MDA Organization Platform: A Holistic Approach for the Management of Model-Driven Architectures

Andreas Emrich[1], Dmytro Panfilenko[1], Sebastian Weber[2]

[1]*German Research Center for Artificial Intelligence (DFKI)*
Stuhlsatzenhausweg 3, Campus D3.2, 66123 Saarbrücken, Germany
{andreas.emrich|dima.panfilenko}@dfki.de

[2]*Fraunhofer Institute for Experimental Software Engineering (IESE)*
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
sebastian.weber@iese.fraunhofer.de

Abstract

Nowadays, there is no integrated system that allows for complete carrying out of the model driven development based on MDA. In addition, transforming CIM to PIM is regarded as manual and not further investigated.

Hence, we propose a holistic approach that facilitates conceptual development of an MDA management platform under change considerations and traceability through CIM, PIM and PSM models and code generation.

Semantic meta descriptions of these models along with facilities for impact analysis and cost estimations allow for keeping track of changes. Moreover, plug-in-based code generation is further core functionality of the system. Finally, through semantic role associations the responsible personnel can be contacted in case of need for the manual changes on the system.

Overall, the platform allows for the execution of defined change processes with both automatic and manual transformation tasks. It regards not only technical processes but also focuses on the integration of relevant personnel.

Keywords: *model-driven architectures, MDA management, SPACE*

1. Introduction

In the field of model-driven architectures there are a lot of approaches for the setup and execution [1]. Nevertheless, it is often a view that is based on a single creation process; the entire software lifecycle is seldom addressed [2] [4]. Consequently, changes should be tracked between different artifacts and throughout different model layers. The human perspective is also important, as manual transformation steps should be addressed to the respective process owners.

Accordingly, we try to adapt the concept of traceability to the domain of model-driven architectures. Traceability enables engineers to keep track of all changes, which occur in a model-driven architecture. We will analyze, what impact is caused by changes and how consequent transformation tasks can be triggered. Semantics will be used to create a semantic metadata infrastructure for model-driven architectures, which allows for the seamless, semantic capturing of the overall model.

To achieve this, we will adapt the SPACE (Semantic Process- and Artifact-oriented Collaboration Environment) concept to the area of model-driven architectures. This paper will highlight the MDA Organization Platform (MOP) as an implementation of the SPACE concept. Different use cases throughout the software development lifecycle will be analyzed. It will be shown, how full traceability over all artifacts contained in a model-driven architecture can be achieved and what potential benefits it brings throughout the software lifecycle.

Related work will be discussed in section 2 with a focus on change management and cost and impact analysis. In section 3 we will describe contemporary MDA tools and analyze their suitability from a change management perspective. Section 4 will discuss some relevant aspects of the Semantic Process- and Artifact-oriented Collaboration Environment and how they are applied to the context of model-driven architectures. Section 5 will show different scenarios in terms of changes that occur in the context of model-driven architectures, and how the proposed solution could improve the management of model-driven architectures in this area. We will end the paper with some conclusions and directions for further research.

2. Related Work

This section provides an overview on the known related research efforts on the topic of this paper and the nearest adjacent ones. The work related to the problematic of this paper comprises not only the general question about how to model the CIM-level in MDA and to transform developed models into PIMs and PSMs with the resulting code. Even more arises the following question: how to assess the changes a user of an integrated MDA management system makes, how to track them inside a modeling level and moreover between levels as well as how to model the role structure and dependencies in order to make the changes transparent for the according users and to notify them in case the manual changes have to be performed. The code generation via plug-in mechanism is another important question apart from the alreadz mentioned. Thus, the relevant research work being overviewed here addresses the conceptual framework development for MDA support or parts of this problem.

The work of Gruhn [5] stands closely to the topics being under concern in this paper with its activity and artifacts produced during the engineering process aligned with MDA as well as the role structure for the project being under development.

The activities Gruhn elicitates in his work are: 1. Qualify and 2. Analyze the domain of interest, 3. Implement the framework, 4. Model the system; 5. Execute the transformation between different levels of modeling and 6. Finally get the feedback about the completed steps.

