BUSINESS PROCESS MODEL TRANSFORMATION ISSUES
*The Top 7 Adversaries Encountered at Defining Model Transformations**

Marion Murzek
Women's Postgraduate College for Internet Technologies (WIT), Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria

Gerhard Kramler
Business Informatics Group (BIG), Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria

Keywords: Model Transformation, Business Process Modeling Languages.

Abstract: Not least due to the widespread use of meta modeling concepts, model transformation techniques have reached a certain level of maturity (Czarnecki and Helsen, 2006). Nevertheless, defining transformations in some application areas in our case business process modeling is still a challenge because current transformation languages provide general solutions but do not support issues specific to a distinct area. We aim at providing generic solutions for model transformation problems distinct to the area of horizontal business process model transformations. As a first step in this endeavor, this work reports on the most pressing problems encountered at defining business process model transformations.

1 INTRODUCTION

As companies discovered the benefits of Business Process Modeling (BPM), the use of Business Process (BP) models moved from a "luxury article" to an "everyday necessity" in the last years. Meanwhile many companies own thousands of models which describe their business. Since business changes over the years, e.g., business to business interoperability came up with new inventions in communication and companies merge with others, there arises a need to keep existing business models up-to-date and to synchronize or translate them into a contemporary BPM language. To facilitate these scenarios, a model transformation technique for BP models is needed.

At the moment much work is done in the area of model transformations (MTs). The main research interest lies on the technical aspects of MTs, for example transformation languages and verification of model transformations. Model transformation languages like ATL (Bézivin et al., 2005), QVT (OMG, a) or frameworks for general purpose programming languages like Java provide very good solutions for 1:1 and 1:n correspondences, thus they seem to be well suited for horizontal transformations as is the case when transforming BP models. Defining a transformation between any two BPM languages, however, is still a difficult task as several domain-specific problems remain to be solved.

BPM languages are used for a distinct purpose, namely the illustration processes in a company. So they all provide very similar concepts to the modeler. But the particular elements and attributes which are used to express a concept are more or less different in the BPM languages. Based on the assumption that there exist generic solutions for these often occurring problems, we are currently developing a framework which provides solutions for typical transformation problems applied on a generic meta model which contains all concepts of the participating BPM languages.

To the best of our knowledge there is no work on supporting the definition of transformation problems in a distinct domain. The contribution of this work is an overview of the problems one is confronted with when defining model transformations in the area of BPM. In a first step, we focus on the control flow part of BPM languages, as this is perhaps the most critical part of defining BP model transformations.

This reminder of this work is structured as follows. The next section puts our work into context

*This research has partly been funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

Murzek M. and Kramler G. (2007).
BUSINESS PROCESS MODEL TRANSFORMATION ISSUES - The Top 7 Adversaries Encountered at Defining Model Transformations.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 144-151
DOI: 10.5220/0002383201440151

of related work. Section 3 introduces the four different Business Process Modeling (BPM) languages we inspected, and in section 4 we discuss the seven most pressing model transformation issues. The conclusion in section 5 summarizes this work and gives an insight into ongoing work.

2 RELATED WORK

There has been and still is much research on the differences and equalities of Business Process Modeling and Workflow languages. They all have a different focus why and how they analyzed and compared some of the languages.

Wil van der Aalst et al. concentrated on Workflow Systems and developed the workflow (van der Aalst et al.,), resource (Russell et al., 2005) and data patterns (Russell et al., 2004). These patterns have been and are still used to build or analyze workflow and BPM languages concerning their expressiveness. For example Wohed et al. (Wohed et al., 2005), (Russell et al., 2006), (Wohed et al., 2006) and White (White, 2004) inspected the Business Process Management Notation (BPMN) and the UML 2.0 Activity Diagram (AD) to find out how far the WF patterns can be represented in these languages, with the aim of assessing the weaknesses and strengths of the BPMN and AD and its coverage of business process modeling problems. In (Mendling et al.,), EPCs have been analyzed with regard to the representation of the WF patterns. Additionally, an extended EPC, "yet another EPC" (yEPC) (Mendling et al., 2005) has been developed which covers all WF patterns.

Zachman et. al developed a framework (Zachman, 1987) which defines the system architecture of an enterprise. He uses the five basic questions "What, How, Where, Who, When, and Why" to achive a comprehensive view of the whole enterprise. This framework has also been used to evaluate BPM languages, for example UML 2.0 in (Fatolahi and Shams, 2006).

