



## COVER SHEET

---

Recker, Jan and Mendling, Jan (2006) On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages. In Latour, Thibaud and Petit, Michael, Eds. *Proceedings 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortiums*, pages pp. 521-532.

**Published by Namur University Press**

Accessed from:

[https://eprints.qut.edu.au/secure/00004637/01/ReckerMendling\\_emmsad\\_new.pdf](https://eprints.qut.edu.au/secure/00004637/01/ReckerMendling_emmsad_new.pdf)

# On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages

Jan Recker<sup>1</sup> and Jan Mendling<sup>2</sup>

<sup>1</sup> Queensland University of Technology  
126 Margaret Street, Brisbane QLD 4000, Australia  
j.recker@qut.edu.au

<sup>2</sup> Vienna University of Economics and Business Administration  
Augasse 2-6, 1090 Vienna, Austria  
jan.mendling@wu-wien.ac.at

**Abstract.** Business practice shows that, often, different process models are employed in the various phases of the Business Process Management life cycle, each providing a different paradigm for capturing and representing the business process domain. Recently, significant efforts have been made to overcome the disintegration of process models by providing complementary language standards for process design (BPMN) and execution (BPEL), based on the claim that these languages are semantically integrated. However, the conceptual mapping between both languages remains unclear, thus it is undecided whether any BPMN diagram can be transformed to BPEL. In this paper we argue that there is conceptual mismatch between BPMN and BPEL that needs to be identified in order to guide the language integration process semantically. In our analysis we take into account the various perspectives of the Business Process Management life cycle, in particular business and technical analyst perspectives. Our approach is generic and can also be utilized as a guiding framework for identifying conceptual mismatch between other business process modeling languages.

## 1 Introduction

In theory, Business Process Management (BPM) efforts follow a certain life cycle [1] that idealizes the phases of development and deployment of business processes into the stages of design, implementation, enactment, and evaluation. In principle, the design phase involves the development of conceptual process models from a *business analyst* perspective. As a second step, these models serve as input to *technical analysts* concerned with the development of technical process models, i.e., implementation models in the form of executable workflow specifications. These specifications then serve as templates for the enactment of process instances deployed on a workflow engine. Lastly, the execution of a process is monitored and evaluated to guide the revision and improvement of the process models as part of another iteration of the life cycle.

In business practice, however, the transition between these phases is often broken. As the process design and execution stages usually employ different modeling languages, this translation is prone to semantic ambiguities [1]. This may cause the loss

of design considerations within the execution models. We refer to such undesirable cases as *conceptual mismatch* between process modeling languages deployed in different phases of the BPM life cycle. Accordingly, the transition between the phases seems to be an important prerequisite to make the process management life cycle work, in particular, between business analyst and technical analyst models [1, 2].

The Business Process Modeling Notation (BPMN) [3] has been developed with the ambition to bridge the gap between business analyst and technical analyst by providing a standard visual notation for executable BPEL processes and by also specifying a formal mapping between BPMN to BPEL. In fact, the specification document states that "BPMN creates a standardized bridge for the gap between the business process design and process implementation." [3, p. 1]. However, as we will discuss during the course of this paper, the translation of BPMN to BPEL is far from trivial.

Specifically, we argue that mapping issues arise from conceptual mismatch between the two process modeling languages based on the assumption that the languages differ in expressive power, which in turn hinders the translation of models between these languages. Accordingly, *the first and foremost objective of this paper* is to discuss how conceptual mismatch between business analyst and technical analyst process models can be identified. Despite the focus on BPMN and BPEL we seek to deliver a generic solution that builds on established evaluation theories in the field of process modeling. Forthcoming from this discussion, as a second contribution of this paper we provide guidance for the translation of process models in the form of abstract transformation strategies that we deem promising for overcoming the identified mismatch.

