RIGA TECHNICAL UNIVERSITY
Faculty of Computer Science and Information Technology
Institute of Applied Computer Science

Vladimirs NIKULSINS
Computer Systems doctoral programme student

APPROACH TO TRANSFORMATION OF SOFTWARE DEVELOPMENT LIFE CYCLE MODEL IN THE CONTEXT OF MODEL-DRIVEN ARCHITECTURE

Doctoral thesis summary

Scientific adviser
Dr. sc. ing., professor
O.NIKIFOROVA

Riga 2011

UDK 004.2(043.2)
Ni 805 a

Nikulsins V. Approach to Transformation of Software Development Life Cycle Model in the Context of Model-Driven Architecture
Doctoral thesis summary. -R.:RTU, 2011.-25 pp.

Printed in accordance with the decision of the Institute of Applied Computer Systems on May 23, 2011, the minutes No 71.

This work has been partly supported by the European Social Fund within the National Programme “Support for the carrying out doctoral study programmes and post-doctoral researches” project “Support for the development of doctoral studies at Riga Technical University”.

ISBN

DOCTORAL THESIS
SUBMITTED FOR THE DOCTORAL DEGREE
OF COMPUTER SYSTEMS
AT RIGA TECHNICAL UNIVERSITY

The assertion of the thesis submitted for the doctoral degree of computer systems will take place at an open session on November 7, 2011 in Room 202, Meza 1/3, at Riga Technical University, Faculty of Computer Science and Information Technology, Riga, Latvia.

OFFICIAL OPPONENTS:

Professor, Dr.habil.sc.ing. Janis Grundspenkis
Riga Technical University

Professor, Dr.habil.sc.comp. Audris Kalnins
Latvian University

Professor, Dr.sc.comp. Victor Taratoukhine
The National Research University Higher School of Economics (Russia)
University Alliance Program Director, CIS, Nordics and Baltics at SAP

CONFIRMATION

I confirm that I have developed this thesis submitted for the doctoral degree at Riga Technical University. This thesis has not been submitted for the doctoral degree in any other university.

Vladimirs Nikulsins _____ (Signature)

Date: 13.6.2011

The doctoral thesis has been written in Latvian, and includes Introduction, 5 Parts, Conclusion, 3 Appendices, Bibliography, 47 figures and illustrations, 7 tables, in total 147 pages. Bibliography includes 151 information sources.

GENERAL CHARACTERISTICS OF THE RESEARCH

Research motivation

The world as we know it is dynamic by nature. Based on the philosophical statements by Alfred North Whitehead, our world can be described/ as a union of interconnected small and large systems, which are constantly changing. From the point of view of the process philosophy, the world around us is not static, it is constantly evolving [PAL 2006].

Software development is also dynamic by nature. It is based on the process oriented approach, called software development process. Software development process reviews aspects related to software development, including management discipline and process enhancement. Software development process is one of the software engineering disciplines. It is a relatively new field, and scientists are still discussing whether software engineering conforms to classical engineering or not [Vliet 2006].

Software development methodologies define the application of software engineering principles that can be described as coordinated actor driven activities set with the ultimate goal to create software development products [LAN 2004]. Modern software development methodologies are structured, disciplined and iterative by their nature [KRU 2000]. The methodologies structure and formalize software development in a way that allows it to be quantitatively assessed in terms of required resources, planned software functionality, and required time [MSF 2003]. Modern software development methodologies can be divided into heavyweight and agile. Heavyweight methodologies are based on planning, detailed documentation and design. Agile methodologies are generally goal-oriented, where the key factors are the efficiency of interaction between the software development team members (instead of following strict process guidelines) and the working code (instead of documentation) [KHA 2004].

Instead of various attempts to generalize and formalize software development process, modern heavyweight and agile software development methodologies support a so called traditional viewpoint to software development, when the code is the essential deliverable of the software development. The main problems of traditional software development are the following [KLE 2003]:

- The productivity problem (developers often consider the code is to be productive, (whereas writing models and documentation are not)
- The portability problem (a frequent update or change of technology)
- The interoperability problem (the complexity of communication between systems)
- The maintenance and documentation problem (documentation has no influence on the code and vice versa)

A promising approach to resolve the above problems is the Model Driven Architecture (MDA), a framework for software development defined by the Object Management Group (OMG) in 2001 [SIE 2001]. In contrast to the traditional software development, MDA as a main result considers system model developed, instead of software code [KLE 2003]. The models are being transformed into the working code through the various automated and semi-automated transformations. The system model is independent from the implementation, which allows its deployment into different platforms. MDA is introduced new software development era in software development process. It can be assumed that currently software development industry remains in a transition period, with the old methodologies unable to overcome the complexity of software systems, and the new methodologies not being mature enough to be applied. This actualizes the need for the algorithm development, which could enhance the existing mature software development methodologies with the artefacts and principles from some new methodology. In other words, there is a need for the transformation approach from the traditional software development into the model driven software development.

Relevance of the thesis

The evolution of the software development process stabilized conventional software development practices [PEI 2010]. Introducing new technologies into the current process is often quite complicated and labour-consuming c. It is impeded by both formal and objective factors, and also by the conservatism and internal resistance to changes [SER 2005]. Methodologies, that support traditional code-oriented development, are subject to enhancement. They should be flexibly moved to the MDA oriented process by changing current software development process. The implementation of a new MDA based process conforms to the goals of software improvement, including the enhancement of both quality and productivity, as well as the decreased development time [STE 1999].

Changes in software development to use MDA process reflect also architectural changes. From the financial viewpoint it is advantageous to use business models, e.g., UML (*Unified Modelling Language*), that forms a part of MDA [STA 2006]. They either are not changed, either evolving independently from the changes in technological platforms [ERI 2004]. In software engineering, it is not only possible to define system behaviour with the help of models, but also to define the software development process itself. Models can formally define actions (current or required states of the system), however the changes in the process can not be expressed with the help of models. For example, CMMI or ISO 9000 define the required states of the system [STE 1999], [CMM]. Namely, the methodologies define how these changes should happen [SWE 2004].

As every "revolutionary" approach, MDA is still on its way to perfection, currently being in the beginning of its evolution [GUT 2007]. 10 years have passed since MDA was declared as an alternative to the code-oriented development. MDA tools and methods are evolving, and there is still a need for new tools, standards and best practices [FAV 2010]. It is applicable to both the analysis of the real life MDA projects and to theoretical ground [HUS 2011].

Problem definition

MDA defines the standards and principles for the model transformations, but at the same time it does not prescribe a development methodology and its related activities. MDA technologies are not explicitly related to the identifiable activities within the software development processes, as these technologies are developed to be generally applicable in combination with development processes that may be already anchored in organisations [GAV 2004]. OMG consortium gives no guidelines for MDA usage in terms of the processes such as activities and phases, roles and responsibilities that are involved in the software development and are used for formal representation of the software development process. Therefore, the **research subject** of this thesis is the organization of the software development process and the approach for transition from one process organization to another, keeping the artefacts and their content per se unaffected.

Taking into account the trend or Latvian software development companies to use mainly code-oriented development [NIK 2006a], [NIK 2006b], the task of defining the transition methodology to adopt MDA within the software engineering context in Latvia is especially important. Additionally, it is valuable to define the framework and apply this approach in practice for one of the Latvian software development companies.

Related works

There is no complete and unified methodology at the moment, that would guide the transition from the traditional software development process into the model driven. There are various specific or general researches that to some extent cover some parts of model driven process. There also exist some specific process researches, e.g., MASTER project (*Model-driven Architecture inSTrumentation, Enhancement and Refinement*) deliverable called Process Model to Engineer and Manage the MDA Approach, MODA-TEL project [ESI 2003], [BEL 2002], [STE 2003]. However, the application of these researches is not universal – it is not possible to determine how specific activities of some software development company are related to MDA activities.

The basis of MDA is the object-oriented approach and the component-based development. Therefore, theoretically, it can be unified with various software development processes [CHI 2007]. According to the vision of OMG consortium, MDA is not limited to be used in any software development process. Though, the transition between processes is not formalized. As one of the possible solutions to solve this problem OMG suggests to use the support of OMG FastStart program consultants. Consultants are supposed to analyse every process in the organization and provide recommendations and the support required to apply MDA [GUT 2004].

