MDSD light for ERP

Stephan Roser, Florian Lautenbacher, Bernhard Bauer

Angaben zur Veröffentlichung / Publication details:

Roser, Stephan, Florian Lautenbacher, and Bernhard Bauer. 2008. "MDSD light for ERP." In *Proceedings of the 2008 ACM Symposium on Applied Computing - SAC '08, Fortaleza, Ceara, Brazil, March 16 - 20, 2008*, edited by Roger L. Wainwright and Hisham M. Haddad, 1042–47. New York, NY: ACM Press. https://doi.org/10.1145/1363686.1363930.

MDSD light for ERP

Stephan Roser
Programming Distributed Systems Lab
University of Augsburg
Germany
roser@ds-lab.org

Florian Lautenbacher
Programming Distributed Systems Lab
University of Augsburg
Germany
lautenbacher@ds-lab.org

Bernhard Bauer
Programming Distributed Systems Lab
University of Augsburg
Germany
bauer@ds-lab.org

ABSTRACT

Fully-fledged model-driven software development is often too expensive for small and medium sized enterprises (SMEs). Therefore we propose MDSD light which reduces the available adjustment possibilities, but offers an easy way of workflow and process management. We introduce a tool-suite which realizes the aspects of MDSD light and show the advantages of this approach in a short example.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Computer-aided software engineering (CASE)*; K.6.3 [**Management of Computing and Information Systems**]: Software Management—*Software development*

General Terms

Design

Keywords

MDSD, model-driven, process, service-oriented, ERP, SME

1. MOTIVATION

Model-driven software development (MDSD) has evolved by applying the concept of model-driven engineering (MDE) to software development [2]. MDSD is a promising software engineering approach that focuses on the creation of models rather than on programming code. In MDSD models are used to describe business concerns, user requirements, information structures, components, component interactions, and any other issues that are relevant in a software development process. These models govern the system development because they are transformed into executable code.

Model Driven Architecture® (MDA®) [11] is the OMG's effort to realize MDSD. Its primary goals are to achieve portability, interoperability and reusability through an architectural separation of concerns between the specification and implementation of software [5]. The link between models representing different concerns is provided by so-called model transformations. In MDA for example, multiple model transformations have to be developed transforming and integrating knowledge captured in the various models (CIM, PIM, PSM, architectural model, platform model, etc.).

However, model transformations are a kind of metaprogramming. The needed skills and knowledge for the implementation of model transformations are very difficult to achieve and too expensive for small- and medium-sized enterprises (SMEs). There is a need for a model-driven way with less models and model transformations which can be applied to the concerns of SMEs. In this paper we introduce the concept Model-driven Software Development light (MDSD light) which aims to fill this gap. We present how MDSD light can be applied to the ERP domain and illustrate the relevant aspects via a case study. The open-source tool AgilPro will be introduced and aspects of MDSD light that are realized in AgilPro are highlighted.

This paper is structured as follows: In Section 2 we review current software development, the barriers SMEs face in applying MDSD and propose MDSD light. In Section 3 we present the application of MDSD light for the ERP domain and describe its implementation in the AgilPro tools. This implementation is illustrated via a case study in Section 4. In Section 5 we discuss related work and in Section 6 we provide a discussion and conclusions about our approach.

2. MDSD AND MDSD LIGHT

Having a look at a common usage of the MDA approach, one can find three abstraction levels (compare Figure 1): business managers model their business requirements and flows in the computation independent model (CIM) using the Business Process Modeling Notation (BPMN) [13] , Event-driven Process Chains (EPCs) [19] or simply UML2 activity diagrams [16]. Via model transformations these aspects are transformed into the platform independent model (PIM). A PIM includes more architectural aspects, but still stays independent of any implementation details. In this technical view software architects mostly use UML2 diagrams for the documentation and communication about the system. Using another model transformation one can include platform aspects. The resulting platform specific model (PSM) is often manually refined with implementation details. This model can directly be mapped to code for the respective languages and technologie (e.g. WebServices, EJBs, POJOs, WSBPEL [10], etc.).