The results of the activities should be accordingly: 1. The economic model of a developed system, 2. The domain specific languages (i.e. constructs for modeling CIMs, PIMs and PSMs), 3. The implemented framework prototype or even a product, 4. The set of all models for the system (i.e. all CIMs, PIMs and PSMs not created through transformations), 5. All the models and artifacts generated out of higher-level models and system source code, 6. An evolution document comprising changes to the system and assessment of the completed work.

The role structure and assignment of the activities mentioned are basically divided into two not strictly separated pools of domain and application engineers, thus showing the difference between domain experts having the knowledge about the environment and the technical experts finally implementing the modeled system. As to this extent, the system proposed in this paper should comprise the mentioned functionalities and extend them with at least impact and cost analysis, traceability and notification mechanisms.

Another research effort in this area is the CASSANDRA research platform developed by KnowGravity Inc. in Zurich, Switzerland (see also section Existing Tools for the tool description) [11]. The core of this system and the main idea about it is that it is assistant-based and guides the users along the MDA line helping them in their tasks by giving hints in business, systems and software engineering. Its functionality is based on the project data analysis from other similar CASE tools and generation of the according questions or hints for the next steps for making progress forwards in the development process. CASSANDRA comes in two variants. CASSANDRA's core components are: REMEMBER – an active declarative database repository of all project information, KNOW – an optional component that is accessing the domain model of the world and THINK – a rule-based engine that is able to infer the propositions for the next steps, questions and other general information related to the project and also explain the information provided. There are also additional component as interface and application agents in the CASSANDRA architecture providing adaptation functionalities for different external CASE tools and service components respectively. The platform for which an approach is proposed in this paper has a different architecture maybe not that much like the human brain components drawn in CASSANDRA architecture. The knowledge, remembering and thinking functionalities are also present in the MOP architecture in another way, providing abilities to trace the model changes throughout the MDA-based engineering process and estimate the costs emerging from a single model element addition on one of the levels, as well as the role management features.

The last work in this overview concerns the approach to enterprise architecture models and their maintenance proposed by Fischer [6]. Its main thought turns around alignment of the business domain experts and IT specialists, as it usually can be seen in much research approaches concerning separation between the business- and technology-oriented architectures. The interesting point in this work is that it emphasizes the importance of the permanence of the enterprise architecture management process and the certain and unavoidable need for a well-thought concept for keeping the enterprise models aligned, up-to-date and ready-to-use. For this, states the work, there should exist elaborated concepts that are not only process-oriented, but also include role assignments and role management and pay attention to scheduling of the maintenance processes. The authors also agree that the interfaces to different other architectures are needed in order to have a complete approach for enterprise architecture, among others to the data and metrics architectures. Although the ideas in this work are very related to the concepts of the current paper, we choose a holistic approach instead of the federated one chosen in the references work for the reasons that can be seen in sections 4 and 5, where the motivation is given for the approach and its methodology is presented.

3. Existing Tools

In this section we provide an overview over existing tools that partly or to some extent support the MDA idea through providing modeling features on different levels as CIM, PIM or PSM, M2M-transformations on or between these levels or even organizational features that are helpful for managing development processes.

There are five tools we provide overviews for in this paper, namely AndroMDA [7], PowerDesigner [8], Rational family [9], Modelio [10], and KnowGravity's CASSANDRA [11] and KnowEnterprise [12]. For each of these tools, the feature highlights are addressed first and then the support for modeling on different MDA-levels as well as for M2M-transformation is questioned. In this way, we will see which aspects of MDA are mostly covered in the practice of model-driven engineering. Of course, there are many other tools we can't afford to give a survey on in this paper, but the tool choice here should give a good snapshot of the existing tool support for MDA.

AndroMDA (pronounced "Andromeda") is an open source tool supporting many features including UML 1.4 modeling (UML 2.0 is under development) and deployment of the modeled content onto different platforms (J2EE, Spring, .NET). For the latter a mechanism dealing with so-called cartridges is implemented in this tool. This is basically a set of transformation prescriptions targeting different platforms as Spring, EJB, Hibernate, Struts, JDF etc. In addition to UML modeling support, AndroMDA supports other existing UML-modeling tools like MagicDraw, Poseidon and Enterprise Architect. That said, we can see that the AndroMDA, as the tool description itself states, is basically a transformation engine offering modeling support for PIM- and PSM-levels as well as transformation to code for a number of platforms that can be additionally defined by users. The CIM-level modeling is not explicitly supported or mentioned.

