Usage of Model Driven Engineering in the context of Business Process Management

Pascal Bauler, Fernand Feltz, Etienne Frogneux, Benjamin Renwart, Céline Thomase

Informatique, Systèmes, Collaboration
Centre de Recherche Public – Gabriel Lippmann
41, rue du Brill
L-4422 Belvaux, Luxembourg
{bauler, feltz, renwart, frogneux, thomase}@lippmann.lu

Abstract: This paper shows how Model Driven Engineering (MDE) and Business Process Management (BPM) can be combined when generating executable BPEL processes out of BPMN models. By adding appropriate meta-data information to the BPMN models, business patterns like model fragments or reference process building blocks and technical patterns are applied during the BPMN to BPEL transformation phase. The proposed solution offers a common modeling environment and facilitates collaboration between the business analysts and the technical team. A practical use case in the context of a research project realized by the Centre de Recherche Public – Gabriel Lippmann in collaboration with the Luxembourg National Family Benefits Fund, shows the benefits of the combination of MDE and BPMN.

1 Introduction

In the early 1990's, business processes became popular in the context of enterprise modernization and reorganization. Business process management was mainly used at management level, to document the current activities and to plan business improvements through reengineering projects [HC93]. Information Technology (IT) was mainly considered an enabling technology offering adequate tool support for management. Over the last years, Business Process Management (BPM) got more and more integrated into IT projects and workflow engines able to directly execute business processes emerged. Process Aware Information Systems (PAIS) became popular and as a consequence several IT focused standards emerged. We may mention standards like Business Process Modeling Notation (BPMN), Business Process Execution Language (BPEL) and XML Process Definition Language (XPDL), which are discussed in more detail in [Cha06]. These standards can mainly be divided into two categories:

- Process description languages, convenient for business analysts, need to be manually reworked before being executed.

- Formal Execution languages directly executable by workflow engines, which however are too technical for business people

In addition, Service Oriented Architectures (SOA) use BPEL as a de facto orchestration layer. The BPMN specification is mainly focused on business analysts but introducing transformation mappings with BPEL keeps a link with executable workflows. As BPMN and BPEL operate at different abstraction layers, these transformation mappings have some limitations and the resulting BPEL models still have to be manually adapted by the technical team. This results in misunderstandings and inefficiency. As a consequence, the business analysts, when defining business processes in direct collaboration with the business owners, continue to work at a high abstraction level. The same processes are handed over to the technical team for integration into the workflow engines and enterprise applications.

Recent research projects, in view of optimizing the modeling activities by reusing recurring constructs, focus on identifying reusable model fragments, business oriented workflow patterns [TIR07] and Reference Process Building Blocks (RPBB) [BRW07].

This paper follows a similar approach and by applying concepts of Model Driven Engineering (MDE) to BPM, tries to reduce the gap between process description languages and formal execution languages. The idea consists in having the technical team enhance the business processes handed over by the business analysts, by means of annotations and meta-data. These enhanced business processes still remain readable for business people. The technical annotations are used to apply project specific business and technical patterns while hiding the technical complexity and are taken into account when generating executable workflows. This paper relies on BPMN as high level modeling language and executable BPEL processes are generated. The proposed solution relies on two transformation phases. A first transformation phase applies specific business patterns by including previously defined model fragments and RPBB. A second transformation phase handles the technical aspects by helping to overcome language incompatibilities between high level and low level modeling languages. The resulting tool chain is able to practically apply concepts like business patterns and RPBB as well as technical patterns. This is illustrated in the context of a research project realized in collaboration with the Luxembourg National Family Benefits Fund.

The following text gives an overview of the involved concepts before showing the underlying IT architecture. The paper concludes after having discussed the proposed solution and exposed some use cases.

2 Technical aspects

This section details the two main topics of the paper, Business Process Management and Model Driven Engineering.