As already mentioned above, there exist certain barriers

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

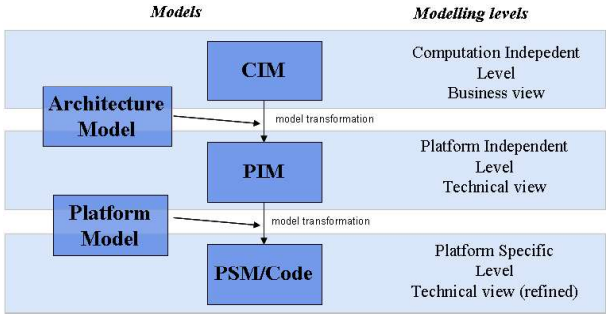

Figure 1: The model-driven architecture

for SMEs to apply this classical approach of MDSD:

- *Model transformations need highly skilled experts:* Languages and frameworks for model transformation like OMG's standard Query/View/Transformations (QVT) [12] or the Model Transformation Framework (MTF) have a very steep learning curve and need expert knowledge. Especially SMEs can't afford these experts.
- *Set up and maintenance:* MDSD solutions, i.e. models and model transformations, have to be adjusted to enterprises' development environments. There are also maintenance efforts when development styles change or new versions of metamodels are released (e.g. UML).
- *Modelling overhead:* Often the architectures and platforms used in the IT landscape of SMEs do hardly change over a long time. Thus, the added value by additional architecture and platform models does not justify the efforts needed for implementing und maintaining them.
- *General purpose modelling languages:* General purpose languages like the UML provide various ways for modelling the same thing or system. Additional information like modelling guidelines or constraints have to be added in order to allow effective model transformation and code generation.

In order to reduce such obstacles, we introduce and promote the usage of the MDSD light approach. MDSD light is not defined in an absolute manner, but through a set of paradigms that represent this approach. The more a MDSD process fulfils those characteristics, the more MDSD light it is.

- Only the really necessary models for a specific enterprise environment and application area are incorporated in an MDSD light development approach.
- Model transformations and code generation are reduced to an absolutely minimum.
- As much knowledge as possible is directly encoded in transformations or captured by the semantics of domain specific modelling languages.

In MDSD there are several adjustment screws which can be regulated so that a MDSD process meets the requirements of specific projects or application domains. The adjustment screws can offer a maximum of flexiblity for the software development and the modeller can constrain software development to very specific and predefined scenarios. However, the greater the flexiblity the more effort and skills are needed to perform MDSD. With the goal of MDSD light to reduce barriers for applying MDSD, we can attempt to reduce complexity by the price of less flexibility. The main decisions that have to be made in MDSD determine also the available adjustment screws:

- *Technology:* In MDSD the model transformation from CIM to PIM needs to include information about the specific technology and architecture that should be used further on.
- *Platform:* The information about the specific platform is included in the platform model which is needed for the model transformation from PIM to PSM.
- *Domain:* The domain is included in all models, starting with CIM down to code. Domain information are elements of the models and metamodels that represent concepts specific to e.g. the ERP domain. These elements also contain information that is relevant for the execution of the information systems.

3. APPLYING MDSD LIGHT TO ERP

During the last decade process modelling and agile processes in the ERP domain got more and more important. In order to improve existing products most changes need to be done not on the services that are offered, but on the flow and composition of services instead. This results from changes in jurisdiction, new products or standards.

3.1 Approach

We apply the paradigms of MDSD light to develop a model-driven solution for ERP processes of SMEs. First, we have to gather important requirements and constraints of this application area:

- The architecture for current and future development is the service-oriented architecture (SOA) [4] as it is state-of-the-art right now and supports flexible reuse and integration of components.
- The platform is a process engine which can be invoked using existing web service orchestration standards such as WSBPEL and which can be combined with JavaEE.
- The proposed solution needs to be able to realize scenarios dealing with ERP processes from SMEs. In such scenarios normally only a few people are involved in developing the processes by taking up the roles of 'business expert' and 'IT expert'.

These assumptions are used to narrow down the approach we take for developing and deploying processes in the ERP domain. This is done by regulating the adjustment screws of MDSD light and proposing an appropriate solution concept (see Figure 2).

- *Technology:* The architectural paradigm used for process execution from an IT point of view is service-orientation. Service-orientation and the composition of services is also the main point of interest from the business point of view. This is the reason why the

abstraction gap between these two view points consists mainly of technical details about service components. Hence, our solution is to use one domain specific model, a domain specific model (DSM) for ERP, where services and service compositions are the central modelling elements. This model encodes information about architecture and different views are provided on this model.

