Process Driven Architecture: A Model Driven Development Approach for Process Support Software

Sascha Müller[1], Stefan Jablonski[2] and Matthias Faerber[2]

[1]Ansbach University of Applied Sciences, Residenzstr. 8, 91522 Ansbach, Germany

[2]Chair of Applied Computer Science IV, University of Bayreuth
Universitätsstrasse 30, 95447 Bayreuth, Germany

Abstract. In this paper we propose a new model based development method specialized on the efficient production of process support software, called the Process Driven Architecture (PDA). It is based on our experiences in the domain of clinical process support and a comprehensive review of related model driven software approaches, such as the OMG's Model Driven Architecture (MDA) or the Software Factories approach. The exemplary implementation of a clinical process illustrates the feasibility of our PDA approach.

1 Supporting Clinical Research Processes

The clinical application domain has been a great challenge for information technology solutions for decades [2][5][15][3] and is sometimes even considered as the "killer application" in the field of process support [5]. This estimation is based on the high complexity of the medical domain [2][15], which results from multiple factors. According to [2][4][14], these factors can be divided into two groups: technical and socio organizational. One of the most challenging technical factors is flexibility with respect to process modeling and execution [5]: process management must not restrict physicians or nurses in their decisions about the next treatments to undertake, but must silently assist and support decisions of the medical staff. A second major technical factor is fast adaptation to new requirements. Typical reasons for this are new medical research findings and organizational or juridical amendments [16].

We have discussed in several publications that the above mentioned requirements can be met using domain-specific process models [10] [12]. Due to their inherent flexibility domain-specific process models require sophisticated tool support. This starts with the modeling and ends with the execution of tailored domain-specific processes. In order to cope with this challenging technological need we propose to implement domain-specific process models according to the MDSD (Model Driven Software Development) methodology [20]. This facilitates rapid provision of tools for process modeling and execution that are tailored to specific application domains. After a short introduction into MDSD (Section 2) we present our approach that is called Process Driven Architecture (PDA, Section 3). Section 4 then exposes an example from a clinical application implemented according to the PDA approach.

Müller S., Jablonski S. and Faerber M. (2007).
Process Driven Architecture: A Model Driven Development Approach for Process Support Software.
In *Proceedings of the 3rd International Workshop on Model-Driven Enterprise Information Systems*, pages 73-82
DOI: 10.5220/0002432000730082
Copyright © SciTePress

2 Model Driven Software Development

The notion of MDSD comprises several software development approaches sharing the same core idea: Efficiently develop software based on models using a domain-specific modeling language (DSL). Our PDA approach is especially influenced by two MDSD approaches: The Model Driven Architecture (MDA) and Software Factories (SF) [8].

The Object Management Group's (OMG) MDA [19] is an industry standard for the model driven development of software based on the MOF standard [18]. In contrast to the fuzzy definition of MDSD, the MDA standard is more concrete and prescriptive. The main difference between the MDSD and the MDA can be summarized by three observations:

- Predefined development process: The MDA proposes a model driven software development process with four layers of abstraction.
- MOF compliant DSL: MDA requires the conformity of the DSL to the MOF.
- Specification languages for transformation: The transformations between the language layers must be described by standard languages (e.g. QVT).

One of the cornerstones of the MDA development process is that specifications on the abstraction layers are completely based on models. However, these models not just serve presentation purposes but represent code. In [KlWB03] the roles of these models are characterized as "different abstraction levels in the system specification".

With their concept of SF, Greenfield and Short [8] pick up the basic idea of McIlroy's "component factory" [17], but move the emphasis away from the mass production of software, which is not suitable for software development [1], to build-to-order software development, i.e. the efficient development of a software product line (SPL). The SF approach is divided into three stages [8]: First, the development of a SPL, i.e. the tools, languages, patterns and frameworks needed for software development, called the Software Factory Schema (SFS). Second, the configuration of the SPL, i.e. the domain-specific adoption of the SFS, called the Software Factory Template (SFT). Finally, during the Product Development, the application is developed using the SFT.

Contrary to the MDA approach, the SF approach considers both the development of a software development infrastructure and the domain-specific configuration of this software development infrastructure.

3 The Process Driven Architecture

The Process Driven Architecture (PDA) is a comprehensive approach for the model based development of process support software. It covers not only the application development stage, but as well the method and tool development stage and the domain-specific customization stage. Yet, the PDA's application development stage is built upon the core concepts of the MDA. Section 3.1 exhibits the similarities and differences of both methods. Section 3.2 then broadens the view of the development of process support software. The last two sub-sections expand on the models of the PDA (Section 3.3) and the transformation support offered by the PDA (Section 3.4).