We proceed as follows: in Section 2 we briefly introduce BPEL and BPMN. Also, in Section 2.3 we discuss existing studies on the correspondence between BPMN and BPEL, which show that there appears to be significant mismatch between the languages that hinders if not counteracts translation specifications. In Section 3 we then derive a multi-perspective method for identifying conceptual mismatch between business process modeling languages and apply it to BPMN and BPEL (Sections 3.1, 3.2, and 3.3). We close in Section 4 by drawing some conclusions from our work.

## 2 Background & Related Work

### 2.1 BPEL4WS

The Business Process Execution Language for Web Services [4], in its essence, is an extension of imperative programming languages with constructs specific to the BPM domain, in particular web service implementations. Version 1.1 of BPEL was released in 2003 and its version 2.0 is currently in process of standardization with OASIS. A BPEL process definition specifies the technical details of a workflow that offers a complex Web Service build from a set of elementary Web Services. The most important concepts of a BPEL process are variables, partnerLinkTypes, basic and structured activities, as well as handlers. *Variables* store process data and messages that are exchanged with Web Services. *PartnerLinkTypes* define the mutual required port types of a message exchange by declaring which partner acts according to which role defined in a partner link. *Basic Activities* specify the operations which are performed in a process.

These include Web Service operations like invoke, receive, or reply. There are further activities for assigning data values to variables (assign) or wait to halt the process for a certain time interval. *Structured Activities* are utilized for the definition of control flow, e.g., to specify concurrency of activities (using flow), alternative branches (e.g. via switch), or loops (while). Structured activities can be nested and links can be used to express synchronization constraints between activities. *Handlers* can be defined in order to respond to the occurrence of a fault, an event, or if a compensation has been triggered. For further details on BPEL refer to the specification [4].

## 2.2 BPMN

The nature of executable languages such as BPEL renders them less suited for direct use by humans to design, manage, and monitor the business processes that are enacted by process-aware information systems. In order to provide a standard visual notation for business processes defined in an executable process language, the Business Process Modeling Notation in its version 1.0 was first released in May 2004 and in February 2006 approved by OMG as a final adopted specification [3]. It has been the intention of the BPMN designers to develop a modeling technique that supports (a) typical process modeling activities for both *business* and *technical* analysts and (b) the straightforward mapping to executable workflow specifications in BPEL.

The complete BPMN specification defines thirty-eight distinct language constructs plus attributes, grouped into four basic categories of elements. *Flow Objects*, such as events, activities and gateways, are the most basic elements used to create Business Process Diagrams (BPDs). *Connecting Objects* are used to inter-connect Flow Objects through different sorts of arrows. *Swimlanes* are used to group activities into separate categories for different functional capabilities or responsibilities (e.g., different roles or organizational departments). *Artefacts* may be added to a diagram where deemed appropriate in order to display further related information such as processed data or other comments. For further details on BPMN refer to the specification [3].

## 2.3 Related Work on the Correspondence between BPMN and BPEL

The recent momentum on BPMN and BPEL in industry practice has triggered significant related research on these languages. In this section we focus on work that studies the correspondence between these two seemingly complementary languages.

Trying to support the claim that BPMN provides a visualization mechanism for BPEL, subsection 11 of the BPMN specification [3, pp. 137–204] presents a mapping between BPMN and BPEL; however, it is rather informally given in prose; a precise algorithm and a definition of required structural properties is missing. An example for how a mapping could work is given in [5], however, it is rather simple and the feasibility of such a mapping in the general case has not been demonstrated yet. It is also worthwhile noting that some available software such as Telelogic's System Architect ([www.telelogic.com/popkin/](http://www.telelogic.com/popkin/)) support the generation of BPEL code from BPMN diagrams, but only for a limited subset of BPMN.

From an academic perspective, recent work has led to the proposal of transformation strategies for process models, with focus often given to the case of BPMN and BPEL.