2.1 Business Process Management

A key benefit of integrating BPM solutions into enterprise applications is the decoupling of the application code and the underlying workflows. As a consequence, the adaptability and traceability of the Process Aware Information Systems are improved, which results in more sustainable IT solutions [DAH05]. A close collaboration between business analysts, taking advantage of high level modeling languages, and the technical team, working with executable modeling languages or application code, is mandatory when designing PAIS. This paper mainly focuses on BPMN as high level modeling language and on BPEL as orchestration language of executable business processes. These two technologies operating at different abstraction layers are geared to different user groups and have different origins. Articles like [RM06] even identify a conceptual mismatch between them. The proposed solution provides some useful elements addressing these issues, without however ignoring the underlying concepts.

Business Process Modeling Notation (BPMN)

BPMN is well accepted by the business analysts, who take advantage of convenient modeling tools respecting the visual BPMN specification[1]. BPMN, specified by the Business Process Management Initiative founded in 2000, was first released in 2004. In February 2006, the OMG approved the final specification. BPMN is basically a graph-oriented language offering many facilities and flexibilities to business analysts. The specification also defines mappings with BPEL, which however have some limitations [Woh06].

Business Process Execution Language (BPEL)

OASIS elaborated BPEL as an XML based specification for the orchestration of long running processes based on Web Services. The BPEL 1.1 specification was released in 2003 and version 2.0 was approved by OASIS in 2007[2].

BPEL is a block structured process definition language [LR04], primarily focused on the execution/automation of business processes. It offers a technology-centered approach for defining business processes, which can directly be executed by adequate workflow engines. BPEL is mainly useful for technical people and the BPEL diagrams are hardly meaningful to business analysts.

2.2 Model Driven Engineering

Model Driven Engineering became popular over the last years and Model Driven Architecture (MDA) and Model Driven Software Design (MDSD) got heavily used in application design. The key benefit of these approaches is the possibility to hide the technical complexity of the underlying solutions and frameworks by defining Domain Specific Languages (DSL) especially adapted to the specific project needs. The classical

[1] BPMN: Business Process Modeling Notation, (http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf)
[2] BPEL: Business Process Execution Language, (http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html)

MDA approach, as introduced by the OMG, distinguishes between the Platform Independent Model (PIM) and the Platform Specific Model (PSM). The PIM offers a high level, technology neutral view of the IT solution. By means of various Model To Model (M2M) transformations based on the underlying meta-model, the PIM is systematically enhanced to result into a PSM. This PSM is directly linked to the underlying technology and can be transformed into executable code. As a complete generation of application code out of a model seems impossible with today's technologies [PM06], existing frameworks introduce special concepts like protected regions or specific patterns. By means of these workarounds, generated and manually developed code segments are combined to obtain the final application. MDSD realizes a similar but more flexible approach. While MDA, as defined by the OMG, fully relies on UML based modeling tools, MDSD puts no constraints on the input models. This identifies MDSD as a good starting point when working with XMI[3] / XML based modeling languages like BPMN and BPEL. MDSD also introduces concepts like M2M transformations as well as several model transformation techniques ([EHV06], [CH03]) (like model merge, enhancement, linking,…), which are heavily used by the proposed solution.

From an operational point of view, the proposed solution takes advantage of OpenArchitectureware[4] (OAW), an Open-Source Model Driven Software Design framework offering generic model-to-model and model-to-code transformation possibilities, based on underlying meta-models. The OAW framework introduces XPAND and XTEND, which are powerful and easy to manipulate uni-directional transformation languages.

3 Architecture

The proposed architecture describes a tool chain built around three key components, an enhanced BPMN editor with meta-data support, a BPMN to BPEL transformation algorithm and the OAW MDSD framework. As shown in figure 1, the BPMN model, output of the BPMN editor, is stored in an XML file. The MDSD framework uses this file as input and applies business oriented model transformations based on project specific business patterns, to produce an enhanced BPMN model. This output is viewable by the editor, but is also used as input by the BPMN to BPEL transformation algorithm, which produces a pseudo BPEL process. As BPMN and BPEL operate at different technical abstraction layers, this BPEL process has to be enhanced in a second transformation phase. As a consequence, the MDSD framework is invoked a second time. During this second invocation it relies on a BPEL based meta-model, performs technical model transformations and applies technical patterns in order to produce executable BPEL processes.