3.1 The PDA as Adaptation of the MDA

The basic concepts of the MDA and the PDA are very similar: Starting with a relatively coarse application / process model a detailed model of the application is derived by stepwise refinement. The structural similarities become very obvious when comparing the application development steps (Fig. 1). In the following paragraphs we discuss common characteristics and differences of both approaches.

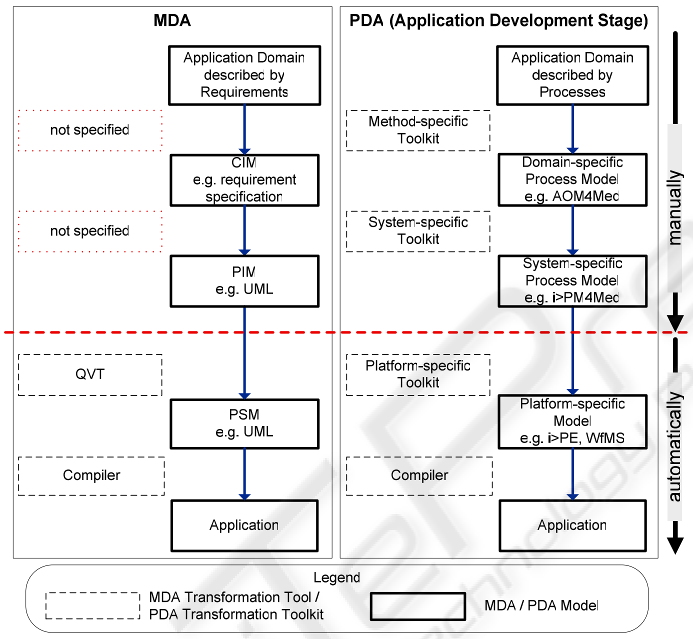
MDA
PDA (Application Development Stage)
Application Domain described by Requirements
not specified
CIM e.g. requirement specification
not specified
PIM e.g. UML
QVT
PSM e.g. UML
Compiler
Application
Application Domain described by Processes
Method-specific Toolkit
Domain-specific Process Model e.g. AOM4Med
System-specific Toolkit
System-specific Process Model e.g. i>PM4Med
Platform-specific Toolkit
Platform-specific Model e.g. i>PE, WfMS
Compiler
Application
manually
automatically
Legend
MDA Transformation Tool / PDA Transformation Toolkit
MDA / PDA Model

Fig. 1. The structural similarities of the MDA and the PDA Application Development.

□ *Shift of Focus*
The central idea of the MDA is to describe an arbitrary application by its data, functions and necessary communication, i.e. by interacting objects. By contrast, the PDA shifts the modeling focus to the processes as the fundamental elements of a (process-oriented) application. Thus, the PDA is a sort of result of the logical merger of Business Process Reengineering [9] and MDA.

□ *Clear and Explicit Structures*
An important advantage of the MDA is the definition of a clearly structured process for application development. This is of utmost importance, as it allows for the development of tools that may be applied for many kinds of MDA-based software development projects. The PDA builds upon this experience by defining specific models and transformations while keeping the same clear development process (Fig. 1).

□ *Domain-specific Customizability*
The MDA focuses on model based generation of object-oriented software. It is generic and adoptable to different application domains only to a restricted degree using UML profiles or stereotypes. In contrary, the PDA inherently picks up the domain's

requirements, as the domain-specific customization of the process modeling method, tools and development environment is a prerequisite for the PDA's application development stage (Section 3.2). In contrary to the resulting lack of transformation support between the application domain and the Computation Independent Model (CIM) and the CIM and the Platform Independent Model (PIM) respectively, the PDA's specialization allows for significantly improved transformation support (Section 3.4).

 Domain-specific Language
The PDA offers a high degree of adaptability to an application domain, by using a process modeling method that can be customized to make use of domain-specific concepts. The combination of the domain-specific language and the paradigms of process orientation enable the PDA to involve users much tighter in the application development process. Compared to the MDA, the PDA's domain-specific process model is for the users much more intuitive and understandable than an object-oriented representation of the process support system.

To summarize, the MDA is used for the development of arbitrary software systems whereas the PDA is used specifically for the development of process based applications. In other words, the PDA application development stage might be regarded as a specialization of the MDA for process based software. The next section deals with the prerequisites to allow for a domain-specific application development in the PDA.