Various researches within the problem domain provide specific theoretical and technological solutions. They have contributed to the formation of the solution described in this thesis. [FEN 2006] proposes its own solution on how to map the SPEM (*Software and Systems Process Engineering Metamodel*) and XPDL (*XML Process Definition Language*), naming their approach SPEM2XPDL. [COM 2006] suggests enhancing SPEM with OCL [OMG 2006], because it lacks a formal description of its semantics that makes it difficult to use. Another problem is that using SPEM is difficult because the OMG proposal is rather generalist and provides no directives on how to use it. [DEB 2007] is investigating SPEM transformations into BPMN (*Business Process Modelling Notation*) by using workflow automation. Some of the authors are investigating the specifics of model transformations, and their solutions can also be useful for the process model transformations. [TRA 2004] investigates different approaches to model transformations. Particularly, the attention is paid to OMG's Queries/Views/Transformations (QVT) [MOF 2008]. Authors also review the problems of tracing activities and model validity, when changes are made in both source model and target model. Unification of declarative and imperative transformation languages on model transformations is proposed by adopting a new approach to the model transformations [TRA 2004].

[BRE 2001] investigates the integration process of metamodels. Even though this paper regarding process-centred model engineering was written in 2001, it has become even more of a problem today due to different implementations of MOF (*Meta-Object Facility*). [DIA 2010] investigates insufficient model-driven support in process modelling tools. SPEM 2.0 is not supported with MDE (*Model-driven Engineering*) aspects on the metamodel level. One of the solutions proposed by the authors involves using the custom SPEM metamodel and the corresponding tool enriched with MDE specifics.

Goal of the thesis

The **goal of this thesis** is to define an approach, that could allow making a transition from one software development process into another by using formal principles of transformation, and demonstrate the suggested approach with an example of such transition.

In order to achieve this goal it is necessary to investigate which MDA artefacts and processes can enhance corresponding software development processes and artefacts of traditional development, as well as define the methodology for the transition from traditional software development to the MDA based development.

Tasks of the thesis

1. Define the traditional software development process and standards.
2. Investigate the essence and concepts of the model driven development.
3. Investigate the possibilities of integrating the traditional software development methodologies with the introduction of MDA principles.
4. Define a formal approach for the transition from the traditional software development into MDA oriented.
5. Make testing of suggested software development approach within one of the software development projects and assess possible opportunities, perspectives and constraints.
6. Draw conclusions about MDA usage possibilities, problems and perspectives in software development when doing transition from the traditional software development to the model-driven.

Research methods

The analysis of the information sources is based on available problem domain literature, which includes books, magazines and conference materials. This allows getting the actual status and latest trends of the problem domain under investigation. Such information can help in generating ideas for possible mappings of the model driven software development components into traditional software development. Software development life cycles (and their process model representations) should be analyzed for investigating information flows and steps in software development process. With the help of the analysis methods (by merging MDA specific activities with traditional software development) it is possible to define the guidelines for transition from any software development process.

This thesis describes the modelling methods of the software development process (analysis, modelling and assessment of results). For model transformations the author used imperative approach, which defines state changes (transformations are described in *QVT Relations* language). Viability assessment of the suggested approach and the architectural solution proposed is practically verified in real project.

As a base of research author of this thesis took part in several scientific and information system development projects, that were organized based on different software development life cycle models. Author of this thesis also took part in multiple international industry projects (starting from early inception phases till to transition and support) in *Accenture* where he works as SAP consultant. Also he participated as performer or general performer in the following research and methodological study projects:

1. Research project of Latvian Ministry of Education and Science Nr. 09.1269 "Methods and Models Based on Distributed Artificial Intelligence and Web Technologies for Development of Intelligent Applied Software and Computer System Architecture" (2009-currently);
2. Research project of Riga Technical University Nr. FLPP-2009/10 „Development of Conceptual Model for Transition from Traditional Software Development into MDA-oriented" (2009);
3. Participation in ESF funded project „Development of Study Module on Model Driven Software Development Technology within the Study Programme "Computer Systems""(Contract Nr. 2007/0080/VPD1/ESF/PIAA/06/APK/3.2.3.2./0008/0007) (2006-2008);
4. Participation in ESF funded project 3232/15 „Modernization of RTU Study Programme "Computer Systems" with the Aim to Decrease Professional Competitiveness" (2006-2008);
5. Research project of Riga Technical University ZP - 2005/02 „Application of Two-Hemisphere Approach for Development of Agile Software Engineering Knowledge Architecture" (2005-2006);

Scientific novelty of the thesis

This thesis introduces the following innovations:

1. Mapping of model-driven software development artefacts into the traditional software development process.
2. Solution architecture and set of transformations for the process model enhancement using QVT Relations language.
3. Demonstration example and approbation within one of the Latvian software development companies.

Practical significance of the thesis

The practical significance of the suggested approach is proved with companies PF example. It confirms the possibility to use this approach in other software development companies that have a different software development life cycle.

Research publications and presentations at the scientific conferences

The results of the research are obtained and published from 2003 till 2011, when the author of this thesis was studying at Riga Technical University Computer Systems programs, with the aim to complete the bachelor, master and doctoral degrees. The publications include 10 research

publications in the internationally edited conference proceedings. Additionally, there are two thesis publications in the proceedings of the International Scientific Conferences in Riga Technical University. The artefacts created and described during the writing of the thesis are published in five learning materials and used in several courses by the Institute of Applied Computer Systems of Riga Technical University. The results of the application of the suggested approach are published in the report on the scientific project [NIK 2009e]. A list of publications is attached in the end of this summary.

The main results of the research were presented by the author at 7 international conferences:

1. Nikulsins V. Transformations of Software Process Models to Adopt Model-Driven Architecture. ENASE 2010: 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development MDA&MTDD. July 22-24, 2010, Athens, Greece
2. Nikulsins V., Nikiforova O. Tool Integration to support SPEM Model Transformations in Eclipse. The 50th RTU International Scientific Conference. October 12-16, 2009, Riga, Latvia
3. Nikulsins V., Nikiforova O. Transformations of SPEM models using query/view/transformation language to support adoption of model-driven software development lifecycle, ADBIS-2009. Model – Driven Architecture: Foundations, Practices and Implications (MDA) workshop. Riga Technical University, September 7-10, 2009, Riga, Latvia
4. Nikulsins V., Nikiforova O. Adapting Software Development Process towards the Model Driven Architecture. The Third International Conference on Software Engineering Advances, ICSEA 2008, October 26-31, Sliema, Malta
5. Nikulshins V., Nikiforova O., Sukovskis U. Mapping of MDA Models into the Software Development Process, The 8th Biennial International Baltic Conference on Databases and Information Systems, Baltic DB&IS, July 2-5, 2008, Tallinn, Estonia
6. Nikulshins V., Nikiforova O., Sukovskis U. „Analysis of Activities Covered by Software Engineering Discipline", The 7th Biennial International Baltic Conference on Databases and Information Systems, Baltic DB&IS, July 3-6, 2006, Vilnius, Lithuania
7. Nikulshin V., Nikiforova O. "Software Development Teams Organization", The 46th Scientific Conference of Riga Technical University, Computer Science, Applied Computer Systems, October 13-14, 2005, Riga, Latvia

Structure of the thesis

The structure of this paper is organized as follows. The introduction describes the relevance and novelty of the research, the goals and tasks, and the scientific and practical significance of the thesis.

Chapter 1 systematizes and describes the knowledge about the software development process, its definitions, standards and the modern technologies used in the software development.

Chapter 2 clarifies aspects of the modernization of the software development process in relation to the application of MDA in that. The general MDA principles and concepts are defined there.

Chapter 3 describes author's efforts in integrating the software development process and MDA artefacts.

The theoretical material described in the first three chapters, together with the results of the research allows author to suggest a hypothesis about a possible solution and a transition approach from the traditional software development organization into the model drives software development organization, which is described in chapter 4.

Chapter 5 describes the practical approbation and the tool developed to support the suggested approach with several examples.

In the last chapter the general achievements of the thesis are summarized and the conclusions are made.

1. METHODOLOGIES AND STANDARDIZATION OF SOFTWARE DEVELOPMENT PROCESS

The first chapter describes a history of software engineering and its advances since the declaration of the software engineering as an engineering discipline at NATO conference in 1968 [NAU 1969], [VLI 2008]. A short vocabulary with some of the most essential definitions of the terms used in this thesis, is also provided. This chapter describes the most eminent software development methodologies and standards.

1.1. Terms and definitions

The elaboration of software engineering has some effect on its terms and definitions. Some of the terms become ambiguous. For example, the definitions of the the terms "software development process" and "methodology" are usually mixed. Some of the terms translated to Latvian and can be found from [CAU]. In this thesis they are supplemented or improved (in Latvian).