Korherr et al. (List and Korherr, 2006) compared seven different BPM languages. Their focus lay on the Business Process Context Perspective and they evaluated how expressive these BPM languages are for modeling business goals and metrics.

The recent work (Störrle, 2006) compares EPCs and UML 2.0 AD. The comparison is structured by syntax, semantic and pragmatic of the two languages. The central question which is left open in this work is if UML 2.0 AD will replace EPCs in the long run.

In our work we focus on what is important to keep care of, in case of transforming business process models between different BPM languages. So this work compares the syntax and semantics - by means of the meta model elements - of the concepts provided in the control flow part of four different BPM languages, with a specific focus on the differences between languages and how these differences affect model transformation.

3 BUSINESS PROCESS MODELING LANGUAGES

In the following the four BPM languages, ADONIS® Standard Modeling Language, Business Process Modeling Notation (BPMN), Event-driven Process Chains (EPC) and UML 2.0 Activity Diagrams (AD), which we have inspected concerning BP model transformation issues are described.

The concepts illustrated in the following four meta models capture the basic control flow part (as defined by Aalst et al. - basic control flow patterns) of each language. This small core part of each BPM language was taken to demonstrate the differences between these four languages concerning model transformation. Inspections of the remaining concepts of the BPM languages are out of scope of this work.

3.1 ADONIS Standard Modeling Language

The ADONIS® Standard Modeling Language (BOC, 2005) provides different kinds of model types which cover different business aspects. The BP model is used to model the business processes itself, i.e., its control flow aspect. Furthermore it is used to integrate the organizational and the information aspect. Since our work focuses on the control flow aspect, we concentrate only on the BP model (see Fig. 1).

ADONIS® is a graph-structured BPM language, which implies for example that there could be more than one end element in a process model. The integral model element is the *Activity*. A sequence of activities is modeled by means of the *Successor* which represents the control flow in the ADONIS® BP model. To depict a process call within a process, the element *Sub Process Call* is used. The *Control Objects* are used to model the flow of control.

The ADONIS® BP model provides no special element for modeling merges of alternative control flows. Furthermore, the decision element - *Decision* - does not distinguish between alternative split and multiple alternative split.

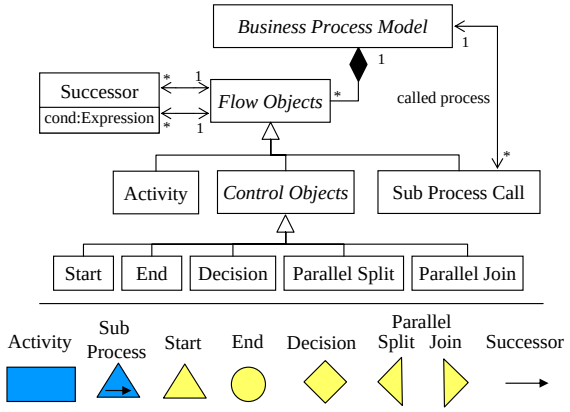

Figure 1: Part of the ADONIS® BP meta model and the concrete syntax.

3.2 UML 2.1 Activity Diagram

The UML 2.1 AD (OMG, b) is also a specification of the OMG. The meta model in Fig. 2 contains a very small part of the UML 2.1 language, the basic control flow elements which could be used for modeling BP models.

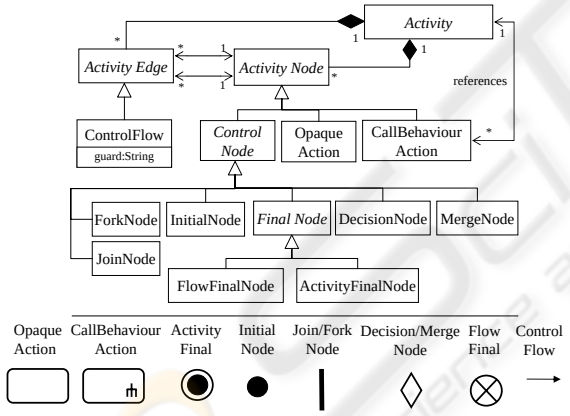

Figure 2: Part of the UML 2.1 AD meta model and the concrete syntax.

