# HORIZONTAL BUSINESS PROCESS MODEL INTEROPERABILITY USING MODEL TRANSFORMATION

Harald Kühn
BOC Information Systems GmbH
Product Development
Rabensteig 2
A-1010 Vienna
Austria
Tel.: ++43-1-513 27 36 10
Fax: ++43-1-513 27 36 28
E-Mail: harald.kuehn@boc-eu.com

Marion Murzek
BOC Information Systems GmbH
Tool Development
Rabensteig 2
A-1010 Vienna
Austria
Tel.: ++43-1-513 27 36 10
Fax: ++43-1-513 27 36 28
E-Mail: marion.murzek@boc-eu.com

Franz Bayer
BOC Information Systems GmbH
Service Development
Rabensteig 2
A-1010 Vienna
Austria
Tel.: ++43-1-513 27 36 10
Fax: ++43-1-513 27 36 28
E-Mail: franz.bayer@boc-eu.com

## 1. INTRODUCTION

The complexity in developing enterprise-spanning applications is continually growing. Amongst others, a vital field of delivering technical concepts and technologies for integrating heterogeneous applications and components to support interoperability in inter-organisational business processes is the area of Enterprise Application Integration (EAI). The main idea of EAI is to provide technical solutions to integrate workflows and heterogeneous parts of enterprise applications in a continuous business application [6]. A common characteristic of all EAI approaches is their focus on *technical and runtime aspects* of integration, i.e. on the level of target systems and execution environments.

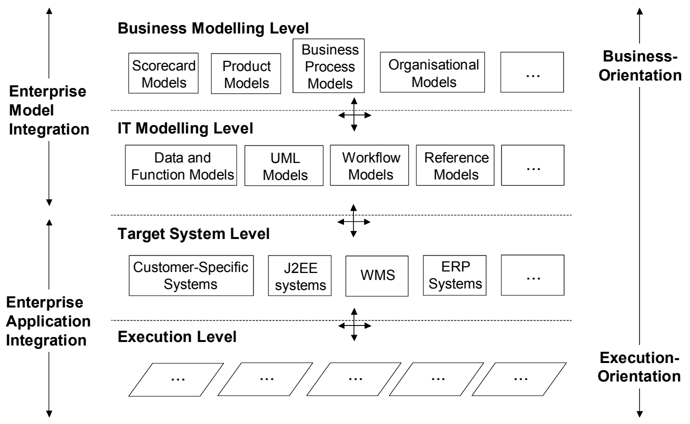

Figure 1: Levels in Modelling Enterprise Systems

From our project experiences in developing and modelling large enterprise applications, it is necessary to integrate applications on the *business and conceptual level* as well. Nevertheless, standardisations on modelling languages and model exchange formats (incl. their industrial implementation) still can rarely be found. Almost the only, but prominent, example is the UML and XMI for modelling object-oriented systems [10, 11]. Other examples from industrial research area are BPML/BPMN [2, 3] and UEML [15]. In addition, the OMG community started to establish a relatively new vision with the Meta Object Facility (MOF) and the Model Driven Architecture (MDA) to improve productivity in software development, applying object-orientation, meta level concepts and modelling [12, 14]. Because of today’s diversity of models and heterogeneous modelling languages for developing enterprise applications, we apply the *Enterprise Model Integration (EMI)* approach [8]. This approach is based on object-oriented metamodelling concepts to describe context-specific, integrated modelling languages and on model transformation concepts for model exchange.

The remainder of the paper is organised as follows: In chapter 2 the BOC Model Transformer (BMT), which is part of the EMI approach, is described according to its requirements, its architecture, and its rule file design. Chapter 3 shows a case study applying BMT in the context of business process model interoperability. The paper concludes with a summary and a position statement.

## 2. BMT: BOC MODEL TRANSFORMER

In the following, the requirements, the architecture and the rule file design are described. Some typical application areas of BMT are:

- Model transformations from modelling language A to modelling language B.
- Model exchange between different instances of the metamodelling platform ADONIS.
- Model exchange between other modelling environments and ADONIS.

### 2.1 Transformation requirements

As a result of long lasting manual model transformations the following requirements for automated model transformation arise:

- It must be possible to navigate in the source- and target models during transformation.
- This leads to condition two, it must be possible to selectively search for elements, using expressions and constrains.
- Copying, deconstructing, merging, filtering and computing of values must be possible.
- Creating elements like instances, relations, attributes, etc. in the target models.
- Dependencies between elements must be preserved after the transformation. Therefore, a history which stores transferred elements and their relations is required. Furthermore this history must be accessible during transformation.
- Functions concerning arranging objects or sorting attributes should be executed after all models are transferred.

