Gov, SMARTGOV, TERREGOV, R4eGov, Access-eGov, etc. (see *http : //europa.eu.int/comm/research/fp6/index_en.htm*)

[2] These directives are to be seen as framework and are translated into national law by the member states

[3] In Germany, business registers are organized on "Länder" (federal state) level, not on national level

a project co-funded by the EC within the 6th framework program of IST[4]. Based on the legal groundings such as the 11th Directive of the EC, BRITE aims to build an agile and dynamic governmental environment for the emerging European market place. The approach pursued in BRITE will heavily build on semantic technology, especially ontologies, to facilitate interoperability and process-oriented information exchange [9]. The next section therefore introduces the methodological framework to develop a BR ontology. By developing this methodology we emphasizes the importance of not being too general and focusing on the migration tasks [8].

3 Methodological Frame for BR Ontology Development

In the context of guaranteeing interoperability among systems, services and organizations, ontology development has become an important issue. We understand ontologies as a key enabling concept for the Semantic Web. Ontologies 'interweave human understanding of symbols with their machine processability [and] promise a shared and common understanding of a domain that can be communicated between people and application systems' [6]. The general term *ontology* is further detailed in various literature. Guarino for example defines domain ontology as [7]:

- constituted by a specific vocabulary used to describe a certain reality
- a set of explicit assumptions regarding the intended meaning of the vocabulary.

According to that, a domain ontology is extended by some specifications:

- ontologies describe a formal specification of a specific domain
- share understanding of a domain of interest
- represent a formal and machine executable model of a domain of interest.

This article focuses on the use and development of ontologies in respect to the intended interoperability of eGovernment applications and public administration systems especially in the context of European Business Registers (EBR). In this context, already implemented domain ontologies based on different geographical, organizational and historical roots have to be faced and harmonized resp. linked up with each other. Instead of "reinventing the wheel", the re-usability of existing ontologies has to be checked and aggregation of existing data, document and process schemata should be aspired on an overarching level. Furthermore, an ontology has to fulfill criteria such as openness, dynamics and flexibility in order to allow for future changes and integration of laws to come. The aim of BRITE is to combine Domain Ontologies and Process Ontologies in a way to achieve maximum productivity. See fig. (**??** This combination is necessary in order to a) harmonize the vocabulary toward a common upper level conceptual standard, b) to get an understanding of the individual, national processes and

[4] see $http://europa.eu.int/comm/research/fp6/index\_en.htm$

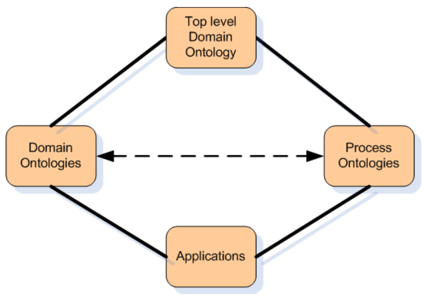


Fig. 1. Combination of Domain Ontologies and Process Ontologies

c) to integrate the corresponding processes correctly. In order to achieve the goals mentioned above, we introduce a methodological framework to secure real interoperability between different institutions, even with language barriers and massive process diversities. The approach has the advantage of not having to change legacy systems, but to link them up via standardized overall, domain and process ontologies. It consists of the following steps:

- Defining a high level domain ontology (HLDO)
  - Identifying Domain ontologies of interest
  - Selecting subset of vocabulary of Domain Ontology
  - Merging subsets
  - Mapping merged subsets to HLDO
  - Adding semantics to the HLDO
  - Provide interoperability layer through the usage of the HLDO
- Identifying national BR process ontologies (BRPO)
- Developing high level BR process ontologies
- Integrating national processes

4 Defining the High Level Domain Ontology (HLDO)

This paper is based on the idea that a general eGovernment ontology is considered as top level Domain Ontology (figure (reffig:hierarchy)). Technically, the HLDO is an aggregation of at least two Domain Ontologies based on similar semantic backgrounds. This definition bears on the idea of generating semantic interoperability between different Domains without destroying historical data architectures. Legacy systems and data structures can work without any restrictions or losses, because data structures are not changed but enlarged. Additionally, we see it as basis for the development of a standardized process ontology within BR environment. The methodology how to define an HLDO is described in the following figure (3. All steps mentioned in this model are to be done iteratively and are to be repeated until all participating project members feel comfortable with the solution.

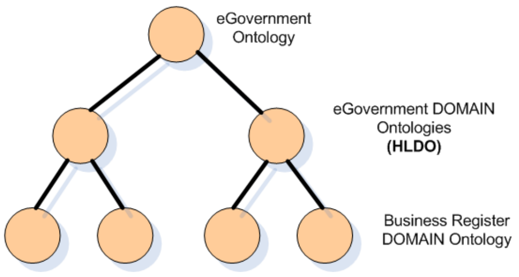

eGovernment Ontology
eGovernment DOMAIN Ontologies (HLDO)
Business Register DOMAIN Ontology

Fig. 2. Hierarchy of Ontologies within eGovernment

4.1 Identifying Domain Ontologies of interest

Domain Ontologies are - in most cases - not explicitly described in the Domains of interest. Mostly they hidden within legacy systems, databases and in people's minds. The last place mentioned is the most interesting but also most difficult to exploit. The methodologies to identify these various ontologies are as different as the ontologies themselves. Concerning the legacy systems and databases, field and data descriptions can be extracted and then imported into an ontology modeling tool. Getting ontologies out of people's head is much more difficult. Brainstorming, Soft System Methodologies, and Mind Maps are examples of how to retrieve knowledge residing in human brains.

4.2 Selecting subset of vocabulary of Domain of Interest

This task is easy to be done. Basically, all BR Domains are of interest for the development of the HLDO. Starting with these project partners that already have explicitly defined their very own ontologies seems to be a reasonable approach.

4.3 Merging subsets of national vocabulary

The idea of merging is to finally build an ontology combining all relevant subsets of vocabularies known in the Domain of BR within the project. Merging subsets includes deleting redundant entries. Historical sources are to be tracked by the help of semantics in order to re-assign them to their source of origin.

4.4 Mapping merged subsets to HLDO

After having defined the HLDO by merging different subsets, a mapping is to be done in order to validate the new version of the HLDO against the formerly existing Domain Ontologies. Basically, vocabularies and semantics from different domains are to be harmonized. The mapping process carrying out the development of the HLDO is described in figure 5.A simple example shall demonstrate this: "$< Firmenname >$" (domain ontology A) as well as "$< name\_of\_company >$" (domain ontology B) are mapped to "$< BRITE\_company\_name >$". In this

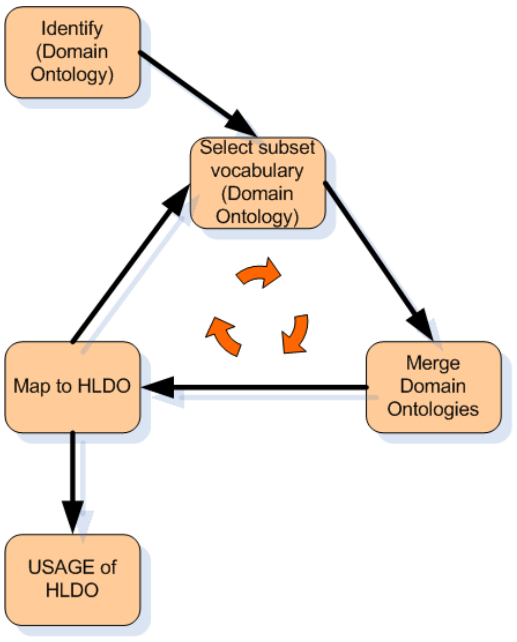

Identify (Domain Ontology)
Select subset vocabulary (Domain Ontology)
Merge Domain Ontologies
Map to HLDO
USAGE of HLDO

Fig. 3. Methodology to define a HLDO

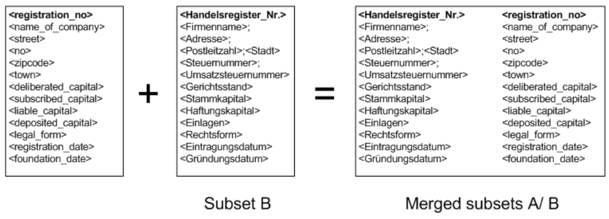


Fig. 4. Merging subsets of national ontologies

way, the new concept just needs to store the original identifier (in order to reverse the process) and its origin domain (for allocation reasons).

On the left side of the figure, a subset of a domain ontology is presented. One can easily recognize some major differences: a) linguistic (English and German), b) structural (database management) and c) organizational (mainly content based). According to the definition of the HLDO provided above, an ontology is more than just translating or mapping data and information. Making this information valuable for the processes to be executed, semantics need to be added. First of all, information of the mapping process needs to be stored.

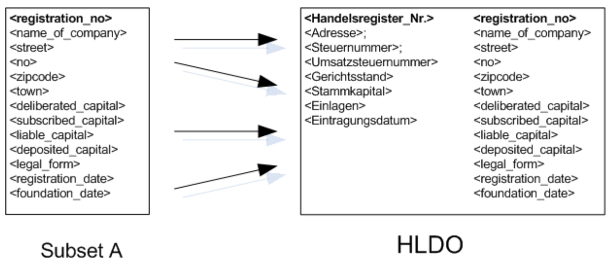


Fig. 5. Mapping national domain ontologies to HLDO

4.5 Adding Semantics to the HLDO

Next step is to add semantics to the HLDO. The reasons for adding semantics are several. First of all, these semantics will support in building inferences between independent data sets. Apart from that, advanced search mechanisms can be applied to get qualified retrieval results. Finally, semantics allow to introduce automatic machine processing of applications. A concept of Wimmer [10] is used to introduce the application of ontologies within eGovernment contexts. The concept presented in figure (6) consists of classes (topic) and their concrete instances (individuum). Adapting this concept to the vocabulary introduced

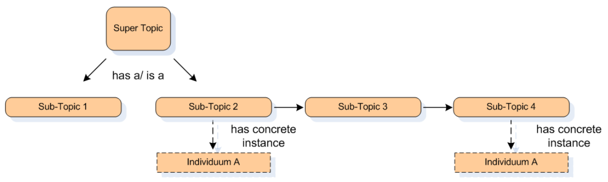


Fig. 6. Concept for the abstract ontology

before, a basic knowledge map can be build (7), in which simple inferences are possible. As one can easily see in the figure, the class "Supertopic" is -according

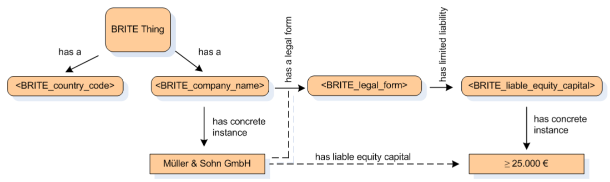


Fig. 7. Adding semantics to facilitate automatic machine processing and inferences

to the syntax of OWL- translated into a "BRITE Thing". "Subtopics" are named

"Company Name","Registration Number", etc. Important for the idea of an ontology concept is that we can now build inferences like: Instance A ("Müller und Sohn GmbH") *must* have Instance B ("a liable equity capital of at least 25.000 EUR"). This example does not only show the inference mechanism but also demonstrates the possibilities of using incremental logical conclusions (*here:* Country code AND Liable equity capital).

4.6 Provide interoperability layer through the usage of the HLDO

After having developed a HLDO, the domain ontologies need to be revisited because in some cases, adaptations may be needed to exploit the full potential of the common overall ontology. There are two ways to approach the interoperation of domain specific ontologies: In some cases, even legacy systems might have to be adapted or reengineered based on the new vocabulary in order to smoothly interoperate. However, this approach would take enormous amounts of personpower and time, apart from any technical and organizational constraints probably occurring. On top of that, actualizations of the HLDO could not be done synchronously. The second solution proposed is to make the common ontology available in a central storage and make it functioning as intermediary medium. This approach would allow the individuality of each domain ontology to be maintained. Next step is to integrate processes within a distributed eGovernmental environment.

5 Identifying national BR process ontologies (BRPO)

Identifying corresponding process ontologies is a key success factor for BRITE. In order to retrieve comprehensive information and generate applicable knowledge out of it, processes need to be well described and modeled. A business process is described as "[..] *representation of a business process in a form that supports automated computation, such as modeling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc.*" [5] This definition contains all relevant elements of a business process modeling approach. On another descriptive level, processes have a desired *output* generated by an *input* and a *throughput.*[6] This classical idea of processes makes clear distinctions between the process stages. Please refer to figure (**??**). Exploring and describing national BR processes is to be done along the cases provided in the description of work of the BRITE project. Important in the context of this paper are:

[5] WfMC Glossary - WfMCTC- 1011

[6] E.g. transfer of a registered office to another country within the EU. *Output*: establishment of registered office in another country. *Input*: all relevant data from source country. *Throughput*: all relevant processes and information enabling the transfer.

