Feasibility of EPC to BPEL Model Transformations Based on Ontology and Patterns

Lucas O. Meertens, Maria-Eugenia Iacob, and Silja M. Eckartz

University of Twente,
Drienerlolaan 5, 7522 NB, Enschede, The Netherlands
{l.o.meertens,m.e.iacob,s.m.eckartz}@utwente.nl

Abstract. Model-Driven Engineering holds the promise of transforming business models into code automatically. This requires the concept of model transformation. In this paper, we assess the feasibility of model transformations from Event-driven Process Chain models to Business Process Execution Language specifications. To this purpose, we use a framework based on ontological analysis and workflow patterns in order to predict the possibilities/limitations of such a model transformation. The framework is validated by evaluating the transformation of several models, including a real-life case.

The framework indicates several limitations for transformation. Eleven guidelines and an approach to apply them provide methodological support to improve the feasibility of model transformation from EPC to BPEL.

Keywords: Model transformation, EPC, BPEL, Guidelines.

1 Introduction

In most traditional software application development practices, the ultimate product of the design process is the "realization", deployed on available realization platforms. In several model-driven approaches, however, intermediate models are reusable and are also considered final products of the design process. These models are carefully defined so that they abstract from details in platform technologies, and are therefore called computation-independent models (CIMs) and platform-independent models (PIMs), in line with OMG's MDA [1][2]. MDA (Model-Driven Architecture) has emerged as a new approach for the design and realization of software, and has eventually evolved into a collection of standards that raise the level of abstraction at which software solutions are specified. Thus, MDA fosters a design process and tools, which support the specification of software in modelling languages such as UML, rather than in programming languages such as Java.

The central idea of MDA is that design models at different levels of abstraction are derived from each other through model transformations. More specifically,

S. Rinderle-Ma et al. (Eds.): BPM 2009 Workshops, LNBIP 43, pp. 347–358, 2010.

different platform-specific models (PSMs) can be derived (semi-) automatically from the same PIM, making use of information contained by a platform model. Thus, MDA eventually advocates the principle that models can automatically be made directly executable, instead of being delivered to programmers, only as a source for inspiration or requirements, in order for them to create the real software [3]. The complete route from business model to executable code requires model transformations that function as a bridge between business process modelers and the IT department, and actually bring us one step closer to real and (partially) automated business-IT alignment. In this paper, we focus on a specific model transformation, namely the transformation from EPC (Event-driven Process Chains) [4] to BPEL (Business Process Execution Language version 1.1) [5]. The business uses EPCs to model its processes. BPEL serves as the executable code used by IT, in order to manage the control flow, for example to invoke web services.

The contribution of this research is threefold. First, we propose an approach to evaluate to what extent model transformation between two process modeling languages are possible. Secondly, we apply the proposed framework to the specific case of EPC to BPEL transformations. Furthermore, we evaluate the accuracy of these transformations as implemented in the Oracle BPA Suite, and uncover some of the limitations one may expect when using the above-mentioned implementation in practice. Finally, we propose several practical modeling guidelines and an algorithmic approach, which allow modelers to improve the feasibility of EPC to BPEL transformations.

The paper has the following organization. Section 2 briefly explains the research method that we use in this paper for analyzing and evaluating model transformations. Section 3 is devoted to the presentation of the theoretical framework and its application to the case of EPC and BPEL. Transformation of several diagrams in Section 4 puts the framework to the test and reveals several practical issues. In Section 5, we propose our guidelines and approach to improve the feasibility of EPC to BPEL model transformation. Section 6 discusses the results and relates our findings to previous research. Finally, in Section 7, we present our conclusions and pointers to future research.

2 Methodology

In order to analyze to what extent transformation from EPC to BPEL is possible, a theoretical framework is developed first. Then, in order to validate the framework, several models are transformed from EPC to BPEL using the Oracle BPA Suite, and the practical results are compared to the expectations (as resulted from the application of the framework). Finally, guidelines are devised to provide methodological support to improve the feasibility of EPC to BPEL transformation.

The theoretical framework consists of two components that combined form an approach for the analysis of model transformations between process modeling languages in general. The first component is represented by the Bunge-Wand-Weber (BWW) representational model [6]. The BWW model defines the

concepts that modeling languages should be able to represent. Evaluating the languages according to this model indicates their completeness and clarity. The second component entails the workflow control patterns (WFCP), proposed by Van Aalst et al. [7]. These represent the patterns that commonly occur in business processes. Both EPC and BPEL have been evaluated separately already with respect to the BWW model and WFCPs [8][9][10]. In this research, we compare the evaluations of the two languages with each other in order to discover the theoretical limitations of transformation.