3.2 The PDA Development Approach

The PDA promotes a comprehensive development approach which covers not only the application development, but also the prior development of adequate tools, methodologies and necessary customizations. Accordingly, the overall PDA development process is divided into three stages (cf. Fig. 2): Tool & Method Development, Customizing and Application Development.

During the first stage the basic process modeling methodology, e.g. the perspective oriented process modeling method [11], and associated tools are developed. This is done in three steps: First, a method developer, typically embodied by an industrial consortium (e.g. the OMG) or a research group, designs a process modeling method (cf. Fig. 2, ❶). Second, software developers create specialized tools to be able to efficiently perform modeling tasks using the new process modeling method (cf. Fig. 2, ❷). Finally, an execution environment is developed that can implement processes modeled using the newly designed method (cf. Fig. 2, ❸). The Tool & Method Development stage is generic by nature, as it is independent from any application domain and the results.

The second stage of the PDA approach adopts the results of the first stage to a certain application domain, e.g. the clinical field of application. Yet, it is still independent of a concrete application process and thus belongs to the infrastructure development. The Customization stage is split into three steps, as well: In the first step, the process modeling method is customized to the application domain (cf. Fig. 2, ❹). Typically this involves the cooperative design of modeling artifacts that adhere to the terminology of the application domain. Actually, a new idiom of the modeling language, called the domain-specific language (DSL)[1] [21], is created that serves as a communication basis between the domain experts and the application developers. This step is critical to many model driven software development approaches [20] and thus has to be performed in close cooperation between experienced domain experts and application developers. The newly introduced modeling artifacts and all other modifications of the modeling methodology subsequently have to be reflected in the modeling tools (cf. Fig. 2, ❺). The application developer is responsible to configure the tools accordingly. It is important to note that this step also comprises the provision and preparation of other required tools like wrappers, mediators or middleware. The process execution environment has to be customized to fit into the application domain, as well (cf. Fig. 2, ❻). The necessary actions at this step range from the installation of basic infrastructure such as a web server or a database system to the provision of network access to required information systems, such as a hospital information system. At the end of the Customization stage all tools and the modeling methodology are fitted to the intended application domain, i.e. the development infrastructure is ready to be used for the "production" of process-oriented applications.

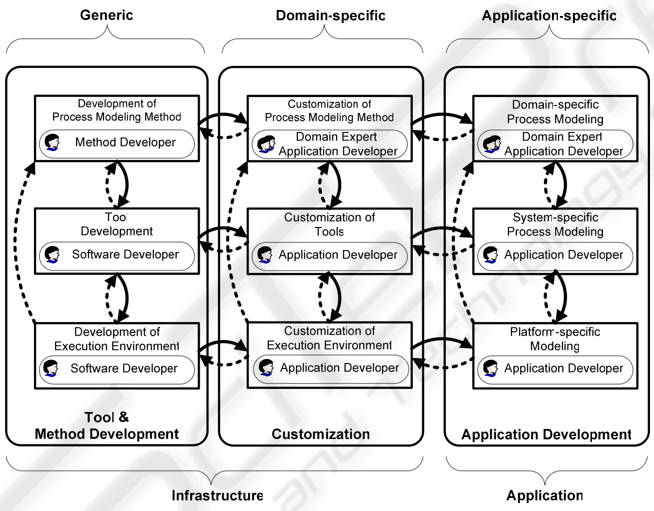
Generic
Domain-specific
Application-specific
Development of Process Modeling Method
Method Developer
Tool Development
Software Developer
Development of Execution Environment
Software Developer
Tool & Method Development
Customization of Process Modeling Method
Domain Expert Application Developer
Customization of Tools
Application Developer
Customization of Execution Environment
Application Developer
Customization
Domain-specific Process Modeling
Domain Expert Application Developer
System-specific Process Modeling
Application Developer
Platform-specific Modeling
Application Developer
Application Development
Infrastructure
Application

Fig. 2. The multi layered PDA approach.