Sybase provides a commercial modeling tool *PowerDesigner* for enterprise architecture modeling, which supports several modeling techniques as data, application and business process modeling on different levels of abstraction as conceptual, logical and physical. The standards PowerDesigner uses for the above mentioned modeling are among others UML, BPMN and BPEL4WS. In addition, there is a repository storing the models created with help of PowerDesigner, which supports standard features like version control and merging as well as advanced features like team solution (multiple users on the same model at the same time), meta-data management and security. Overall, it is a powerful tool offering modeling on the CIM-, PIM- and PSM-levels as well as a bridge to the execution environments through support of the BPEL export.

IBM's *Rational family* is a well-known commercial tool family supporting modeling of the different aspects of the enterprise architecture with established standards like UML targeting different programming languages (e.g. Ada, ANSI C++, Java, Visual C++ and Visual Basic). The requirement modeling is paid special attention with the Requirements composer facilitating integrations between modeled requirements, defect and change tracking. The next feature supported by the Rational family is its configurable process, which selects only the process components needed for the development process. It also supports Model-Driven Development with patterns identification and provides functionality for round-trip engineering – enables to model the application, generate the code elements, then modify and implement the code as necessary. This tool offers support for modeling on CIM-, PIM- and PSM-levels with code generation to different programming languages and some support for horizontal and vertical traceability.

Modelio is a famous commercial modeling tool with explicit model-driven development support from Modelio software, including extensive UML and BPMN support as well as some basic features for enterprise-level modeling. Not like AndroMDA, Modelio already provides support for graphical modeling of the UML profiles using UML 2.0 diagrams, thus exploiting further features of this modeling language with respect to adaptability and traceability. Its support for BMM [13], BPMN and SoaML [14] states its strong tendency to work with SOA applications. Especially the BMM support is currently under development and extension, including goals, rules and organization modeling and standards support. The generators Modelio is using are at the time targeting Java, C# and C++, which allow for code generation without any programming efforts. In addition, teamwork solutions are well supported by a unique shared repository throughout the whole development cycle. As we can see, Modelio offers support for modeling on CIM-, PIM and PSM-levels, for code generation to different programming languages and some support for role management in the team solution.

The last tool presented in this overview is a pair of KnowGravity's CASSANDRA and Know Enterprise – a research platform series of the modeling tool including support for CIM-level modeling and transformation support from PIM to PSM. Here we have a division of the enterprise modeling features as motivation, vocabulary, rule and process views for business and requirements, xUML, architecture and deployment views for IT in KnowEntreprise on the one hand and the assisted business, systems and software engineering in CASSANDRA on the other hand. In the enterprise modeling tool KnowEnterprise there is support for the relevant modeling standards like BMM for motivation, SBVR for business rules and BPMN for business processes as well as OSM for organization modeling. At the same time teamwork relevant features are supported, too: model version control, sharing and multi-user access authorization as well as fine-grained change logging, which is useful for further research tasks addressed in CASSANDRA. The latter makes estimation of the usages of the models and changes on them for the basis of the generation of the next steps proposals for the progress in the engineering process. KnowEnterprise and CASSANDRA provide modeling support on CIM- and PIM-levels, M2M-transformation definition and change analysis as well as next step proposals as the basis for teamwork solution, which is interesting in the research aspect and not to be found in other commercial tools.

4. SPACE - The Semantic Process- and Artifact-oriented Collaboration Environment

The Semantic Process- and Artifact-oriented Collaboration Environment (SPACE) is a concept for a semantic meta-model infrastructure that aims at semantically describing the whole perspectives of a collaboration context [16] [17]. In our scenario, this would be a management environment for model-driven architectures. It features the process and artifact models. The relationships among processes and artifacts can be used to run analyses across different kinds of information artifacts. Overall, it will facilitate end-user friendly creation, management, and execution of process and artifacts models.

The objective of SPACE is to facilitate the management of meta-models and the model instantiation with the help of visual editors and templates, respectively. Templates are generated on the basis of the modeled artifacts. See Fig. 1 for an example of an artifact described on meta-level and the corresponding artifact template.