The models used during the evaluation cover the patterns and concepts that EPC is able to represent according to the framework. We compared the resulting BPEL specifications to the code fragments documented by Mulyar [11], who analyzed the capability of Oracle-BPEL to represent patterns. Furthermore, we transformed a composite model from a real-life case to discover additional, practical limitations.

Based on the uncovered limitations, we devised guidelines. More precisely, they resulted from workarounds to the limitations and ways to avoid the limitations altogether. We validate the guidelines by applying them to the composite case. As both EPC and BPEL focus on the static flow of control, this research only deals with the control flow aspects of both languages. Other aspects, such as data and resources, fall outside our scope.

3 A Framework to Evaluate Model Transformation

This section presents the framework, which provides a method for evaluating model transformation. Language evaluation using ontological analysis (using the BWW model) and workflow patterns form its basis, and comparison based on those two components completes the framework. We argue that it is possible to use this framework to evaluate the model transformation from any business process modeling language to another. As explained in the sequel, when applying this framework to EPC and BPEL we conclude that it is possible to map most patterns and constructs from EPC to BPEL. However, our research shows that one pattern is impossible to transform, and several constructs cause ambiguities.

3.1 Ontology

As part of this research, ontology provides a theoretical foundation, as it studies the way the world, business processes in this case, is viewed, and especially modeled. The BWW representational model [6] is one of the two components that we selected for the framework.

An ontological analysis of a modeling language consists of checking which concepts in the BWW model, the language is able to represent through its constructs, and how. Any deficiency (no language constructs exist to represent a certain BWW concept) found during such an evaluation renders the representation less complete. Three other types of "defect" affect the clarity of the representation: redundancy (more than one language construct for a BWW concept),

overload (more than one BWW concept for a language construct), and excess (a language construct that has no related BWW concept).

Not all cases of lack of clarity and completeness in a modeling language lead to problems for model transformation. It mainly depends on whether the source or the target language contains the issue. For example, in the case of a deficiency in the source language, a certain BWW concept is impossible to model in that language. Therefore, it will never be necessary to transform that particular concept (since it does not exist in the source) and, consequently, no issue arises. Similarly, if both languages have some excess construct with the same meaning, but no related concept in the BWW model, it is still possible to map the constructs to each other.

The BWW model was used to evaluate both EPC and BPEL separately already [8]. However, it was not yet used to compare the two languages to each other, which is done in the remainder of this section.

Table 1 shows how many of the EPC and BPEL constructs have been found to represent the concepts in the BWW model. We left out the concepts that EPC is unable to represent, as it will never be necessary to transform them. Noticeably, EPC has no redundant constructs; all concepts are represented in EPC by a single construct. BPEL, on the other hand, has redundant constructs for several concepts. Especially the availability of eleven constructs for Transformation stands out. Besides redundancy, BPEL also lacks constructs for several concepts that EPC is able to represents, such as State Law.

Table 1. Ontological completeness and redundancy

BWW Concept	EPC	BPEL 1.1
State	1	1
State Law	1	
Stable State	1	
Event	1	4
External Event	1	1
Internal Event	1	3
Well-Defined Event	1	1
Transformation	1	11
Lawful Transformation	1	3
Level Structure	1	

Table 2. Ontological excess and overload

Excess		Overloaded	
EPC	BPEL	EPC	BPEL
AND-connector	Empty	Function	Partners
OR-connector	Message property	Event	
XOR-connector	Message definition		
	Sequence		
	Flow		
	Scope		

Table 2 shows the overloaded and excess constructs in each of the languages. For EPC all three connector types are considered excess, as they are not strictly needed to model a process. However, with the exception of the OR-connector, they directly match to BPEL constructs. Both other EPC constructs are overloaded. The small number of constructs in the language explains both this, and the lack of redundancy.

3.2 Patterns

A second approach to evaluate and compare modeling languages, as well as their mappings to other languages, is to identify their support for patterns. For this research, the applicable patterns appear in workflow literature, specifically the patterns by Van Aalst et al. [7]. Only the twenty standard static workflow control patterns (WFCP) are the patterns considered as a component for this research, as opposed to data, resource, and advanced patterns. Patterns were used to evaluate both EPC [9] and BPEL [10] separately already. However, a comparison of the two languages, EPC and BPEL, in order to detect which patterns may cause problems in case of a transformation, was not done before.