For every process that is to be supported, only the application development stage has to be passed. Three steps constitute the PDA application development: At first, a domain-specific model of the process to support is drafted cooperatively by the domain experts, e.g. the medical personnel, and the application developers (cf. Fig. 2, ❼). This step is of utmost importance for the quality of the resulting process support application and has to be done very carefully. Exactly at this point, a model based approach like the PDA offers significant advantages regarding the development efficiency, as it allows for rapid application development and immediate user feedback. This ability to semi-automatically generate an application directly results from the comprehensive efforts during the Customization stage. Right after the domain-

[1] Often a graphical DSL is also referred to as domain-specific modeling language (DSML)

specific process modeling is finished, the application developer takes care of system specific adoptions and enrichments of the process model by simply wiring the previously configured and prepared tools with the elements in the process model (cf. Fig. 2, ❽). All actions that have to be taken after this step may be automated and do not need any further human interaction if prepared well during the Customization stage. The last application development step comprises the generation, compilation and deployment of the application (cf. Fig. 2, ❾). What steps are exactly necessary depends on what kind of process execution environment was selected, i.e. the generation of software is treated differently than the deployment of a workflow in a Workflow Management System. Respectively, the degree of manual intervention strongly depends on the specific situation, especially if a well organized test and release cycle is followed. The Application Development stage results in an executable process-oriented application.

3.3 Models of the PDA

During the execution of the Application Development stage three types of models are used: the domain-specific process model, the system-specific process model and the platform-specific model. Two more representations are relevant for application development: the description of the application domain and the resulting application.

- *Application Domain Description*

The starting point of the PDA is a detailed description of the application domain. The PDA Application Development stage requires a process-oriented description, i.e. the process that is to be supported must be documented in any way and the users should be accustomed to it. In our clinical application domain medical guidelines or clinical pathways are common.

- *Domain-Specific Process Model*

The domain-specific process model is created according to a comprehensive process analysis of the application domain. The transformation between the application domain description and the domain-specific process model hast to be done manually by the application developer and the domain expert. It is the first step towards a machine readable representation of the process and serves as communication basis between the application developer, the domain experts and the users. The domain-specific modeling artifacts (the DSL) created during the Customization stage are used to record the process model, this allows for a tight integration of the users.

At this level the model is not detailed enough to be executed, but it contains all relevant information of the application domain to correctly and comprehensively describe the process.

- *System Specific Process Model*

The system specific model contains all details necessary to automatically generate the platform-specific model. The application developer is responsible to enrich the domain-specific model with the necessary technical details, e.g. where the used data is stored. Typically this is done by specifying missing attribute values or adding special modeling artifacts.

The professional content of the process model must not be changed at this level; only system-specific changes may be performed. After the system-specific model is complete, all following transformations are performed automatically.

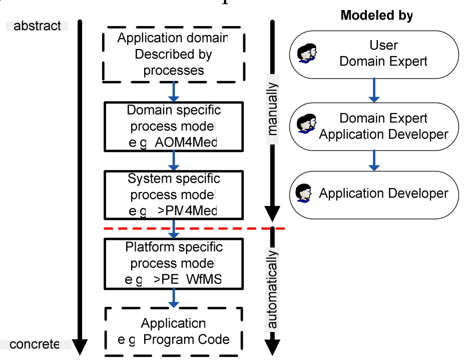
Modeled by
abstract
concrete
Application domain Described by processes
Domain specific process model e g AOM4Med
System specific process model e g >PM4Med
Platform specific process model e g >PE WfMS
Application e g Program Code
manually
automatically
User Domain Expert
Domain Expert Application Developer
Application Developer

Fig. 3. The models of the PDA Application Development Stage.

 Platform-Specific Model
The platform-specific model is tightly connected to the used process execution platform and thus cannot be used for a different execution platform. It is no longer human readable, and it is stored in a format that is perfectly suited for the following code generation steps. As the lack of the term "process" in the name of this model implies, the process is no longer "visible" in this representation, i.e. an object-oriented model might be used to describe the process support software to be generated.

The software that transforms the system-specific process model in the platform-specific model typically inserts modeling artifacts in the model, that are very specific for the execution platform, such as artifacts needed to generate the user interface or code to access subsystems like a database. The platform-specific model is subsequently transformed into a representation that is suitable to support the user during process execution, e.g. the application is compiled and then deployed.

 Application
The application is the result of the PDA Application Development stage and supports the process, exactly as previously modeled in the domain-specific process model. How this is done, depends of the chosen process execution platform, e.g. it might be deployed as a web based application.

To convert one model to the next, model transformations have to be applied. The overall quality of the generated process support application directly depends on the quality of these transformations (Section 3.4).