Artifacts are organized in artifact models that contain related artifact types (e.g., requirements engineering artifacts, such as use cases) but also in process models that contain process artifacts (e.g., requirements elicitation, use case creation, etc.). SPACE allows the definition of models on any granularity level. As an example, there might exist an artifact model that contains only the artifacts of any software phase required for the distribution to the customer (e.g., requirements specification or the documentation of the software system). These artifacts are connected to other artifacts of other artifact models (e.g., the requirements specification artifact is connected to use case artifacts, interview artifacts, etc., that are used as input).

Process models can be seen as specialization of artifact models because they base on the same concepts but have a few more characteristics, which are fully described in [17]. The semantic relationships in the process description also offer many other opportunities. The semantic information moreover can be used to include process-related experiences and best practices in the platform view of the processes.

The inner structure of artifacts (i.e., attributes) as well as the relationships to other artifacts (see Figure 2 as an example) constitutes semantic data that are implicitly part of the generated visual templates. In the artifact instantiation process, the users complete the templates and link artifact instances and might provide manual semantic annotations via tags. The semantic data are the foundation for the proactive recommendation facilities [16]. The templates support an easy way to capture and package experience.

The MDA Organization Platform (MOP) will be a model-specific implementation of the SPACE approach. Overall, process models will be used for software development processes as well as change management or knowledge management processes. Especially, transformation tasks are in the focus of these process models. First of all, all model artifacts specified and generated in an MDA are artifacts in the context of MOP. However, as MOP seeks to be a holistic approach, also any relevant object in the context of the original artifacts will be also regarded as an artifact in the context of MOP. The respective process owners for transformation tasks will be modeled and assigned to their respective transformation processes and the associated model artifacts.

In the context of the Software Organization Platform (SOP) [19] a prototype has been developed that facilitates the management of artifacts throughout the entire software lifecycle. The concepts of SOP will be partially adapted to MOP.

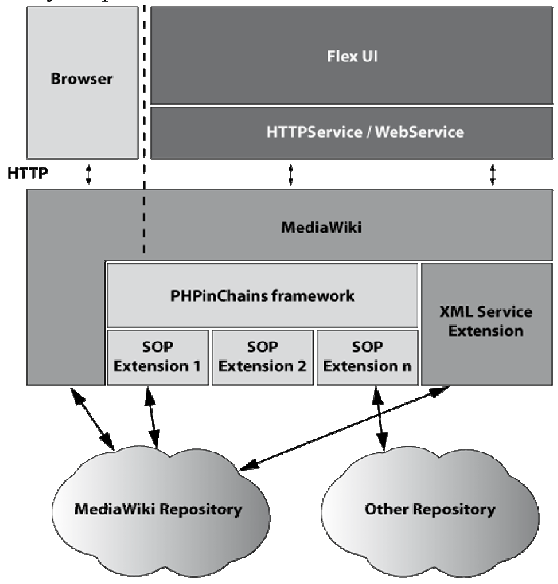

Fig. 1. Architecture of the Software Organization Platform [18]

The artifacts are represented by wiki documents stored in MediaWiki. The extension Semantic MediaWiki [20] facilitates the semantic annotation of these artifacts, and allows for the linkage of related MDA artifacts. The user interface empowers software engineers to create, modify and associate artifacts. The Adobe Flex UI facilitates end-user-friendly support for the graphical modeling of process and artifact models. The PHPinChains [18] framework along with the XML Service Extension allows for the integration with any kind of applications. By this means, code generation or external tools can be triggered within the MDA Organization Platform.

The focus of this paper is to show the concept of the MDA Organization Platform. Further research should describe the processes and artifacts for model-driven architectures that need to be modeled within MOP.

5. Second and following pages

As stated in the introduction change management implies the need for a flexible and simple yet completely traceable management of artifacts within a model-driven architecture. In this section, we will show what traceability means within the context of the MDA Organization Platform (MOP). Then according to that, we will demonstrate how traceability can improve transformations corresponding to changes that occur during the lifecycle of a model-driven architecture.