– Transfer of registered office
– Opening a branch in another EU Member State

5.1 Developing high level BR process ontologies

We define process ontology as a conceptual description framework for business processes. This definition is built on the following basic assumptions:

– Processes ontologies are abstract and generalized
– Processes have interfaces to any repository (databases, other processes etc.)
– Processes have formerly defined inputs [*throughputs*] and outputs
– All processes described are based on the HLDO

Process ontologies have clearly defined borders between the process stages (*input, throughput* and *output*). Dependent on national conditions, interfaces to adequate sources (databases, other process in different domains) are to be defined, as figure (8) illustrates. In this model, an input (e.g. application for a

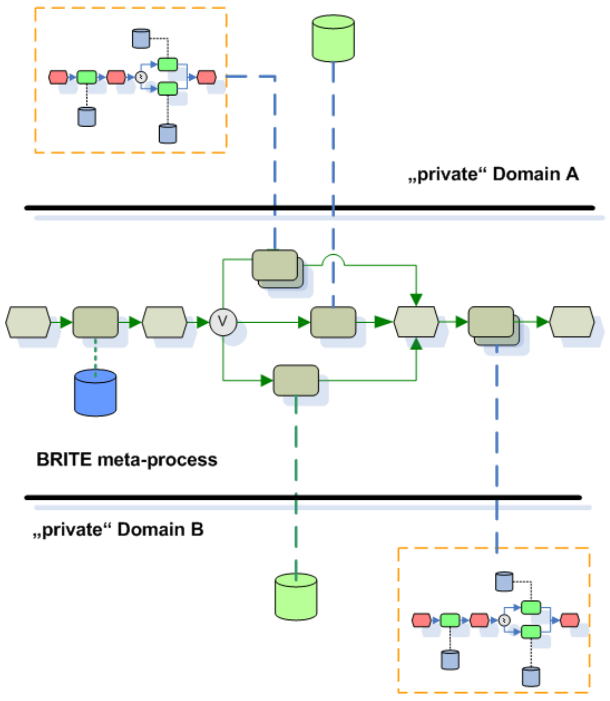


Fig. 8. Domains as boundaries

transfer of seat) and an output (e.g. transfer of seat) is given. All intermediary stages (interfaces to legacy systems, data storage, archiving) are defined in the process ontology but executed on legacy systems. According to that idea, a limitation is also shown in the figure, one of the major threat to interoperability, the domain boundary. In the figure we refer to *"private" Domain A and B*, which are representatives for two separate Business Registers. The information flow is shown, but the integration level is very low, because of the clearly defined domain borders. In order to built real interoperability, these boundaries are to be broken - as far as possible. Therefore, the abstract process must be prepared for different kinds of input, due to particular situations and their context.

5.2 Integrating the processes

In the last step of our approach we are now integrating the process ontology into the heterogeneous landscape of European Business Registers. In order to protect the idea of an holistic integration and interoperability approach we are considering all integration possibilities. Depending on language, country, use case or any other influence, the abstract process can become concrete. By doing so, the process remains dynamic and adaptable. The HLDO is an indispensable prerequisite for this procedure. Figure 9 illustrates such an "opened-up process".

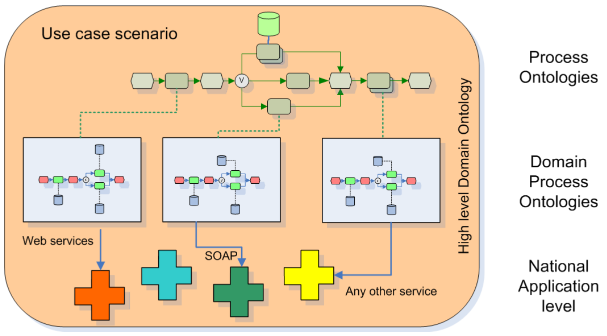

Use case scenario
Process Ontologies
Domain Process Ontologies
National Application level
High level Domain Ontology
Web services
SOAP
Any other service

Fig. 9. Different interoperability layers of a Process Ontology

Figure 9 show three different layers: Process ontologies, domain process ontologies and national applications. They are linked together by the usage of different protocol and services, depending on the legacy systems and applications. All processes in this figure are based on the HLDO which is stored in the knowledge repository of BRITE. The figure shows a use case scenario which could be every use case, e.g. "transfer of seat". The domain process ontologies refer to the national application level using their specific protocols and service in order to retrieve, process and store data. Using this architecture, BRITE is enabled to use every source of all BR involved in the project - directly or indirectly.

- Databases and processes of BRITE
- Databases and processes of national Business Registers
- Third party repositories

DB and processes of BRITE The BRITE database contains the HLDO and all abstract processes. Only selected "live" data is stored centrally (as it is already done in the EBR) [7] in order not to get in conflict with the national regulations. The abstract processes are just the framework to enable the desired functionality described in the use cases.

Databases and processes of national BR These databases provide the essential information for the BRITE processes to work. They provide static information about companies as well as detailed information about processes (e.g. for registering, increase of capital stock etc.) which have important contributions to the BRITE process. Major constraints of proper integration are identification (authorisation/ authentication) of users, security constraints and legal restrictions. Detailed investigations in that field are done in literature.

Third party repositories Third parties can be assurances, finance providers, SCHUFA [8], IHK [9] and others. The (partial) integration of their databases is to be discussed. But the general architecture of the process ontology should be open enough for a latter integration of these.

These interfaces may vary from process to process, depending on all parameters possibly changing: country, legal form, use case scenarios etc. The usage of semantics (which are developed within in the HLDO) allows such a diversification. As a desired consequence, local variations are not affecting the functionality of the system. The depth of integration also depends on multiple parameters.

6 Summary

In this contribution, we have introduced an approach to develop a common ontology for European Business Registers that links up national Business Register ontologies. The development of a High Level Domain Ontology for European Business Registers allows to conciliate national Business Registers and, via the common high level concept, allows to fully interoperate among national or even local applications. The work is being carried out within the EC co-funded project BRITE, an integrated project within FP 6 of IST.
In order to design interoperability between different European Business Registers, the common high level ontology was used as a mapping tool. It was further used as a means to narrow down process ontologies and, based on that, to develop an intermediary that links up distinct national applications.
In our approach we firstly define a HLDO which enables a communication between all project partners and BR involved in the project. The development

[7] For an example browse to: http://www.ebr.org
Navigate to: products/view example
[8] Schutzgemeinschaft für allgemeine Kreditsicherung [german] Community to protect general capital interests
[9] Industrie und Handelskammer [german] Chamber of Industry and Commerce

is described as an iterative and self-repeating methodology consisting of selecting subsets of domain ontologies, merging them in order to delete redundancies and finally mapping the outcome against the domain ontologies. This approach maintains functionality of legacy systems and keeps them operating.

In the next step we are supporting the identification and description of processes related to the use cases. These domain process ontologies are abstracted to a general process ontology. By the help of this high level process ontology, legacy systems can be addressed over the borders of domains. The outcome of our approach is interoperability between different legacy systems on technical, organizational and semantical levels.

The first step we undertook was to resolve the differences within the vocabulary in order to get a common understanding of Business Register's topics. Then, semantics were added to the vocabulary in order to build a knowledge map and enable inferences. Finally, processes were modeled on a high level and were opened to other inputs (such as foreign or own data storages and processes) and created consequently the basis for interoperability. Final step will be to integrate these processes into the legacy systems, varying in depth and depending on legal, organizational and technical constraints.

References

1. Modinis Programme. Study on Interoperability at Local and Regional Level. 2006. http://www.egov-iop.ifib.de/index.html.
2. eeurope 2005. an information society for all. eGovernment, *http : //europa.eu.int/ information_society/eeurope/news$_l$ibrary/ documents/eeurope*2005/*eeurope*2005*_en.pdf*, 2002.
3. Commission of the European Communities. Linking-up Europe: The Importance of Interoperability for e-Government Services. Staff Working Document, 2003.
4. Decision of the European Parliament and of the Council on Interoperable Delivery of pan European Services to Public Administrations, Businesses and Citizens (IDABC). Decision 2004/387/, *http : //europa.eu.int/idabc/*, 2005.
5. i2010 Eine europäische Informationsgesellschaft für Wachstum und Beschäftigung: Mitteilung der Europäischen Kommission und Nachfolgeprogramm von eEurope 2005. *http : //europa.eu.int/information_society/ eeurope/i*2010/*i*2010/*index_en.htm*, 2005.
6. J. Davies, D. Fensel, and F. Van Harmelen. In *Towards the Semantic Web Ontology-driven Knowledge Management. Sussex: John Wiley and Sons Ltd.*, 2003.
7. N. Guarino. Formal ontology and information systems. In *In Nicola Guarino, editor, Formal Ontology in Information Systems - Proceedings of FOIS'98, Trento, Italy, pages 3-15. IOS Press, Amsterdam*, 1998.
8. R. Klischewski. Migrating small governments' websites to the semantic web. In *Sematic Web meets eGovernment, Papers from the AAAI Spring Symposium, Technical report SS-06-06*, 2006.
9. L. e. a. Van Elst. Business register interoperability throughout europe: The brite project. In *Semantic Web Meets eGovernment - Papers from the AAAI Spring Symposium*, 2006.

10. M. Wimmer. Implementing a knowledge portal for egovernment based on semantic modelling: The e-government intelligent portal (eip.at). In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, 2006.

Semantic Reference- and Business Process Modeling enables an Automatic Synthesis

Florian Lautenbacher, Bernhard Bauer

Programming of Distributed Systems
Institute of Computer Science, University of Augsburg, Germany
[lautenbacher|bauer]@informatik.uni-augsburg.de

Abstract. The optimization of business processes is a necessary prerequisite to reduce transactional costs in and between enterprises. Though the modeling of processes is supported by a variety of graphical notations and tools, changes to sub-processes often require the adaptation of the whole process. Thus, methods are necessary to support automated actualization of process models omitting this time-consuming manual task. As a result, business models need to be extended with information describing the semantics of the processes. Machine-understandable information based on standards of the Semantic Web can be applied to automate this task. Describing each process with semantic information enables an automatic synthesis of processes, calculating the optimal combination of them. This paper shows a first approach how to annotate process models with semantic data for a synthesis, describes synthesis algorithms and evaluates a prototypical implementation.

1 Introduction

Over the past few years, enterprises have been undergoing a thorough transformation in reaction to challenges such as globalization, unstable demand, and mass customization. A key to maintain competitiveness is the ability of an enterprise to describe, standardize, and adapt the way it reacts to certain types of business events, and how it interacts with suppliers, partners, competitors, and customers. In the context of process orientation, enterprises today describe these procedures and interactions in terms of business processes and invest huge efforts to describe and standardize them. Business processes are either notated only on a textual basis or graphically with models. During the last decades several graphical model standards emerged like event-driven process chains (EPC) or the Unified Modeling Language (UML2) [11] which is more established in computer science. In particular, the UML 2.0 standard with its extended activity diagrams supports an elegant modeling of business processes.

Nowadays, the creation of business process models can be done with several tools. All of them provide the creation and modification of models, some of them support the users with wizards, but none of these tools currently provides an assistance how the modeled processes should be combined for the optimal flow of the process.

This is, in particular, important for reference processes. Reference process models are the basis for many companies to develop their own business processes. Currently lots of reference processes are available (for a detailed list see e.g. [21]), but each one uses a different language (EPC, UML, OMT, IDEF0, etc.). Additionally, it is very difficult to find a reference process which is applicable for the scope of the business area used in a company. Therefore, these reference processes should be annotated with semantic information to increase their usability and re-use.

An annotation also supports the adaptation of a reference process. Currently, several reference processes are available and companies are using these reference processes and adapting them, but very often after a few years with the changing demands of customers and IT, these reference processes need to be actualized. Annotating the reference processes and the business processes with semantic information, this actualization can be automated. To describe the (reference and business) processes, one can use the annotation that has been developed for several semantic web service standards. These are part of the platform specific model in the model-driven architecture (MDA, [22]) as can be seen in Figure 1. Both, the reference process model (RPM) and the business process model (BPM) would be layered in the Computation Independent Model-layer (CIM). Using similar annotation for RPM and BPM on the one hand and Semantic Web Services in a platform specific view-layer on the other hand will enhance the transformation from CIM to PSM later.

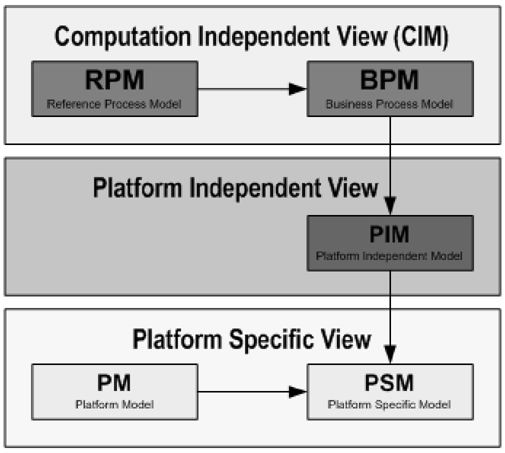


Figure 1: Reference process models in the MDA

The upcoming standards of the semantic web provide a set of concepts that can be used to annotate processes in a way machines can analyze. These concepts are summarized in an ontology which can be built using the resource description framework (RDF), RDF Schema (RDFS), the web ontology language OWL [1] or other standards. Based on these standards web services can be augmented to enable an automatic publication, discovery, interoperability and access. These so-called semantic web service standards (e.g. OWL-S [2] or WSMO [3]) contain parameters for describing the functional behavior of a web service including input and output parameter, preconditions and effects, description of the quality of a service and others. Most of these concepts can be applied to business processes as well.

This paper describes an approach to annotate business and reference processes with machine-understandable information based on standards of the semantic web and semantic web services. Using these metadata a PC is able to compute a (semi-) automatic composition of all processes (called synthesis). Therefore, synthesis algorithms were developed and are surveyed to calculate an optimal composition of all processes (concerning the given parameters).