- *Platform:* To produce executable workflow code from the DSM, knowledge about the target platform (in our case JBoss jBPM) is added via model transformation and graph transformations are applied on platform specific level. For example an arbitrary process graph has to be transformed into block-structured WS-BPEL code manually. Our solution uses a code generation framework that provides an integrated solution to these challenges and reuses standard components for e.g. graph transformation. With the combination of domain knowledge (see next point) code generation can be fully automated and the manual refinement step to the platform specific level becomes obsolete.

- *Domain:* A deep knowledge of the application domain allows to specify constraints and checks on the DSM. The constraints and checks are used to control what is allowed to be edited in a certain view and when it is allowed to switch views. They also ensure that the models are in a form so that the (highly-specific) automated code generation can be applied.

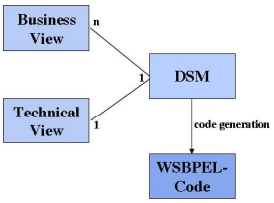

Figure 2: The MDSD light approach

3.2 Implementation

The described approach has been implemented in the AgilPro tools suite. The AgilPro Light Modeller (LiMo) is based on a metamodel that was specificly developed to realize MDSD light. It offers the possibility to easily provide different views for business managers and IT experts. The modelled processes can be simulated and executed on a process engine. Figure 3 depicts the architecture of AgilPro.

The AgilPro LiMo offers technical and business views that do not only consist of different sets of modelling elements. Each view is also connected with a distinct concrete syntax, i.e. notation. For example, in a business view the business manager models with BPMN. He is only concerned about which steps in a process are taken and who is responsible. He is not interested in the technical details such as which web services are invoked, how the data mapping between different applications works, etc.. This is part of the technical view where an IT expert can specify the relevant data for an execution of the process. In the LiMo one can switch between these views and, of course, other views like for the certification for ISO 9000, ITIL processes, etc. are also possible.

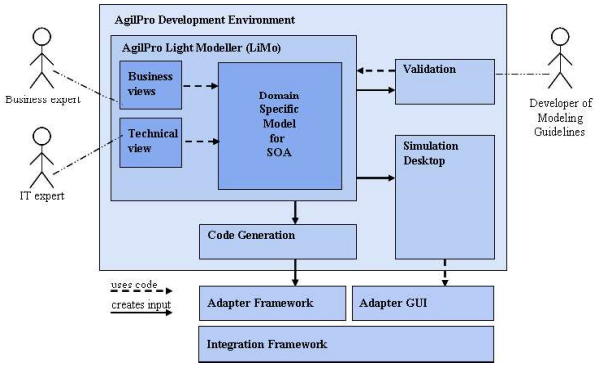

Figure 3: Architecture of AgilPro

The AgilPro Simulation Desktop enables users to simulate and preview processes straight from the model files without any server infrastructure (neither application server nor process engine). There exist several adapters in the Adapter Framework which offer the usage of common tools like Office products, PDF viewer, internet browser, or AJAX. Back end adapters for web services, JDBC, JMS, etc. are displayed in the simulator using the Adapter GUI. It is also possible to validate the modelled processes and to generate code. The Code Generation is a framework the combines standard components for graph-transformation and highly platform specifc generation templates to produce executable workflow code. The integration framework is an abstract API that provides means to run these WSBPEL [10] or XPDL [21] processes on a process engine.

3.2.1 AgilPro LiMo

The AgilPro modelling tool was developed as an Eclipse Rich Client Platform (RCP) application. It uses the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF) to create, display and modify business processes. The metamodel has been implemented as an EMF Ecore model. Ecore is an implementation of EMOF which is part of the well-known MOF [14]. The EMF framework supports automatic regeneration and adaptation of the metamodel implementation whenever changes to the metamodel become necessary. The AgilPro metamodel supports the usage and nesting of sub-processes, grouping of process steps, parallel and alternative branches in a process model, and many more things that are necessary for modelling a business process.

As already describe above, the AgilPro LiMo separates modelling views. These views can be defined in an extra tool (AgilPro Views). One can restrict the graphical presentation of models to some parts of the specific view using the model-view-controller pattern [6] that is realized in GEF. Everyone can create his own view using the plugin infrastructure of Eclipse. One can write additional plugins that use the models to generate code, create documentation for the modelled processes or for any other purpose.

3.2.2 AgilPro Code Generation

The generation of executable workflow code from higher-level, platform-independent process models is challenging.

It can often only be realized be highly complex transformation and generation patterns. However, this complexity arises mostly from the combination of problems that concern different aspects of MDSD. The following list describes the most important problems and how our solution deals with them.