The Activity denotes different from ADONIS® and BPMN the whole BP model. The central element is the Opaque Action which is used to model the activities within a process. The Call Behavior Action represents the concept of a sub process call. Control Nodes are used to structure the process. There is a Fork Node and a Join Node provided to express a concurrent flow and a Decision Node and a Merge Node to model an alternative flow. The Initial Node marks the begin of a process model. The AD differs between two final nodes, the Flow Final Node (FFN) and the Activity Final Node (AFN). The FFN is used to mark the final of a distinct flow, that means if it is reached the remaining tokens in the process will proceed. Whereas the AFN marks the end of the whole process which means if it is reached the remaining tokens in the process are killed immediately. The only Activity Edge we considered here is the Control Flow which is used to connect the Activity Nodes to form the flow of control of a process.

3.3 Business Process Modeling Notation

The BPMN (OMG, 2006) was primarily introduced by the Business Process Management Initiative (BPMI.org) and is now a finally adopted specification by the Object Management Group (OMG). It has been designed as a graphical language to describe business process models and map them to the Business Process Execution Language (BPEL) for Web Services 1.1 (IBM,). The meta model illustrated in Fig. 3 shows the elements which are used to model the control flow aspect.

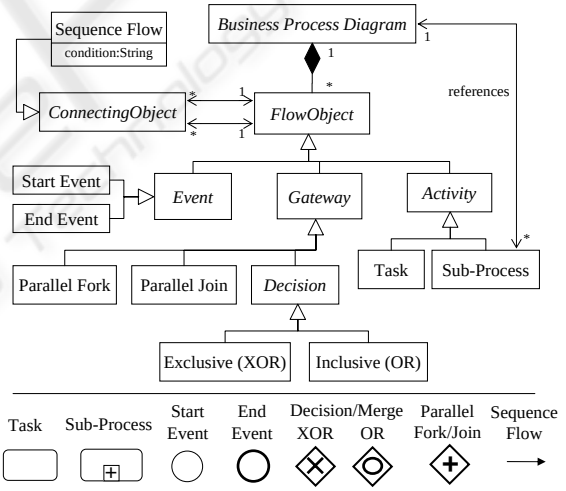

Figure 3: Part of the BPMN meta model and the concrete syntax.

In the BPMN the *Task* is the central element of the process model. The *Sub-Process* is used to model a reference to a sub process within a process model. *Gateways* are used to depict different flows (parallel, alternative etc.) of the process. To model the begin and the end of a process model *Events* are used. The *Core Elements* can be connected by the Sequence Flow to form the process model.

3.4 Event Driven Process Chains

Event Driven Process Chains (EPCs) (Keller et al.,) have been introduced by Keller, Nüttgens and Scheer in 1992. EPCs are basically used to model processes. We focus on the main elements which are used to model the integral and control flow aspect of a BPM (see Fig. 4).

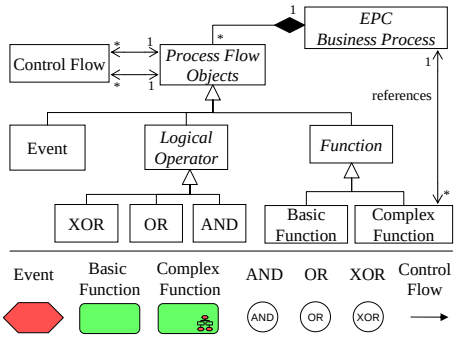

Figure 4: Part of the EPCs meta model and the concrete syntax.

The *Function* describes an activity. It creates and changes Information Objects within a certain time. The *Event* represents a BP state and is related to a point in time, it could be seen as passive element compared to the function as an active element (compare (Mendling and Nüttgens, 2003)). To model a sub process call the *Complex Function* is used. The *Logical Operators* elements are used to structure the proceed of the BP model.

EPCs do not provide a specific element to indicate the begin and the end of a BP model, therefor the *Event* is used. Event elements are not allowed to be in front of an OR and XOR element. Function and event elements must alternate in the proceed of the BP model and are connected via the *Control Flow*. Another restriction in EPCs is that branches parallel as well as alternative must be split and merged with the same kind of Logial Operator.

4 MODEL TRANSFORMATION ISSUES

Although we have taken the control flow aspect which is a small part of each BPM language and their descriptions may sound very similar, there are a lot of differences concerning the representation of concepts.