In [6] a general approach is presented to translate standard workflow models (refer to [7]) to BPEL by exploiting the BPEL construct 'event handler'. However, as the authors admit, this approach only holds for a core subset of BPMN and UML Activity Diagrams. Later, this approach has been adopted to the specific context of BPMN and BPEL [8]. Again, this approach relies on the discovery of process patterns in BPMN models, which are tried to be mapped onto BPEL structured activities. While this approach, too, is not yet at a stage where it holds for more advanced BPMN models, it is closely related to our forthcoming discussion as we specifically take into account the mismatch between BPMN and BPEL with respect to the representation of such control flow patterns. Another interesting approach is discussed in [9], where the authors discuss different strategies for translating graph-oriented models (like BPMN) to block-oriented specifications (like BPEL). These strategies have different perks and perils, nevertheless, we deem them a suitable starting point for devising concrete mappings based on an identification and understanding of the mismatch between the languages; hence, we will refer back to them later in this paper.

### 3 Conceptual Mismatch between BPMN and BPEL

As the discussion of related work reveals, existing transformation strategies falter when it comes to defining general mappings. We argue that conceptual mismatch exists that we assume to be a root cause for the translation problems. Our forthcoming discussion rests on two observations in this context:

- BPEL and BPMN come from different backgrounds (technical analyst versus address business analyst). Thus, they employ different paradigms for capturing relevant aspects of business processes, which in turn leads to the manifestation of conceptual mismatch with respect to the semantic expressiveness of these languages.
- BPEL and BPMN are employed in different stages of the BPM life cycle. Hence, the requirements of both stages need to be taken into consideration when identifying potential conceptual mismatch.

Based on these observations we argue that, specifically, the different BPM lifecycle perspectives need to be taken into consideration when devising a transformation between process models.

In particular, from a business analyst perspective, the transition between BPMN and BPEL must preserve the semantic information about the represented domain, viz., it should minimize if not avoid loss of semantic representation information. In this regard, Wand and Weber's work [10] is widely acknowledged as a framework of real-world domains concepts that modeling languages should be able to represent. In other words, a transition between languages should establish a high extent of *matching domain representation capabilities* between the two languages. From a technical analyst perspective, the underlying workflow execution engine determines the specification of processes. In this regard, Kiepuszewski et al. [7] state that control flow is a central aspect of a business process that needs to be sufficiently supported by any given language. Therefore, a transition between languages should establish a high extent of *matching*

*control flow support*. Beyond that, several authors in the field [6, 9] state that the underlying *process representation paradigm*, i.e., block-oriented vs. graph-oriented process representation, is another source of conceptual mismatch between process modeling languages. While both domain representation capabilities and control flow support permit statements about **whether** certain relevant aspects of a process can be expressed, the process representation paradigm influences **how** such aspects can be expressed.

Forthcoming from this argumentation, a method for identifying the conceptual mismatch between business and technical analyst process models must be able to identify all three types of conceptual mismatch. We will employ two established evaluation frameworks, namely *Representation Theory* [10, 11] (Section 3.1) for the specification of domain representation capability mismatch, and the *Workflow Patterns framework* [12] (Section 3.2) for the specification of control flow support mismatch. In addition to these established theories, Section 3.3 introduces a mismatch identification method based on a set of *transformation strategies* [9] that can potentially be used to translate process models into another.

The selection of the mentioned evaluation frameworks can be reasoned by their reasonable maturity, their rigorous development, their structured evaluation approach and foremost by their established track record in the field of process modeling. For overviews refer, for instance, to [13] and [14], respectively. In particular, as we seek to deliver a general contribution beyond the case of BPMN and BPEL, the high level of dissemination of these theories in the field of process modeling reasons our selection, as it allows for a wider uptake of our approach to cases of other process modeling languages that have previously been evaluated (e.g., BPML and WSCI).