Below the various sub-components are described in detail.

[3] XMI: XML Metadata Interchange format: http://www.omg.org/technology/documents/formal/xmi.htm
[4] OpenArchitectureWare is an Open Source MDSD framework: http://www.openarchitectureware.org

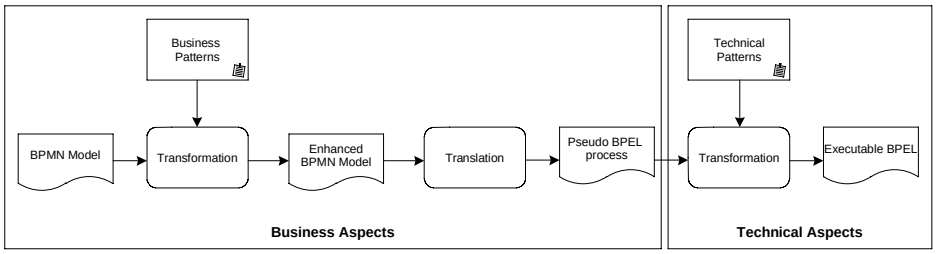

Figure 1: Architecture overview

3.1 BPMN editor

The first component in the tool chain is an enhanced BPMN editor based on the Eclipse EMF[5] and GMF[6] frameworks. This editor offers adequate modeling possibilities to the business analysts. It also supports additional meta-data to be specified by the technical team. These meta-data, mandatory to generate executable workflows, contain valuable information used during the transformation steps. For instance the precise service calls invoked out of BPEL and references to specific RPBB are set by means of meta-data tags. The proposed editor offers two views: the view dedicated to the business owners and business analysts hides the technical aspects; the view dedicated to the technical team gives access to all available features. The editor supports agile development involving the business analysts and the development team, by enabling an iterative collaboration between the various stakeholders, supports agile development involving the business analysts and the development team. Refactored or adapted processes, relying on a same data exchange format can be interchanged between developers and business analysts in order to validate the final version of the business processes. From a technical standpoint, both views of the editor rely on the same meta-model. This meta-model is used to export/import the business processes represented in XMI format and it is generic enough to store business process descriptions as well as specific meta-data. The same meta-model is used as source model in the first model transformation phase.

3.2 OpenArchitectureWare: first invocation

The XMI output file, produced by the editor, is used as input by the OAW MDSD framework. During this first transformation step, business related transformation patterns and RPBB are applied by means of scripts written in the XTEND model-to-model transformation language. For instance, a generic building block in charge of handling the Luxembourgish allowances can be integrated and by specifying the required meta-data, this building block is adapted to the particular context of the family allowances. This first MDSD step also helps to overcome conceptual mismatches between BPMN and BPEL. For instance, the first MDSD step will adapt BPMN input models composed of multiple swimlanes (not supported in BPEL) in order to obtain independent BPEL

[5] EMF: Eclipse Modeling Framework http://www.eclipse.org/emf
[6] GMF: Graphical Modeling Framework http://www.eclipse.org/gmf

processes during later transformation steps. After applying several model transformations, the OAW framework will generate two outputs:

- A BPMN model file, which is ready to be used as input by the BPMN to BPEL transformation algorithm.

- An enhanced XMI output file respecting the initial meta-model. This file can be used by the BPMN editor for further enhancements or by the business owners for the validation. It also contains additional meta-data, which are mandatory for the enhancement of the generated BPEL process during the second transformation phase.

As the BPMN editor is able to use the XMI output file, an agile modeling approach can be adapted. Business owners, business analysts and the development team iteratively take ownership of the business processes to review them and to work on them. The final business processes are passed to the BPMN to BPEL conversion algorithm.