Before introducing the transformation issues themselves we first have to consider how differences between BPM languages should be handled by a model transformation. Model transformations aim at *preserving the semantics* of a model. Unfortunately this high ambition can not always be achieved. If this is not possible, for example if there is missing a corresponding semantic concept, then the next stage of requirements is to *avoid loss of information* during the transformation. This can be obtained by annotating target model elements, for example inserting information into a annotation attribute of an element, for reasons of documentation or back-transformation. If this is also not possible then another alternative is to ask the user - *provide user interaction* - to decide what should happen in a distinct case. It can also happen that the target model must be *semantically enriched*, because of mandatory elements in the target meta model. In this case new information based on the existing information has to be created.

The following transformation issues have been encountered during the definition of transformations between the BPM languages introduced in Section 3. Each issue is described by its name, the participating elements and the problem description. Different kinds of examples are provided to establish understanding of the transformation problem.

To illustrate some of the solutions, ATL has been used because it is one of the most well-known transformation languages. The solutions would look different in other transformation languages such as QVT but the problems would remain the same.

4.1 Decision (Un)Ambiguity

In ADONIS® there is only one element (Decision) to express inclusive and exclusive alternatives. In ADs an exclusive split is modeled by a Decision Node and the inclusive split by a Fork Node with guard conditions on the departing Control Flows. BPMN and EPCs provide one distinct element for each concept Inclusive(OR)/Exclusive(XOR) in BPMN and OR/XOR in EPCs.

The transformation problem in case of transforming ADONIS® models into EPC models is to decide if a Decision is used as an exclusive or an inclusive split. The distinction can be made depending on the conditions on the outgoing edges of the Decision in ADONIS® . The main problem is to decide whether the value of the cond attribute on the departing Successors of a Decision are exclusive or inclusive.

In the following the transformation problem is illustrated by means of an example transformation code in ATL where a ADONIS® Decision should be transformed to an EPC OR or XOR:

```
rule decision2xor{
from:
    d: adonis!decision
```

```
    ((d.outgoing.conditions).ifExclusive())
to:
    x: epc!xor
}

rule decision2or{
from:
    d: adonis!decision
    (not (d.outgoing.conditions).ifExclusive())
to:
    x: epc!or
}
```

The function ifExclusive() decides whether the conditions are exclusive or not. Unfortunately there is no straightforward implementation because there are various kinds of exclusive conditions. For example: True/False, 0/1, X/not X and antonyms in general.

One possibility could be to check the conditions as words via an encyclopedia that contains antonyms, for example WordNet (Miller et al., 1998).

4.2 Invisible Merger

Two of the BPM languages we inspected offer the possibility to model a merger of a Alternative split implicitly, i.e. without using a distinct element. In ADONIS® the mergers of flows which have been split by a Decision are implicitly modeled, there is no explicit merge element provided. In BPMN the choice for modeling an explicit exclusive(XOR) merger is left to the user.

In this case the transformation problem is to decide on which position in the process model an implicit (exclusive or inclusive) merge is used (for example see Fig. 5). On this position an XOR or OR merge must be inserted in the target model.

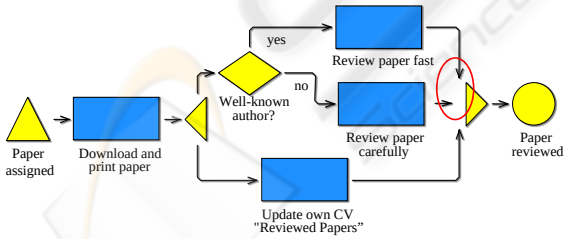

Figure 5: Invisible Merge in ADONIS® .

The central questions to solve this transformation problem are ”How to find invisible mergers?” and ”How to find their corresponding split elements?”.

One suggestion to solve this problem is an algorithm which detects the positions in the model where more than one Successor leads into an element. Then the algorithm must follow all the control flow backwards until it finds a common split element. This is easy in case of block-structured models. But if we are faced with a BPM language which allows the modeling of graph-structured models as it is the case in our four languages, then this algorithm provides only a partial solution. A more powerful solution approach for this can be found in (Murzek et al., 2006).