### 3.1 Identifying Domain Representation Capability Mismatch

Evaluation using Representation Theory, exemplarily [10, 11], rests on the assumption that computerized information systems are essentially representations of real world systems and that IS models must hence contain the necessary representations of real world constructs including their properties and interactions. The BWW representation model (short: the BWW model) contains four clusters of constructs that are deemed necessary to faithfully provide *complete* and *clear* representations of the semantics of information systems domains: things including properties and types of things; states assumed by things; events and transformations occurring on things; and systems structured around things. For a more complete description refer, for instance, to [13].

Evaluation of modeling languages by means of Representation Theory seeks to reveal *construct deficit* within languages, which inhibits them from making statements about certain domain aspects [11]. We use this type of evaluation to investigate only those types of construct deficit in a particular language (e.g., BPMN) that another language (e.g., BPEL) is able to express and argue that this particular form of deficit constitutes a mismatch that in turn potentially impacts the translation of models between these languages. This means that if a more expressive process modeling language features a representation construct that is not supported in a less expressive process modeling language, then the translation of the modeled process to the less detailed language will be at cost of losing expressive power and thus, semantic information about the represented domain.

For the purpose of this paper, we draw on the individual analyzes of BPMN [15] and BPEL [16] and provide a cluster-oriented discussion of the differences between BPMN and BPEL in terms of their construct deficit (see Table 1). Representation Theory offers a systematic analytical method, overlap analysis [10, 16], for a thorough and more detailed evaluation of the completeness and overlap of domain representations in a combination of languages. We must consider such an evaluation out of scope for this paper; however, we see an interesting and important research challenge in such an overlap analysis in order to comprehensively clarify the type of mismatch between BPMN and BPEL.

**Table 1.** Support for the BWW model constructs in BPMN and BPEL. Adapted from [15, 16]

<b>BWW Construct</b>	<b>BPMN</b>	<b>BPEL</b>
<i>Things, including their Types and Properties</i>		
THING	++	-
PROPERTY	N/A	N/A
in general	++	+
in particular	-	-
hereditary	-	-
emergent	-	+
intrinsic	-	-
mutual: non-binding	-	-
mutual: binding	-	-
attributes	-	+
CLASS	++	+
KIND	+	-
<i>States assumed by Things</i>		
STATE	-	+
CONCEIVABLE STATE SPACE	-	-
LAWFUL STATE SPACE	-	-
STATE LAW	-	-
STABLE STATE	-	-
UNSTABLE STATE	-	-
HISTORY	-	-
<i>Events and Transformations occurring on Things</i>		
EVENT	++	++
CONCEIVABLE EVENT SPACE	-	-
LAWFUL EVENT SPACE	-	-
EXTERNAL EVENT	++	+
INTERNAL EVENT	++	++
WELL-DEFINED EVENT	++	+
POORLY-DEFINED EVENT	++	++
TRANSFORMATION	++	++
LAWFUL TRANSFORMATION	++	++
stability condition	++	+
corrective action	++	-
ACTS ON	+	+
COUPLING	+	+
<i>Systems structured around Things</i>		
SYSTEM	++	+
SYSTEM COMPOSITION	++	+
SYSTEM DECOMPOSITION	++	-
SYSTEM STRUCTURE	-	+
SYSTEM ENVIRONMENT	++	-
SUBSYSTEM	++	-
LEVEL STRUCTURE	++	-

Table 1 summarizes the findings from the analyzes in [15, 16]. In this table, a “+” indicates that the respective language provides one construct supporting the representa-

tion of the respective BWW model construct, a “++” indicates a support for the BWW model construct by more than one language construct and a “-” indicates a lack of support for the respective BWW model construct.

As can be seen from Table 1, there are a number of potential domain representation capability mismatches between BPMN and BPEL, indicated by varying support for the BWW model constructs. The following paragraphs discuss some of these discrepancies with respect to a potential translation of process models from BPMN to BPEL.