3.3 BPMN – BPEL transformation

The BPMN-BPEL transformation algorithm directly relies on the work realized by [Ouy06]. This paper gives a detailed explanation of the transformation algorithm taking valid BPMN files as input and producing pseudo BPEL constructs as output. The conversion algorithm takes advantage of a pattern matching approach, distinguishing between simple workflow structures (named well-structured) and more complex constructs (non well-structured). In the specific context of this paper, the underlying algorithm is reused as is, with only minimal adaptations. Unfortunately, the available implementation is limited to a sub-set of the BPMN specification and BPMN specific structures like pools, swimlanes and sub-processes are not covered. In addition, during the translation placeholders replace the variables used inside invoke or branch operations. The generated output file of the BPMN to BPEL transformation algorithm is a BPEL XML file, which however is not yet ready for deployment. Several technical aspects are not yet handled and further model transformations are required. This explains the need for a second invocation of the OAW MDSD framework.

3.4 OpenArchitectureWare: second invocation

This transformation step uses two input files, the XMI output file resulting from the first invocation of the OAW framework and the output file of the BPMN to BPEL conversion algorithm. These two files are merged in order to populate the BPEL specific meta-model defined in OAW. After this model merge, OAW holds all data to generate, by means of the XPAND model-to-code transformation language, a deployable BPEL output file and the deployment descriptors. The two input files are mandatory because BPMN and BPEL do not operate at the same abstraction layer. The conversion algorithm only sets default values for several technical BPEL parameters. Additional technical information, like precise service invocations, are extracted from the XMI input file.

OAW also applies technical model enhancements and transformations as well as a set of technical patterns. The implemented patterns handle BPEL specific issues and help to overcome limitations of the language. Section 5 shows for instance how an error-handling scenario is implemented by simulating the “arbitrary cycle” workflow pattern [Aal03], which is not supported in BPEL. The technical aspects of this error handler are generated during this transformation step.

4 Discussion

The proposed tool chain illustrates how MDE can be applied in the context of BPM. The resulting solution goes beyond the transformation of BPMN business processes to BPEL processes, by enabling the concrete implementation of business patterns like model fragments or RPBB. Standardized and recurring model constructs can be added and customized by specifying appropriate meta-data in the enhanced BPMN model.

The technical complexity can also be significantly reduced, by applying specific patterns through model transformations. These advantages of the MDE approach in the particular context of BPM reduce the gap between high-level process definitions and operational/executable business processes. This gives an overview of the overall situation to the project team and is also a good starting point for a documentation of the business processes. High-level models and process definitions are a good basis for discussions between the application architects and the business owners. A close coupling between high-level BPMN models and executable BPEL is maintained during the whole project. The automated generation of deployable BPEL processes avoids problems, which could result from incorrect interpretation during manual translations and adaptations.

In the context of BPM, MDE enables the automation of repetitive tasks. This automation improves efficiency of the project team and significantly reduces debugging activities. The learning curve is also improved, as the technical details of the BPEL language are not mandatory when working on the business processes. Deep BPEL or BPMN knowledge is only required when specifying new technical or business patterns to be applied during the two model transformation phases. While classical MDSD can enforce the enterprise architectures, MDE in the context of BPM enforces the enterprise modeling standards. The generated workflows are free of technical errors and the deployment is automated, which again hides many technical details.

Business patterns and technical transformation patterns facilitate the encapsulation of recurring sub-models. This results in a clearer representation of the core business processes and is a starting point for defining domain specific modeling languages, where domain specific modeling constructs directly take underlying business constraints into account.

5 Use case

In this section we show two practical use cases extracted from a research project in collaboration of the Centre de Recherche – Gabriel Lippmann and the Luxembourg National Family Benefits Fund. The main objective of this project is to automate the payment of family allowances in Luxembourg, by focusing in a first project phase onto the French borderline commuters. This automation is part of the complete modernization of the Luxembourg National Family Benefits Fund, introducing a Service Oriented Architecture [Bau06] integrating legacy applications and state of the art enterprise solutions based on Java Enterprise Edition technology.