4.3 Mandatory Events

In EPCs Events are mandatory. It is required that before and after a function there must be an Event, i.e. that Events and Functions have to alternate during the flow of the process model. A further restriction in EPCs is, that it is not allowed to model an Event which is followed by an XOR or OR element (see semantics of EPCs in (Keller et al.,) and (Mendling and Nüttgens, 2003)) There is no semantically corresponding element in ADONIS® . In BPMN (Intermediate Events) as well as in AD (Receive and SendSignal) there are events specified. But the concept of events in BPMN and AD differs from the concept of the Events in EPCs. In EPCs the Events are business states, meaning that a process is in a distinct state, furthermore the events are used to mark the begin and the end of a process. In BPMN and AD events represent external triggers which could be used to model external influences or interaction with the process. As the events in BPMN and AD could be assigned to an interaction aspect of BPM they have not been taken under consideration in this work.

The transformation problem is to decide where to insert events in the process model and to make sure that the target model is in a valid state after the transformation.

A possible straightforward solution is to convert for example an Activity in ADONIS® into an Event and a Function connected by a Control Flow. This would avoid having any Events followed by an OR or XOR. Nevertheless it could result in invalid EPC process models. An example for this problem is shown by the transformation of the ADONIS® model fragment in Fig. 6 into the EPC model fragment in Fig. 7.

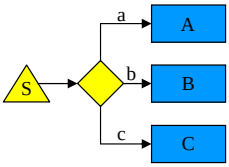

Figure 6: Start with following Decision in ADONIS® .

The model in Fig. 7 violates the syntax of EPC

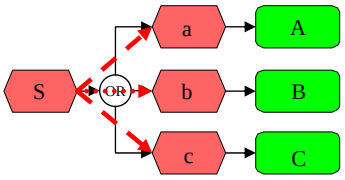

Figure 7: Invalid EPC model - alternating Event/Function condition violated.

models in two ways. First there is an Event followed by an OR split and second the same Event is followed by three Events, but it is mandatory that Events and Functions alternate during the flow of the process model. So this solution is only partially satisfying and a review by the modeler is necessary.

4.4 Different Start Objects

In AD, BPMN and EPCs it is possible to define multiple start objects for one process model. But the semantics of multiple start objects is different. In AD *all* Initial Nodes will be activated if the process starts. This means that the Initial Nodes themselves mark the beginning of a process but do not depict any events which trigger the process. By contrast the activation of *one* of the Start Events in EPCs or BPMN is sufficient to trigger the begin of the process. However the semantics of multiple Start Events in BPMN changes if all of the outgoing flows of these Start Events are leading to the same Activity which implies a Parallel Join in front of the Activity. Then the process starts, if all of the Start Events are activated. In ADONIS® only one common start element is allowed.

This leads to two different semantical problem areas:

1. The difference in the semantics of start objects in AD compared to the start objects in other BPM languages and
2. The different semantics of multiple start objects.

In case of multiple start events with an exclusive alternative semantics in BPMN (see Fig. 8(a)) it is possible to create one common Initial Node in the AD target model with a proceeding Decision Node (see Fig. 8(b)). The Decision should be annotated with a condition that checks for the trigger of the process. This workaround preserves all of the information and as much of the BPMN semantics as possible. When transforming an Activity in AD back to BPMN this workaround needs to be taken into account such that the original model can be reconstructed.

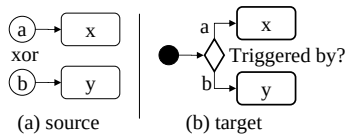

Figure 8: Possible solution in AD for multiple start objects in BPMN.

The model transformation cycle in Fig. 9 shows another problem resulting from a workaround necessary when transforming an AD model part with two Initial Nodes (a) into a BPMN model part (b) and back again (c). The model in Fig. 9(a) is semantically equivalent but syntactically different from the model part in Fig. 9(b).

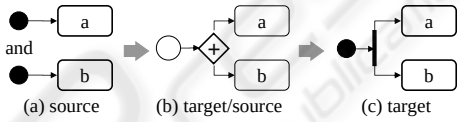

Figure 9: Different model structure AD (c) at back transformation from BPMN (b).

4.5 Split/Merge Unambiguity

This problem has been considered in our work because at first sight the provided Logical Operators (LO) in all four BPM languages seemed to be equivalent but the solution is not a trivial 1:1 transformation.