Software engineering is the application of systematic, disciplined and quantifiable approaches to scientific and technological knowledge, methods and experience usage within functionally effective software development, application and transition process. Also the branch of science about software development [CAU], [SCA 2001], [SWE 2004], [IEE 1990].

Software development process (sometimes *Software process*) – the process of translating the user's needs into a software product. The process involves translating the user's needs into software requirements, transforming the software requirements into specific design, implementing the design in the code, testing the code, and sometimes, installing and checking out the software for operational use. These activities may overlap or be performed iteratively [IEE 1990].

Software life cycle – the period of time of software existence from its early development till the moment when the software loses its value. The essential software life cycle phases are analysis, design, implementation, maintenance and, possibly, also enhancement [CAU]. Sometimes it is assumed that the cycle terminates with the system deployment [IEE 1990]. One of the most widely used industry standard is [IEE 2008].

Software development life cycle (SDLC), sometimes is also called *Systems development life cycle* – in general synonym to software development process, is the structure intended for the development of software products. In contrast to the software life cycle, this term is more specific and is typical to software development, and not Software development life cycle as such.

1.2. Organization of software development process

Fundamental science such as physics and mathematics, as well as social science and economics have stabilized traditions and formal basics to define the corresponding mechanical systems, mathematical expressions or chemical reactions. Though, software engineering is a relatively new and therefore heterogeneous engineering field, where standard specifications and regulations for software development are not yet established. This is the area where boundary definition between the sufficient degree of formalization and the simplicity of problem solving and readability is not yet complete. Research on this area usually requires to take into account not only technological, but also social, economic and information technology related aspects [NIK 2006c]. This problem is vital when trying to formalize software development process.

As software development is an organizational process, it is possible to depict it with the help of a model, thus gaining required formalization level. The base elements in such model are phases, activities, artefacts and roles, which are linked together. One software development life cycle can contain several software development models, that define tasks and activities within the software development process.

The choice of the software development process is related to multiple factors, such as a scale of the project, a software development team and its organization, client's requirements, etc. [COC 1999]. The standards of software development process and body of knowledge allow formal criteria to be defined for both software systems and their development.

1.3. Software development standards

There is a common assumption in software engineering that methodical approaches to software development provide less defects and inaccuracies, and help develop software faster and more qualitatively. Software engineering standards in software development life cycle are considering software development aspects and its standardization in software development activities organization, classification and grouping, as well as their own life cycle organization [SWE 2004]. Chapter 1.3 reviews several software engineering related organizations and their standards.

Software engineering standards can be used for software development life cycle formalization since they give general and structured information about different artefact groups. The standards provide a basis for grouping software development artefacts, their classification, showing required details or abstraction from details etc. Standards are independent from software development methodologies and provide a general view into the process organization, thus ensuring an abstraction level required for software development process modelling.

1.4. Software development life cycle models

Software development life cycles define a strategy for software development. Waterfall [ROY 1970], spiral [JAY 2007], [TSU 2011], incremental [THA 2005], [SCA 2001], [TSU 2011] and V-model [THA 2005], [VLI 2008] are examples of the most typical software development life cycle models. They are described in the chapter 1.4 of this thesis.

1.5. Software development methodologies

According to software engineering and project management disciplines, software development methodology is a recommended set of systematized practices, that can be supported with training materials, formal education programs or tools [ESS 2009]. The chapter 1.5 of this thesis provides an overview of two heavyweight methodologies:

1. *Microsoft Solutions Framework* (MSF) [MIC 2003];
2. *Rational Unified Process* (RUP) [RAT 1998].

The extreme programming (XP) [WAK 2001] is reviewed by the author of this thesis as an example of the agile software development methodology.

Both heavyweight and agile software development methodologies allow to achieve the same goal – create a software product, based on the formal approach and best practices, minimizing costs and controlling the software development process. Heavyweight methodologies explicitly define software development artefacts and activities. In the last years certain principles of agile methodologies have been incorporated into heavyweight methodologies. When developing the process model, it is possible to choose a process abstraction level that would allow to avoid the differences between methodologies. Agile software development can also be formalized with the help of the process models, as the agile development defines similar phases, processes, activities and artefacts, although in a more informal manner.

Software development process models have both declarative (defining how the software development should happen) and descriptive (describing how the software development happens right now) nature. Therefore, when upgrading software development or changing the process, it is possible to use process models, that can formally and declaratively describe such change.

2. MODERNIZATION OF SOFTWARE DEVELOPMENT PROCESS WITHIN THE CONTEXT OF MODEL DRIVEN ARCHITECTURE

The future trends of software development are globalization, scaling and integration of different systems [BOT 2010]. An essential role in this process is assigned to the modelling of systems logic and business process modelling, which currently can be done with various modelling languages like UML or BPMN.

One of the solutions is Model Driven Architecture (MDA), a framework for software development defined by the OMG consortium in 2001. MDA conceptually changes software development priorities from the code oriented approach (when the code is the main artefact) to the modelling approach (when model is the main artefact) [OSI 2010]. Thus, the design becomes as

part of solution and is expressed with the help of the model. Every change in design phase will affect models, which can be then transformed into execution code. In contrast to traditional code oriented development (where the executable code appears in the end of the project), an MDA based approach allows getting the model (which is the code) in the early beginning of the project. This avoids the consumption of both excess resources and the implementation of inappropriate business logic.

MDA allows reviewing of different complicated systems from different abstraction levels both on business model level (independently from technology) and on platform specific level (taking into account specifics of technology), where the latter one is produced from business model [OMG], [NIK 2008c]. MDA defines how the business model or platform independent model can be transformed into the platform specific model.

Similarly, it is also possible to model own development process and link activities, roles, artefacts, and describe their interaction as interrelated network, namely, with the help of process models [SCA 2001]. Process models can be used for the introduction of new software development process or employee training purposes.

2.1. MDA basic concepts

MDA is the architecture based on UML language and other software engineering industry standards for model and design visualisation, storing and interchange. MDA supports creation of high abstraction models, that are independent from execution platform and are stored in specialized standardized repositories.

MDA includes the following technologies: *Unified Modeling Language* (UML), *Meta-Object Facilities* (MOF), *XML Metadata Interchange* (XMI) *and Common Warehouse Metamodel* – (CWM) [OMG].

Model transformations is a unified system process used for converting one model into another, preserving defined equivalency relation between these models. The essence of MDA is a process of modelling and the model transformations. A model can be expressed as UML diagram, OCL specification or text set. MDA separates different model types, that can be abstract (specify functionality of system) and concrete, i.e., linked to a specific platform, technology and implementation. MDA types of models are [OMG], [FAV 2010], [PIL 2005]:

- CIM model (*Computation Independent Model*)
- PIM model (*Platform Independent Model*)
- PSM model (*Platform Specific Model*)
- *Code Model*, sometimes also called ISM model (*Implementation Specific Model*) [BRO 2005a].

MDA assumes that it is possible to perform transformations from CIM to PIM, from PIM to PSM and from PSM to the code model, as well as transformations at the same abstraction level. High abstraction level models can have corresponding models at lower levels. For example, one PIM can correspond to several PSM, which defines system models for different platforms. Transformations between the models happen with the help of marking: the elements of the source model are marked and linked to the elements in the target model.

Model transformations are based on the **metamodelling principles** [EVA 2003]. Metamodelling is one of the MDA base techniques. MDA is based on the platform models, expressed with UML, OCL, which are saved in *Meta-object Facility* (MOF) repository [MOF 2008]. **Metamodel** (or model of the model) is a model of modelling language, that defines syntax and semantics for the modelling language and provides collaboration between the modelling process and transformation tools. **MOF** is OMG consortium standard, which is used for specification of metamodels, their development and management. It defines a language, that defines the modelling construction set, used for definition or usage of collaborated metamodels set. MOF is also an international standard [ISO 2005]. MOF is expressed in UML and is UML 2.x extension [FAV 2010]

2.2. Usage of MDA artefacts in software development phases

From the process point of view MDA is not offering any development methodology, motivating this as a possibility to use MDA in any software development process. This is both an advantage and a shortcoming: the architecture with its properties is defined, but no usage guidelines are

provided. The transition process from the so called traditional code-oriented process into the model-driven is also missing [MEL 2004].

In general, a model driven approach is used in various stages of software development life cycle – in structuring requirements, business analysis, process modelling, system design, service definition, system integration, solution design, source code generation, automatic transformations etc. In all these cases MDA unifies related activities, resulting in a model driven development software development life cycle. Thus, it is possible to separate a high level business process descriptive model from the ones which are defining system architecture and deployment platform, so that later when developing different applications it could be reused again.