The paper is organized as follows: Section 2 provides an overview on current approaches of semantic web service standards. The concepts of these standards are the basis to annotate BPM and RPM as described in section 3. In the same section algorithms for the synthesis of business processes are presented and explained. In section 4 a case study based on a prototype implementation is shown. Section 5 gives an overview about related research. Finally, section 6 summarizes the opportunities of our approach and outlines further research activities.

2 Background

In order to obtain a (semi-)automatic synthesis of business processes, processes have to be annotated with semantic information. This requires a language that computers can decode and analyze, i.e. it has to be machine-understandable data. Here the advantages of semantic web standards come into play. Semantic web constructs can be used to describe the data structure as well as the dynamic behavior of an enterprise.

Ontologies provide a basis for describing e.g. context information in a way that human and machines can read and understand. The semantic web stack (cf. [14]) shows the standards that can be used to describe several concepts. Based on RDF and RDFS most ontologies use the standard OWL. But a description of processes needs additional information e.g. what the process does, when it might be started, which inputs it needs, etc. Therefore, several semantic web service standards which describe web services in a similar way can be used to fill this gap: OWL-S, WSMO, SWSF and METEOR-S. In the following these standards will be explained and compared.

OWL-S

The semantic markup for web services (OWL-S, antecessor was DAML-S [15] which is now standardized by W3C) is based on the Web Ontology Language OWL and supplies web service providers with a core set of markup language constructs for describing the properties and capabilities of their web services in an unambiguous, computer-interpretable form. It describes a service in three different ways: ServiceProfile ("What does the service provide?"), ServiceModel ("How is the service used?") and ServiceGrounding ("How does one interact with it?"). In the ServiceProfile the type of the service is specified with the ServiceCategory. A ServiceCategory describes categories of services on the bases of some classification that may be outside OWL-S and possibly outside OWL.

WSMO
Based on the Web Service Modeling Framework (WSMF) [12], the Web Service Modeling Ontology (WSMO) is a formal ontology and language that consists of four different main elements for describing semantic web services:
- Ontologies that provide the terminology used by other elements
- Goals that state the intentions that should be solved by web services
- Web Services which describe various aspects of a service
- Mediators to resolve interoperability problems.

Each of these wsmoTopLevelElements can be described with non-functional properties like creator, creation date, format, language, owner, rights, source, type, etc. WSMO comes along with a modeling language (WSML) [16], a reference implementation (WSMX) and a toolkit (WSMT).

SWSF
Another W3C submission beside OWL-S and WSMO is the Semantic Web Services Framework (SWSF) [4] which represents an attempt to extend the work of OWL-S and consists of two major parts: the Semantic Web Service Language *SWSL* and the ontology *SWSO* above. Both were developed on basis of two different logics: the first-order logic within *FLOWS* (First-order Logic Ontology for Web Services) and based on logic-programming the Rules Ontology for Web Services (*ROWS*).

METEOR-S
Another proposal for a semantic web service standard is the METEOR-S project [5] of the LSDIS-lab, University of Georgia, collaborating with IBM. METEOR-S is the abbreviation of “Managing End-To-End OpeRations for Semantic Web Services” and focuses the process itself. The main point of METEOR-S is the use of semantics in the complete lifecycle of semantic web processes to represent complex interactions between semantic web services.

Comparison
OWL-S provides a distinction between atomic processes, simple and composite processes. This separation is necessary to enable more complex service interactions. The composition of services is abutted on BPEL [25] which is a commonly used language in the web service community and can therefore be easily adapted. The differentiation between ServiceProfile, ServiceModel and ServiceGrounding makes it easier to describe a service completely. The categorization of a service can be used by a semantic-enhanced service registry to find a corresponding service.

The mediator concept of WSMO is very important to solve interoperability issues. Defining goals will be necessary when an automatic orchestration of services is needed. WSMO separates the provider and requester point of view by defining goals and web service capabilities separately (different to OWL-S where no goals can be modeled). It also offers the possibility to add non-functional properties (which lacks in OWL-S) and is not only restricted to web service composition (modeled by transition rules), but also envisions modeling of orchestration.

SWSF extends the OWL-S standard, enables rule languages and extends the description logic used in OWL-S to a first-order logic in FLOWS which makes it easier to describe concepts and their relationships. It already offers the usage of rule languages (similar to WSMO) which is still missing in OWL-S (an extension of the underlying OWL to SWRL in OWL-S could be used to solve this).

METEOR-S separates different ontologies: data semantics for the definition of the vocabulary, functional semantics to describe the capabilities of web services, execution semantics to represent the flow of services and QoS semantics which explains the quality a service offers. This separation makes it easier to create and modify these ontologies.

We will come back to the advantages and concepts of the different standards presenting our own approach.

3 Overall Approach

This section defines our overall approach, how reference and business process models using UML2.0 syntax can be annotated using the concepts of semantic web services. In particular, we explain how an automated synthesis of processes can be achieved using matrices and algorithms working on the annotation.

3.1 Semantic Modeling of Reference and Business Processes

The modeling of business processes concentrates on the functional view whereas web services are focused on the technical view, i.e. different levels of abstraction. Nevertheless, both services and processes can be described with functional (and non-functional) parameters. The semantic web service standards provide descriptions what a service does and how it interacts with others. These descriptions can be applied to business processes and used to describe their choreography.

Each semantic web service standard has advantages when being used to annotate business processes. We are interested in a general approach for the automated synthesis of reference processes, i.e. pre-defined business processes to be customized and combined to obtain value-added business processes. In order to get a long-time realization of the system we decided to combine the concepts and advantages of the standards presented in section 2 in our approach, hence staying independent in the realization being able to switch after it is foreseeable which one is accepted most.

OWL-S provides the distinction between atomic processes, simple and composite processes. Business processes as modeled e.g. in an UML2 activity diagram have different levels of detail. A general manager is only interested in the high level process description whereas a project manager needs a more detailed view. Composing several atomic processes to a composite one enables this requirement. Thus we will keep the distinguished views provided by OWL-S. OWL-S also has a possibility to categorize a service in a taxonomy. This will be important for reference processes, too. **WSMO** enables the definition of goals and several tools are already using the web service descriptions and goal description to create a plan for achieving this goal (e.g. IRS-III [19]). These tools and algorithms might be useful for an improvement of our work, but are currently limited to semantic web services in WSMO. It is the only standard that offers the modeling of non-functional properties which is important for reference and business processes as well. Similar to OWL-S **SWSF** describes each action with four different parameters: input, output, preconditions and effects (in short: IOPEs). This description and abbreviation will be used to describe business processes which will be modeled through actions in UML2.0 activity diagrams. **METEOR-S** separates different ontologies, namely data semantics, functional semantics, execution semantics and QoS semantics. Our approach will use the *data semantics* for the description of the underlying ontology, the *functional semantics* for the description of each process and the *execution semantics* to specify how the processes are related to each other[1].

We adopt and merge the advantages of all standards and build the following architecture which shows how the semantics are interrelated (see Figure 2).

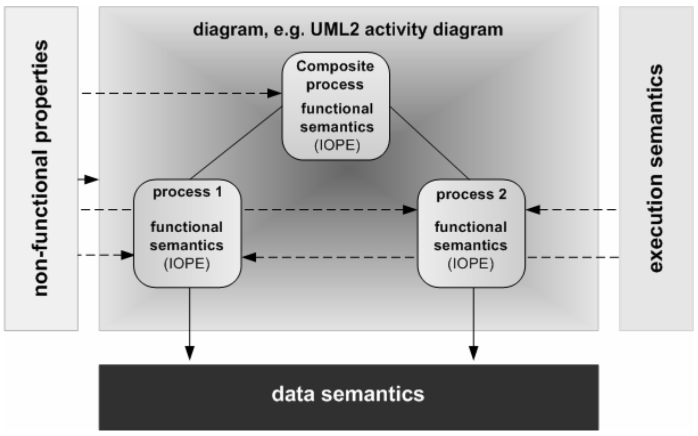


Figure 2: Useful semantics for the synthesis

Our business process model is defined as follows: BPM = (D, F, E, N), whereas D is the data semantics, F are all processes (containing the functional semantics) and relations, E describes the execution semantics and N the non-functional properties. A formal description of a diagram would be $F = \cup_{0 \le i \le n} P_i \cup \cup_{0 \le i, j \le n} Con_{Pi, Pj}$ where n is the number of processes, P_i is a process and $Con_{Pi,Pj}$ is a connection between the two processes P_i and P_j. A connection $Con_{Pi,Pj}$ describes that after process P_i has been executed, process P_j follows.

The ontology used to define all concepts of the company and corresponding business processes will be called **data semantics** and provides a basis for the modeling of processes. It defines all concepts and their relationships that describe the enterprise, the departments and their tasks which are required to annotate BPM and RPM. Addi-

[1] These data, functional and execution semantics can be modeled using each of the presented semantic web service languages.

tionally, (global) variables can be defined to be used in preconditions and effects for describing the change to a global state, e.g. changes in a database, etc.

Each process is described in the **functional semantics** using IOPEs. Therefore, the concepts defined in the data semantics are used for the input and output section. The variables can be tested for specific values in the preconditions and new values can be assigned in the effects. Each process P_i includes the functional (and non-functional) semantics for this process, $P_i = (I_i, O_i, P_i, E_i, N_i)$. I_i and O_i describe the inputs and outputs of this process and contain concepts that are defined in the data semantics ($I_i \subseteq D$, $O_i \subseteq D$). P_i and E_i are well-formed formulas (of an arbitrary logic) describing preconditions and effects and use variables defined in the data semantics for Boolean expressions and assignments. N_i is the summary of all non-functional properties of this sub-process.

To improve the automatic synthesis it is possible to define the necessary chronology of some processes. Following the METEOR-S approach this is captured in the **execution semantics** E. Using the following statements the user can define his/her preferences for the composition of the processes. Possible statements for two processes P_1 and P_2 are:

- P_1 next P_2 ($P_1 \circ P_2$, means P_2 follows directly on P_1),
- P_1 eventually P_2 ($P_1 \lozenge P_2$, sometime after P_1, P_2 will be executed),
- P_1 previous P_2 ($P_2 \circ P_1$),
- P_1 before P_2 ($P_2 \lozenge P_1$),
- P_1 parallel P_2 ($P_1 \| P_2$, means P_1 and P_2 are in two parallel threads).

The **non-functional properties** N can be modeled using a categorization of the service similar to OWL-S and the values in WSMO. The whole process should at least be annotated with the version of the model, when it was created, the name and the subject it is about, the type and format it uses (e.g. UML2) and which business areas it covers. Every sub-process should be annotated with the topic, a short description of the steps in this sub-process, how much it costs and how long it will normally take or after which period of time it must have been finished.

3.2 Synthesis of Semantic Process Models

Having defined all pre-defined processes, i.e. modeled the data semantics, functional semantics and execution semantics, a synthesis task can be started. The non-functional properties are currently not used for the synthesis. These will be used in a future version to optimize the synthesis from an economic point of view.

The synthesis works incremental: first, the functional semantics of each process is compared with the functional semantics of all other processes in a reasoner. The results of these queries are converted to numbers and stored in a synthesis and identity matrix. These matrices are then interpreted as a directed and weighted graph. Based on these graphs the synthesis algorithms can then compute bottom-up what the optimal composition of all processes would look like. We have developed two different synthesis algorithms (Modified Prim and RandomWalk) which will be described in section 3.2.2. The following section describes how the synthesis matrix and identity matrix are built. A more detailed description about the synthesis can be found in [23].

3.2.1 Synthesis Matrix and Identity Matrix

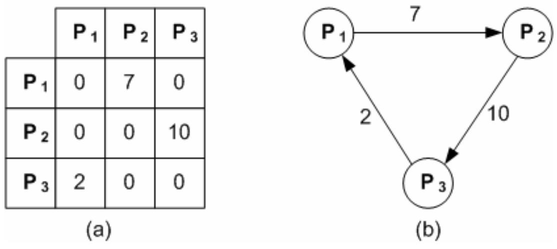
The synthesis and identity matrix are responsible to store the values that are computed in the inference engine. The inference engine has loaded the data semantics and gets queries whether the parameters of two processes fit together. It tests the congruence of the outputs of the first process with the inputs of the second whether they are equal or the inputs are a subset of the outputs. Furthermore they are checking whether the preconditions of the second process are satisfied with the effects of the first. The more these tests are successful, the higher is the value in the synthesis matrix.

The **synthesis matrix *synmat*** is a `n×n` matrix (whereas `n` is the number of processes in the business or reference model) with

$$synmat_{Pi,Pj} = \begin{cases} 0, \text{ if } (P_i \equiv P_j) \text{ or noMatch}(P_i,P_j), \\ 2, \text{ if Input}(P_j) \subset \text{Output}(P_i), \\ 4, \text{ if Input}(P_j) = \text{Output}(P_i), \\ 5, \text{ if Preconditions}(P_j) \subset \text{Effects}(P_i), \\ 6, \text{ if Preconditions}(P_j) = \text{Effects}(P_i), \\ 7, \text{ if } (\text{Input}(P_j) \subset \text{Output}(P_i)) \wedge (\text{Preconditions}(P_j) \subset \text{Effects}(P_i)), \\ 8, \text{ if } (\text{Input}(P_j) \subset \text{Output}(P_i)) \wedge (\text{Preconditions}(P_j) = \text{Effects}(P_i)), \\ 9, \text{ if } (\text{Input}(P_j) = \text{Output}(P_i)) \wedge (\text{Preconditions}(P_j) \subset \text{Effects}(P_i)) \\ 10, \text{ if } (\text{Input}(P_j) = \text{Output}(P_i)) \wedge (\text{Preconditions}(P_j) = \text{Effects}(P_i)). \end{cases}$$

We have chosen these numbers for simplification of the implementation: every entry in the matrix bigger than six can be created by adding the smaller numbers and is unambiguously factorable into these smaller numbers.