Beside these functional requirements there are additional usability requirements [9].

### 2.2 Architecture of BMT

The core element of the BMT is the rules file. It contains all instructions to fulfil the requirements for transforming models. Figure 2 shows the interactions in BMT during model transformation.

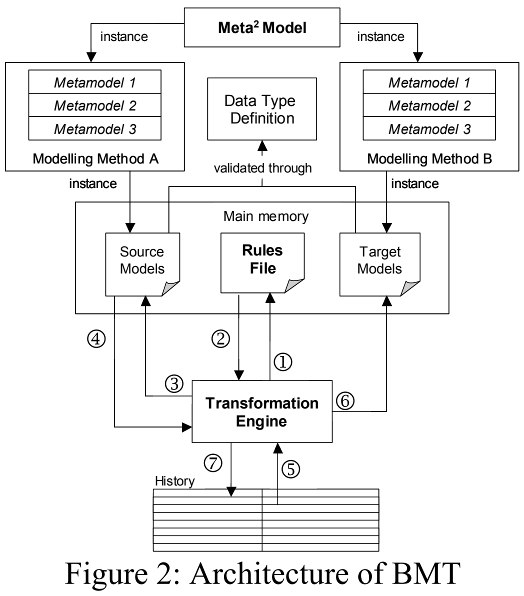

Figure 2: Architecture of BMT

1. The Transformation Engine (TE) reads a part of the rules file, containing an instruction tag (e.g. XML tag).
2. The information within the matched tag is being processed in the TE.

3. If it was a navigation tag, the TE searches through the source models until it finds one of the demanded elements. In case of a namevaluemap- or rulescontainer tag (defined XML tags) the TE stores this information in program variables or containers (see table 1).
4. The selected elements will now be transferred to the TE. If no elements match 5-7 will be skipped and the process continues with step 1.
5. In case elements have already been created, it is necessary to check for existing references in the history.
6. Now the TE navigates either explicit or implicit to the predetermined position and creates the elements as instructed (within the scope of the instruction tag – step 1).
7. In the last step of the pass the new elements and their references to the source elements have to be registered in the history.

**2.3 Design of Rules File of BMT**

This section describes the transformer elements used in the rules file. There are six main elements: conditions, definitions, functions, navigations, rules and postprocessor actions.

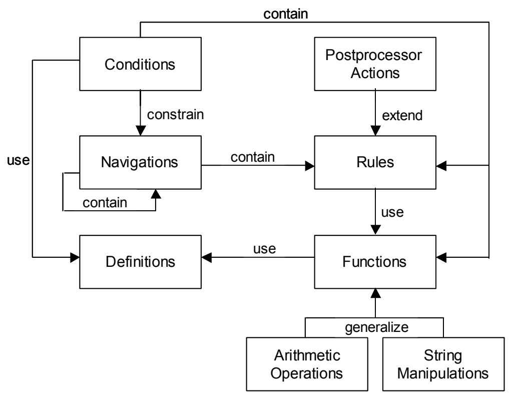

Figure 3: Design of BMT rules file

- *Conditions* are used to constrain the selection of elements within navigations. They may contain functions and reuse definitions.
- *Definitions* are used to save information about recurrent elements, using variables annotated with a unique name. In the course of transformation this information can be recalled with the unique name given to the variable. Definitions are used by conditions and functions.
- *Functions* are used to manipulate strings or numbers. Moreover, there are functions converting time, shift or invert bendpoints. Functions also reuse definitions and are used by the rules.
- *Navigations* offer the possibility to navigate within the source models and target models. For every model element there is an appropriate navigation element, such as `navigate_modeltype` for “model”, `navigate_instance` for “instance”, `navigate_attribute` for “attribute” etc. The navigations are constraint by conditions. They may contain further navigation elements and rules.
- *Rules* describe where and how elements should be created. There are two sorts of rules: one for copying and one for creating elements. Rules use functions and they are contained in navigations and may be contained in conditions.
- *Postprocessor Actions* are performed at the end of the transformation. These actions include sorting of attribute values, arranging objects within a model or shifting objects due to changes in size of other objects in the same model. Postprocessor actions extend the rules.