- There exists a hugh diversity of modelling languages and projects providing means to model processes (like UML activities, BPDM, PIM4SOA [1], AgilPro, etc.). Our generation solution decouples code generation from the format of the input models, allows the reuse of code generation algorithms and patterns, and facilitates the coupling of these to input models of arbitrary format with minimal effort.
- An important issue of code generation is the translations of arbitrary process descriptions into constructs that are provided through the target process execution language. WSBPEL for example is a representative of so-called blockstructured languages. Thus we apply more or less complex graph transformations from graph-based to block-based structure (see also [9, 17]).
- Depending on the process execution environment the same external behaviour of process engines is achieved by different process execution code. In many cases the code generation has to encode complex invocation patterns that include knowledge about the respective workflow execution engine. Our solution provides means to easily describe, maintain and reuse these generation patterns independent of other information that is necessary for the code generation.

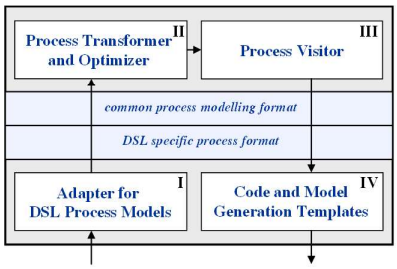

Figure 4: Code Generation Framework

The workflow code generation framework (for more details see [18]) allows to address these problems separately. Figure 4 depicts a structural view on the code generation framework. The *process transformer and optimizer* (II) and the *process visitor* (III) address general graph transformation problems and graph traversing independent to any concrete process modelling language. These two components access process descriptions that are represented in a common process modelling format defined by the framework. For arbitrary processes the framework uses a process modelling format that was derived from the Standard Workflow Models [8]. For block-structured graphs it provides a common process modelling format that is based on WSBPEL. The *adapter for DSL process models* (I) and *code and model generation templates* (IV) directly access the process modelling format of the DSL that is used for modelling the input model.

The workflow code generation framework combines a visitor-based and a template-based code generation approach (see [3]). The process visitor traverses the process flow of the input model and calls templates for workflow code generation. The framework allows graph transformation and flexible code generation for different workflow languages, engines and use cases.

4. CASE STUDY

In the AgilPro project[1] we have applied our approach to automate ERP processes with a real ERP system. In this case study we describe the application of MDSD light by the means of the information that is relevant for business managers, modellers, and software developers. The case study consist of a typical ERP process: the creation of a tender.

The business view in Figure 5 depicts the information that is important for business managers. These are mainly the tasks that are performed, their sequence, and the people that perform the task. In the tender process the sales assistant first creates a tender including addresses, contact person, etc.. Next, he adds all items to the tender. If the total value exceeds 1,000 EUR the sales manager must check the tender conditions, otherwise it is directly uploaded into a B2B portal.

As one can see in Figure 6 the technical view contains more information like the processed data and the executing applications. To insert the tender address the tender header data and a specific ID of the tender is needed. Both is input to the tender management software. This returns the tender draft which is the basis for including the tender items in the next step. Afterwards, the tender is printed to a PDF file and given to the sales manager who can check the tender using a PDF viewer. If he does not stop, the process the tender is uploaded to a B2B portal with a specified URL which is inserted into a browser.

One can easily see that in the business view irrelevant detail is hidden. The technical view displays the same underlying process extend with the details that are necessary for code generation and execution with a process engine. The AgilPro LiMo also allows to use different symbols or presentation froms in the business and the technical views.

Finally WSBPEL code is automatically generated from the refined technical process model. This workflow code is deployed to the process engine of the integration framework and executed via web services. An excerpt of the code for the JBoss jBPM process engine is shown in Listing 1. First a sequence is generated which includes all invoke actions. At the decision a switch block is started that includes two different cases: e.g. for tender with a *value<1000* it does nothing.