Since MDA does not prescribe software development process, in order to apply MDA to traditional software development process it is necessary to analyse the aspects related to the process integration [CHI 2007]. It is possible to investigate and generalize various researches [GAV 2004], [MAN 2006], [BEL 2002], [ESI 2002], [ESI 2003], [VOG 2006]. [MEL 2004] reviews MDA from activities and their collaboration point of view. These activities are reviewed by analysing a simple hypothetical system with one source model and one target implementation model. Later, this hypothetical system definition is extended, reviewing iterative development of MDA and model implementation on various platforms [MEL 2004].

By analysing multiple literature sources, the author of this thesis defined MDA activities with their inputs and outputs (these are defined in chapter 2.2). The information in the tables provides a basis for the integration of MDA activities into activities of the traditional software development process (which are described in chapters 3.1 and 3.2 of this thesis).

2.3. MDA maturity levels

In order to make a successful transition to the model-driven development within a selected organization, it is mandatory to assess to which extent its current software development process corresponds to the model driven development, and the roles of models and modelling activities. It is possible to assess maturity levels for the model driven software development within the organization [RIO 2006] based on Forrester classification [FOR 2009]. In this thesis this information is described in chapter 2.3.

In real life, following MDA principles into the development process is not a trivial task, since various problems are possible. When creating a fully-fledged MDA software development life cycle, both OMG consortium recommendations (including standards, guidelines, life cycle stages etc.) and general aspects of model driven process (provide model repository, corresponding modelling environment etc.) must be taken into account. However, a successful compliance with these two conditions will not obligatory guarantee that MDA adoption into traditional software development process organization will work out in real life. Various MDA artefacts and their integration with the traditional software development process must be checked for conformance, how the developed artefacts are linked, and what their interface points are.

3. INTEGRATION OF MDA ARTEFACTS INTO SOFTWARE DEVELOPMENT METHODOLOGIES

Traditionally, software development life cycle can be divided into six phases: requirements, analysis, design, coding, testing and maintenance. The problems can happen if, for example, the existing platform has to be replaced with the different one, or when the final result is different from the requirements. Model driven architecture is based on model development and their transformations. Each software development phase can be mapped to some model [NIK 2008a]. The author of this thesis pays attention to heavyweight methodologies and its most significant representatives – RUP and MDA. Both methodologies can include MDA by depicting them together, so that it can be possible to identify which phase corresponds to which MDA model [NIK 2008b], [NIK 2009c]. Searching for the interconnections in activities helps in the development of the integrated approach. Since it is possible in MDA to identify interconnections with traditional software development, the identified shared activities can be grouped by MDA principles.

3.1. MDA implementation in RUP process organization

RUP is not providing official guidelines on how to integrate MDA into their software development process with the following justification: RUP represents current best practices in software engineering and does not include approaches that are not widely used and accepted. MDA is a young and still future oriented approach, therefore the complete MDA and RUP integration process is yet to be defined. The RUP fundamental, an architecture-centric and iterative development process, is highly consistent with MDA concepts. Various researches exist on MDA integration with RUP, that use different methods [ESI 2002], [ESI 2003], [BRO 2005b]. By unifying multiple project experiences, the recommendations for MDA aspect adoption are provided in chapter 3.1of this thesis. It is still possible to perform some phase mappings from RUP to MDA (as shown on Figure 1) [NIK 2008c], even though there are no official recommendations. More information about MDA principles integration into RUP software development process organization can be found from chapter 3.1 of this thesis.

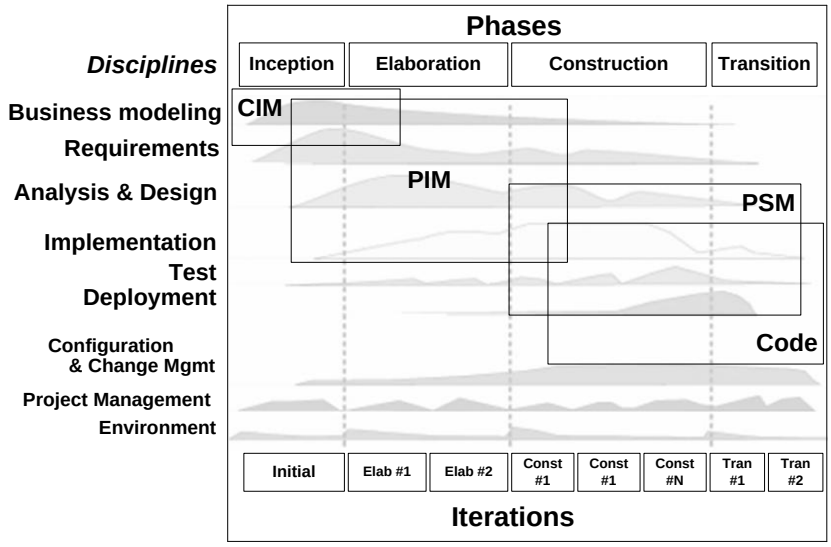


Figure 1. MDA and RUP phase mappings

3.2. MDA implementation in MSF process organization

Officially, MSF is not supporting MDA. Also, in comparison to RUP, there is no research available on how to unify both processes. However, with the help of the analysis of both technologies, it is possible to define general mapping rules for MSF process model (as in Figure 2) [NIK 2008c].

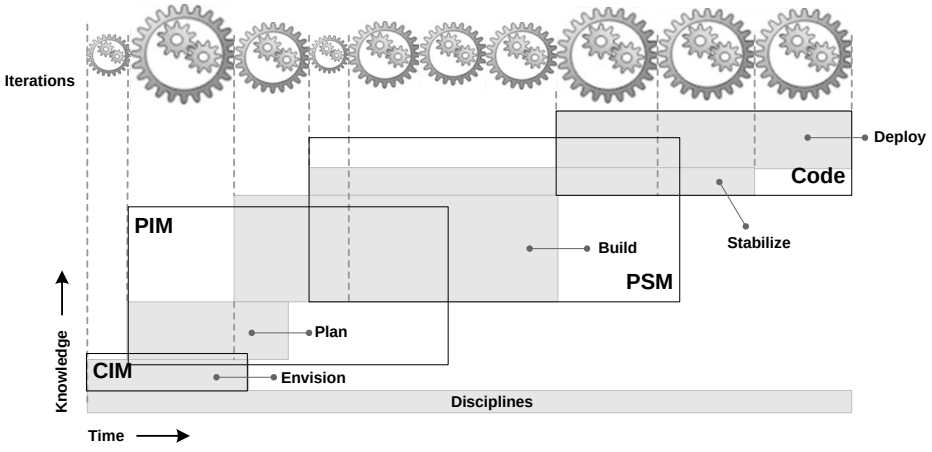


Figure 2. MDA and MSF phase mappings

More information about MDA principles integration into MSF software development process organization can be found from chapter 3.2 of this thesis.

3.3. Integration of MDA activities into RUP and MSF processes

MDA principles can also be applied to both methodologies (namely, RUP and MSF). Chapter 3.3 of this thesis contains descriptions on how models and its transformations correspond to their phases. Also, the MSF and RUP methodology processes corresponding to the MDA processes are described. These processes also show which artefacts are being replaced or enriched with the model-driven specifics.

3.4. SPEM concept

Software and Systems Process Engineering Metamodel (*SPEM*) is OMG consortium specification, that represents software development processes and related process groups. OMG standards that are related to the usage of MDA are defined at metamodel level. In a similar way, OMG defines SPEM metamodel. SPEM 2.0 is defined as a metamodel, as well as UML 2 profile (that, in its turn, is defined as MOF meta-metamodel instance).

SPEM 2.0 can define any software and system development process and its components. SPEM is limited with the minimal element amount, that is required for software development process representation. It does not include the usage of specific elements for specific domains or disciplines (e.g., program management). The goal of SPEM is to adapt different development methods, processes, formalization levels, life cycle models and workflows. The general topic of interest in SPEM is software development projects. SPEM 2.0 is not a general process modelling language and is not offering own modelling concepts.

SPEM allows defining the software development process using roles, activities, tasks, artefacts and work products. Additionally, SPEM provides a generally accepted syntax and structure for various tools. Therefore, by using MDA transformation possibilities (e.g., MOF and QVT), SPEM models can be changed and transformed into other process languages, for example, BPMN.