## 2.4 Related Work

Recently, a large number of model transformation approaches were proposed. Additionally, the OMG issued in 2002 a Request for Proposal (RFP) on Query/Views/Transformations to establish standardized model views, model transformations and model synchronizations. This RFP further pushed the number of model transformation approaches [5]. [4] provides an extensive overview of model transformation approaches and categorizes into model-to-code approaches and model-to-model approaches. BMT uses a model-to-model approach.

## 3. CASE STUDY: BUSINESS PROCESS MODEL EXCHANGE

### 2.1 Levels in Business Process Modelling

In modelling and implementing business processes at least three levels can be distinguished:

- *Business process model*: The business process model describes enterprise processes from the business point of view. A business process model includes manual activities, business descriptions and business performance parameters such as to-be execution times, to-be cycle times etc.
- *Workflow model ("macro flow")*: The workflow model describes the enterprise processes from the technical point of view. A workflow model includes technical activities (no manual activities!), specification of called programs and modules, necessary data units, rules concerning access rights etc.
- *Orchestration model ("micro flow")*: The orchestration model describes the flow of control and the runtime information within a "chunk" of executable code, e.g. within a program, a module, web service etc.

The following section focuses on the business process model level.

### 2.2 Business Process Model Transformation and Horizontal Model Exchange

In the following, the BMT is applied for business process modelling language transformation and XML-based model exchange. The model exchange is between a business process modelling tool supporting the event-driven process chain modelling language (EPC) [7] and the metamodelling platform ADONIS supporting the ADONIS standard process modelling language [1]. The model exchange is called horizontal, because the model exchange is on the same level of abstraction.