Listing 1: Sample WSBPEL code

```
1  <sequence>
2   <invoke name="TenderAddresses"
     partnerLink="TenderAddresses_Prov"
4    portType="TenderAddresses"
     operation="in"
6    inputVariable="TenderID"
     inputVariable="TenderHeader"
8    outputVariable="Tender"
    </invoke>
10  <invoke name="IncludeTenderItems" ... />
    <switch>
```

[1]http://www.agilpro.eu/

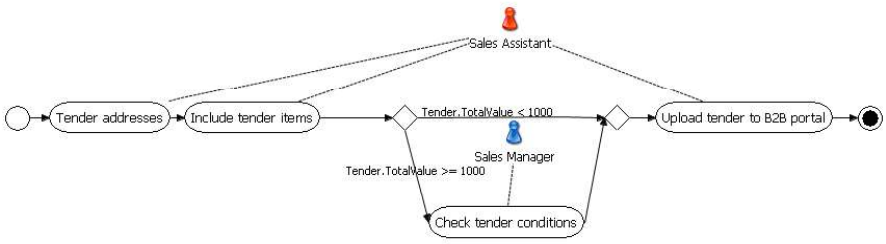

Figure 5: Business view of a sample ERP tender process

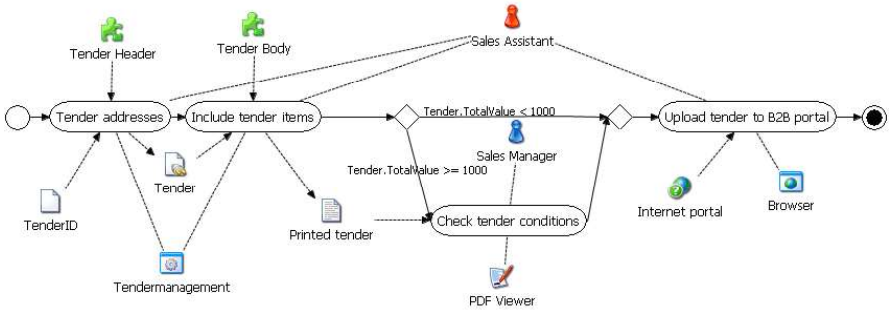

Figure 6: Technical view of a sample ERP tender process

```
12   <case "Tender.TotalValue < 1000">
      <empty/>
14   </case>
     <case Tender.TotalValue >= 1000">
16    <invoke name="CheckTenderConditions" ... />
     </case>
18  </switch>
    <invoke name="UploadTenderToB2BPortal" ... />
20 </sequence>
```

5. RELATED WORK

The problems that SMEs face when trying to introduce model-driven software development in their companies and an iterative solution have already been described in [20]. However, the authors describe only initial results in the introduction of a fully-fledged MDA approach for SMEs, but do not restrict MDA to the relevant parts such as we do.

We are aware of different tools and tool suites that offer similar functionality as the described approach. There are a variety of process and workflow modelling tools as well as WSBPEL modellers: The IBM Tool Suite allows to model processes with the WebSphere Business Modeler which can be exported to WSBPEL using the WebSphere Process Server. However, these suite only supports a fully-fledged MDSD solution with multiple model transformations and human refinement steps. Other tools (e.g. Oracle BPEL Process Manager or ActiveEndpoint's ActiveBPEL) that offer direct WSBPEL code generation, restrict process modelling to block-structured graphs. They do not provide the abstraction level for process modelling that seems adequate for business managers. Tools that provide higher abstractions for process modelling like the TIBCO Business Studio or CARNOT Process Workbench do not provide or provide only limited export of WSBPEL code. Other examples are the BONAPART BPEL-Modeler and the tool objectiF by the microTOOL company.

As the AgilPro metamodel and also the integration framework is based on services we also had to take into account service-oriented modelling approaches. In the development of the metamodel we also profited from service modelling and composition approaches and UML-profiles like PIM4SOA [1] or UPMS [15] where we were partially involved in. Within AgilPro we integrate existing WSDL services, but we don't generate WSDL descriptions since we focus on service composition. Modelling the web service interface and generating WSDL code would for example be possible using PIM4SOA or UPMS.

The MDSD light approach and its application for the ERP domain and SMEs shares some characteristics with other domain specific modelling approaches. It provides a highly customized solution to a problem domain. It also supports direct and automated code generation from models from an abstraction level that represents domain and not only IT concepts. Compared to other approaches like software factories [7] or the MetaCase tool suite MDSD light builds on the OMG's MOF standard for metamodelling that allows a seamless integration with other MDA and UML tools.

6. CONCLUSIONS

This paper presented the MDSD light approach and described its characteristics and adjustment screws. We presented its application and implementation via the AgilPro tools suite. This has been illustrated with a tender process of the ERP domain.

We are well aware that our approach and its realization depends on further requirements and also has some limitations we did not discuss in this paper yet:

- The domain needs to be clearly defined and the DSM needs to be created for one domain and is often not applicable to other domains.
- A methodology is needed for process modelling.
- Special tool development or adjustment is necessary.
- Within MDSD light solutions there are only few variation points, so the possibility of making changes is limited.
- As the models get more and more complex, the costs for creation and maintenance increase.