The automation of the payment of family allowances is realized in a multi-layer architecture, where a BPEL engine is used as orchestration layer handling the execution of the underlying business process. The main objective of this approach is to clearly decouple application code and business processes. This innovative approach published in [Bir07], improves the flexibility and adaptability of the overall solution and significantly facilitates the integration into the new SOA of the Luxembourg National Family Benefits Fund.

Experience has shown that BPEL is an adequate solution to orchestrate business processes, but several technical issues or language constraints increase the complexity of the BPEL processes. A negative side effect is that the resulting processes are hardly understandable for business people and require deep technical BPEL skills.

In this section we rely on a simplified BPMN business process coordinating the payments of the Luxembourgish Family Allowances for French borderline commuters, as shown on figure 2. The first process step consists in computing the Luxembourgish allowances. In case of problems, the workflow ends. Otherwise the data exchange with France is triggered. Based on the French feedback, the internal computations are initiated and executed. The final process step consists in preparing the concrete payment of the family allowances. This BPMN model is used to illustrate how business patterns and RPBB are applied, by introducing the concept of a generic Luxembourg allowances building block. A second scenario shows how technical patterns are implemented in the context of a convenient error handling solution based on the “arbitrary cycle” workflow pattern.

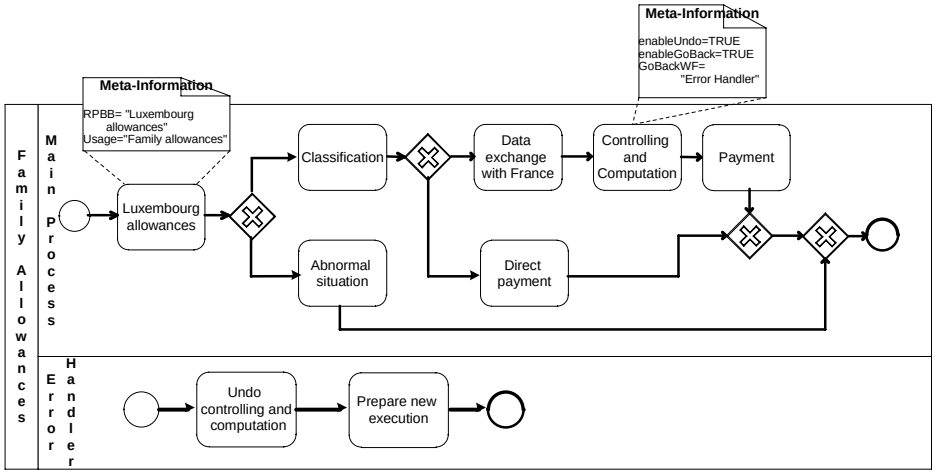

Figure 2: BPMN Example workflow

5.1 Business patterns and Reference Process Building Blocks

In this paragraph we concentrate onto the computing of the Luxembourgish family allowances. To do so, a generic allowances RPBB (Figure 3) is introduced. This building block contains the required logic to handle various types of allowances. Specific meta-information refer to the appropriate RPBB during the first invocation of the OAW framework. In addition, the context specific usage is also specified and refers to the family allowances, without considering other types like social allowances or subsistence allowances. The BPMN model, resulting from the first invocation of the OAW MDSD framework, is not overloaded by the complete generic allowances RPBB, but focuses onto the family allowances sub-model. The resulting BPMN model is used as input in the subsequent transformation steps when generating executable BPEL processes.