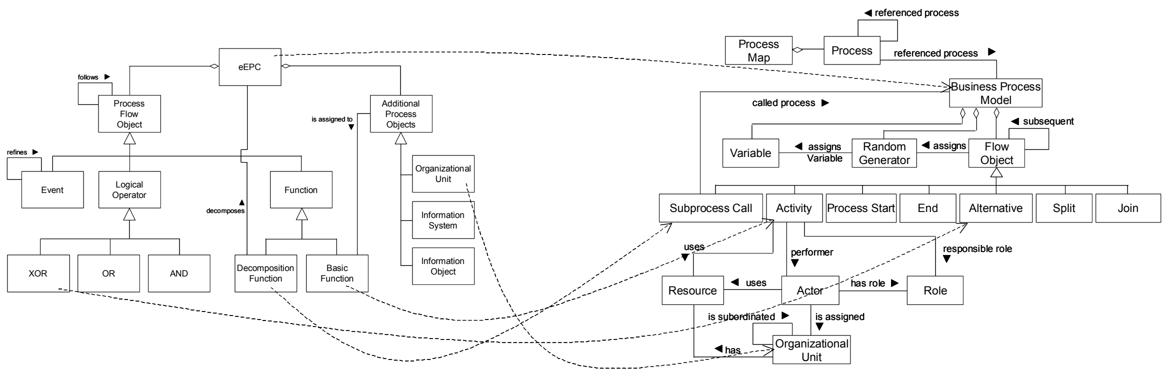

Figure 4: Metamodel mapping and transformation rule definition within BMT

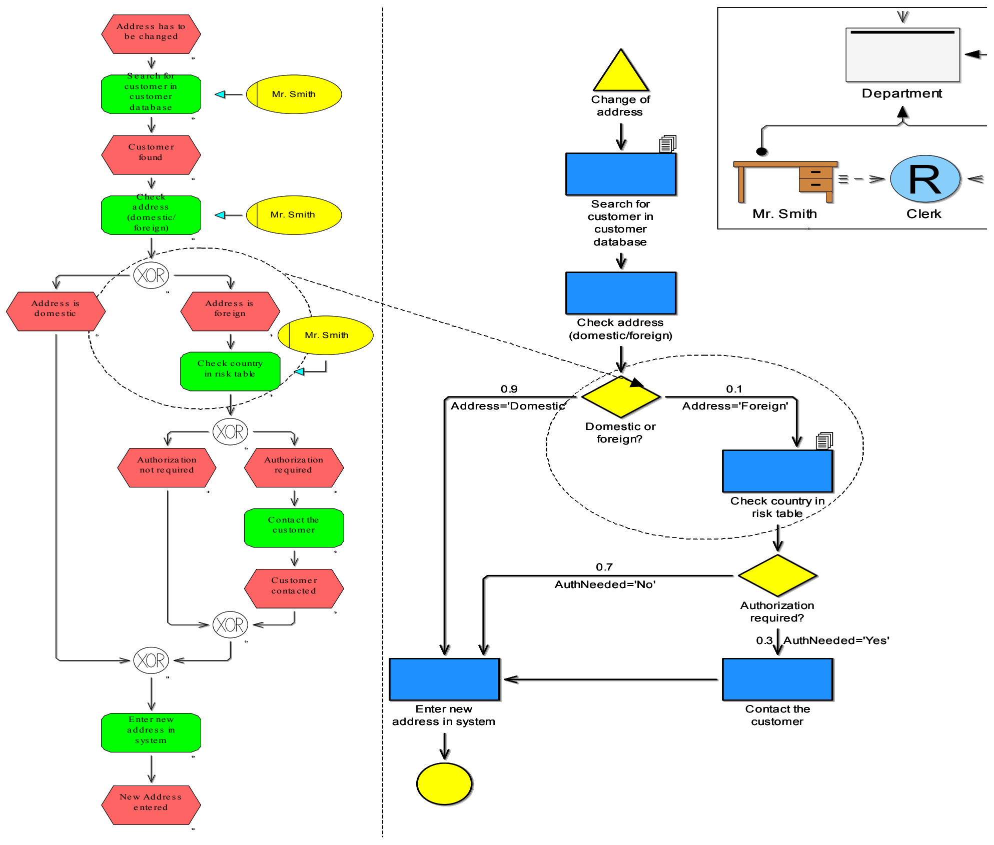

Figure 5: Transformation from EPC to ADONIS Standard Process Language

As a starting point, the metamodels of the source business process modelling language and the target business process modelling languages are defined (see figure 4). Structural correspondences of the metamodels can be mapped directly. More complex transformations, such as the transformation of control structures of the process flow definition are specified by generation rules. Figure 5 shows a business process model in EPC language (left side) and a business process model in ADONIS standard process modelling language (right side). The modelling language on left side will be automatically transformed into modelling language (right side).

Table 1 (left side) shows the generation rule for transforming a XOR control structure from EPC language to a decision structure in ADONIS language. Both events “Address is domestic” and “Address is foreign” after the XOR object are transformed to transition conditions “Address=’Domestic’” and “Address=’Domestic’” following the decision object. A precondition for generating the transition conditions is the proper naming of the event objects. A detailed description of the transformation rule can be found in table 1 (right side).

If an element of the source metamodel should be transformed context-sensitively to different concepts in the target metamodel, BMT offers the “mode” transformation concept to distinguish context and to assign several transformation rules. For describing a complete model transformation, for each element of the source metamodel, one or more rules have to be defined. For a detailed description of rule definition within BMT see [9].

| RULE | DESCRIPTION |
|---|---|
| `<NAMEVALUEMAP>`<br>`  <ELEM name="n_xor">`<br>`    <select-value>`<br>`      <PARAM name="attribute">name</PARAM>`<br>`    </select-value>`<br>`  </ELEM>` | • In this section two variables are created. The first variable n_xor is initialised with the name of the XOR-element taken out of the XML attribute "name". |
| `  <ELEM name="n_funct">`<br>`    <NAVIGATE type="follow-connector">`<br>`    <PARAM name="class">subsequent</PARAM>`<br>`    <PARAM name="target">event</PARAM>`<br>`    <NAVIGATE type="follow-connector">`<br>`    <PARAM name="class">subsequent</PARAM>`<br>`    <PARAM name="target">function</PARAM>`<br>`      <select-value>`<br>`        <PARAM name="attribute">name</PARAM>`<br>`      </select-value>`<br>`    </NAVIGATE></NAVIGATE></ELEM>`<br>`</NAMEVALUEMAP>` | • The second variable n_funct is initialised with the name of the function-element. Therefore, we navigate from the XOR object to the next event on the outgoing connector. Then it is navigated from the event to the next function, also on the outgoing connector. Then the name of the function-element is assigned to the variable n_funct. |
| `<RULE type="create-instance">`<br>`  <PARAM name="Instanceclass">decision</PARAM>`<br>`  <PARAM name="name"><get-map-value>n_xor`<br>`                    </get-map-value></PARAM>`<br>`</RULE>` | • Create a new instance of class "decision" and assign the value of the n_xor variable. |
| `<RULE type="create-connector">`<br>`  <PARAM name="Relationclass">subsequent</PARAM>`<br>`  <PARAM name="Convertmethode">create</PARAM>`<br>`  <PARAM name="from-class">decision</PARAM>`<br>`  <PARAM name="from-name"><get-map-value>`<br>`        n_xor</get-map-value></PARAM>`<br>`  <PARAM name="to-class">activity</PARAM>`<br>`  <PARAM name="to-name"><get-map-value>n_funct`<br>`                       </get-map-value></PARAM>`<br>`</RULE>` | • Then create the connector from the new decision to the new activity, which has not been created yet.<br><br>Note that at the end the final "from" and "to" of the connector will be refreshed according to the reference history (see figure 2). |