Similar to EPC completeness with respect to the BWW model, only those patterns that the source language is able to represent are of interest. Table 3 lists those patterns for EPC. Problems only arise when the target language is not able to represent one of those patterns. For the case of EPC to BPEL transformation, the only problematic pattern is WFCP 10, Arbitrary Cycles.

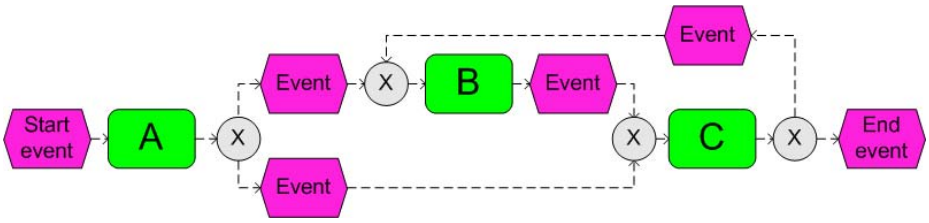

Fig. 1. Workflow control pattern 10: an arbitrary cycle

Table 3. Capability of representing patterns

Patterns		EPC	BPEL
WFCP 1	Sequence	+	+
WFCP 2	Parallel Split	+	+
WFCP 3	Synchronization	+	+
WFCP 4	Exclusive Choice	+	+
WFCP 5	Simple Merge	+	+
WFCP 6	Multi Choice	+	+
WFCP 7	Synchronizing Merge	+	+
WFCP 10	Arbitrary Cycles	+	-
WFCP 11	Implicit Termination	+	+

Fig. 1 shows a variant of this pattern with a loop that has two entry points. It is just one of the many possible variants of this pattern. Thanks to its graph-structure, EPC is able to represent this pattern inherently. On the other hand, BPEL has a block-structure, which does not allow most variants of this pattern.

3.3 Issues and Solutions

Table 4 shows all issues for transformation from EPC to BPEL based on the framework. For each of these issues, a solution must be proposed when trying to implement a (partially) automated transformation. Two of the most straightforward solutions in nearly any situation are to forbid the use of the problematic construct or pattern, or to leave that part of the transformation to a human developer. A third solution is to disregard the part altogether. The best solution depends on the situation.

Table 4. Theoretical issues for transformation

Criteria		Issues
Ontology	Deficiency	State Law Stable State Level Structure
	Excess	OR-connector
	Overload	EPC Event EPC Function
	Redundancy	BWW Event BWW Internal Event BWW Transformation BWW Lawful Transformation
Patterns	WFCP 10	Arbitrary Cycles

4 Validation of the Framework

The Oracle Business Process Analysis (BPA) Suite contains an implementation for model transformation from EPC to BPEL. In order to both validate the theoretical framework and to assess the ability of this tool to perform the transformation correctly, we used the Oracle BPA Suite to transform a set of small models and a larger composite real-life case from EPC to BPEL. The component of the suite used for modeling and transformation is the Business Process Architect. The version of the tool used in this research is 10.1.3.4. We found no tools that provide EPC to BPEL transformation, other than the Oracle BPA Suite, and IDS Scheer's SOA Architect, which is its basis.

4.1 Pattern Transformation

The set of small models consists of relatively small EPC models based on the patterns that EPC is able to represent, as listed in table 3. Besides the patterns

themselves, these models also contain all the concepts and issues that we found in the ontological analysis. We compared the resulting BPEL code of the pattern fragments to the BPEL code documented by Mulyar [11]. For several patterns, he proposes two possible mappings, one with link-constructs, and one without. Use of the link-construct allows for a transformation, which is applicable for more cases. This is good for automatic transformation. However, it has several drawbacks. Especially for understandability, the mapping may be unacceptable.

Patterns 1 to 5 and 11 transform successfully, and preserve their semantics. Two things stand out in the resulting BPEL code. Firstly, the models transform to the proposed variant without the use of link-constructs. Secondly, the transformation of the function-construct of EPC, which represents the BWW concept of transformation, catches the eye. As table 1 shows, BPEL overloads this concept with eleven constructs. In the tool, the choice is made to transform the concept to an invoke activity within a sequence within a scope.