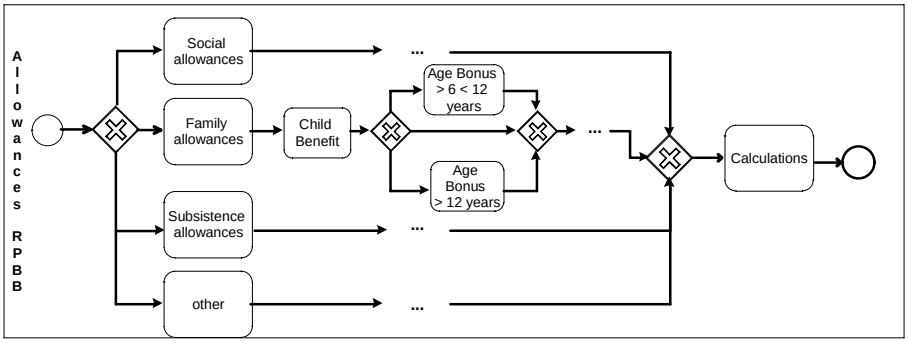

Figure 3: Business pattern example; Allowances RPBB

5.2 Error handling mechanism

The proposed error handling mechanism relies on the “arbitrary cycle” workflow pattern, which is not supported by BPEL [Woh03]. This specific workflow pattern is simulated, by adding a specific error handling workflow to the “Controlling and Computation” process step.

The proposed solution (Figure 4) consists in including a custom defined fault handling process. After the execution of the fault handling process, the main process is restarted at any previously defined position and thus requires the implementation of the “arbitrary cycle” workflow pattern. The process steps of the fault handler are specified in a dedicated BPMN model. Additional meta-data in the enhanced BPMN model associate the error handling workflow to a specific task of the main process by generating an appropriate fault handler. During a first model transformation phase (first invocation of OAW), the two BPMN workflows are properly decoupled, adapted and passed individually through the BPMN to BPEL transformation algorithm. During the second invocation of the OAW framework, the two resulting BPEL processes are merged, by adding appropriate constructs. The fault handler contains the appropriate BPEL constructs to realize some internal housekeeping tasks at runtime, before executing the error handling workflow specified in BPMN. After completion, the current BPEL process is cloned with the appropriate start state and the current process is terminated.

As final output, a deployable BPEL process and the deployment descriptors are generated. The proposed solution is flexible enough to enable repetitive invocations of the undo operation and is able to undo a complete business process.

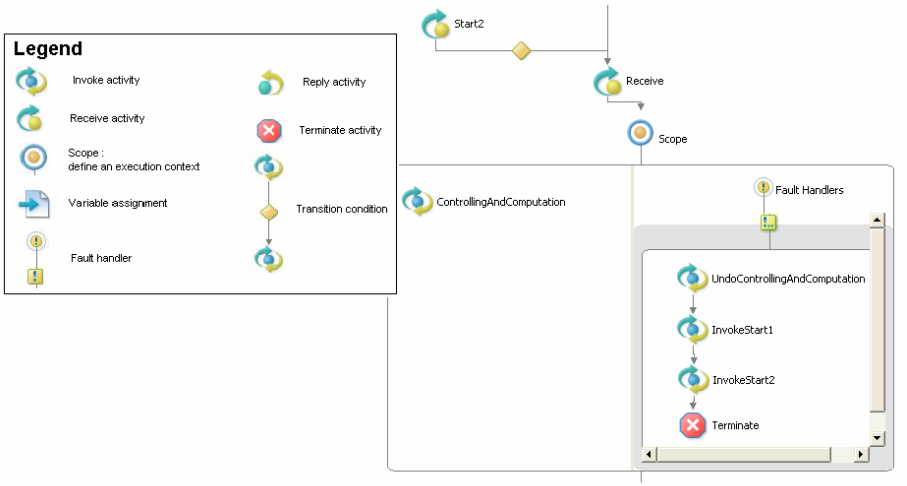

Figure 4: Extract of the generated BPEL process

In the context of the project with the Luxembourg National Family Benefits Fund, the main objectives of the above-mentioned solution are to accelerate and optimize the modeling activities by reusing recurring RPBB and to solve technical issues by trying to overcome technical limitations and complexities of the different modeling languages. The available transformation patterns can be extended and adapted to fit special business

needs, improve the efficiency of the project team or impose certain modeling standards. As the transformation process generates valid BPEL processes similar to those already in use, the implementation of the solution has only minimal impact onto the overall development process.