In EPCs there is only one element used to depict splits and mergers for a distinct LO. In BPMN it is the same, except for the parallel fork and join there are two different elements.

The problem in case of models of these BPM languages is to decide whether a LO is the begin or the end of a branch of the control flow in a process model when transforming to a language which requires to make that distinction.

Once more the Control Flow elements - leading to or departing from a LO - have to be taken into consideration. Three different cases can be distinguished:

1. LO with one incoming and more than one outgoing Control Flows,
2. LO with more than one incoming and one outgoing Control Flow and
3. LO with more than one incoming and outgoing Control Flows.

In case of (1) we have to transform it into a split object. In case of (2) the LO has to be transformed to a join or merge object. In case of (3) the AND Operator

has to be split into three elements: one split element (Parallel Split in ADONIS® , Parallel Fork in BPMN and Fork Node in AD) and one join/fork element (Parallel Join) connected by a control flow edge. If the LO is an XOR then the transformation leads to one Decision element with more than one incoming and outgoing Successors in case of ADONIS® and BPMN and to three objects, Merge Node, Control Flow and Decision Node in AD.

The following ATL transformation code example defines the above solutions for a transformation from EPC to ADONIS® :

```
rule and2parallelSplit{
from  a: epc!AND (a.incoming->size() = 1 and
                  a.outgoing->size() > 1)
to    ps: adonis!ParallelSplit(...) }

rule and2parallelJoin{
from a: epc!AND (a.incoming->size() > 1 and
                 a.outgoing->size() = 1)
to   pj: adonis!ParallelJoin(...) }

rule and2parallelJoinSplit{
from a: epc!AND (a.incoming->size() > 1 and
                 a.outgoing->size() > 1)
to  psp: adonis!ParallelSplit(...),
    suc: adonis!Successor(...),
    pjo: adonis!ParallelJoin(...)
}
```

The transformation in the reverse direction may be optimized to reestablish the original model. In this case a corresponding transformation to the ”and2parallelJoinSplit” rule, which merges a Parallel Split, a Successor and a Parallel Join to an AND element, has to be found. As a rule in ATL can only match one element in the ”from” part this will require a more complex solution algorithm in ATL.

4.6 Join Specification Problem

This problem can only be observed in ADs. Contrary to ADONIS® , BPMN and EPCs which provide a possibility to express inclusive alternative merge in ADs there is no semantically correct way to depict such mergers. So the problem is how to express a inclusive alternative merge in ADs.

In (White, 2004) a possible but as per (Wohed et al., 2005) incorrect (in terms of the semantics of an executable workflow model) attempt to cover an inclusive alternative merge in ADs has been made (see Fig. 10).

In this solution the problem of the transformation definition lays on the derivation of the ”Join Spec” expression, which could be derived from the departing edges of the corresponding inclusive alternative split. The ATL transformation code for this example is to comprehensive to illustrate it here.

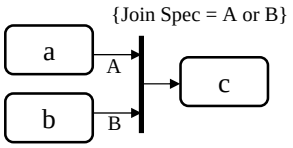

Figure 10: Incorrect attempt to capture a synchronising merge - Figure taken from (Wohed et al., 2005).

Another alternative would be to ask the user to decide how to transform this problem.

BP models are not designed for executing them directly. They are used to support the understanding of business processes and furthermore they can provide an adequate documentation of the business processes in a company. Therefore the solution in Fig. 10 can be satisfying for business people.

4.7 Different Final Nodes

AD provide the possibility to model different kinds of end nodes, the Activity Final Node (AFN) and the Flow Final Node (FFN). The semantics of these nodes is as follows: When a FFN is reached the remaining tokens in a model will succeed. In contrary if the AFN is reached all remaining tokens in the processed will be ”killed” immediately. In the other BPM languages the end element is used corresponding to the FFN. There is no semantically equivalent to the AFN.

The problem regarding the FinalNodes in AD is to decide how to transform the AFN. A FFN can be transformed into an end element in either of the three BPM languages. But if an AFN is transformed into an End element in ADONIS® , BPMN or EPC the semantics of the process model is different.

There are two supposable solutions: First an annotation of the end object in the target language if an attribute for annotations is provided. Second the creation of an additional Activity, Function or Task which is called ”Terminate process” and which is integrated in front of the end object.