3.5. Conceptual Solution for software development process transition

The author of this thesis suggest performing MDA implementation into traditional software development with the help of formal models and transformation approaches. The following research paper by the author can be used as a basis for this approach: analysis of software development process in context of SWEBOK [NIK 2006b], software development structure analysis, taking into account software engineering standards (ISO, CMMI) in the context of software development teams are described in [NIK 2006c], and some results about the usage of RUP and MSF can be found in the author's master thesis [NIK 2008a], [NIK 2008b], [NIK 2008c]. The diagram of the hypothetical solution proposed is shown in Figure 3.

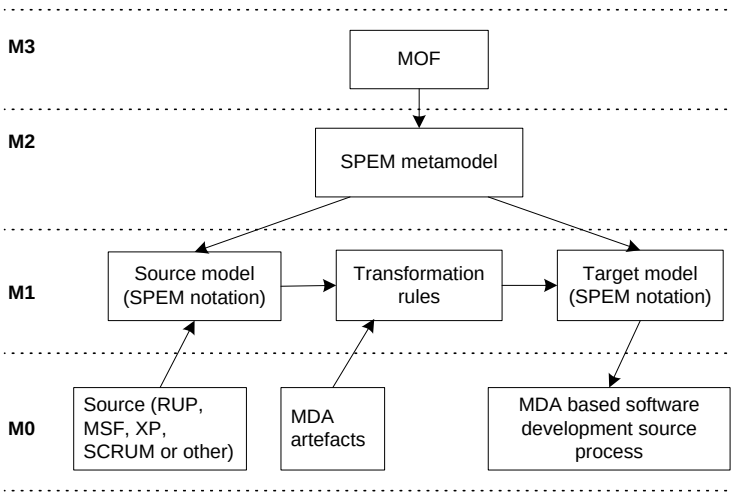


Figure 3. Hypothetical solution mapped to MDA four-layer architecture

SPEM is OMG standardized and formal software development process modelling notation, though it is not intended for software development process modification or transition from one

software development process into another. The author of this thesis believes that the elements of SPEM can also be used in the solution which will help with the transition of such processes.

Taking into account that the main artefacts of SPEM are models, an assumption is that the model driven development principles can be used also in this context. The author proposes a hypothesis that it is possible to define an approach for software development process transition which could solve the process adoption problem by integrating two solutions, namely SPEM and MDA.

In the MDA context a source model represents a problem domain, and target model represents software components (CIM to PIM or PIM to PSM). With the help of transformation one model can be transformed into another.

When choosing the software development life cycle model as a source model (which conforms to some specific traditional software development process), it can be assumed that some other software development process also exists, and the latter describes the software development from the model driven paradigm point of view. It is assumed as a target model. In addition, SPEM allows to represent any software development process with the help of the model [SPE 2008], therefore both source and target model can be expressed in SPEM notation, since both models correspond to SPEM metamodel.

Every software development process can be specified in terms of both static (activities, artefacts, roles) and dynamic (cycles, phases, iterations) aspects. By comparing two development processes it is possible to find some process conformance, shared elements and define general process groups [NIK 2006a], that provide logical association for them. Logical association of elements within the context of MDA is the basis for transformation definitions. By reviewing process aspects as transformation attributes it is possible to define the preconditions for executing transformations, i.e. define the transformation rules for SPEM models. In order to define such rules it is necessary to get formalized aspects of the traditional software development process, that can be typical for all software development processes and the elements of which can be clearly linked to MDA based software development process elements.

To solve this kind of task both theoretical knowledge about the software development process specifics and technological support for SPEM model formal representation and transformation implementation between the models are required.

The suggested approach allows to define any traditional software development process in the source model (e.g., RUP, MSF, SCRUM), which corresponds to the chosen organization with its specifics. With the help of MDA transformations it is possible to obtain the model driven software development process or the target model, thus defining the adoption process from one software development process into another and formally supporting this transition process. This approach completely matches MDA four-layer architecture, since the real software development process is being formalized in SPEM model, which conforms to SPEM metamodel, and the latter is represented with MOF meta-metamodel.

4. APPROACH TO TRANSFORMATION OF SOFTWARE DEVELOPMENT PROCESS

Existing MDA solutions for working with models also allow using these solutions in some different context by replacing business models with software development life cycle models. Based on MDA framework, the first step is the development (or adoption of the existing one) of SPEM metamodel (Figure 4). SPEM is UML profile, that is defined with the help of MOF. SPEM can represent any software development process [SPE 2008]. When practically implementing this concept, SPEM is intentionally limited with the most essential concepts of software development life cycle in order to make models as simple as possible and decrease the number of transformations. If needed, it is possible to add extra elements and define additional transformations since it does not affect the general solution concept, but only extends that with additional artefacts.

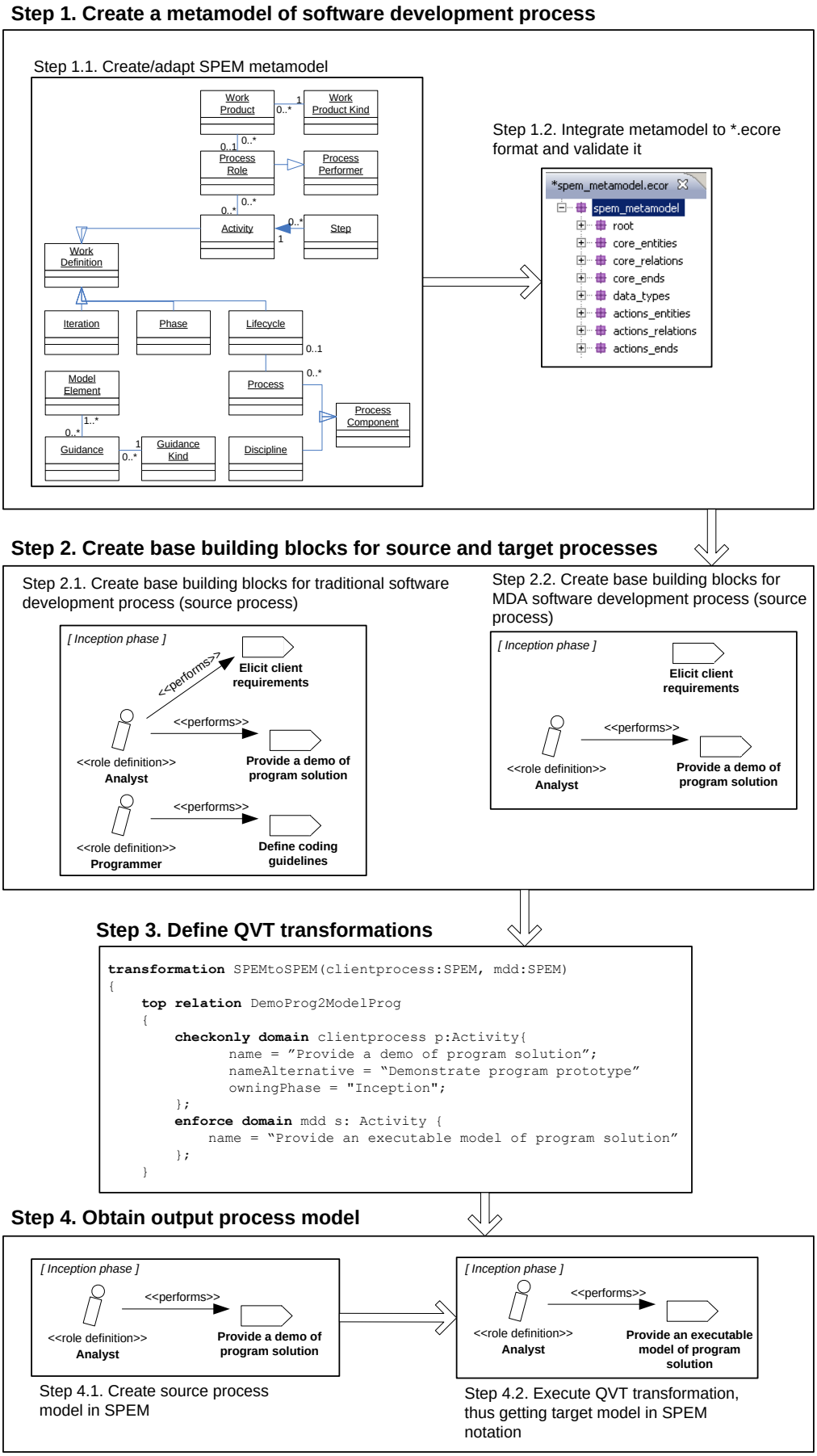


Figure 4. General schema of algorithm

The author of this thesis suggests using open source platform Eclipse and *Eclipse Modeling Framework* (EMF) data modelling and integration framework [EMF], [NIK 2010c] as the solution

architecture for this MDA based approach. EMF conforms to one of the MOF evolution branches – Essential MOF (EMOF). Within EMF architecture Eclipse models are stored in ECore format. In addition to ECore metamodelling language, EMF also supports code generation framework, which can generate Java code from ECore models.