The tool fails to transform 'multi choice' and 'synchronizing merge' patterns. It provides the error message that this version (10.1.3.4) of the tool does not support the OR-connector. While Mulyar [11] provides a mapping to BPEL code for these patterns, the OR-connector is indeed a possible issue according to the framework.

The tool succeeds in transforming only certain variants of the 'arbitrary cycle' pattern. More specifically, we refer to those variants that are possible to model with a while-loop. Other variants of the pattern are impossible to transform, for example loops with multiple entry and exit points. By using the framework, we predicted that this pattern cannot be transformed. For the general case, this holds true.

4.2 Composite Case Transformation

To further validate the framework and investigate the feasibility of the model transformation in practice, we also transformed an existing EPC model from a real-life case. The case was the monthly "Accounting Close"-process in order to bill customers, as modeled for a large, Dutch insurance organization. It involves several departments and many information systems but, as this research focuses on the control flow, these resource and organizational entities are not present in the diagrams. Originally, the process was modeled as a composition of several smaller business processes. The main process consists of six sub-processes, which differ in size and complexity.

Fig. 2 shows the full composite case diagram. It shows all the sub-process combined. Together, the sub-processes cover all patterns and BWW concepts, which EPC is able to represent.

Before attempting to transform the full composite process, we transformed the sub-processes one by one. Some of the diagrams transform successfully without changes, while others need modification first. They require modification, either because part of the transformation is theoretically impossible, or because the Oracle BPA Suite does not (yet) support the structure or construct.

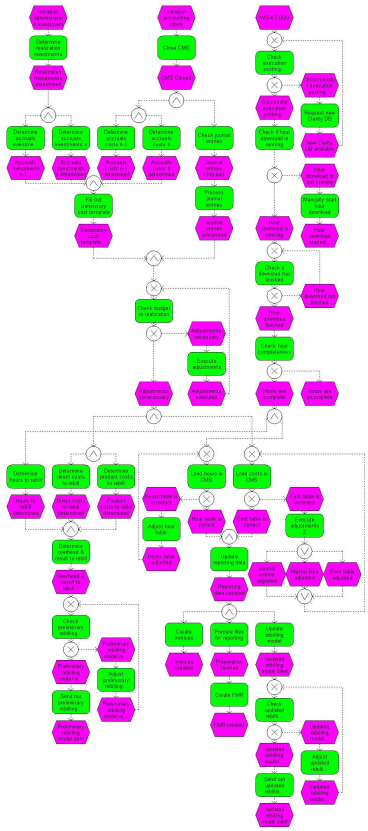

Fig. 2. Full composite case

Successful transformation, according to the Oracle BPA Suite, does not always indicate that the control flow in BPEL is the same as in EPC. Multiple start-events occasionally results in exclusion of a branch of the EPC diagram. While the transformation happened successfully, this is clearly not correct. Therefore, *successful* indicates that the Oracle BPA Suite performs the transformation, and *correct* indicates that the resulting BPEL retains the meaning of the input EPC diagram. While successfulness is apparent from the message given by the tool, we checked correctness by manually evaluating the resulting BPEL.

As opposed to the transformation of the patterns, the case from practice shows more limitations than theory predicted, as well as confirming those acquired from theory. Several things that should be possible, according to theory, are not possible for the transformation in the Oracle BPA Suite. This mainly includes diagrams with while-loops. These loops are possible, but often cause unexpected, incorrect results in practice, especially when contained by parallel branches. Multiple start-events may lead to similar issues. They are possible but, under certain circumstances, branches are missing. We encountered further limitations

with multiple end-events and unstructured ("spaghetti") structures. Of all these limitations, we consider unstructured structures and certain cases of multiple end-events related to "unstructuredness" as issues for transformation from EPC to BPEL in general. The other limitations we consider as issues for the particular implementation of the Oracle BPA Suite.

5 Guidelines

Based on the issues encountered during the application of the framework and during transformation in practice, we devised a set of guidelines. Together with a sequencing, specifying the order in which the guidelines should be applied, they provide methodological support to improve the feasibility of EPC to BPEL transformation. Table 5 shows the guidelines. The priority indicates the necessity to apply them for model transformation. We determined this by assessing whether it is possible to transform the model without the modification. For example, the tool does not transform the model at all if it contains an OR-connector, however using descriptive labels only improves communication.