Traceability in Model-driven Architectures

Traceability originates from the domain of requirements engineering and describes the relationships between requirement artifacts. Literature distinguishes between horizontal and vertical traceability [15]. Horizontal traceability analyses the relationships among requirements, whereas vertical traceability explores how requirements are used in consequent phases of the software development process. Newer approaches also define traceability more shallow, in a sense that it analyses the traceability among all artifacts.

In the context of the MDA Organization Platform traceability is the main property among different artifacts that allows for analyses according to changes. The artifacts can be different types of artifact documents as in the commonly known software engineering understanding of traceability. Moreover, it enables to incorporate additional information from the context such as associated roles, persons or help documents, etc. Thus, traceability is the key enabler for the MDA Organization Platform.

MDA Management Use Cases

Changes are considered to be an important cost factor. Often, maintenance is considered to be the most expensive phase in the software development lifecycle. With MDA as a top-down approach, this is often neglected. [21]

The following use cases will demonstrate how change management can be applied to model-driven architectures, and how MOP supports change management scenarios for model-driven architectures.

Horizontal Traceability

In the context of model-driven architectures we define horizontal traceability on artifacts within a specific model level. All semantic relationships that are analyzed with respect to horizontal traceability describe two or more artifacts on the same model layer.

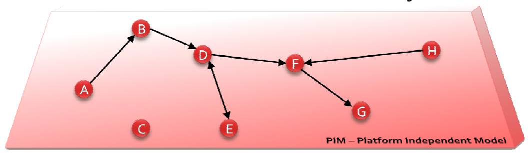

Fig. 2. Horizontal Traceability in the MDA Organization Platform

The figure shows how the different artifacts are interrelated in a Platform Independent Model (PIM). Artifact B depends on artifact A, i.e. if a change occurs to A, B must be adapted accordingly. However, there are also artifacts that do not participate in any kind of semantic relationship on a specific model layer and thus do not cause any effects on other artifacts, e.g., C.

The semantic relationships can be unidirectional or bidirectional. For outgoing unidirectional and bidirectional relationships the related artifacts must be changed as well. E.g., if B changes, the related artifacts D, E, F, and G must be changed accordingly. This is not the case for incoming unidirectional relationships such as for B and A. If B changes, A does not need to be updated. In case of the bidirectional relationship between D and E, a change on E would also cause changes for D, and consequently, also for F and G.

Vertical Traceability

From our point of view, vertical traceability addresses not a single model layer, but semantic relationships between artifacts on different model layers. E.g., a specific goal defined in the Computation Independent Model can be represented by different model elements, such as classes, processes, etc., on the Platform Independent Model.

The following figure shows how vertical traceability can be shown between artifacts on different model levels in a model-driven architecture. The figure depicts just the CIM and PIM level for simplicity reasons.

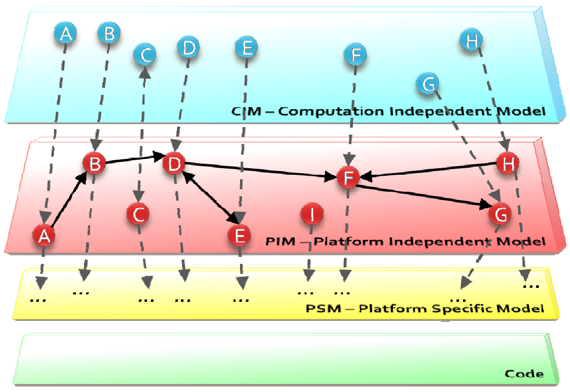
A
B
C
D
E
F
H
G
CIM – Computation Independent Model
B
D
F
H
A
C
E
I
G
PIM – Platform Independent Model
...
PSM – Platform Specific Model
Code

Fig. 3. Vertical traceability in the MDA Organization Platform

The figure shows unidirectional as well as bidirectional semantic relationships between artifacts on different model levels of the model-driven architecture.

Due to the nature of these relationships, there are significant differences for the classical top-down approach or for the bottom-up perspective. The following subsections will describe, in which scenarios the respective approach is useful and how traceability is supported in these scenarios by the MDA Organization Platform.