In the example in Figure 3a the outputs and effects of process P_1 are sufficient (but not equal) for the inputs and preconditions of process P_2 (a value of 7). The outputs and effects of process P_2 are equal to the inputs and preconditions of process P_3 (value: 10). Additionally the outputs of P_3 are similar to the inputs of P_1 (value of 2).



	P_1	P_2	P_3
P_1	0	7	0
P_2	0	0	10
P_3	2	0	0

Figure 3: A synthesis matrix and the corresponding graph

Similar to the synthesis matrix the **identity matrix *idenmat*** contains the results of the comparison of two processes, but this time the equivalence of them is tested. If parts or all inputs are the same (alternatively outputs, preconditions or effects) there is a value bigger 0 (maximum 16), otherwise with a value of 0 the two processes are not parallelizable or the process was compared with itself.

3.2.2 Modified Prim and RandomWalk

Having created both matrices these can be interpreted as a directed and weighted graph G = (N, E). The nodes N are the processes and there are edges if the entries in the synthesis matrix between two processes are bigger than 0 (cf. Figure 3b). Based on these graphs the synthesis algorithm can be started.

Modified Prim is an adapted Prim- (or Dijkstra-) algorithm and creates a graph choosing one edge randomly first. Then the edge with the highest value will be added to the solution if it fulfills given constraints. This is repeated until all processes have been visited. After visiting each process, the operation is terminated and the solution is evaluated. This algorithm is executed with each existing edge as start edge and after all solutions are computed, the one with the best rating is returned to the user.

The synthesis algorithm **RandomWalk** operates on basis of an existing solution (e.g. generated via Modified Prim). RandomWalk was primarily invented to optimize mathematical problems and tries to converge to a local (or better: global) maximum by changing the solution iterative. Every solution is rated (using the value of the applied edges) and computed whether there exist solutions with a better rating. Therefore, in our case, edges are removed by accident and others (with a better value) are added. After a period of time the operation is stopped and the so far best solution is returned which represents the best computed workflow of the annotated processes. Of course, it is also possible to specify a goal (this would be the effects or outputs of the whole activity) and the algorithm will reject all solutions which don't give the required outputs or effects.

4 Case Study

To test the semantic process modeling and to compare the different synthesis operations, a prototypical tool was developed [6]. It offers the modeling of a semantically-enriched UML2 activity diagram and testing the synthesis with the operations explained above.[2] The big advantage of the synthesis comes to the fore using it on extensive reference process models. However, due to space limitations we only demonstrate the synthesis of a small business process.

Let us assume a purchase process where a customer buys a product which has to be adapted to his needs. First, the warehouse has to be checked whether the necessary raw product is available or perhaps a comparable product can be found. If this is not the case, then the raw product is ordered. With the raw product the end product can be manufactured, a bill can be written and the product can be delivered to the customer. Without semantic enhancements, the system would not be able to decide whether the process "Order product" or "Write bill" should come first. Therefore every process can be annotated with inputs, outputs, preconditions and effects.

A part of the semantics for this example would be the following: Figure 4 shows a part of the data semantics for one of these processes ("Order product"). The functional semantics for the same process would include (informal):

Precondition: ProductInStock hasValue FALSE,
Output: Order,
Effect: ProductOrdered setValue TRUE.

[2] The case study is based on OWL and OWL-S to present the ontologies, but every concept can be adapted to other semantic web service languages, too.

```
<semBPM:Object rdf:ID="Order"/>
<semBPM:SemDataProperty rdf:ID="executedFor">
        <rdfs:domain rdf:resource="#Order"/>
        <rdfs:range rdf:resource="#Customer"/>
</semBPM:SemDataProperty>
<semBPM:globalVariable rdf:ID="ProductInStock"/>
```

Figure 4: Data semantics for the process "Order product"

Having modeled all processes (without transitions!), annotated them with semantics and started the synthesis, one gets the result of Figure 5. Both algorithms achieve the same result and the solution the user might have probably expected. If one of these processes changes or needs to be deleted, one can simply start a new synthesis to get the new optimal combination and no further action is required.

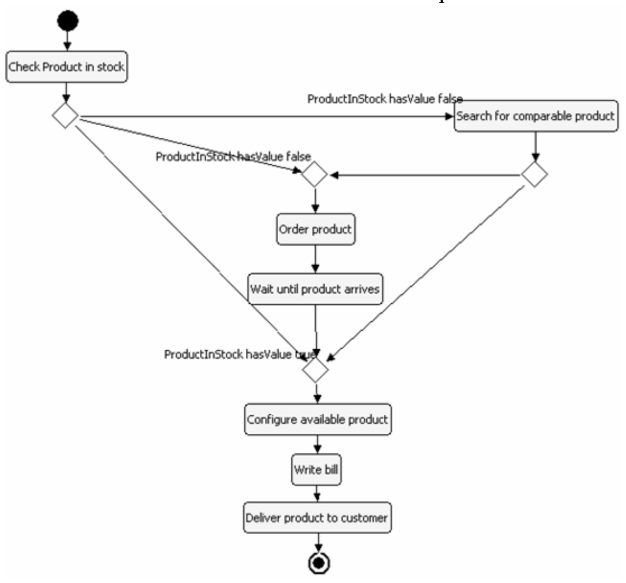

Check Product in stock
ProductInStock hasValue false
Search for comparable product
ProductInStock hasValue false
Order product
Wait until product arrives
ProductInStock hasValue true
Configure available product
Write bill
Deliver product to customer

Figure 5: Result of the synthesis of our business trip example

Testing more complex cases the algorithms sometimes find different solutions which are syntactically and semantically correct, but not always the optimal and expected results concerning the given parameters. The creation of the synthesis and identity matrix has a time and space complexity of $O(n^2)$. We are currently working on optimizing Modified Prim, which has a time complexity of $O(n^3)$, whereas RandomWalk only needs $O(n \cdot \log n)$.

5 Related Work

This is – to our knowledge – the first approach to annotate business processes and use these annotations to make an automatic synthesis of the modeled processes. Other institutions are describing business processes as well: [24] describes an approach to use ontologies for business process modeling in Petri nets. Jenz & Partner is developing a tool named BPEdit [9] which enables users to graphically define and edit business process definitions using the Business Process Modeling Notation (BPMN) [8]. BPEdit relies on an open and extensible ontology-based information model which is defined in OWL and offers the generation of deployable process definitions in process definition languages such as BPEL. The METEOR-S development team is developing a Process Designer [10] to generate BPEL Web Processes based on the METEOR-S Web Service Annotation Framework (MWSAF) [13] and which supports dynamic discovery of partner services using semantics. Both tools enable the annotation of business processes and generate process definitions in BPEL. They use semantic web languages like OWL and develop concepts to annotate business processes with semantic information. But at present they are not using these annotations to make an automatic synthesis of the described processes.

There are several groups working on composition of web services: they either require static information (e.g. Golog-based [17] or based on Hierarchical Task Networks [18]). Some approaches use ontologies [19], others agent technologies [20]. But none of them currently considers business processes and the automatic composition of UML2 activity diagrams.

6 Conclusions and Outlook

The presented approach is a first step to use annotated business process models to make an automatic synthesis and compute the best composition of the modeled processes. This enables a modeling tool to assist humans in modeling and optimizing a model when changes to the corresponding actions have occurred. Our approach enables the usage of ontologies to describe business processes and to use this information to compute an optimal workflow of business processes which is necessary for an automatic adaptation of reference processes and will increase their utilisability and usage.

Our approach combines the advantages of the discussed semantic web service standards, but stays adaptable to each of these standards. We implemented a prototypical application which was developed as an eclipse plug-in and provides therefore an opportunity to be connected with other plug-ins.

Our prototype only offers the modeling of UML2 activity diagrams; other diagrams are currently not supported. Especially diagrams to cover organizational or resource aspects should be included and additional semantics and ontologies created (e.g. for the organizational structure of the company). We will create a UML2 profile for the usage of ontologies in business process models and try to combine the higher-level business process descriptions with lower-level web services. The synthesis will be extended not to consider optional elements and to use actions (e.g. for consistency checking) several times.

We are aiming to develop a methodology to annotate reference processes, whereas we will consider current approaches to model business processes like BPMN and approaches of business process ontologies.

References

1. Guiness, D. and Harmelen, F.: “OWL Web Ontology Language - Overview”, February 2004, W3C Recommendation.
2. Martin, D. et al: “OWL-S: Semantic Markup for Web Services”, November 2004, W3C Member Submission.
3. Lausen, H., Polleres, A. and Roman, D. (Eds.): “Web Service Modeling Ontology (WSMO)”, June 2005, W3C Member Submission
4. Battle, S. et al.: “Semantic Web Services Framework (SWSF) Overview”, September 2005, W3C Member Submission.
5. LSDIS, University of Georgia, “METEOR-S: Semantic Web Services and Processes”, Homepage: http://lsdis.cs.uga.edu/projects/meteor-s/
6. University of Augsburg, Plug-In available at http://pvs.informatik.uni-augsburg.de/eclipse
7. Axenath, B., Kindler, E. and Rubin, V. “the Aspects of Business Processes: An Open and Formalism Independent Ontology”, Paderborn, April 2005.
8. BPMI.org, “Business Process Modeling Notation (BPMN) – Version 1.0”, May 2004.
9. Jenz & Partner GmbH: “BPEdit – What is it?” available online at http://wa0529.dw10.de/MamboV4.5.2/content/view/15/28/
10. METEOR-S Download and Release Page: “METEOR-S Process Designer” available online at http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=4
11. Object Management Group (OMG): “Unified Modeling Language (UML) Specification”.
12. Fensel, D., Bussler, C.: “Web Service Modeling Framework WSMF”, ECommerce, 2002.
13. LSDIS, University of Georgia, “MWSAF: METEOR-S Web Service Annotation Framework”, Homepage: http://lsdis.cs.uga.edu/projects/meteor-s/mwsaf/
14. Berners-Lee, T. “Putting the Web back in Semantic Web”, In: ISWC, Galway, 2005.
15. The DAML Services Coalition: “DAML-S: Semantic Markup for Web Services”, 2003.
16. deBrujin, J., Lausen, H.: “Web Service Modeling Language”, W3C Submission, 2005.
17. McIlraith, S. and Son, T.: “Adapting Golog for composition of semantic web services”. In: KR-2002, Toulouse, France, 2002
18. Au, T.C., Kuter, U. and Nau, D.: “Web Service Composition with Volatile Information”, In: ISWC05, Galway, November 2005.
19. Domingue, J., Galizia, S. and Carbal, L.: “Choreography in IRS-III – Coping with Hetereogeneous Interaction Patterns in Web Services”, In: ISWC05, Galway, 2005.
20. Greenwood, D., Buhler, P. and Reitbauer, A.: “Web Service Discovery and Composition using the Web Service Integration Gateway”, in: EEE’05.
21. Fettke, P. et al.: “Business Process Reference Models: Survey and Classification”, BPRM05, Nancy, September 2005
22. Miller J. and Mukerji J. (Eds.): “MDA Guide Version 1.0.1”, June 2003, omg/2003-06-01.
23. Lautenbacher, F., Bauer B. “Automatic Synthesis of Reference Processes Using Semantic Concepts”, Poster at I-ESA06, Bordeaux, March 2006, available online at www.ds-lab.org/publications/proceedings/2006_I-ESA2.html
24. Koschmider, A. and Oberweis, A.: “Ontology based Business Process Description”, in: EMOI-INTEROP’05, Porto, June 2005.
25. OASIS: Business Process Execution Language For Web Services, Version 1.1, 5 May 2003.

Towards Business Level Verification of Cross-Organizational Business Processes

Kioumars Namiri[1], Nenad Stojanovic[2],

[1] SAP Research Center CEC Karlsruhe, SAP AG, Vincenz-Prießnitz-Str.1
76131 Karlsruhe, Germany
Kioumars.Namiri@sap.com

[2]FZI Karlsruhe, Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany
nstojano@fzi.de

Abstract. In this paper we present a novel approach for the verification of a business process configuration. The approach is formal, so that logic mechanisms are used in the verification process. The approach has been applied in the scope of the project ATHENA for the verification of cross organizational business processes.

Keywords: Formal verification, Business process modeling, Business configuration, Cross-organizational business processes

1 Introduction

Today most enterprises do not implement their software applications from scratch on their own. They rather decide to buy pre-built IT-Solutions from software vendors, where the software applications are built on top of it. This is especially the case in the area of ERP Software. The software has to be adapted in such a way that the implemented business processes there meet the needs and requirements of the customer enterprises. Usually, same business processes differ from company to company as a result of different and changing business environments. Changing business environment is caused by frequently changing internal local business practices and capabilities of an enterprise, its partner ecosystem and the local legal regulations. That means that customers have to configure the functions in the purchased solutions accordingly. Business configuration is part of every implementation project for customers who have bought an ERP product. Soffer et al. describe in [1] configuration as an alignment process of adapting the enterprise system to the needs of an enterprise. Typically in praxis, technology and application consultants take care of an initial business configuration, and the customer business department is responsible for maintenance.