Table 1: Fragment of Transformation Rule in BMT for Process Flow

## 4. SUMMARY AND STATEMENT

Because of current diversity of modelling languages in the domain of business process modelling, we propose a business process model transformation approach. Especially in large organisations or in networks of corporations, often a number of different business process modelling languages and/or process modelling tools can be found. To ensure the exchange and interoperability of business process models in the design and implementation of large, heterogeneous enterprise systems, BMT is based on metamodelling and model transformation techniques. Our experiences applying BMT and model transformation approaches are considerable savings in costs and time, and improved quality in system specification. Nevertheless, the effort in specifying proper rule definitions should not be ignored.

As model interoperability within enterprise systems specification is a crucial aspect, we are convinced the BMT approach is a suitable contribution for the INTEREST workshop.

## REFERENCES

[1] BOC: ADONIS 3.7 - User Manual III: ADONIS Standard Modelling Method. BOC GmbH, May 2003.
[2] BPMI.org: Business Process Modeling Language - Specification Version 1.0. http://www.bpmi.org/bpml-spec.esp, access 12 April 2004.
[3] BPMI.org: Business Process Modeling Notation - Specification Version 1.0. http://www.bpmi.org/bpmn-spec.esp, access 12 April 2004.
[4] Czarnecki, K.; Helsen, S.: Classification of Model Transformation Approaches. OOPSLA'03, Workshop on Generative Techniques in the Context of Model-Driven Architecture.

[5] Gardner, T.; Griffin, C.; Koehler, J.; Hauser, R.: A review of OMG MOF 2.0 Query / Views / Transformations Submissions and Recommendations Towards the Final Standard. OMG, IBM, July 2003.
[6] Johannesson, P.; Wangler, B.; Jayaweera, P.: Application and Process Integration – Concepts, Issues, and Research Directions. In: Brinkkemper, S.; Lindencrona, E.; Solvberg, A. (Eds.): Information Systems Engineering Symposium CAiSE 2000, Springer-Verlag, 2000.
[7] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozessmodellierung auf der Basis "Ereignisgesteuerter Prozeßketten (EPK)". Publication of Institute for Business Informatics, No. 89, University of Saarbruecken, 1992. http://www.iwi.uni-sb.de/iwi-hefte/heft089.zip, access 12 April 2004.
[8] Kühn, H.; Bayer, F.; Junginger, S.; Karagiannis, D.: Enterprise Model Integration. In: Bauknecht, K.; Tjoa, A M.; Quirchmayr, G. (Hrsg.): Proceedings of the 4th International Conference EC-Web 2003 - Dexa 2003, Prague, Czech Republic, September 2003, LNCS 2738, Springer-Verlag, pp. 379-392.
[9] Murzek, M.: Methodenübergreifende Modelltransformationen am Beispiel von ADONIS. Diploma Thesis, University of Vienna, April 2004 (in German).
[10]Object Management Group: OMG Unified Modeling Language Specification, Version 1.4, September 2001. http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf, access 12 April 2004.
[11] Object Management Group: OMG XML Metadata Interchange (XMI) Specification, Version 1.2, Januar 2002. http://www.omg.org/cgi-bin/doc?formal/02-01-01.pdf, access 12 April 2004.
[12] Object Management Group: Meta Object Facility (MOF) Specification, Version 1.4, April 2002. http://www.omg.org/cgi-bin/doc?formal/02-04-03.pdf, access 12 April 2004.
[13] Object Management Group: MOF 2.0 Query / Views / Transformations RFP. OMG Document: ad/2002-04-10, April 2002.
[14] Object Management Group: MDA Guide, Version 1.0.1, 12. Juni 2003. http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf, access 12 April 2004.
[15] UEML Version 1.0, http://www.ueml.org, access 12 April 2004.