Table 5. List of Guidelines

Nr.	Priority	Guideline
1	Must	Do not use the OR-connector.
2	Could	Avoid loops.
3	Must	If loops are necessary, then use only while-loops with a single exit.
4	Should	Avoid multiple start events.
5	Should	Converge multiple end events.
6	Could	Minimize the amount of arcs attached to a construct.
7	Must	Create structured models.
8	Should	Decompose processes containing problematic structures.
9	Should	Always follow XOR-splits with events.
10	Could	Alternate functions and events.
11	Would	Use clear, descriptive labels.

The guidelines are best to be applied in the order in which they are listed in the table. In this order, the first six guidelines handle issues for transformation that are relatively simple to solve. These issues are the OR-connector, loops, multiple start- and end-events, and constructs with a too high degree of incoming or outgoing arcs. Solving these issues also makes the next steps easier. The next two steps (guidelines 7 and 8) solve harder issues for transformation, such as unstructured models. These two guidelines should be used iteratively, if complex issues remain after a first iteration. The final three guidelines are mainly for communication. These steps require that the other steps completed already.

The above steps can be viewed as a normalization algorithm. It is a model transformation on its own, as it transforms one EPC model, which cannot transform correctly, to another EPC model, which can. Further details and (partial) automation of the algorithm remain an issue for future research.

By applying them to the composite case, we validated the guidelines. This resulted in a modified model, which the Oracle BPA Suite was then able to transform successfully. We checked the resulting BPEL specifications manually, and found them to be transformed correctly too.

6 Discussion and Related Work

Rigorous prior research provides the foundation for the framework. Both the BWW model and the workflow patterns proved their use individually. Combined, they provide us with a more powerful instrument. Not only does it allow us to establish correspondences between the constructs and relationships of the two languages, but it also facilitates the mapping of complex structures, and the assessment of whether the two languages are comparable in terms of expressive power. The predictions we made by using the framework show its correctness for the case of EPC to BPEL transformation. As these predictions resulting from theory match the results obtained during the empirical validation, the framework appears to be valid and lead to accurate results. As both parts of the framework, ontology and patterns, were used individually for other languages, literature confirms with high probability that the approach can be applied to assess transformation between any other process modeling languages too [8][12].

While the combination of workflow patterns and the BWW model provides a complete framework, the two components also overlap on some points. Some of the concepts of the BWW model inherently cover parts of the workflow patterns. In the case of control flow, these include the law concepts, which overlap with the splitting and joining patterns. If data patterns would be included, the overlap is even clearer, as data by nature are the thing and property concepts. Removing the overlap improves (the clarity of) the framework. This is necessary to use the framework as an evaluation tool to judge the quality of model transformations.

The choice for the BWW model to evaluate a single business process modeling language is debatable [13], as it has excess on several parts, and is incomplete on some others [14]. For evaluating the capabilities of transformation, this is less of an issue. The combination with workflow patterns solves the incompleteness, and the workflow patterns fill the gaps. Evaluation of both languages solves the excess, as only the comparison is important for the transformation.

While no other tool was found for direct EPC to BPEL transformation, transformation from EPC to BPEL is also indirectly possible in two steps; It is possible to go from EPC to another language, and then from that other language to BPEL. Theory development shows promise in this area, especially if using EPML (EPC Markup Language) or AML (ARIS Markup Language) as intermediary language [15]. Tools are available to do these individual steps.

To enable model transformation, a conceptual mapping from one language to the other is necessary. A conceptual mapping is also a way to deal with the lack of clarity revealed by the framework. A prerequisite for a conceptual mapping is the formalization of the languages. Van der Aalst [16] has done this for EPC by mapping it to Petri nets. Formal semantics are not yet complete

for BPEL [17]. However, one of the concepts that received adequate attention in the literature is the formalization of the control flow [18]. As this research focuses on the transformation of the control flow, it suffices that the control flow semantics are formalized. For the case of EPC to BPEL transformations, Ziemann and Mendling [19] propose a conceptual mapping. Several assumptions limit this mapping. These assumptions resemble the issues we encountered when applying the framework and carrying out the validation.

7 Conclusion and Future Research

This research has proved that automated transformation from EPC models to BPEL specifications is possible to a large extent. Conceptual mappings from EPC to BPEL exist. They indicate which concepts, constructs, and patterns can transform from EPC to BPEL. We encountered several problems that impose limitations on the structure of the models that can be transformed, according to the presented framework. The problems most difficult to solve have to do with the graph-structure of EPC versus the block-structure of BPEL. It makes transformation of arbitrary cycles and some other structures hard or even impossible. Besides this, several ambiguities exist when defining a mapping from EPC to BPEL. Lack of clarity is the cause of them. Therefore, it is hard to define a normative mapping.