At the same time, most enterprises focus on specific parts of their business process and depend on partners in a market to perform the additional parts of the process required to achieve a complete end-to-end business process. A common business paradigm is that of service outsourcing, in which an enterprise focuses on its core business process and has secondary process parts enacted on its behalf by service provider organizations.

In the EU-Project ATHENA [2], these kinds of business processes are called Cross-organizational Business Processes (CBPs) [3], i.e. processes that cross two or more enterprises. The ATHENA project deals with the problem of interoperability between enterprise information systems. Solutions to problems associated with CBPs are the main goal of the project. Support for the semi-automatic modeling and automatic execution of these processes are the focus of study in the different research groups, which investigate the problem at different levels: business, technical and execution.

We state that in real world situations not fully configured ready to run CBPs are delivered by ERP vendors to customer enterprises, since the vendors aim to cover the requirements of their different customers. These requirements are first known when introducing the CBPs at a customer company. By business configuration of CBPs we particularly mean in this paper that the customer company is forced to integrate the set of available external business functionalities provided by partners and internal systems in not configured CBPs in such a way, that it fulfills the enterprises situation. Nowadays, both the interface to provide business functionalities provided by partners and the interface to internal systems of enterprises are exposed mostly as services, even as web services. The task of business level adaptation of CBPs according to the available set of existing web services is a costly and time consuming task in most non trivial cases, since it is dependent on frequently changing business environment of a customer. A rule based approach is required to express the existing business environment of an enterprise in terms of business rules to influent the instantiation and execution of a business process.

One challenging need in this context is to provide a mechanism to ensure that CBPs are configured and upgraded consistently. The vision is a top-down deployment of business level requirements on an enterprise application spanning over different processes and a bottom-up verification of already configured processes making sure they steadily fulfill the business level requirements.

In this paper we address the problem of verification of a configured CBP whether it meets the requirements of an enterprise. On requirements level we focus on the mapping between a process step and the available web services. We will see by an example that there are different mapping variants dependent on each enterprise demands how web services are mapped in the same CBP. We will express these dependencies as declarative rules specific for each enterprise representing its business requirements in this scope. We propose to use semantic technologies based on SWRL [4] for representing the configuration constraints on the CBPs. We will use the reasoning technologies provided in Description Logics [5] to inference the configuration of a CBP according a knowledge-base in terms of SWRL-rules, thus we will be able to verify the current CBP Configuration of each enterprise.

This paper is structured as follows: First we take a look at the related work in this area. Following we introduce the concepts developed in ATHENA in the area of business processes collaboration. In the next chapter we introduce a motivating use case for illustrating the need for verification of CBPs. Based on the introduced scenario we show a solution approach how to semantically verify the configuration of CBPs. Finally we present the planed research and conclude.

2 Related Work

Business Process Execution Language for Web Services (WSBPEL) [6] aims to provide an XML based language for describing business processes and how web services are composed together. ATHENA provides the tool Maestro, which is used for orchestrating business processes. They can be enacted by Nehmiah, a Process Execution Engine. We argue that both approaches, using Maestro or a BPEL-based description of a process does not solve the problem of different enterprise-dependent process configurations since they can be characterized as static in that sense that once a process is implemented, it runs always in the same way and is not aware of changing business environments.

As stated clearly in [5], “Configuration” can be considered as one successful domain for knowledge-based applications built using Description Logics, which includes application that support the design of complex systems created by combining multiple components. While there is industry-wide accepted related work in the area of DL-based product configuration and verification [7] [8], there is less industry-wide accepted research work done in the area of configuration and verification of process-based software applications.

In [9] an approach is introduced to express configurable EPCs (CEPCs). The semantic of business level requirements on business process, which is matter of frequent change, is here explicitly modeled in CEPCs and not separately outside of the business processes, as required by our approach.

There are several works done in the area of integrating business rules and service compositions [10] [11]. While the business rules there address more the runtime behavior of processes, we concentrate more on the design time of processes and offer a verification mechanism for the processes whether they fulfill a set of predefined SWRL-rules. In [12] a formal framework is provided to represent business requirements and goals of an organization and how they can be operationalized in terms of BPEL-Processes. The verification mechanism in this framework is based on model checking.

Finally in [13] with TOVE-Ontology an extension of First-Order-Logic (Calculus Logic) is provided to verify if business processes in a producing enterprise fulfill ISO9000 Quality Norms. In this ontology concepts are provided to describe business processes of an enterprise, but the ontology does not take collaborative aspect of business processes and web service integration into account.

3 Business Process Concepts and Tools in ATHENA

3.1 CBPs

A concept is developed in ATHENA to classify process types pursuing different goals. Processes are divided in three levels of abstraction: a level suited for business analysts, an intermediate level suited for process analysts, and a level suited for IT-experts. At this last level the processes may be executed by computer systems. Furthermore, ATHENA presents a concept to model cross-organizational processes without having to reveal internal, private information of enterprises. This concept includes three different process types that vary in their degree of providing information about a single enterprise and in their degree of providing information about the whole collaborative process:

- Cross-Organizational Business Process: This process type is intended to explain the whole collaborative process and contains mainly abstract information about the roles the involved enterprises play
- Private Process: This process type is used only internally by an enterprise and contains all information regarded as necessary by internal users
- View Process: This process type hides sensitive information contained in the private process of an enterprise and provides partners with information on how to interact with the enterprise owning this private process.

Based on these concepts, modeling tools were enhanced to support collaborative business processes on each level; the approach of having three different levels of abstraction was implemented, too. Thus it is possible to model processes at the business analyst level and to transform them to the technically detailed BPDM-models used in Maestro. Apart from this, based on formal operators a method was developed to enable horizontal transformation. Thus automatic transformation from view process to private processes and vice versa is supported by the Maestro tool.

To complete the model framework, a modeling procedure was established that identified three possible procedures for the creation of views and CBPs. In a *bottom-up* approach each company starts with the identification of their private processes and the creation of interaction-specific views which then are combined into CBPs. In a *top-down* approach, the partners start identifying a common picture of the interaction in terms of a CBP model. Each partner then creates its views according to the process steps that will be executed. As a last step the partners have to define their private processes. The third scenario ("*middle-out*") is that of one partner starting with its private processes and offering a view process to its partners. The partners can link this process-based interface to their internal processes via view processes. This corresponds to a bottom-up approach for one partner and a top-down for the others.

3.2 Business Processes vs. Services

In ATHENA a Tool called Gabriel is provided to bridge the gap between the business process world in terms of CBPs and the SOA world in terms of web services provided by business partners. Gabriel is responsible for defining and enacting the concept of business process task particularly as web services. Maestro allows a user to model business processes as a set of tasks and dependencies between those tasks. Nehemiah allows the execution of business processes modelled in Maestro; such processes are exported from Maestro to Nehemiah via a business process repository, which therefore provides a link between modelling time and runtime. Because both those tools were originally aimed at simulating business process execution, Maestro and Nehemiah do not focus on modelling and enacting the business action that should be performed for a given task. With the aim of cleanly separating modelling and enactment, Gabriel has two distinct parts. The modelling side allows defining so-called task profiles, which are a set of attributes that describe what action has to be performed for a given task. Such an action can be user interaction or service invocation. Task profiles as well as organisational data are stored in a repository that connects modelling time and runtime. When the process enactment engine (Nehemiah) notifies Gabriel that a task is available, Gabriel looks up the task profile that was associated with the corresponding task model in Maestro, and based on the type of profile, puts the task at the top of some appropriate principal's task list, or causes a web service invocation related to that task.

4 Scenario: Integration of Carrier Web Services in the Shipping Process

Many enterprises with a need for shipping functionality use various online-services and tools provided by companies in the transportation industry to facilitate the shipping process for their customers. For instance United Parcel Service of America (UPS) provides several on-line XML Tools [14] or FedEx offers various APIs for integrating their business functionality into a client shipping application [15]. These Tools can be integrated into the Order-To-Cash-Process of a shipper. Integration software vendors have been addressing this issue by providing Shipping applications, but they are mostly carrier dependent and force shippers to implement a specific business process pattern supported by the solution. The shipping process at a shipper company, as well as the problems in its configurations is described in following subsections.

4.1 Description of collaborative Shipping Process

An Order-to-Cash process at a shipper company is generally constituted by following process steps:

A **Sales Order** contains information on ordered Goods, shipping address etc. and is the starting point for the process. The Sales Order can be changed as long as there is no subsequent Delivery processing started. Until then all fields are changeable. The **Delivery** is created from a Sales Order. Depending on customer specific criteria (e.g. delivery dates / shipping mode) several Deliveries can be created from one Sales Order. Merging of several Sales Orders into one Delivery is possible. From the Delivery the **Picking and Packing** Information is created (Pick List / Handling Units). During this Phase quantities are confirmed. The information is written back to the Delivery. Depending on the physical availability of goods the Delivery can change several times (e.g. only a partial shipment is possible, thus correction of delivery is needed). From a Delivery a **Shipment** can be entered. Several Deliveries can be aggregated to one Shipment. The Shipment is the foundation for the manifest / shipping instructions. After the goods are shipped, a commercial **invoice** for the customer is created. The shipper will be invoiced by the selected carrier and is therefore out of the scope of the shipping process. During **After Sales**, tasks related to eventual Return Management are done.

The Shipping Process is handled by several services. In this paper we will concentrate on core shipping services provided by a carrier as web services:

- **Generate Routing Code:** The routing code is the carrier specific representation of the route over which a parcel is shipped.
- **Calculate Rate:** Rate calculation takes as input the selected shipment type, the route and some more information on the parcel and calculates the rate for the shipment.
- **Generate Label:** This service creates the label for the parcel in the carrier specific format.
- **Manifest:** In this service the current shipment is added to the manifest for the day and sent to the carrier.

4.2 Shipping Process Configuration Variants

Considering the abstract shipping process previously described, the process is instantiated and executed in such a way that in each process step different internal systems/services or external carrier services are called.

In case of a shipping process realized through a purchased solution at a shipper company, the business configuration of the solution according to the requirements of that enterprise has direct impact on the way, at which process step or activity which service(s) is called. Different variants occur as web services can be called from different process activities of the standard order-to-cash process executed at the shipper. Furthermore, there are particular call dependencies between some of the services. For instance, a routing code is required to calculate the rate and the label can only be printed after the rate has been determined.

To exemplify this situation, three different business situations for three different enterprises are described, which lead to different configurations of the core-shipping services calculate rate, generate routing code and generate label web services of carriers into the same standard order-to-cash-process of each shipper. This is a result of a real world analysis of different shipping companies regarding their interoperability requirements.

- **Process Variant 1:** In this case, the shipping condition selection, routing code calculation and rate calculation are all done during sales order. This is possible if the total weight of the parcels is already known during sales order entry. After the goods have been picked and packed a label is printed and finally the manifest is generated. As the manifest is not generated after each individual shipment this results in adding a new entry to the manifest for the daily manifest.

- **Process Variant 2:** In this scenario the shipper only selects the carrier and the desired shipping condition during sales order entry. Routing code, rate calculation and label generation are performed after the goods have been packed. This is either the case if the shipper does not need rate estimations during sales order entry or if the shipping process starting with picking and packing is run by a different system.

- **Process Variant 3:** If the shipper wants to use the option of combining several sales orders in one shipment the routing code and the rate cannot be calculated until this combination has been done. Thus, the services for calculating the routing code and the rate are called during the delivery step. This is possible if the final weight of the freight is already known during delivery. Label and manifest are still called after packing.

Figure 1 illustrates the 3 different scenarios:

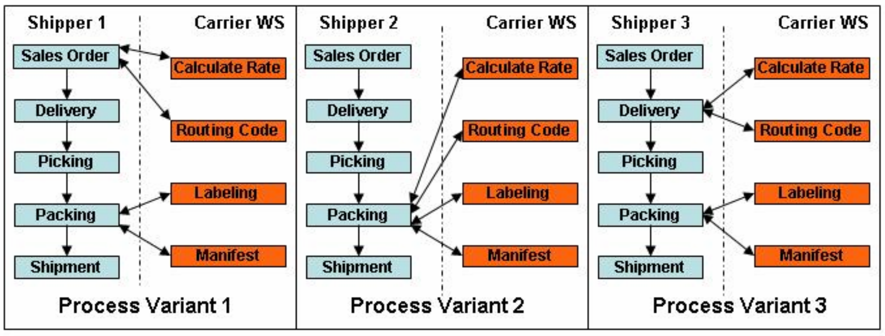


Fig. 1 Different Process Variants in the Scenario

We can see that there are 3 different variants how external web services can be integrated into the same business process. Usually the process is purchased by a customer enterprise (the shipper) and may be preconfigured in one of these variants. In this case only a technical configuration is necessary in terms of registering the technical endpoints of the provided carrier web services in the solution. In other cases the shipping process is not configured at all and must be first configured according to the appropriate process variant reflecting the current business demands of that customer enterprise.

In both cases there is no way to verify if the current configuration in fact satisfies the business level requirements defined by stakeholders of that process, which are often non-technical persons. This is insofar critical, since either the business level requirements may change after the process is already configured or the business process may be reengineered on a technical level. In both cases a mechanism is required to bridge the gap between the technical and business level in terms of to verify whether the technical implementation of the business process always reflects the business level requirements.