5 CONCLUSION

In this work there have be introduced transformation problems observed at defining model transformations between four BPM languages. Although we only focused on the transformation of the basic control flow aspect of four BPM languages we encountered seven non-trivial problems. As they are similar between different BPM languages we suggest to provide general solutions for often arising problems. This will reduce the effort of defining BP model transformations and

the number of times that some of these wheels have to be reinvented.

The transformation issues illustrated can be categorized into two different areas:

1. Transformation requirements which raise problems at the implementation level, they have *technical complexity*, for example the *Decision (Un)Ambiguity* and the *Invisible Merger* problem.
2. Transformation requirements which raise problems concerning their feasibility are situated at the application level, they contain *application complexity*, for example the *Different Final Nodes* and the *Join Specification Problem*.

The technical complexity affects model transformation languages themselves. General transformation languages like ATL (Bézivin et al., 2005), QVT (OMG, a) or programming languages like Java provide very comprehensive but unspecific possibilities to define such transformations. Unspecific in the sense that they provide a language to define "everything" but nothing in special.

The problems on the application level are of conceptual nature and so they could hardly be solved with technical inventions. But there can be provided second level mechanisms like user interaction which does not solve the problem, but supports the user at transformation. The "adversaries" on the technical level can be attacked, for example by providing reusable general solutions for distinct problems in the area of BP model transformation.

REFERENCES

Bézivin, J., Jouault, F., and Touzet, D. (2005). An Introduction to the ATLAS Model Management Architecture. Technical report, LINA.

BOC (2005). ADONIS 3.7 - User Manual III: ADONIS Standard Modeling Method. BOC Ltd.

Czarnecki, K. and Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, pages 621–645.

Fatolahi, A. and Shams, F. (2006). An investigation into applying uml to the zachman framework. *Information Systems Frontiers*.

IBM. *Business Process Execution Language for Web Services version 1.1*. IBM.

Keller, G., Nüttgens, M., and Scheer, A.-W. Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". Technical report, Institut für Wirtschaftsinformatik Universität Saarbrücken.

List, B. and Korherr, B. (2006). An evaluation of conceptual business process modelling languages. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*.

Mendling, J., Neumann, G., and Nüttgens, M. Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). In *Proceedings of the 2nd GI Workshop XML4BPM at the 11th GI Conference BTW 2005*.

Mendling, J., Neumann, G., and Nüttgens, M. (2005). Yet another event-driven process chain.

Mendling, J. and Nüttgens, M. (2003). EPC Modelling based on Implicit Arc Types. In *Proceedings of the 2nd International Conference on Information Systems Technology and its Applications (ISTA)*.

Miller, G. A., Fellbaum, C., and Tengi, R. (1998). Wordnet: A lexical database for the english language.

Murzek, M., Kramler, G., and Michlmayr, E. (2006). Structural patterns for the transformation of business process models. *EDOCW'06*, pages 18–28.

OMG. *MOF QVT Final Adopted Specification*. Object Management Group.

OMG. *UML 2.1 Superstructure Specification*. Object Management Group.

OMG (2006). *Business Process Modeling Notation Specification*. Object Management Group, http://www.bpmn.org/.

Russell, N., ter Hofstede, A. H. M., Edmond, D., and van der Aalst, W. M. P. (2004). Workflow Data Patterns. Technical report, Queensland University of Technology.

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., and Edmond, D. (2005). Workflow resource patterns: Identification, representation and tool support. In *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE05)*.

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., and Wohed, P. (2006). On the suitability of uml 2.0 activity diagrams for business process modelling. In *APCCM '06: Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling*.

Störrle, H. (2006). A Comparison of (e)EPC and UML 2 Activity Diagrams. In *EPK 2006 Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*.

van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, A. P. Workflow Patterns. *Distributed and Parallel Databases*.

White, S. A. (2004). Process Modeling Notations and Workflow Patterns. *BPTrends*.

Wohed, P., van der Aalst, W. M., Dumas, M., ter Hofstede, A. H., and Russell, N. (2005). Pattern-based Analysis of UML Activity Diagrams. In *Proceedings of the 25th International Conference on Conceptual Modeling (ER'2005)*.

Wohed, P., van der Aalst, W. M., Dumas, M., ter Hofstede, A. H., and Russell, N. (2006). On the Suitability of BPMN for Business Process Modelling. In *Proceedings 4th International Conference on Business Process Management*.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM System Journal*.