Class names of ECore metamodel are close to UML terminology, the basis of ECore is some UML subset, constrained with its specific application within EMF.

4.2. Step one. Create a metamodel of software development process

The output of the first step is the complete SPEM metamodel, which is described with ECore metamodel. The use of this metamodel ensures generating SPEM models for formalization of software development life cycle.

4.3. Step two. Create base building blocks

During the second step there is a need to create base building blocks for both source and target processes. Within the context of this thesis, a process base block is some part of software development process, which groups some activity set within the transformation context. By decomposing software development life cycle, it is possible to acquire a set of process base blocks. And vice versa – a process can be interpreted as interconnected set of process base blocks [NIK 2010b], [NIK 2011]. Input process is the source process, which conforms to a specific organization's process (traditional software development process). Output process is the target process, which is the goal of transformation and which conforms to MDA based process.

Software development processes are unique within every software development organization. The same process elements can be named differently. However, they own the same standard activities which can be acquired from predefined process building blocks [NIK 2006a]. A similar approach is implemented in various tools - Eclipse EPF (*Eclipse Process Framework*), IBM Rational Method Composer, Enterprise Architect, MagicDraw UML (SPEM Package) and Objecteering SPEM Modeler [EPF], [MAG], [OBJ], [NIK 2010c].

The output of the second step is both source and target processes unified set of base building blocks, which describes both traditional software development process and MDA based process.

4.4. Step three. Define QVT transformations

The third step is about defining the transformations. One of the QVT family languages is used for the definition of transformations – QVT Relations. QVT Relations is OMG standard declarative language, which is intended for model-to-model transformations. MDA for QVT Relations defines both textual and graphical syntax [MOF 2008], [RED 2006].

The source of information for the definition of transformations is the knowledge on how the source elements from the process model can be mapped into the target model elements [NIK 2009a]. As described in step 2, the transformations are defined for the base building block set. Both processes are analysed with a goal to identify the impact of the model-driven development to the traditional software development process.

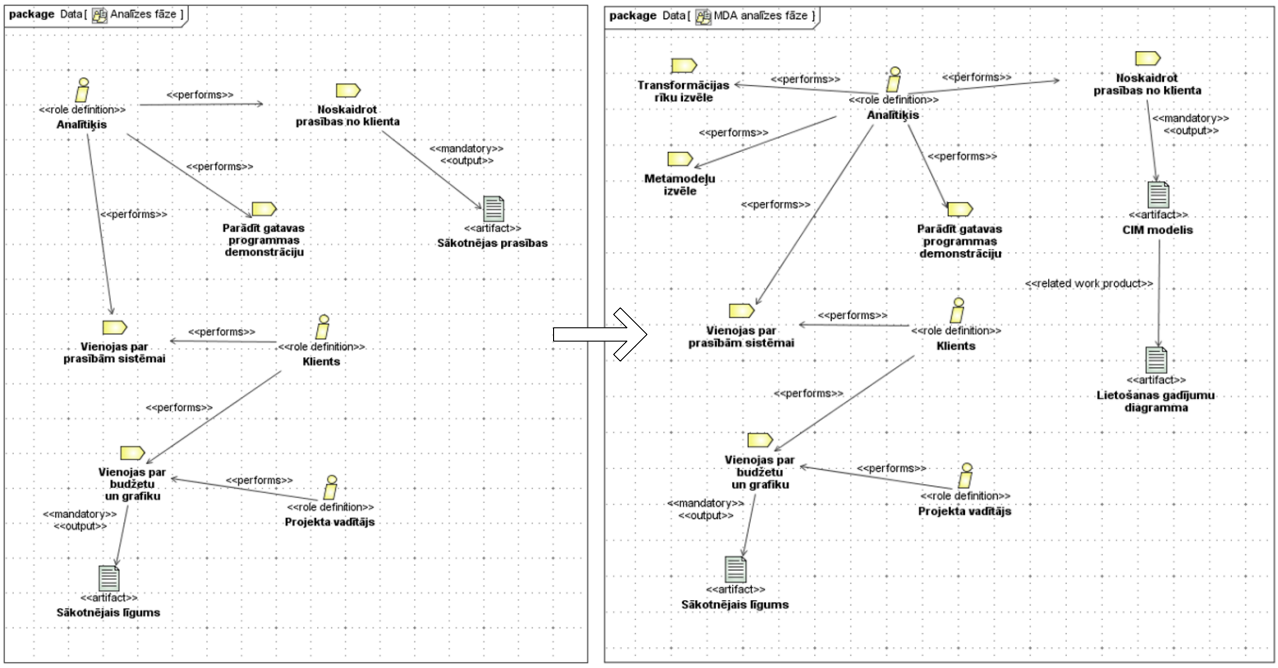
4.5. Step four. Obtain output process model

The fourth step is basically a practical application of the suggested approach. By using the software development process base building blocks the SPEM model that conforms to the traditional software development process in the chosen organization is created. This SPEM model must conform to ECore metamodel. By executing QVT Relations transformations, the source model is transformed into the target model. The target model, that conforms to the model-driven process, is the main deliverable of the fourth step.

5. PRACTICAL APPLICATION OF SUGGESTED APPROACH

The approach described in chapter 4 of this thesis is practically applied in PF software development company, reaching the goal of transforming this organizations' software development process into the model-driven [NIK 2010b]. The author's approach was applied to the process model for the existing software development process in the company, resulting in the target process

model, enriched with the model driven software development artefacts [NIK 2009e]. A fragment of the source and the target models is shown on Figure 5.



5. att. Demonstration of a of source and corresponding target process model fragments

The author of this thesis has created the prototype of the process modelling tool, that allows to automate the proposed approach (as shown on Figure 6). This prototype is based on Eclipse GMF (*Graphical Modeling Framework*) technology, which is aimed at model-driven development of graphics editors. Transformations are defined with other Eclipse plug-in mediniQVT.

This prototype can be used as a source for fully functional plug-in development which can support software development process transformations.

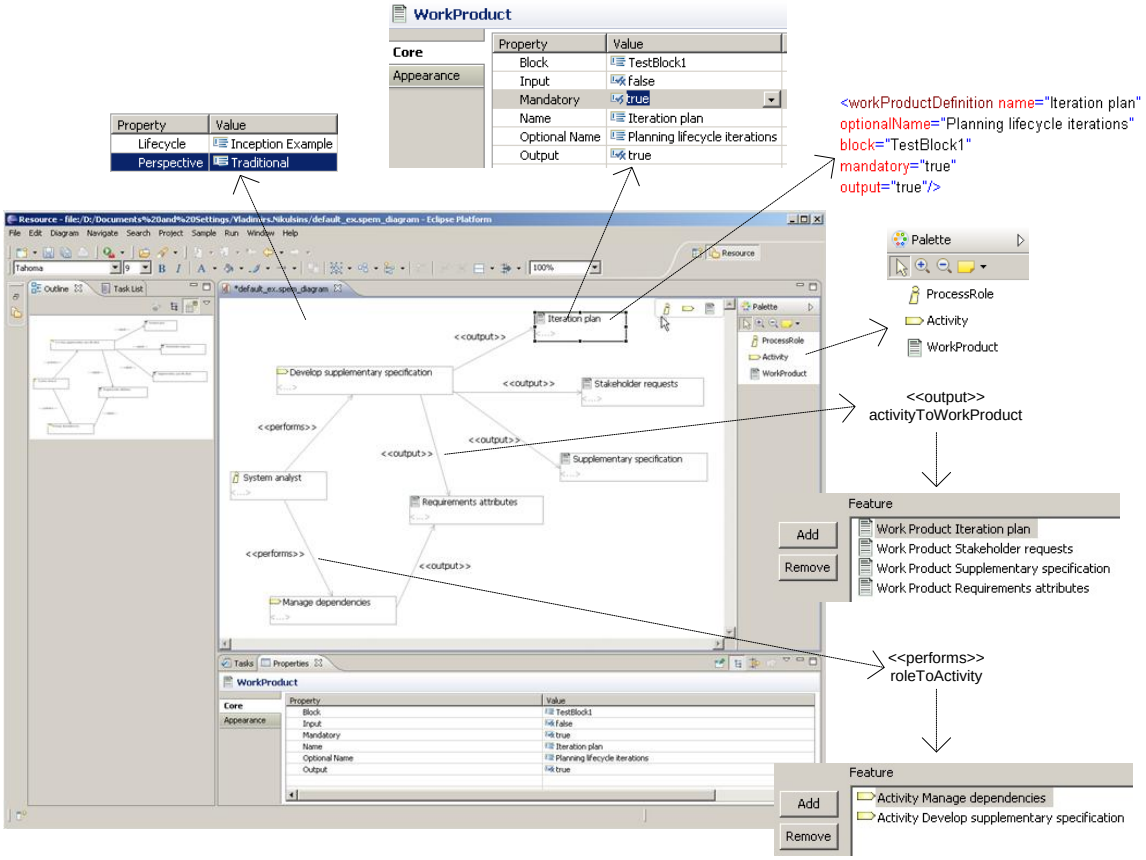


Figure 6. RUP activity group „define system" base block (prototype developed by the author of this thesis)