Top-Down Traceability

The traditional MDA approach derives the system via transformations on different model layers. The business goals and user requirements (CIM level) are aggregated in a conceptual design of the system (PIM). This design is refined for the ultimate goal of implementation on the PSM level and then code is generated according to the information provided in the models.

When developing with MDA, the traceability of artifacts can be helpful in different situations. It can identify further courses of action that can be taken by the respective engineer / developer. Furthermore, it can provide a full overview of all artifacts that have to be changed according to the changes already applied.

Overall, the top-down traceability enables to apply the needed transformation tasks whenever changes occur.

Bottom-Up Traceability

Despite the nature of model-driven architectures, it is also desirable to have bottom-up views on the model-driven architecture and its contained artifacts. First of all, there are bidirectional relationships that could cause changes on higher-level model layers. E.g., in figure 6, the artifacts C in the CIM and PIM level are connected by a semantic relationship; i.e., if a change occurs to C on the PIM level, it must also be applied to C on the CIM level.

Moreover, this implies a new question: Can we (automatically) decide what to do on higher-level model layers. Model-driven architectures heavily take advantage of automatic transformation tasks and code generation. However, this works fine for top-down approaches, as information gets extended but not reduced. When looking at the bottom-up approach, there are both fuzziness in the transformation, and manual transformation tasks. The fuzziness results from the fact, that lower-level information are usually richer in content than information on the higher levels. Therefore it is hard to decide automatically, what do in these cases.

One approach to overcome this problem would be to refine the metadata representation, i.e. the templates in MOP, for describing the MDA artifacts. It could be defined, which data can be transferred from one level to the next. However, this leads to a considerable amount of effort for describing the artifacts. Moreover, it is questionable, whether such a solution could cover the majority of use cases in this area.

When also considering manual transformation tasks, obviously human decisions are necessary to proceed.

People in MDA: Manual Transformation Tasks

Manual transformation tasks are a crucial part of transformations in model-driven architectures [3]. Many transformations cannot be decided automatically, because the underlying information is too abstract to be formalized in a machine-readable way.

Thus, people have to perform these tasks themselves. As MOP seeks to be a collaboration platform that allows for editing the model-driven architecture and its related artifacts in groups, it incorporates role concepts. The roles can either be defined explicitly or can be gathered manually by the actions of the user. E.g., if a user creates an artifact, he is automatically assigned as the contact person for this artifact, unless it is explicitly changed.

These role concepts can be linked to certain processes, i.e. transformation tasks, and artifacts. So whenever a change occurs to related artifacts respective transformation tasks are triggered. Whenever a transformation needs human input, the role information along with the semantic relationships to artifacts and processes, can be leveraged to proactively notify the process owners.

This is not only helpful in change management scenarios, but also for developers in a collaborative project, that need input to a specific transformation task or artifact.

Analysis of Impacts and Costs

As we already saw, semantics do not only support better opportunities for handling change management issues in model-driven architectures, but also helps to analyze the model-driven architecture itself and to organize its project context. In software projects the monetary costs and also timeliness are crucial success factors [21].

Traceability provides the technique to analyze all changes that have to be performed, when a certain event occurs. However, it does not make any assumptions about the costs.

Costs models can either be explicitly assigned to certain transformation tasks or can be gathered by monitoring the actions of users within the MDA Organization Platform. Consequently, the costs in terms of money and time can be estimated for (a set of) transformation tasks.

Especially, when design decisions have to be made, this can be a helpful assistance. If transformation task A and transformation task B would both resolve a conflict, a cost estimation could help to decide, if it is better to perform task A or task B.

6. Conclusions and Outlook

The MDA Organization Platform allows for the seamless management of artifacts of a model-driven architecture throughout the entire software development lifecycle. It empowers developers and software engineers to keep track of all the changes occurring in the context of a model-driven architecture. Furthermore, it helps to understand the MDA much better through the extensive traceability support. It aids with identifying appropriate contact persons for problems and enables easy-to-use cost estimations for transformation tasks. Overall, this makes decisions more transparent for the user.

For future work additional considerations have to be made. The better the context of MDA artifacts is described, the more effective the support provided by MOP can be. Thus, the context must be described with appropriate artifact models and must be semantically intertwined with the descriptions of MDA artifacts.