**Translation of things** A *thing* denotes the elementary notion in Representation Theory. The perceived world is constituted of things, either imaginary or real, that can be grouped into sets and species of things (*class* and *kind*, respectively). Table 1 reveals that BPMN is capable of representing things, classes, and kinds of things. However, BPEL only supports the representation of classes of things, viz., BPEL can only make semantic statements about groups of things but not specific instances. This means that object instances in a BPD, e.g., a specific organizational entity, a specific business partner or a specific application system, possibly need to be generalized to classes of instances, i.e., to a more aggregate level. On the other hand, the rather limited and general representation of properties of things in BPMN can be broken down into more specialized subtypes of properties in BPEL (see Table 1).

**Translation of states** A *state* of a thing is a vector of all the property values of a thing at a given point of time. Table 1 reveals that both BPMN and BPEL lack expressive power for modeling states assumed by things. While this finding may be problematic in general [13], it does not denote a area of concern with respect to translating BPMN diagrams to BPEL as both languages basically share the same (in-) capabilities with regards to explicit state representation.

**Translation of events and transformations** The occurrence of an *event* changes the state of a thing. A *transformation* is the mapping between two states of a thing. Table 1 reveals that BPMN has more expressive power than BPEL for the representation of events and transformations occurring on things. However, there seems to be a high extent of redundancy of BPMN in terms of transformation and event modeling [15], viz., BPMN offers many overlapping constructs and thus lacks orthogonality. This goes alongside with the finding that a translation to BPEL potentially needs to map certain dedicated event subtypes within BPMN to a single event type of BPEL (for instance, external events). Transformations, however, are more differentiated in BPEL, implying that transformation representations in BPMN potentially need to be annotated with further information or attributes to sufficiently specify a mapping to an appropriate BPEL construct.

**Translation of systems** Things can be composed to a *system*, which may have subsystems and interfaces to the environment of the system. Table 1 reveals that BPMN’s support for the modeling of systems structured around things excels the support provided by BPEL. Thus, a BPMN specification of the system to be developed, especially

the demarcation from its environment (*system environment*) and its disaggregation into subsystems (*system decomposition*), might not be unambiguously translatable into executable BPEL specifications and may thus require extra modeling and specification effort to avoid misinterpretations of the resulting BPEL models. In particular, the mapping of the BPMN Pool and Lane constructs to the BPEL Partner construct will require attention as the semantics of Pool and Lane seem to be more extensive than any BPEL counterpart.

### 3.2 Identifying Control Flow Support Mismatch

The development of the Workflow Patterns framework ([www.workflowpatterns.com](http://www.workflowpatterns.com)) was triggered by a bottom-up analysis and comparison of workflow management software. The goal was to bring insights into the expressive power of the underlying process execution languages. This work identified 20 *control flow patterns* [12] that specify atomic chunks of behavior capturing some specific process control requirements. The identified patterns span from simple to complex control flow scenarios and provide a taxonomy for the control flow perspective of workflows and processes. This taxonomy, in turn, has been widely used as a benchmark for analysis and comparison of process specification and execution languages.

Here we use the Workflow Patterns framework to draw conclusions as to the control flow support mismatch between BPMN and BPEL, drawing on the individual analyses of BPMN [14] and BPEL [17]. Table 2 summarizes the findings from both analyses. In this table, a “+” indicates a direct support for a pattern, a “+/-” indicates a partial support and a “-” indicates a lack of support.

**Table 2.** Support for the control flow patterns in BPMN and BPEL. Adapted from [14, 17]

Workflow Patterns	BPMN	BPEL	Workflow Patterns (ctd.)	BPMN	BPEL
<i>Basic Control Flow</i>			11. Implicit Termination	+	+
1. Sequence	+	+	<i>Multiple Instances Patterns</i>		
2. Parallel Split	+	+	12. MI without Synchronization	+	+
3. Synchronization	+	+	13. MI with a priori Design Time Knowledge	+	+
4. Exclusive Choice	+	+	14. MI with a priori Runtime Knowledge	+	-
5. Simple Merge	+	+	15. MI without a priori Runtime Knowledge	-	-
<i>Adv. Synchronization</i>			<i>State-Based Patterns</i>		
6. Multiple Choice	+	+	16. Deferred Choice	+	+
7. Synchronizing Merge	+/-	+	17. Interleaved Parallel Routing	+/-	+/-
8. Multiple Merge	+	-	18. Milestone	-	-
9. Discriminator	+	-	<i>Cancellation Patterns</i>		
<i>Structural Patterns</i>			19. Cancel Activity	+	+
10. Arbitrary Cycles	+	-	20. Cancel Case	+	+