Following the presented guidelines results in EPC models, which the Oracle BPA Suite can transform to BPEL specifications automatically. While some of the guidelines can be applied in general, several of the guidelines are specific for EPC modeling and models, which are meant for transformation to BPEL. This provides methodological support to improve the feasibility of EPC to BPEL transformation.

Much research on transformation from BPMN to BPEL also exists. A comparison of the two transformations based on the presented framework is now a possibility. The comparison may lead to a more founded choice for the use of BPMN or EPC over the other. It may also shed light on general issues of BPEL, which need improvement.

The main limitation for applying this research in practice is that the Oracle BPA Suite does not deliver executable code. In order to arrive at executable code, the modeler has to provide more than just the modeled control flow. For example a data model, and interaction with partners. Therefore, a question for future research is "What does transformation to executable code require from the input model?"

We based the framework on the basic workflow control patterns. Further research can also handle the other patterns in the same manner. This includes the advanced control flow patterns, as well as the data and resource patterns.

Acknowledgement. This work is part of the IOP GenCom U-Care project which is sponsored by the Dutch Ministry of Economic Affairs under contract IGC0816.

References

1. Miller, J., Mukerji, J.: MDA guide version 1.0.1. Technical Report doc. no. omg/2003-06-01, Object Management Group (2003)
2. Soley, R.: The OMG Staff Strategy Group: Model driven architecture. OMG white paper, Object Management Group (2000)
3. Bézivin, J.: In search of a basic principle for model driven engineering. Novatica Journal, Special Issue (March-April 2004)
4. Scheer, A.W., Schneider, K.: ARIS (Architecture of integrated Information Systems). Springer, Heidelberg (1992)
5. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S.: Business process execution language for web services, version 1.1. Standards proposal by BEA Systems, IBM Corporation, and Microsoft Corporation (2003)
6. Wand, Y., Weber, R.: An ontological model of an information system. IEEE Trans. Softw. Eng. 16(11), 1282–1292 (1990)
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), 5–51 (2003)
8. Rosemann, M., Recker, J., Indulska, M., Green, P.: A study of the evolution of the representational capabilities of process modeling grammars. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 447–461. Springer, Heidelberg (2006)
9. Mendling, J., Neumann, G., Nuttgens, M.: Towards workflow pattern support of Event-Driven process chains (EPC). In: Proc. of the 2nd Workshop XML4BPM, pp. 23–38 (2005)
10. Wohed, P., van der Aalst, W.M.P., Dumas, M., Hofstede, A.H.M.: Analysis of web services composition languages: The case of BPEL4WS. LNCS, pp. 200–215. Springer, Heidelberg (2003)
11. Mulyar, N.A.: Pattern-based evaluation of Oracle-BPEL (v.10.1.2). Technical report BPM-05-24, BPMcenter.org (2005)
12. Wohed, P., van der Aalst, W.M.P., Dumas, M., Hofstede, A.H.M., Russell, N.: Pattern-Based analysis of the Control-Flow perspective of UML activity diagrams. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 63–78. Springer, Heidelberg (2005)
13. Wyssusek, B.: On ontological foundations of conceptual modelling. Scandinavian J. Inf. Syst. 18(1), 63 (2006)
14. Gehlert, A., Esswein, W.: Toward a formal research framework for ontological analyses. Adv. Eng. Inform. 21(2), 119–131 (2007)
15. Mendling, J., Nuttgens, M.: Transformation of ARIS markup language to EPML. In: Proc. of the 3rd GI Workshop on EPCs, Luxembourg, pp. 27–38 (2004)
16. van der Aalst, W.M.P.: Formalization and verification of event-driven process chains. Information and Software Technology 41(10), 639–650 (1999)
17. Reichert, M.U., Rinderle, S.B.: On design principles for realizing adaptive service flows with BPEL (2006)
18. Ouyang, C., Verbeek, E., van der Aalst, W.M., Breutel, S., Dumas, M., ter Hofstede, A.H.: Formal semantics and analysis of control flow in WS-BPEL. Sci. Comput. Program. 67(2-3), 162–198 (2007)
19. Ziemann, J., Mendling, J.: EPC-Based modelling of BPEL processes: a pragmatic transformation approach. In: Proc. of MITIP 2005, Italy (2005)