3.4 Supporting Transformations: The PDA Toolkits

In order to offer optimal support for the mission critical model transformations the PDA introduces the concept of so called "toolkits". Three toolkits are used in the PDA Application Development Stage (Fig. 1): the Method-specific, the System-specific and the Platform-specific toolkit. The toolkits are the result of the Customization stage and offer the application developer and the domain expert a specific collection of methods, procedures, modeling artifacts, tools to perform the transformation effectively and efficiently. A toolkit does not contain a fixed set of methods or tools; rather it might be regarded as (formal or informal) domain-specific knowledge that is to be applied during the transformation. Thus, the existence of a toolkit does not necessarily imply the possibility to automate a transformation. Yet, the availability of the Method-specific and the System-specific toolkit are a significant advantage of the PDA.

The concrete content of the PDA's toolkits cannot be specified in advance, as the contents heavily depend on the project context and are developed during the Customization stage. Nevertheless, the following exemplary lists of typical contents, derived from our experience, give a brief impression of the PDA's toolkits:

- *Domain-specific toolkit*
  - Domain-specific modeling artifacts (e.g. the artifact to represent a clinical checklist or an evidence based medical decision)
  - Domain-specific modeling conventions (e.g. "medical decisions are always modeled using the artifact for an evidence based medical decision")
  - Modeling handbook (e.g. an introduction on how to apply the used process modeling method in the project, modeling conventions)
- *System-specific toolkit*
  - System-specific modeling artifacts (e.g. a representation of the used medical examination tools)
  - Configurations of used software tools (e.g. configuration files, parameters, etc.)
  - System-specific modeling conventions (e.g. "every process can only have one entering data flow")
  - Rule sets to allow for automated model checking (e.g. "the definition of data containers must not be recursive")
- *Platform-specific Toolkit*
  - Mediators and wrappers (e.g. to connect to external applications, medical examination tools, data source, etc.)
  - Installed and configured process execution environment (e.g. an installed and usable workflow management system)
  - Code generation templates and deployment descriptors (e.g. templates for web pages to be generated)

As soon as the toolkits are defined the PDA Application Development stage can be initiated. For the clinical projects (cf. Section 1), a model based software generation and process execution environment has been developed, called i>ProcessExecution (i>PE). The next section highlights its usage for a glaucoma screening process.

4 Example: The A4 Glaucoma Screening Process

After having designed the system specific process model, the manual work steps are completed and an executable application can be generated in three steps: First, the process model is exported as an XML document. This export is then used as input for the i>PE compiler and is transformed into a deployable module. During the compiler run, the system specific process model is split up again into the domain-specific part, which is responsible for the core business logic and the system specific part, which is responsible for connections to other systems. In a final step the executable module, here a web application, can be deployed in an application server and made available to the medical staff.

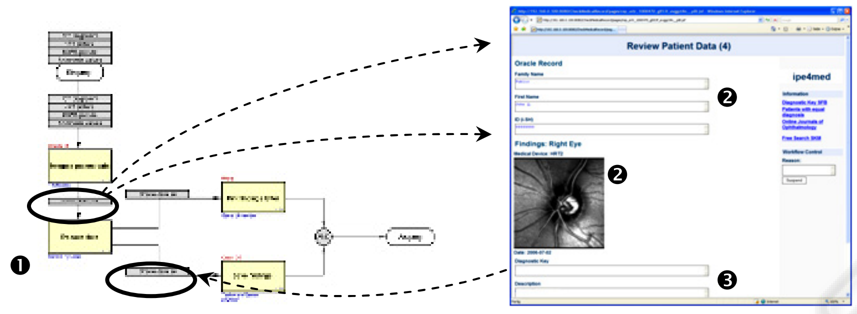

Fig. 4. Transformation of a Process Model into an executable application.

As a short example, we will discuss an application that is used at the ophthalmic department of the University of Erlangen for a glaucoma screening examination. This screening process is described in detail in [13], hence we will omit details here.

In Fig. 4 a small part of this screening process is shown. This process describes the work steps that have to be performed after all medical data has been collected and the patient's data has to be reviewed by a physician. All data that is relevant to assess the patient's eye (e.g. patient's medical history or images taken in previous work steps) have to be presented to the physician.

Each process step (❶) in the process model is mapped to a separate web page (e.g. "Review Patient Data"). The process's input data is transformed into elements of the web page (❷). In the example above, the patient's medical history together with the image from one the medical device are displayed. For all output data (i.e. data that is generated in the process step), empty fields are generated (❸).

5 Conclusion

The PDA is a process model driven software development approach that describes how to efficiently and effectively create process-oriented software. It combines the advantages of the structured application development process of the MDA with the holistic understanding of software development as promoted by other model driven approaches (e.g. SF) and enriches it with the specific concepts of process modeling.