Nevertheless, the MDSD light approach is perfectly usable to model and execute processes in several domains. In AgilPro we currently offer the adjustment screw for the domains, so a model can be adjusted to ERP, SCM, Logistics, etc. The technology on the other hand is predefined and the platform normally won't be changed by the modeller. However, through an adjusted code generation the AgilPro solution could be configured to support other platforms. AgilPro differs in the following from fully-fledged MDSD solutions:

- AgilPro has one DSM and supports several views (business views, a technical view, etc.).
- There is no need to perform separate model transformations between the three abstraction levels of MDA.
- The architecture model and the platform model are captured in the domain specific model and in the code generation framework.

AgilPro is one of the cornerstones of the Eclipse Technology project Java Workflow Tooling (JWT) and will be developed and enhanced by a bigger community in the future. In JWT we not only envision the continuous usage of business process models from design to execution, but also a general mechanism and interoperability format to work with e.g. different process engines and different modelling notations.

7. REFERENCES

[1] Benguria, G., Larrucea, X., Elvester, B., Neple, T., Beardsmore, A., and Friess, M. A Platform Independent Model for Service Oriented Architectures. In *Enterprise Interoperability: New Challenges and Approaches, Proceedings Of the 2nd International Conference on Interoperability of Enterprise Systems and Architecture (I-ESA'2006), Bordeaux, France* (April 2007), Springer, pp. 23–34.

[2] Bézivin, J. On the unification power of models. *Software and System Modeling 4*, 2 (May 2005), 171–188.

[3] Czarnecki, K., and Helsen, S. Feature-based survey of model transformation approaches. *IBM Systems Journal 45*, 3 (July 2006), 621–645.

[4] Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall International, 2005.

[5] Frankel, D. S. *Model Driven Architecture - Applying MDA™ to Enterprise Computing.* John Wiley & Sons, Inc., New York, NY, USA, 2003.

[6] Gamma, E., Helm, R., and Johnson, R. E. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley Longman, 1995.

[7] Greenfield, J., Short, K., Cook, S., and Kent, S. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools.* Wiley Publishing Inc., 2004.

[8] Kiepuszewski, B., ter Hofstede, A. H., and van der Aalst, W. M. Fundamentals of control flow in workflows. *Acta Informatica 39*, 3 (March 2003), 143–209.

[9] Mendling, J., Lassen, K., and Zdun, U. Transformation Strategies between Block-Oriented and Graph-Oriented Process Modelling Languages. In *Multikonferenz Wirtschaftsinformatik (MKWI)* (2006), vol. 2, GITO-Verlag, pp. 297–312.

[10] OASIS. Web Services Business Process Execution Language Version 2.0. wsbpel-primer, May 2007.

[11] OMG. MDA Guide Version 1.0.1. omg/2003-06-01, June 2003.

[12] OMG. MOF QVT Final Adopted Specification. ptc/05-11-01, November 2005.

[13] OMG. Business Process Modeling Notation Specification, Final Adopted Specification. dtc/06-02-01, February 2006.

[14] OMG. Meta Object Facility (MOF) Core Specification, version 2.0. formal/06-01-01, January 2006.

[15] OMG. UML Profile and Metamodel for Services - for Heterogeneous Architectures (UPMS-HA). ad/2007-06-02, June 2007.

[16] OMG. Unified Modeling Language: Superstructure, version 2.1.1. formal/07-02-05, February 2007.

[17] Ouyang, C., Dumas, M., ter Hofstede, A. H., and van der Aalst, W. M. Pattern-based translation of BPMN process models to BPEL web services. *International Journal of Web Services Research (JWSR)* (2007).

[18] Roser, S., Lautenbacher, F., and Bauer, B. Generation of Workflow Code from DSMs. In *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling* (October 2007).

[19] Scheer, A.-W. *ARIS Modellierungs-Methoden, Metamodelle, Anwendungen.* Springer, 2001.

[20] Vogel, R., and Mantell, K. MDA adoption for a SME: evolution, not revolution - Phase II. In *Second Workshop "From code centric to model centric software engineering: Practices, Implications and ROI"* (July 2006).

[21] WfMC. Process Definition Interface – XML Process Definition Language. WFMC-TC-1025, October 2005.