It is obvious that in a shipping process not only the carrier web services matter. For example during sales order activity first the stock management systems will be queried to make sure that the ordered goods are already available. If not, the subsequent processes in Supply Relationship Management system will be triggered to order the missing goods. The same applies after the goods are shipped. At this stage the invoicing processes will be triggered to send the customer an invoice according to the applicable customer invoicing process variant for that shipper enterprise. These aspects are out of scope in this work, since we concentrate only on carrier web service integration, but they are rather mentioned here to emphasize that the mentioned problematic pattern in area of different variants on carrier web service integration in a shipping process currently occurs in nearly every other business process in an enterprise.

We should additionally remark that we assume that each shipping enterprise uses one carrier company, thus a dynamic carrier selection from a market-place according to the current shipment conditions is outside of our scope.

5 Formal Verification of CBPs

In this section we present a general approach for the formal verification of a process configuration that can be used in the verification of the CBPs. The approach enables automatic discovery of all parts of a business process that do not satisfy a predefined business requirement. This process we will call inconsistency detection.

We base our work on the semantic description of a business process introduced in [16]. Very briefly, a process is a sequence of activities connected through several types of connectors (join, split, switch). For each activity a set of properties can be defined, like input, output, assigned resources. Finally, for each activity a set of (web) services can be bound.

Although the approach is general, we focus on the type of business requirement mentioned in section 4.2. More precisely, requirements for the service binding of a process can be defined as:

$$M: \text{Property}(A) \rightarrow \text{Property}(WS)^{n}$$

, where
A is the set of activities from a business process
WS is a set of web services

Property(x) is a function that retrieves characteristics of an entity x. A characteristic is defined according to the underlying process model.

Note that the given mapping directly expresses the original constraints mentioned in section 4.2:

$$A \rightarrow WS^n \subset Property(A) \rightarrow Property(Ws)^n$$

5.1 Formal method for Inconsistency Detection

Verification of process configuration is realized using formal methods. These methods seek to establish a logical proof that a system works correctly, i.e. that it is correctly configured. A formal approach provides:

(1) a modelling language to describe the system;
(2) a specification language to describe the correctness requirements; and
(3) an analysis technique to verify that the system meets its specification.

The model describes the possible behaviours of the system, and the specification describes the desired behaviours of the system. The statement the model P satisfies the specification α is now a logical statement, to be proved or disproved using the analysis technique.

Since the goal of the inconsistency detection is to check whether a service description satisfy the required specification, it can be treated as a formal verification problem in which a modeling language to describe a system is defined through the above mentioned process model, a specification language corresponds to the consistency constraints that must be preserved and an analysis technique can be treated as inference process. In the rest of this section we give more details about last two issues.

5.2 Compliance representation

To formally prove the correctness of a model, the first decision is about what claims to prove. In our case, the claim is that there is no violation of the requirements regarding binding of services. It means that the system for process verification has to return an error value in the case that an activity does not comply to a predefined binding. That can be formally described as follows:

$$isCompliant(X) \leftarrow \neg ErrorBinding(X)$$

$$ErrorBinding(X) \leftarrow Activity(X) \wedge WebService(Y) \wedge Binding(X, Y) \wedge \neg M(Property(X), Property(Y))$$

From the implementation point of view, these constraints can be formally represented as DL-safe rules and KAON2[1] engine is used to evaluate these rules in the process of model verification.

[1] http://kaon2.semanticweb.org/

5.3 An approach for inconsistency detection / model verification

One of the main advantages of the proposed model, in which everything is defined rigorously and precisely, is the possibility to verify the service descriptions formally. In other words, process verification can be done by using formal methods. Formal methods are those that provide a rigorous mathematical guarantee that a large system conforms to a specification. Formal methods can be roughly classified as:

(1) Proof-theoretic: a suitable deductive system is used, and correctness proofs are built using a theorem prover, and

(2) Model-theoretic: a model of the run-time behaviour of the system is built, and this model is checked for the required properties.

In this work we apply the first method, since once we have a service description plus the formally defined consistency constraints we can automatically prove whether these constraints are satisfied in the service description with the help of the reasoning. The KAON2 inference engine is used, since it implements the proof-theory for DL and DL-safe rules. By performing an efficient exploration of the possible inconsistencies that can be built in the service description, the system is able to verify all the consistency constraints defined for the proposed process model.

The set of the consistency constraints as well as a description of the concrete service are inputs to the KAON2 inference engine that is used to automatically verify whether the service description satisfies the consistency. Practically, a trace of the answer to a query is considered as a model that reflects how different pieces of a service description are put together to generate the answer. If the KAON2 verifies that the consistency constrains are fulfilled (i.e. there is no answer), then the service description is consistent. Otherwise, the KAON2 provides explanation about causes of problems, since it can identify the conditions under which the problem occurs.

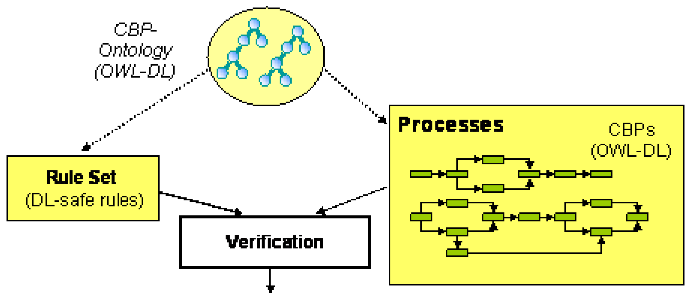


Fig. 2 The illustration of the verification process

Figure 2 summarizes the verification process: The CBP-ontology will be used for the description of the validation rules and the existing processes. The verification module will apply these rules on a concrete process instance in order to determine the validity of the process configuration.

6 Future Research and Conclusion

Changing business environments force software application to be adapted frequently. One challenge is to verify the behavior of a software application if it still reflects the business environments, in which it is executed. In this paper we saw by an example how the business model of a shipper enterprise influences the shipping process composition in terms of the way the carrier web services are integrated into the shipping process. We proposed to use SWRL as a knowledge representation form about the interaction pattern between a shipper and a carrier company. With help of those SWRL-rules the process composition can be verified if it fulfills the required interaction pattern.

By having a formal representation model expressing each enterprise's collaboration configuration for business processes we will be enabled to automatically derive an orchestrated CBP based on that model. A non technical person in an enterprise will be enabled to express and verify the way business processes are composed through available internal or external services if an appropriate user interface is provided to express the rules in an easy way hiding the technical complexity behind those formal models. A very obvious and valuable use case would be then the ability to verify the interoperability configuration of a process with help of reasoning technologies.

As mentioned in the introduction of this paper, one aspect of frequently changing business environments for enterprises is the fulfilling of the growing count of different regulatory compliance requirements. We find nearly in every business area various regulatory requirements, such as Sarbanes Oxley Act, Basel II, HIPAA, ISO 9000, to count a few. One direction of our future work is to research how the content of different regulatory compliance requirements can be captured with semantic technologies and their impact of business processes can be handled more automatically on process-based software applications. Our research will go towards providing a knowledge-based compliance-aware architecture.

Acknowledgment

The research presented in this paper was partially funded by the EC in the projects ATHENA (IST 507312) and SAKE (IST 027128)

References

1. Soffer, P.; Golany, B.; Dori, D.: ERP modeling: a comprehensive approach. Information Systems 28(6) (2003) 673-690
2. ATHENA European Integrated Project, http://www.athena-ip.org/
3. Greiner, U., Lippe, S., Kahl, T., Ziemann, J., Jkel, F.W.: Designing and implementing

crossorganizational business processes - description and application of a modeling framework. In: Interoperability for Enterprise Software and Applications Conference I-ESA. (2006)

4. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web rule language combining OWL and RuleML, version 0.5 of 19 november 2003, http://www.daml.org/2003/11/swrl/ (2003)
5. Baader, F., Calvanese, D., McGuinnes, D., Nardi, D., and Patel-Schneider, P. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press. (2003)
6. Thatte, S.: Business process execution language for web services version 1.1(2003) (2003)
7. Jon R. Wright, Elia S. Weixelbaum, Gregg T. Vesonder, Karen E. Brown, Stephen R. Palmer, Jay I. Berman, and Harry H. Moore. A knowledge-based con-figurator that supports sales, engineering, and manufacturing at AT&T network systems. AI Magazine, 14(3) (1993) 69–80
8. Nestor Rychtyckyj. DLMS: An evaluation of KL-ONE in the automobile industry. In Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96) (1996) 588–596
9. Rosemann, M. and van der Aalst, W. A Configurable Reference Modelling Language. Information Systems, In Press (2005)
10. Charfi, A. and Mezini, M. Hybrid Web Service Composition: Business Processes Meet Business Rules. In Proceedings of the 2nd International Conference on Service Oriented Computing (2004)
11. Orriens, B., Yang J. and Papazoglou M. P. A Framework for Business Rule Driven Service Composition. In Proceedings of the Fourth International Workshop on Conceptual Modeling Approaches for e-Business Dealing with Business Volatility (2003)
12. R. Kazhamiakin, M. Pistore, M. Roveri. A framework for integrating Business Processes and Business Requirements, 9th International IEEE Enterprise Distributed Object Computing Conference (EDOC) (2004)
13. Kim, Henry M. and Fox, Mark S. "Using Enterprise Reference Models for Automated ISO 9000 Compliance Evaluation", In: Proceedings of 35th Hawaii International Conference on Systems Science (HICSS) (2002)
14. UPS Online Tools, http://www.ups.com/content/us/en/bussol/offering/technology/automated_shipping/online_tools.html, Retrieved January 19, 2006
15. FedEx Ship Manager API, http://fedex.com/us/solutions/fsmapi.html; FedEx Return Manager API, http://fedex.com/us/solutions/netreturnapi.html; Retrieved January 19, 2006
16. Stojanovic L., Abecker A., Apostolou D., Mentzas G., Studer R. The role of semantics in e-government service model verification and evolution, Semantic Web meets eGovernment, AAAI Spring Symposia, AAAI. (2006)

Adaptive technologies to address operational complexity in highly configurable value chains

Ray Richardson[1], Aidan Boran[1], Tomas Vitvar[2], Paavo Kotinurmi[2]
David Lewis[3], John Keeney[3], Declan O'Sullivan[3]

Bell Labs, Ireland[1],
firstnamelastname@lucent.com
Digital Research Institute, Galway[2],
firstname.lastname@deri.org
Knowledge and Data Engineering Group, Trinity College, Dublin[3]
firstname.lastname@cs.tcd.ie

Abstract Existing B2B infrastructure is primarily focussed on the secure, reliable, and scaleable transfer of information between business partners. The time and effort taken to establish these B2B connections has meant that the resulting business relationships tend to be long-term and rigid in nature. More recently, however, the configurability of the value chain connecting business partners has been seen as a key to competitiveness. There is increasing pressure to establish more transient ad-hoc relationships whereby dynamic decisions can be made to, for instance, exchange one partner with a more competitive alternative or to introduce new trading partners to improve the robustness of an organisations business model. This new dynamic business model introduces considerable complexity both in the need to deal with heterogeneous partner interfaces and the need to support dynamic decision-making. In this paper we explore how semantic web service technology can be combined with policy-based management to infuse adaptivity into existing B2B infrastructure. The discussion focuses on the use of a high performance policy decision engine to deal with operational complexity introduced as a result of dealing with new business partners.

1. Introduction

In today's business world outsourcing, off-shoring, and partnering have combined to create the virtual organisation where value-adding activities from multiple organisations are connected to form value chains, [1]. There has been considerable investment in efforts to optimise the operational efficiencies of these value chains. Heavyweight e-business frameworks such as EDI, RosettaNet, cXML, and ebXML have been deployed to allow each organisation in the chain to view and modify information inside their own systems rather than have buyers, sellers, assemblers, and engineers work over the phone, fax, or e-mail, [3]. A consequence of these considerable investments to connect businesses is the fact that partnerships, once established, tend to be rigid and long term. More recently, however, the configurability of the value chain has been seen as a key to competitiveness. For many organisations there is increasing pressure to establish more transient ad-hoc relationships whereby decisions can be made to, for instance, exchange one partner with a more competitive alternative or to introduce new business partners in an effort to lessen an organisations dependence on one or more trading partners [1,5].

This dynamic business model introduces considerable complexity both in the need to deal with heterogeneous partner interfaces and the increased operational complexity associated with dynamic decision-making, [7]. Organisations participating in these fluid value chains need a highly adaptive B2B infrastructure to address the additional complexity. In our research we are investigating the use of an integrated semantic web service (SWS) and policy-based management approach to infusing adaptivity into existing e-business framework implementations. Semantic web service technology is used to mediate process and data conflicts within partner interfaces. Policy based management techniques are used to automatically or semi-automatically enforce human governance on dynamic decisions relating to service selection, pricing, levels of service, and so on. Specific contributions of the paper include:

- an overview of a system architecture which incorporates an integration of semantic web services (SWS) technology, widely deployed e-business frameworks, and policy engineering techniques (section 3.1).
- an account of how the approach delivers adaptivity in a specific use case scenario (section 3.2)
- an insight into how a high performance decision engine can be utilised to to enforce organizational policies in decision making related to service selection, parameter setting, and constraint enforcement (section 4).

2 Problem domain - Use Case Scenario

To illustrate the problem domain we consider a use case in which a small-scale supplier, organisation A, supplies widgets to a considerably larger electronics manufacturer, organisation B. Organisation A and B have a long term B2B relationship with fixed terms (pricing, service level, shipping locations, etc..)

and IT support through a RosettaNet e-business framework. Organisation A would like to lessen its dependence on organisation B as its main customer and has recently been approached by other electronics manufacturers requesting quotes for its widget product.