References

1. Aaen, I.; Bøttcher, P.; Mathiassen, L.: The Software Factory: Contributions and Illusions. In: Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia, Oslo, (1997).

2. Anderson, J. G.: Clearing the way for physicians' use of clinical information systems. In: Communications of the ACM, 40(8) (1997) 83-90.
3. Berg, M, & Toussaint, P.: The mantra of modeling and the forgotten powers of paper: a sociotechnical view on the development of process-oriented ICT in health care. In: International Journal of Medical Informatics 69, Elsevier (2003).
4. Berg, M.: Patient care information systems and health care work: a sociotechnical approach. In: Int J Med Inf 1999 (55), Elsevier, (1999) 87-101.
5. Dadam, P., Reichert, M., & Kuhn, K.: Clinical Workflows - The Killer Application for Processoriented Information Systems? In: Abramowicz, W.; Orlowska, M.E. (Eds.): BIS - Proc. of the 4th Int'l Conference on Business Information Systems, Poznan, Poland, Springer, (2000) 36-59.
6. Eisenecker, U. W.; Czarnecki, K.: Generative Programmierung, Addison-Wesley, München, (2000)
7. Fowler, M.: Language Workbenches: The Killer-App for Domain Specific Languages?. Online Article. http://martinfowler.com/articles/languageWorkbench.html, retrieved 3/5/07.
8. Greenfield, J.; Short, K.: Software Factories: Assembling Applications with Patterns, Models Frameworks and Tools. Wiley, Indianapolis, (2004).
9. Hammer, M.; Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. Harper Collins Publishers Inc., New York, (2005).
10. Jablonski, S.: Process Based Data Logistics: Data Integration for Healthcare Applications. In: Proceedings of the European Conference on eHealth (ECEH06), Fribourg, Switzerland, (2006).
11. Jablonski, S.; Bußler, C.: Workflow management - modeling concepts, architecture and implementation. London. International Thomson Computer Press, (1996).
12. Jablonski, S.; Lay, R.; Meiler, C.; Müller, S.; Hümmer, W.: Data Logistics as a Means of Integration in Healthcare Applications. Proc. of the 2005 ACM Symposium on Applied Computing (SAC) - Special Track on Computer Applications in Health Care, Santa Fe, New Mexico, (2005).
13. Jablonski, S.; Lay, R.; Müller, S.; Meiler, C.; Faerber, M.; Derhartunian, V.; Michelson, G.: Building a Generic Platform for Medical Screening Applications based on Domain Specific Modeling and Process Orientation. Proceedings Second International Workshop on Data Integration in the Life Sciences (DILS 2005), LNBI 3615, 2005, San Diego, (2005), 257-265
14. Kuhn, K.A., Giuse, D.A., Bakker, A.R., Ball, M.J., Gell, G.: Challenges in Deploying Health Information Systems. In: Patel, V.L.; Rogers, R.; Haux, R. (eds): Medinfo 2001, Proceedings 10th World Congress on Medical Informatics, (2001).
15. Lenz, R., Elstner, T., Siegele, H., Kuhn, K. A.: A Practical Approach to Process Support in Health Information Systems. In: J Am Med Inform Assoc., 9(6), (2002) 571-585.
16. Lenz, R.; Kuhn, K. A.: Towards a continuous evolution and adaption of information systems in healthcare. In: I J Med Inf 73, Elsevier, (2004) 75-89.
17. McIlroy, M. D.: Mass-Produced Software Components. In: Buxton, J. M.; Naur, P.; Randell, B. (Eds.): Proc. of Software Engineering Concepts and Techniques, 1968 Nato Conf. on Software Eng., (1969) 138–155. http://www.cs.dartmouth.edu/~doug/components.txt, retrieved 3/5/07.
18. Object Management Group (OMG): Meta Object Facility (MOF), (2006). http://www.omg.org/mof/, retrieved 3/5/07.
19. Object Management Group (OMG): Model Driven Architecture (MDA), (2006). http://www.omg.org/mda/, retrieved 3/5/07.
20. Stahl, T.; Völter, M.: Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. 1. Auflage 2005, dpunkt.verlag, Heidelberg, (2005).
21. van Deursen, A.; Klingt, P.; Visser, J.: Domain-Specific Languages: An Annotated Bibliography. In: ACM SIGPLAN Notices, Vol. 35, Issue 6, (2000) 26–36.