Table 2 reveals a number of mismatches between BPMN and BPEL with regards to the support for various control flow concepts. The following paragraphs discuss some of these discrepancies, again in a cluster-oriented manner, with respect to a potential translation of process models from BPMN to BPEL.

**Translation of basic, state-based, and cancellation patterns** Table 2 reveals that BPMN and BPEL both support patterns 1–5 and 16–20 in the same manner. This means

that the representations of these control flow patterns in BPMN should be unambiguously translatable to BPEL. This finding supports the approach taken in [6, 8], in which mappings between BPMN and BPEL are defined based on their support for various control flow patterns.

**Translation of advanced synchronization patterns** Table 2 reveals that BPMN provides almost full support for patterns 6–9. BPEL, however, lacks support for multiple merges and discriminators. In particular, BPEL does not support the invocation of sub-processes [17], which, however, can be supported by BPMN. A specific problem is BPEL’s missing support for the discriminator pattern, i.e., points in the workflow process that wait for one of the incoming branches to complete before activating the subsequent activity. Hence, discriminators used in BPMN require considerable effort in translating them to statements that (a) are expressible in BPEL and (b) bear the same semantics as to the handling of control flow.

**Translation of structural patterns** BPEL does not support arbitrary cycles. The *While* activity can only capture structured cycles, i.e., loops with one entry point and one exit point. Again, this is a potential area of concern when translating arbitrary cycles from BPMN to BPEL code with equivalent control flow semantics.

**Translation of multiple instances patterns** Table 2 reveals that BPMN and BPEL both support patterns 12, 13 and 15 in the same manner but not pattern 14. This means that the BPMN representation of a workflow with multiple instances (where a number of instances of a given activity are initiated, and these instances are later synchronized, before proceeding with the rest of the process)<sup>3</sup> needs to be translated into a less expressive form in BPEL4WS and hence, some desired control flow support and design considerations for the modeled process are prone to getting lost.

### 3.3 Identifying Process Representation Paradigm Mismatch

We argue that a transformation of models must consider not only representational capabilities and control flow pattern support, but also the underlying process representation paradigm. In this context, there are essentially two paradigms to depict processes in a process modeling language: graph-oriented and block-oriented representation [6, 9]. BPMN follows a *graph-oriented* paradigm using arcs to define a partial order of activities and gateways to express split and join behavior. BPEL utilizes a *block-oriented* paradigm to express control flow via nested structured activities enhanced with some restricted graph concepts: in a BPEL process, arbitrary synchronization can be expressed with links as long as the links are acyclic. Cycles are only allowed if they are modeled as structured loops using the ‘While’ activity.

In [9], graph-based languages like BPMN and languages similar to BPEL are abstracted to so-called *process graphs* and *BPEL control flow*, respectively, in order to

---

<sup>3</sup> For pattern 14, the number of instances is known at some stage during run time, but before the initiation of the instances has started.

identify transformation strategies and constraints for the application of these strategies. In this context, a process graph is called *structured*, if split gateways match a join of the same type, and if loops are entered at one XOR join and exited at one XOR split. Furthermore, a process graph is *acyclic* if no node can be reached from itself. A BPEL process is *structured* if it does not include any links. Some transformation strategies are only applicable for process models that fulfil certain properties (see Table 3). For a formal definition of structured and cyclic process graphs as well as structured BPEL control flow refer to [9]. This reference also defines algorithms for each of the four transformation strategy that will be sketched in the following.