The nature of this new business is somewhat different insofar as requests are more ad-hoc and terms more variable, i.e. pricing, service level, shipping locations, and so on can vary greatly. Specific difficulties exist in that the B2B interfaces to each of the electronics manufacturers are different and additional overheads exist due to the requirement to make on-the-fly decisions on pricing, levels of service, etc..

Organisation A's plans to diversify its customer base and grow production is good for business but requires significantly increased adaptivity in its B2B function. The organisation needs to ensure that profits from its new business are not wiped out by the overhead associated with managing the additional operational complexities.

2.1 Interface heterogeneities

The option to standardise B2B interfaces across its customers is not realistic for Organisation A, primarily due to the company's size relative to its customers. B2B interface heterogeneities occur across three layers – network, data, and process. The network layer is perhaps the most straightforward to deal with as many of the manufacturers are willing to provide smaller suppliers with B2B software clients which can handle the encoding and transportation of messages according to their standard protocol.

Data conflicts are more troublesome. Even for manufacturers sharing the same e-business standard it's quite common for data conflicts to arise. For instance one manufacturer may expect contact phone numbers that include area and country codes whereas another simply expects phone numbers to have an area code. Perhaps more serious would be a situation where two manufacturers have different standard units of measure for the same product (e.g. one uses a 5 pack whereas another uses 10 pack). These mismatches are only apparent by inspecting the content of messages or even worse as a result of an investigation following unexpected events.

Process heterogeneities relate to differences in the specific message exchange sequences. One manufacturer may issue multi-line orders as a series of individual requests whereas another may bundle them together into a single request. Again even within the same e-business standards differences can arise.

2.2 Dynamic decision making

In its current business Organisation A has long term agreements with its customer base (Organisation B). Decisions on pricing, levels of service, order volumes, and so on are agreed up front for a period of 12 to 24 months. These 'variables' can be plugged into existing supporting B2B infrastructures. In the extended business model relationships are more ad hoc in nature and operational decisions are much more dynamic.

Human involvement in all decision making is an expensive solution which does not guarantee consistent policy enforcement and fails to scale well. Hard coded application logic or the use of configurable parameters leads to static policies that can be difficult to extend and often result in sub-optimal decisions due to poor models of the real world or poor use of all available relevant information. There is a requirement to support truly dynamic adaptive decision making.

3 Highlevel Architecture Overview

Our approach to addressing the particular demands of the use case scenario is to implement an integration of SWS technology, policy based management techniques, and existing e-business frameworks.

3.1 Overview of Solution

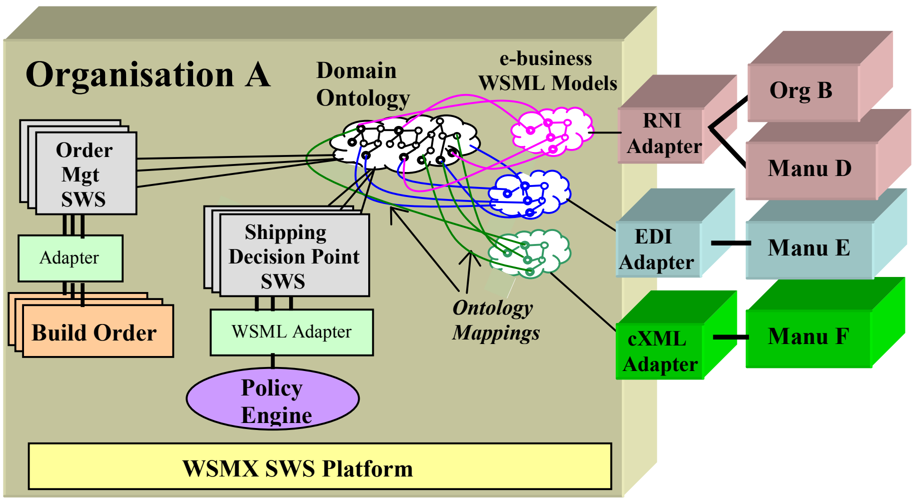


Fig 1: High-level view of architecture

Key elements of the architecture include:

- WSMX - we make use of the Web Services eXecution Environment (WSMX) as our SWS platform, [2]. It provides core support for semantic service discovery, data meditation, process mediation, and service invocation. WSMX resides entirely within organisation A. This allows us to avoid making any assumptions on the semantic technology capability of partner organisations. WSMX makes use of the Web Services Modelling Language (WSML) [8] for all internal processing.
- e-business framework ontologies – these ontologies are flat WSML representations of e-business framework (RossettaNet, ebXML, EDI, etc.) message contents. XSLT is used to automatically construct these ontologies from an XML schema representation of the e-business standard.

```
…
<ProductLineItem>
   <UnitOfMeasureCode>12-pack <UnitOfM../>
   <LineNumber>1</LineNumber>
   <requestedQuantity>
        <ProductQuantity>10</ProductQuantity>
   </requestedQuantity>
…
```

```
…
concept productLineItem
 nonFunctionalProperties
   dc#title hasValue "…"
 endNonFunctionalProperties
 lineno ofType (0 1) _integer
 unitcode ofType UnitOfMsre
 qty ofType  (0 1) Quantity
```

- e-Business Adapters – these exist to translate or 'lift' e-business standard messages into a WSML format, making use of concepts defined in the e-business framework ontologies. For messages going in the opposite direction the WSML concepts are 'lowered' to become e-business standard messages. These adapters are also responsible for creating WSML goals on receipt of 'kick-off' messages.

- Semantic web service descriptions representing back-end information systems (Order Mgt., Shipping, etc..) within organisation A. The descriptions eliminate semantic ambiguity by binding input and output parameters to specific concepts within an accompanying domain ontology. Another adapter exists to 'lift' and 'lower' messages received and sent between back office systems and their corresponding semantic web services.

```
webService OrderMgt

importsOntology { _"http://www.orgA.com/OM" }

capability OrderMgtSWSCapability
 sharedVariables {?request}

 precondition
   definedBy
     //A request to create an order
     ?request memberOf mn#createOrderRequest or
     //A request to add a lineitem to an order
        …

 postcondition
         …
 interface OrderManagementInterface
 choreography OrderManagementChoreography
   stateSignature
       …
```

- Design time ontology mappings - these mappings identify equivalences and relationships between concepts in each of the e-business ontologies and the domain ontology. A data mediation tool exists to support the process.
- The Policy decision engine is used to enforce organisational policies in decision making processes. Decision points are exposed as semantic web services using concepts from the domain ontology (or possibly another ontology linked to the domain ontology via mappings). The policy decision engine is discussed in greater detail in section 4.

3.2 Simple walk through

In this section we provide a simple walkthrough of the architecture described previously. For the walkthrough we assume we are dealing with a purchase order request received from a customer who utilises RosettaNet. We further assume there are backend services to both build an order (*ProcessOrder*) and to deal with shipping. As part of the decision to broaden its business activities org A has introduced a range of external shipping functions that can be used in place of its internal shipping function for certain situations. The basic flow of activity for processing a purchase order is shown below.

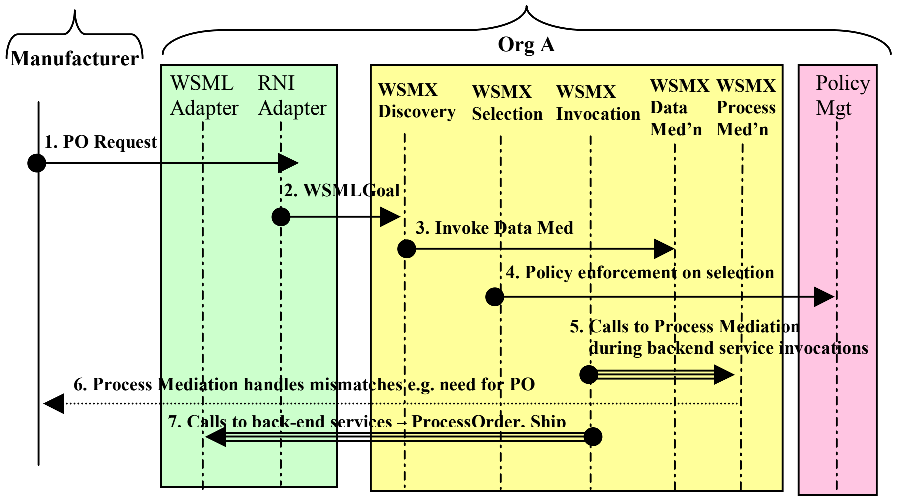

Manufacturer
Org A
WSML Adapter
RNI Adapter
WSMX Discovery
WSMX Selection
WSMX Invocation
WSMX Data Med'n
WSMX Process Med'n
Policy Mgt
1. PO Request
2. WSMLGoal
3. Invoke Data Med
4. Policy enforcement on selection
5. Calls to Process Mediation during backend service invocations
6. Process Mediation handles mismatches e.g. need for PO
7. Calls to back-end services – ProcessOrder, Ship

Fig 2: High-level walk through

Key points include:

- RosettaNet purchase Order request message, PIP 3A1, is 'lifted' by the RNI adapter into an equivalent WSML format. The receipt of this message triggers the adapter to formulate a WSML goal from a set of pre-defined templates. The goal seeks to have a purchase order processed.

- The WSMX Discovery component matches the WSML goal against existing service capabilities. There may be capabilities to quote a price, to create an order, to ship an order, etc.. This match may be quiet simple in practice with inputs/outputs simply checked – data mediation is invoked where necessary. In some cases it may be the case there is no direct match between a goal and the available service descriptions. For instance in this example there is no single service to process an order. However by composing the Build Order and one of the Ship Order services the goal can be achieved.
- The service selection step, in particular which shipping service to utilise can be partially or fully controlled by policy based management. Further detail on how this may take place can be found in section 4. The use of policy based management in this way introduces a significant level of adaptivity into operations management and provides a consistent scaleable solution to the increasing operational complexities found when dealing with additional business partners.
- The mappings defined between the base and domain ontologies are executed as part of the SWS invocations. The mappings ensure RosettaNet concepts/attributes such as *AddressLine1, CityName,* etc., are appropriately translated into their backend equivalents, e.g. *Street, City,* etc..
- Process mediation is a further service offered by the WSMX environment. WSMX is capable of analysing the process choreographies of the goal and the individual service descriptions to identify and reconcile process heterogeneities. For instance the RossettaNet message process purchase order expects to receive an acknowledgement following the issue of a purchase order message (PIP 3A1). This acknowledgment may not be provided by the back-end BuildOrder service, the process mediator is thus responsible for auto-generating the acknowledgement message. Abstract state machines are used at runtime to keep track of process executions.
- During the service execution WSML individuals will be lowered into a message format that can be consumed by the Order Mgt and Shipping back-end services.

4. Policy based management

In our research we propose policy-based management to support dynamic decision making. Declarative rules are used to enforce organizational policies in decision making related to service selection, parameter setting, and constraint enforcement. The Vortex rules engine [10] is being used as the decision engine. Vortex is a high performance, acyclic, forward chaining rules engine that supports reasonably rich policy management for real time environments.

Policies, in their simplest form, are event-condition-action rules. Correspondingly Vortex rules have a simple :

```
If(Condition) then
        action1,
        action2,
        action3,
         ...
    end
```

format. The rules language is strongly typed with support for both atomic and complex typed variables. Vortex is packaged with an extensible range of support functions that can be called from any rule condition or action. Permitted actions include assigning a value to a variable, appending a value to a list variable and removing a value from a list variable. Rules are organized into what are known as rule sets, i.e. the set of rules that should be used for a given 'decision request'. Each rule set has an explicit input/output signature. From an architectural perspective we expose rule sets as individual services with ontologically bound input/output parameters. In some cases data mediation may be necessary as part of decision request processing.

In order to provide an insight into how the decision engine is utilised we build on the scenario developed in previous sections. In this scenario one of the decisions required is to select the most appropriate shipping options. Organisational policies should be adhered to in compiling these options. Information from the purchase order and candidate shipping services are forwarded with the decision request – it itself being a WSML goal. The decision engine is capable of issuing requests to external sources to retrieve additional information required to evaluate the conditions of all rules. The information returned from the decision service consists of a set of shipping options that comply with organisational policies pertaining to the shipping request. A human administrator may make the ultimate shipping decision from this short list of valid choices. Alternatively the selection may be based on some simple criteria such as the cheapest conforming shipping service.

Rule sets begin with a declaration of input, output, and intermediate variables. In our simplified *shipping decision* rule set input variables include the list of concrete shipping candidates, the name of the purchasing organisation, the time the shipment will be available for pickup, the shipment destination, etc..

```
variables:
         purchasingOrgName : string;
        availableForPickup : string;
        shipmentDestination :  list Record of  { location : string };
        shippingCandidates  : list of Record { identity : String;
                                                    pickup_Time : string;
                                                    pickup_Date : string;
                                                    cost : string;
                                                    setdown : string; };
```

An adapter takes care of lowering WSML concepts to become input variables. In some cases the set of input variables are extended as a result of additional domain knowledge held in the ontology. For example a single shipmentDestination of Kista would have the additional locations of Stockholm and Sweden added as Kista is located in Stockholm which is in turn located in Sweden. This expanded list of locations results in more robust rule sets.