6 Conclusion

This paper shows how Model Driven Engineering can be applied in the domain of Business Process Management and illustrates the benefits of this combination. BPMN processes are systematically enhanced by means of meta-data used to include specific business oriented and technical transformation patterns before generating executable BPEL processes. As a result, both the business analysts and the technical team members can use BPMN as a common modeling language. This improves the collaboration between both teams and improves the overall efficiency and transparency of the project. From a business perspective, previously defined model fragments can easily be incorporated into new models. From a technical perspective, most of the complexity of BPEL can be hidden and technical constructs are encapsulated into appropriate model transformations. As iterative enhancement of the business models is possible, the proposed toolset can be used in agile projects. Even if the proposed solution is still in development, concrete benefits in the context of the Luxembourg National Family Benefits Fund are already visible as illustrated by the generic error handling scenario and the reuse of a generic building block in charge of the Luxembourgish allowances.

Bibliography

[Aal03] Wil M. P. van der Aalst, Arthur H. M. ter Hofestede, B. Kiepuszewski, A. P. Barros: Workflow Patterns, Distributed and Parallel Databases 14, pages 5-51, Springer Netherlands, Dordrecht 2003

[Bau06] Pascal Bauler, Fernand Feltz, Nicolas Biri, Philippe Pinheiro,: Implementing a Service-Oriented Architecture for Small and Medium Organisations, EMISA 2006, Hamburg 2006

[BRW07] Lars Baacke, Peter Rohner, Robert Winter,: Aggregation of Reference Process Building Blocks to Improve Modeling in Public Administrations, Sixth International EGOV Conference 2007, Regensburg 2007

[Bir07] Nicolas Biri, Pascal Bauler, Fernand Feltz, Nicolas Médoc, Céline Thomase, Using BPEL as a workflow engine for local enterprise applications, EMISA 2007, St Goar 2007

[CH03] Krzysztof Czarnecki, Simon Helsen,: Classification of Model Transformation Approaches, OOPSLA'03, Anaheim California 2003

[Cha06] James F. Chang: Business Process Management Systems, Auerbach Publications, New York 2006, p. 201-236

[DAH05] Marlon Dumas, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede: Process-Aware Information Systems, Wiley Interscience, New-Jersey 2005

[EHV06] Sven Efftinge, Arno Haase, Markus Völter: Model Transformation, Introduction and Concepts, JAX2006, Wiesbaden 2006 (http://www.völter.de/data/presentations/ModelTransformations.zip) (accessed on 21/9/2007)

[HC93] Michael Hammer, James Champy: Reengineering the Coporation: A manifesto for business revolution, HarperBusiness, New-York 1993

[LR04] Frank Leymann, Dieter Roller: Modeling Business Processes with BPEL4WS, XML4BPM 2004, Marburg 2004

[Ouy06] Chun Ouyang, Marlon Dumas, Arthur H. M. ter Hofstede, Wil M. P. van der Aalst: From BPMN Process Models to BPEL Web Servcies, ICWS 2006, Chicago 2006

[PM06] Roland Petrasch, Oliver Meinberg: Model Driven Architecture, dpunkt.verlag, Heidelberg 2006

[RM06] Jan Recker, Jan Mendling,: On the Translation BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages, CAISE 2006 Workshop Proceedings – Eleventh International Workshop on Exploring Modeling Methods in System Analysis and Design (EMMSAD2006), Luxembourg, 2006

[TIR07] Lucinéia Heloisa Thom, Cirano Iochpe, Manfred Reichert,: Workflow Patterns for Business Process Modeling, CAISE 2007, 8th Workshop on Business Process Modeling, Development and Support (BPMDS'07), Trondheim 2007

[Woh03] P. Wohed, Wil M. P. van der Aalst, M. Dumas, A.H.M. ter Hofstede: Analysis of Web Services Composition Languages: The Case of BPEL4WS, Conceptual Modeling – ER 2003, Chicago 2003

[Woh06] P. Wohed, Wil M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, N. Russell,: On the Suitability of BPMN for Business Process Modelling, BPM 2006, Vienna 2006