**Table 3.** Transformation strategies and applicable models

Transformation Strategies from BPMN to BPEL	Structured BPMN	Acyclic BPMN	All BPMN	Transformation Strategies from BPEL to BPMN	Structured BPEL	All BPEL
Element-Preservation	-	+	-	Flattening	+	+
Element-Minimization	-	+	-	Hierarchy-Preservation	+	-
Structure-Identification	+	-	-	Hierarchy-Maximization	+	+
Structure-Maximization	+	+	-			

**Transformation Strategies from BPMN to BPEL** All four transformation strategies (see Table 3) require that all cycles of the BPMN process model are structured loops with an entering XOR join and an exiting XOR split. The idea of the *Element-Preservation Strategy* is to map all BPMN elements to suitable BPEL elements nested in a BPEL flow and define control flow with links. Gateways are mapped to BPEL empty activities that serve as target and source for multiple input (join) or output links (split). The *Element-Minimization Strategy* takes the result of the Element-Preservation Strategy and replaces the empty activities with links containing transition conditions. The *Structure-Identification Strategy* works similar to the transformation proposed in the BPMN specification [3]. Structured blocks can be identified via graph reduction rules defined in [9]. This strategy is only applicable if all control flow can be mapped to BPEL structured activities. If not, the *Structure-Maximization Strategy* can be applied to derive a BPEL process with as many structured activities as possible nested in a flow for additional synchronization constraints. As Table 3 emphasizes, there is no strategy to generate BPEL from an arbitrary BPMN graph because BPEL does not permit modeling of arbitrary cycles.

**Transformation Strategies from BPEL to BPMN** The transformation from BPEL to BPMN imposes restrictions only for one strategy. The *Flattening Strategy* can be utilized to transform any BPEL control flow to BPMN. BPEL structured activities are flattened to gateways and arcs without any nesting. The *Hierarchy-Preservation Strategy* can be applied if the descriptive semantics of structured activities have to be preserved in the resulting BPMN. Each type of structured activity is mapped to a respective subprocess in BPMN. The *Hierarchy-Maximization Strategy* maps BPEL structured activities to sub-processes if there is no link crossing its borders. Table 3 shows that ar-

bitrary BPEL processes can be mapped to BPMN using the flattening or the hierarchy-maximization strategy.

## 4 Contributions & Conclusions

This paper discussed conceptual mismatch between BPMN and BPEL. We used a general multi-perspective method for identifying conceptual mismatch between process modeling languages employed in different stages of the BPM life cycle. In particular, our identification method applies established evaluation theories and innovative transformation strategies in order to identify potential mapping issues in the form of:

- *Domain Representation Capability Mismatch*: We showed how Representation Theory can be used to compare the representational capabilities of different process modeling languages in terms of divergences in the expressiveness of various aspects of domain semantics.
- *Control Flow Support Mismatch*: We showed how the Workflow Patterns Framework can be used to identify discrepancies between process modeling languages in terms of their support for various aspects of control flow.
- *Process Representation Paradigm Mismatch*: We showed how different representation paradigms underlying process modeling languages require different transformation strategies and we sketched out the implications of four different strategies.

Our analysis of the conceptual mismatch between BPMN and BPEL reveals that BPMN provides a much richer set of modeling constructs. A translation from technical BPEL to BPMN is therefore less a problem than in the opposite direction. On the other hand, BPMN is meant to be utilized as a visual notation for BPEL processes, but, as some of the BPMN constructs cannot be expressed in BPEL, a translation would imply a loss of information. For example, the missing BPEL support for a range of control flow patterns that BPMN can support may, in a translation from BPMN to BPEL, lead to execution semantics that were not intended in the conceptual model. As a consequence, either process modeling in BPMN has to be restricted to those constructs that have an equivalent in BPEL, or a remodeling might be necessary on the level of BPEL in order to handle untranslatable constructs. In order to make the business process life cycle work, it seems to be a better option to restrict BPMN rather than to extend BPEL, as extensions of the latter may not be supported by existing standard compliant process engines.