THE MAIN RESULTS OF THE RESEARCH AND CONCLUSIONS

Software engineering process is in its transitional period, where the traditional or code-oriented software development is unable to deal with increasing software development complexity. There is a gradual trend in moving to the model-driven software development process organization. Thus, a problem of methodological approaches that can define guidelines for the transformation between the traditional and the model-driven software development process becomes actual.

As a result of this research an approach for the transformation of the traditional software development process organization into the model-driven was proposed. The basis of this approach lies in process modelling methods and model formalization principles defined in MDA framework, linked with a set of existing tools, that support the model-driven approach. In order to verify the usability of this approach, its approbation was executed at one of Latvian software development companies. The existing software development process was analysed, and the process model was created. In order to support the transition to the model driven software development process, the transformations defined by the author were applied. To achieve the goal, the following **tasks** were completed:

- the traditional software development principles, standards and SWEBOK framework are described [SWE 2004];
- the principles of model-driven development are outlined, i.e. the definition of the model, model transformations, metamodelling; MDA software development concepts are investigated in relation to software development;
- the possibilities of integration of the traditional software development process with model-driven artefacts are examined;
- a new approach for the transition between traditional software development into MDA oriented is proposed; it is based on formal principles of process modeling, model transformation rules and existing technical solutions which support model-driven architecture;
- proposed approach was used in information technology company to enhance its software development process by introducing process model enriched with MDA artefacts;
- conclusions about MDA application, problems and future use in software development are drawn, in addition to aspects of transition from traditional software development process into MDA process.

The main result of this doctoral thesis *is a developed software development process transformation approach which offers replacement of the code-oriented software development with the model-driven software development artefacts. The proposed approach is based on the formal process modelling principles, theory of transformation rules and is using existing tools, that support process modelling and model transformations. Within the proposed solution, the tool prototype is created.*

The most significant results of the doctoral research are the following:

1. The information about software development history, terminology, software development methodologies and most significant standards and frameworks are systematized and described.
2. The possibility of application of process model and model transformations within the software development process in process handling and control is demonstrated.
3. Model-driven software development terminology and model-driven artefact mapping into the software development life cycle phases and activities are defined.
4. The transformation rules for software development process transformation enriched with model-driven development artefacts are defined.

5. The algorithm for the traditional software development process transition into the model-driven is defined in a way that it can be also modified so that any software development process could be transformed to any other (for example, MSF to RUP).
6. Based on the software development process analysis within selected information technology company, the list of recommendations on how to enrich the software development process in that company with the model-driven software development artefacts, was delivered.

Tests using the applied examples allow to draw the following conclusions:

- Software engineering discipline is still one of the engineering fields that is difficult to formalize. It is especially related to the software development process, where the role of human factor is essential. One of the software development process formalization approaches in this process analysis and handling is the usage of modelling capabilities.
- Process model transformations are especially actual for heavyweight methodology and large scale software development processes, in comparison to the trivial software development project implementation which are used in agile software development.
- There are various notations for software development process modelling. For example, some specifics of software development can be described with SPEM notation. However, SPEM notation contains no built-in mechanisms for model transformations when thinking about the transition from one software development organization into another. The author of thesis identified how the SPEM models can be used in such transition task implementation.
- MDA offers some options on how to avoid dependency on platform specifics, and enhance the abstraction level in software development. The number of available tools that support MDA can confuse end-users, especially taking into account weak tool integration and the imperfect mechanisms for artefacts interchange.
- One of the basic elements proposed by the author is the usage of SPEM base building blocks, since they can define different levels of abstraction, thus describing software development process activities and workflows. Base building blocks can be the foundation for knowledge formalization, e.g. when searching for artefacts that are different by name, but same by the content.
- Analysing the progress of the software development process in relation to model-driven architecture, it is clear that there are not enough high quality tools for end-to-end MDA concepts usage in software engineering. The approach proposed by the author of this thesis is definitely one of the steps forward in the evolution of software development processes, especially in the model-driven branch of software engineering.

The directions of **further research** could be as follows:

- complete implementation of the process model tool prototype;
- SPEM notation enrichment with new elements, that are necessary for the process model transformation tasks

The developed method is recommended by the author to be used in software development companies in order to refine their development processes. It can also be applied in large information technology companies, where software development is organized as software factories (characterized by large number of different projects), and thus the information about typical software systems organization is needed. An example of such company is *Accenture*, where the author of this thesis currently works. In *Accenture*, the initial software development modelling process is extensively used in various management tasks. However, there still is a need for the technology that can provide processing of software development processes and ease their transformation.

BIBLIOGRAPHY

Publications of the author in internationally reviewed scientific papers:

[NIK 2006b] Nikulsins V,. Nikiforova O., Sukovskis U. Analysis of Activities Covered by Software Engineering Discipline // Databases and Information Systems, Seventh International Baltic Conference on Databases and Information Systems, Communications, Materials of Doctoral Consortium, O. Vasilecas, J. Eder, A. Caplinskas (Eds.), pp. 130-138, VGTU Press „Technika" scientific book No 1290, Vilnius, Lithuania – 2006. – pp. 130–138

[NIK 2006c] Nikulsins, V., Nikiforova, O. Review on Allocation of Roles and Responsibilities among Software Development Team // The 46th Scientific Conference of Riga Technical University, Computer Science, Applied Computer Systems, October 13-14, Riga, Latvia, 2005, published in the 5th Series Computer Science. Applied Computer Systems, – 2006. – Vol. 26 – pp. 54–65

[NIK 2008a] Nikulsins, V., Nikiforova, O., Sukovskis, U. Mapping of MDA Models into the Software Development Process // Databases and Information Systems, Proceedings of the Eighth International Baltic Conference Baltic DB&IS 2008, H.-M. Haav and A. Kalja (Eds.), Tallinn University of Technology Press, Tallinn, Estonia, June 2-5, – 2008. – pp. 217–226

[NIK 2008b] Nikulsins V., Nikiforova O. Adapting Software Development Process towards the Model Driven Architecture // Proceedings of The Third International Conference on Software Engineering Advances (ICSEA), International Workshop on Enterprise Information Systems (ENTISY), 2008, Mannaert H., Dini C., Ohta T., Pellerin R. (Eds.), Sliema, Malta, October 26-31, 2008., Published by IEEE Computer Society, Conference Proceedings Services (CPS) – 2008. – pp. 394–399 (available at IEEE Xplore Digital Library; ACM Digital Library)

[NIK 2008c] Nikiforova O., Sukovskis U., Nikulsins V. Principles of Model Driven Architecture for the Task of Study Program Development // Joining Forces in Engineering Education Towards Excellence Proceedings, SEFI Annual Conference, 2008, F.K. Fink (Ed.), CD-ROM of papers – 2008. – paper ID 1162, pp 1–8. (available at SEFI Digital Library)

[NIK 2009a] Nikulsins V., Nikiforova O. Transformations of SPEM Models Using Query/View/Transformation Language to Support Adoption of Model-driven Software Development Lifecycle // The 13th East-European Conference on Advances in Databases and Information Systems (ADBIS), September 2009. Riga, Latvia, JUMI Publishing House Ltd. – 2010. – pp. 416–423

[NIK 2009c] Nikiforova O., Nikulsins V., Sukovskis U. Integration of MDA Framework into the Model of Traditional Software Development // In the series "Frontiers in Artificial Intelligence and Applications", Databases and Information Systems V, Selected Papers from the Eighth International Baltic Conference Baltic DB&IS 2008, Haav H.-M., Kalja A. (Eds.), IOS Press, - 2009. – Nr. 187. – pp. 229–242 (available at Google Books; IOS Press)