Intermediate variables are used to store temporary values during the rule set execution. In some cases these temporary values are populated as a result of rule actions to retrieve information from external sources.

```
shipmentChannel : string;
internalShippingCapacity : string
filteredFromInHrs, filteredFromPerf, filteredFromPickup,
filteredFromPerferred, blacklisted :
                list of Record { identity : String;
                                     pickup_Time : string;
                                     pickup_Date : string;
                                    cost : string;
                                    setdown : string;
                                     priority : string; };
  onTimePerf  : list of Record {identity : string;
                                            channel : string;
                                            perf_Rating : string; };
 preferredVendorList  : list of Record {identity : string;
                                            channel : string; };
```

The single output variable in this case is the list of shipping options that adhere to all organizational policies.

```
validShippingServices  : list of Record { identity : String;
                                      pickup_Time : string;
                                      pickup_Date : string;
                                     cost : string;
                                     setdown : string;
                                      priority : string; };
```

The actual rules are typically organized into groups with the initial group setting intermediate variables, e.g. :

```
shipmentChannel = "lane1";
if(shipmentDestination[$i] == "USA" || shipmentDestination[$i] == "UK") then
            shipmentChannel = "lane2"
end
```

The *shippingDestination[$i]* syntax leads to an evaluation of the rule for each member of the *shippingDestination* list value.

Subsequent rule groups actually enforce the organizational policies. In our simplified scenario we assume the following policies exist :

1. A preferred vendor list exists for each shipping lane. Company policy states for any given shipment the selected shipper must be on the preferred list for the shipments shipping lane.
2. A shipper must have an on time performance of greater than 95% for the the shipping channel in question
3. Shippers are required to make pickups within regular hours
4. The pickup cannot be more than 4 days after *availableForPickup* date
5. Shipments for organization B take priority in the case of the internal shipment service

Generally speaking policies act to filter the allowable list of shipment services. The corresponding rules for each of the policies are presented below :

```
rule: Rule_1
if(shippingCandidates[$i].identity == PreferredVendorList[$j].identity &&
      PreferredVendorList[$j].ShippingChannel == shipmentChannel)
            filteredFromPreferred += ShippingCandidates[#i];

rule: Rule_2
if(filteredFromPreferred [$i].identity == onTimePerf[$j].identity &&
      onTimePerf [$j].channel == shipmentChannel && onTimePerf [$j] > 0.95)
            filteredFromPerf  +=  filteredFromPreferred[#i]

rule: Rule_3
if(Time::between(filteredFromPerf[$i].pickupTime, "08:00", "18:00"))
        filteredFromInHrs += filteredFromPerf[#i];

rule: Rule_4
if(Time::numberOfDaysBetween(filteredFromInHrs[$i].pickupDate,
                                            availForPickup) < 4)
      filteredFromPickup += filteredFromInHrs[#i];
```

```
rule: Rule_5
if(filteredFromPickup[$i].identity == “Internal” &&
                internalShippingCapacity < 0.2 && requestingOrg != “Org B”)
        blacklisted += filteredFromPickup[#i]

rule: Rule_6
if(! (filteredFromPickup[$i] in blacklisted ))
        validShippingServices += filteredFromPickup[#i]
```

Relating organisational policy semantics to the semantics used to define both back-end systems and partner interfaces has obvious benefits in ensuring policy constraints operate as expected. By enforcing policies in service selection and parameter setting an organisation can flexibly and consistently control how it interoperates with partners. Semantically encoded policies are themselves more adaptable to change and heterogeneity and are considerably easier to encode. For instance a policy that states “*during public holidays pickups must take place between 9:00am and 12:00am*” can take advantage of domain knowledge for what constitutes a public holiday to simplify the encoding. Data mediation further enables policies to adapt to heterogeneity, e.g. a concrete service description might encode a pickup time using a 24 hour format in place of the standard 12 hour clock used internally. A mediator can automatically mediate this

5. Conclusions and future work

In this paper we have presented some of the problems facing organisations attempting to participate in configurable value chain partnerships. Increased adaptivity is required within the B2B function to address interface heterogeneities and operational complexities introduced by the more dynamic business model. An integration of SWS technology and policy-based management are proposed to deliver the required adaptivity. Internally deployed semantic web services are employed to address data and process heterogeneities present in partner interfaces. Semantically encoded policies are used to ease difficulties associated with the dynamic decision-making.

The focus of our work is currently on investigating the options available to integrate the policy management and semantic web services onto a single platform, preparing an evaluation framework for the architecture, and building supporting tools. Specific tools in the policy management space include a GUI tool to support the creation of rules and pre-processors to expand rule sets based on domain knowledge.

References

1. T. Friedman, The World is Flat: A Brief History of the Twenty First Century, Farrar, Straus and Giroux, 2005.
2. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX – A Semantic Service-Oriented Architecture. In Proceedings of the 3rd International Conference on Web Services, pages 321 – 328, Orlando, Florida, USA, 2005.
3. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, and A. K. lmagarmid. Business-to-business interactions: issues and enabling technologies. VLDB Journal, 12(1):59–85, 2003.
4. G Olsen, An overview of B2B Integration, eAI Journal, May 2000, p 28-36.
5. Y Sheffi, The Resilient Enterprise - overcoming vulnerability for competitive advantage, The MIT Press, 2005.
6. M. Kerrigan, The WSML Editor Plug-in to the Web Services Modeling Toolkit. In *Proceedings of 2nd WSMO Implementation Workshop (WIW2005)*. Innsbruck, Austria, 2005.
7. C. Preist, J. E. Cuadrado, S. Battle, S. Williams, and S. Grimm. Automated Business-to-Business Integration of a Logistics Supply Chain using Semantic Web Services Technology. In ISWC '05: Proceedings of 4th International Semantic Web Conference, 2005.
8. J de Bruijn, H. Lausen, and D. Fensel, The WSML Family of Representation Languages, http://www.wsmo.org/TR/d16/d16.1
9. E Cimpian, and A. Mocan, Process Mediation in WSMX, http://www.wsmo.org/TR/d13/d13.7/v0.2
10. Richard Hull, Francois Llirbat, Francois Llirbat, Eric Simon, Jianwen Su, Guozhu Dong, Bharat Kumar, and Gang Zhou, *Declarative Workflows that Support Easy Modification and Dynamic Browsing*, International Joint Conference on Work Activities Coordination and Collaboration (WACC) held in San Francisco, February, 1999, pp. 69-78.

Semantic Event-driven Process Chains

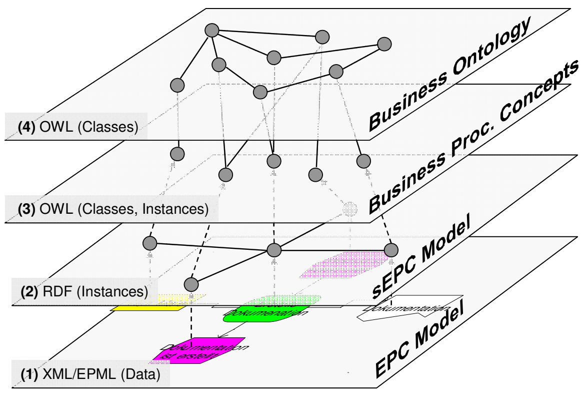
Oliver Thomas, Michael Fellmann

Institute for Information Systems (IWi) at the German Research Center
for Artificial Intelligence (DFKI), Saarbruecken (Germany)
{thomas|fellmann}@iwi.uni-sb.de
http://www.iwi.uni-sb.de

The event-driven process chain (EPC) is a semi-formal modeling language for the description of business processes [1]. It is used for the planning, visualization and analysis of business processes in the realm of business process management. EPC models essentially consist of a set of functions and events, which are connected via a control flow using arcs and connectors. On the one hand, EPC models are used to describe processes from a business perspective. On the other hand, EPC models are used to facilitate the adoption and customization of process oriented information systems, thereby serving as a staring point for the actual implementation [6].

Research regarding the semantics of the EPC so far mainly concentrated on the formal semantics of the available language constructs [2; 3; 4; 5; 7]. The labels of the individual elements of an EPC model were not considered in these investigations so far, although they significantly contribute to the overall use of an EPC model. Usually, the designer of a model adds these labels in a natural language. Hence, a substantial part of the semantics of an EPC model is bound to natural language, introducing a high degree of ambiguity and misinterpretation risk. As long as a model is provided and read only by one individual, this is less problematic. However, if models of different modelers are united, queried and translated, or semantics contained in the models should be validated automatically and leveraged for the configuration of an information system, it is necessary to have clearly defined semantics for each of the model's elements.

This problem can be solved by associating the elements of an EPC model with concepts of a formal ontology, which can be referred to as semantic annotation. In order to make use of this semantic annotation and to shift from event driven process chains to semantic event driven process chains (sEPC), a four-layered approach has been developed by the authors (cp. figure to the right). The different layers are ordered according to the increasing degree of abstraction from the underlying EPC model to more general semantics. The topmost layer 4 contains the business ontology comprising all relevant concepts of an enterprise context and their interrelationships as OWL classes and properties (cp. http://www.w3.org/2004/OWL/). This ontology might be created by merging different ontologies which conforms to the open world assumption of OWL. As a representation language, the OWL-DL subset of OWL is used in order to gain the maximum of expressive power while retaining computational completeness. Going from the top to the bottom, in the next layer 3, these general concepts are used to create new, specialized concepts for the representation of the semantics of individual business process elements, e.g. distinct functions like "order processing" or events like "order received". On this level, additional information can be added like semantic restrictions or details regarding the technical implementation and execution of processes in a centralized and consistent manner. These concepts are instantiated afterwards in the same layer; the instances of business process concepts, produced thereby, can correspond with physically existing entities in the enterprise (e.g. resources). In the underlying layer 2, the instances of the upper layer are used to generate a semantic description of business processes. This is accomplished by establishing a graph based flow



between the instances of the upper layer 3 for each semantic event driven process model, thereby using information of the bottom layer 1 for the concrete flow and the instances involved in this flow. In order to represent a sEPC model, the expressive power of RDF is sufficient. To transform EPC models into sEPC models, the EPC models on layer 1 have to be extended slightly with semantic mapping information. That is, the modeler must associate instances of layer 3 to the EPC process model on layer 1. Technically, this annotation information is added to the XML representation of an EPC model using attributes. For the later transformation, a XSLT stylesheet has been developed which consumes an annotated EPML/XML-model and produces the corresponding RDF/XML representation. For storing and querying the generated sEPC models, a preliminary prototype has been developed at the Institute of Information Systems (IWi) which uses a relational database and the Jena framework. The prototype allows querying sEPC models using the SPARQL query language from the W3C in conjunction with an inference engine.

The overall benefits of our approach are:

– Process models can be queried on a semantic level. With the use of inference engines it is possible to infer new facts that are not contained in the original model. For example, if inventory is defined to be made up of physical things which can be sold to customers, and there is a process which consumes such things, it can be inferred that the process reduces inventory.
– Advanced validation opportunities of process models are achieved. The validation of a sEPC model is done against all restrictions established in the ontology layers 3 and 4. Therefore, it is possible to impose policies for all business processes in a centralized way.
– The execution of processes can be facilitated as the ontology easily can be extended with technical information, thereby bridging the gap between business and technical process models. For example, a BPEL representation can be generated from a sEPC model using execution information added to the ontology classes on layer 3. Consequently, the alignment of business process concepts with the IT-infrastructure can be done in a centralized way without redundancy.
– Queries are possible both on the process concepts level (level 3) and on the instance level (level 2) hence allowing a user or potential business partner to discover available process element types before retrieving instance data from a sEPC repository.
– The expenditure for the internationalization of process models can be reduced as the translation of process model element labels is required only once per process element type on level 3 in contrast to the translation of individual model element labels.

Further research will be done regarding suitable ontologies and tools for the annotation of process models. Therefore, a prototype for a sEPC repository is currently under planning that will provide interfaces or plug-ins for well-established modeling tools.

References

[1] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". In: Scheer, A.-W. (ed.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, No. 89, Saarbrücken : Universität des Saarlandes, 1992 (in German)

[2] Kindler, E.: On the semantics of EPCs: Resolving the vicious circle. In: Data & Knowledge Engineering 56 (2006), No. 1, pp. 23–40

[3] Langner, P.; Schneider, C.; Wehler, J.: Petri Net Based Certification of Event driven Process Chains. In: Desel, J.; Silva, M. (eds.): Application and theory of Petri nets 1998 : 19th international conference ; proceedings. Berlin : Springer, 1998, pp. 286–305

[4] Nüttgens, M.; Rump, F. J.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: Desel, J.; Weske, M. (eds.): Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (Promise '2002), Universität Potsdam, 9.–11. Oktober 2002. Bonn : Köllen, 2002, pp. 64–77 (in German)

[5] Rosemann, M.; van der Aalst, W. M. P.: A Configurable Reference Modelling Language. In: CITI Technical Reports, No. FIT-TR–2003–05, Brisbane : Queensland University of Technology, 2003

[6] Scheer, A.-W.; Thomas, O.; Adam, O.: Process Modeling Using Event-driven Process Chains. In: Dumas, M.; van der Aalst, W. M. P.; ter Hofstede, A. H. M. (eds.): Process-aware Information Systems : Bridging People and Software through Process Technology. Hoboken, New Jersey : Wiley, 2005, pp. 119–145

[7] van der Aalst, W. M. P.: Formalization and verification of event-driven process chains. In: Information and Software Technology 41 (1999), No. 10, pp. 639–650