The integration with existing MDA tools will also be of utmost importance, as MOP will not be focused on the actual modeling within MDA. The MOP concept describes an approach for the semantic management and organization of MDA artifacts. The artifacts themselves are edited with external MDA tools which are integrated with MOP. This also implies that these tools expose appropriate interfaces in order to be interoperable with MOP.

Overall, the concept of the MDA Organization Platform, as presented in this paper, enables models to be steadier against changes throughout the software lifecycle and thus lowers the effort for maintenance of MDAs. It not only seeks the technical integration with several existing MDA tools but also has a strong focus on the user: Users are proactively informed when relevant changes occur and get recommendations for appropriate courses of actions. Therefore, MOP is a holistic approach for the management of model-driven architectures, as it incorporates social and technical solutions for the integration in the respective organization and its tool landscape.

7. References

[1] Fettke, P.; Loos, P.: Model Driven Architecture (MDA). In: Wirtschaftsinformatik, Bd. 45, 2003, Nr. 5, pp. 555-559.

[2] Frankel, D.: Model Driven Architecture. Applying MDA to Enterprise Computing, Wiley & Sons, 2003.

[3] Frankel, D.: Toward a Business Process Platform. In: MDA Journal, July 2005, pp. 1-7, 2005.

[4] Object Management Group: Model Driven Architecture (MDA) http://www.omg.org/docs/ormsc/01-07-01.pdf, 2001.

[5] Gruhn, V.; Pieper, D.; Röttgers, C.: MDA. Effektives Software-Engineering mit UML 2 und Eclipse. Springer, Heidelberg 2006

[6] Fischer, R., Aier, S., Winter, R.: A Federated Approach to Enterprise Architecture Model Maintenance. In: 2nd International Workshop EMISA, pp. 9-23.Gesellschaft für Informatik, Bonn (2007)

[7] AndroMDA MDA Toolkit, http://andromda.org/

[8] Sybase PowerDesigner, http://www.sybase.com/products/modelingmetadata

[9] IBM Rational Software, http://www-01.ibm.com/software/rational/

[10] Modelio Modeling Solutions, http://www.modeliosoft.com/

[11] KnowGravity CASSANDRA knowledge engineering tool, http://www.knowgravity.com/eng/value/cassandra.htm

[12] KnowGravity KnowEnterprise enterprise-scale modeling environment, http://www.knowgravity.com/eng/value/knowEnterprise.htm

[13] Object Management Group: Business Motivation Model (BMM) Specification. http://www.omg.org/spec/BMM/1.0

[14] Object Management Group: Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS). http://www.omg.org/docs/ad/08-08-04.pdf

[15] Gotel, O.; Finkelstein, A.: An Analysis of the Requirements Traceability Problem Proc. of First International Conference on Requirements Engineering, 1994, pp. 94-101

[16] Emrich, A.; Weber, S.; Ras, E.: Towards Proactive and Intelligent Assistance in the Software Organization Platform. 11th Workshop on Learning Software Organizations (LSO 2009). Oulu, Finland, 2009.

[17] Weber, S.; Emrich, A.; Broschart, J.; Ras, E.; Ünalan, Ö.: Supporting Software Development Teams with a Semantic Process- and Artifact-oriented Collaboration Environment. Proc. of the SOFTEAM'09 Workshop. Kaiserslautern / Germany, 2009

[18] Linnenfelser, M.: Realisierung einer flexiblen Pattern- und Plug-in-basierten Architektur für die Software Organization Platform (SOP) zur verteilten kooperativen Softwareentwicklung auf Basis von Adobe Flex. Master Thesis, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern / Germany, 2008

[19] Weber, S; Thomas, L.; Armbrust, A.; Ras, E.; Rech, J.; Ünalan, Ö.; Wessner, M.; Linnenfelser, M.; Decker, B.: "A Software Organization Platform (SOP)," in 10th Workshop on Learning Software Organizations (LSO 2008). Rome, Italy, 2008.

[20] Vrandecic, D.; Krötzsch, M.: *Semantic MediaWiki*. In: Davies, J. et al.: Semantic Knowledge Management. Springer, 2009, pp. 171-179

[21] Sommerville, I.: Software Engineering. Addison Wesley, 2001