As to directions to further research, we perceive this work to be a starting point for a more detailed analysis of BPMN and BPEL (and other combinations of languages) using the approach presented. In particular, we see a need to comparatively assess the varying domain representation capabilities, and control flow support, of BPMN and BPEL in more detail, for example, by means of overlap analysis [10, 16].

## References

1. zur Muehlen, M., Rosemann, M.: Multi-Paradigm Process Management. In Grundspenkis, J., Kirikova, M., eds.: Proceedings of the CAiSE'04 Workshops. Vol. 2. Faculty of Computer

- Science and Information Technology, Riga Technical University, Riga, Latvia (2004) 169–175
2. Dreiling, A., Rosemann, M., van der Aalst, W.M.P.: From Conceptual Process Models to Running Workflows: A Holistic Approach for the Configuration of Enterprise Systems. In: Proceedings of the 9th Pacific Asia Conference on Information Systems, Bangkok, Thailand (2005) 363–376
  3. BPML.org, OMG: Business Process Modeling Notation Specification. Final Adopted Specification, <http://www.bpmn.org/>. (2006)
  4. Andrews, T., Curbera, F., Dholakia, H., Golland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services, Version 1.1. Specification, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems (2003)
  5. White, S.A.: Using BPMN to Model a BPEL Process. *BPTrends* **3** (2005) 1–18
  6. Ouyang, C., Dumas, M., Breutel, S., ter Hofstede, A.H.M.: Translating Standard Process Models to BPEL. In Pohl, K., ed.: 18th Conference on Advanced Information Systems Engineering, Luxembourg, Grand-Duchy of Luxembourg, Springer (2006) forthcoming
  7. Kiepuszewski, B., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of Control Flow in Workflows. *Acta Informatica* **39** (2003) 143–209
  8. Ouyang, C., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Translating BPMN to BPEL. BPM Center Report BPM-06-02, [BPMcenter.org](http://BPMcenter.org) (2006)
  9. Mendling, J., Lassen, K.B., Zdun, U.: Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages. In Lehner, F., Nösekabel, H., Kleinschmidt, P., eds.: Multikonferenz Wirtschaftsinformatik 2006. Band 2. GITO-Verlag, Berlin, Germany (2006) 297–312
  10. Wand, Y., Weber, R.: On the Deep Structure of Information Systems. *Information Systems Journal* **5** (1995) 203–223
  11. Wand, Y., Weber, R.: On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. *Journal of Information Systems* **3** (1993) 217–237
  12. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. *Distributed and Parallel Databases* **14** (2003) 5–51
  13. Rosemann, M., Recker, J., Indulska, M., Green, P.: A Study of the Evolution of the Representational Capabilities of Process Modeling Grammars. In Pohl, K., ed.: 18th Conference on Advanced Information Systems Engineering, Luxembourg, Grand-Duchy of Luxembourg, Springer (2006) forthcoming
  14. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives. BPM Center Report BPM-05-26, [www.BPMcenter.org](http://www.BPMcenter.org) (2005)
  15. Recker, J., Indulska, M., Rosemann, M., Green, P.: Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. In Campbell, B., Underwood, J., Bunker, D., eds.: 16th Australasian Conference on Information Systems. Australasian Chapter of the Association for Information Systems, Sydney, Australia (2005)
  16. Green, P., Rosemann, M., Indulska, M., Manning, C.: Candidate Interoperability Standards: An Ontological Overlap Analysis. Technical report, University of Queensland (2004)
  17. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Analysis of Web Services Composition Languages: The Case of BPEL4WS. In Song, I.Y., Liddle, S.W., Ling, T.W., Scheuermann, P., eds.: Conceptual Modeling - ER 2003. Volume 2813 of Lecture Notes in Computer Science. Springer, Chicago, Illinois (2003) 200–215