[NIK 2010a] Nikulsins V. Transformations of Software Process Models to Adopt Model-Driven Architecture // International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2010), Proceedings of the 2nd International Workshop on Model Driven Architecture and Modeling Theory Driven Development (MDA&MTDD 2010), Osis J., Nikiforova O. (Eds.), Greece, Athens, July 2010, SciTePress, Portugal. – 2010. – pp. 70–79 (available at Thomson Reuters; Inspec; dblp.uni-trier.de Computer Science bibliography)

[NIK 2010b] Nikulsins V., Nikiforova O., Kornijenko J. SPEM model transformations to adopt MDA in practice. Databases and Information Systems // Proceedings of the Ninth International Baltic Conference Baltic DB&IS 2010. Databases and Information Systems, 2010. Barzdins J., Kirikova M. (Eds.). 2010. July, Latvia, Riga. Riga: University of Latvia Press, Riga, Latvia, – 2010. – pp. 295–307

[NIK 2010c] Nikulsins V., Nikiforova O. Tool Integration to Support SPEM Model Transformations in Eclipse // Scientific Journal of Riga Technical University, Computer Science,

Applied Computer Systems. The 50th International Scientific Conference of Riga Technical University, Riga, Latvia: RTU Publishing, – 2010. – Vol. 43. – pp. 60–67 (available at EBSCO)

[NIK 2011] Nikulsins V., Nikiforova O., Kornijenko J. An Approach for Enacting Software Development Process: SPEM4MDA // Frontiers in Artificial Intelligence and Applications, Vol. 224, Databases and Information Systems VI – Selected Papers from the 9th International Baltic Conference, DB&IS 2010. Barzdins J., Kirikova M. (Eds.). Amsterdam: IOS Press, – 2011. Vol. 224. – pp. 55–65 (available at IOS Press; ACM Digital Library)

Research report:

[NIK 2009e] Nikiforova O., Nikulsins V., Kornijenko J. et. al. Implementation of model-driven software development process in software development company „PF", under the research project of Riga Technical University Nr. FLPP-2009/10 „Development of Conceptual Model for Transition from Traditional Software Development into MDA-oriented". – 2009.

Other bibliography used in this summary:

[BEL 2002] Belaunde M. Initial identification of issues for further research. Deliverable 2.2. MODA-TEL. Interactive Objects Software GmbH – 2002. / Internet: http://www.modatel.org/~Modatel/pub/deliverables/D2.2-final.pdf

[BOT 2010] Botteri P., Cowan D., Deeter B. et. al. Bessemer's Top 10 Laws of Cloud Computing and SaaS. Bessemer Venture Partners – 2010. / Internet: http://www.bvp.com/downloads/saas/BVPs_10_Laws_of_Cloud_SaaS_Winter_2010_Release.pdf

[BRE 2001] Breton E., Bezivin J. Process-Centered Model Engineering. Fifth IEEE International Enterprise Distributed Object Computing Conference, 2001. – France: IEEE – 2001. / Internet: http://www.sciences.univ-nantes.fr/lina/atl/www/papers/edoc.pdf

[BRO 2005a] Brown A., Conallen J., Tropeano D. Models, Modeling, and Model Driven Development // Model-Driven Software Development, Springer, Berlin – 2005. – pp. 1–17.

[BRO 2005b] Brown A., Conallen J. An introduction to model-driven architecture. Part III: How MDA affects the iterative development process – May 2005. / Internet: http://www.ibm.com/developerworks/rational/library/may05/brown/index.html

[CAU] Cauna E. Lielā terminu vārdnīca / Internet: www.termini.lv

[CHI 2007] Chitforoush F., Yazdandoost M., Ramsin R. Methodology Support for the Model Driven Architecture // Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. 14th Asia-Pacific Software Engineering Conference APSEC 2007. – 2007.

[CMM] Software Engineering Institute. Capability Maturity Model Integration (CMMI) / Internet: http://www.sei.cmu.edu/cmmi/

[COC 1999] Cockburn A. A Methodology Per Project. / Internet: http://alistair.cockburn.us/crystal/articles/mpp/methodologyperproject.html

[COM 2006] Combemale B., Cregut, X. Towards a Rigorous Process Modeling With SPEM. – France: Rennes University – 2006. / Internet: http://www.combemale.net/research/phd/2006/iceis250406-CCCC-poster401.pdf.

[DEB 2007] Debnath, N., Zorzan, F.A., Montejano et. al. Transformation of BPMN subprocesses based in SPEM using QVT // Electro/Information Technology, 2007 IEEE International Conference. – 2007. – pp. 146–151

[DIA 2010] Diaw S., Lbath R., Thai V. et. al. SPEM4MDE: a Metamodel for MDE Software Processes Modeling and Enactment // Third Workshop on Model-Driven Tool & Process Integration. MDTPI 2010, Paris, France. – 2010. – pp. 109-121.

[ERI 2004] Eriksson H.-E., Penker M., Lyons B. et. al. UML2 Toolkit. Wiley Publishing, Indianapolis, Indiana, USA. – 2004. – 511 p.

[ESI 2002] Steinhau R. Model Driven Architecture Definition and Methodology. Deliverable 3.1. MODA-TEL. Interactive Objects Software GmbH – 2002 / Internet: http://www.modatel.org/public/deliverables/D3.1.htm

[ESI 2003] Process Model to Engineer and Manage the MDA: Model-driven Architecture inSTrumentation, Enhancement and Refinement Approach. European Software Institute – 2003 / Internet: http://modeldrivenarchitecture.esi.es/pdf/Deliverable-D32.zip

[ESS 2009] Essvale Corporation Limited. Business Knowledge for IT in Insurance: A Complete Handbook for IT Professionals. 63 Apollo Building, 1 Newton Place, London E14 3TS. Essvale Corporation – 2009. – 248 p.

[EVA 2003] Evans A., Sammut P., Willans J.S. (eds.) Metamodelling for MDA // First International Workshop Proceedings, York, UK – November 2003. / Internet: http://www.cs.york.ac.uk/metamodel4mda/onlineProceedingsFinal.pdf

[FAV 2010] Favre L. Model Driven Architecture for Reverse Engineering Technologies: Strategic Directions and System Evolution. 701 E. Chocolate Avenue, Hershey PA 17033, United States of America: IGI Global publishing – 2010. – 459 p.

[FEN 2006] Feng Y., Mingshu L., Zhigang, W. SPEM2XPDL: Towards SPEM Model Enactment. Bejing, China: The Chinese Academy of Sciences – 2006 – 6 p.

[FOR 2009] Forrester C., Buteau E. L., Shrum, S. B. CMMI for Services, Version 1.2. Technical Report, Software Engineering Institute – 2009.

[GAV 2004] Gavras A., Belaunde M., Pires L.F. et. al. Towards an MDA-based development methodology for distributed applications – 2004. / Internet: http://eprints.eemcs.utwente.nl/7100/01/paper3-3.pdf

[GUT 2004] Guttman M. Getting Started With OMG's MDA. Application Development. 2004. / Internet: http://www.softwaremag.com/L.cfm?doc=2004-09/2004-09mda

[GUT 2007] Guttman, M., Parodi, J. Real-Life MDA. Solving Business Problems With Model Driven Architecture. Morgan Kaufmann Publishers – 2007.

[HUG 2009] Hug C., Front A., Rieu D. et. al. A method to build information systems engineering process metamodels. Journal of Systems and Software. Volume 82, Issue 10. – 2009. – pp. 1730-1742.

[HUS 2011] Hussman H., Meixner G., Zuehlke D. (Eds.). Model-Driven Development of Advanced User Interfaces. Studies in Computational Intelligence, First Edition. – Berlin, Heidelberg, Germany: Springer-Verlag – 2011. – 324 p.

[IEE 1990] IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology. The Institute of Electrical and Electronics Engineers – 345 East 47th Street, New York, NY 10017, USA: IEEE Standards Board. – 1990. – 84 p. / Internet: http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf

[IEE 2008] IEEE Std 12207-2008. ISO/IEC 12207. IEEE Systems and software engineering - Software life cycle processes. The Institute of Electrical and Electronics Engineers. IEEE Standards Activities Department - 445 Hoes Lane, Piscataway, NJ 08854, USA: IEEE Standards Board. – 2008. – 122 p. / Internet: http://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-2008.pdf

[ISO 2005] ISO/IEC 19502:2005 International Standard. Information Technology – Meta Object Facility (MOF). – Switzerland, Geneva. – 2005.

[JAY 2007] Jayaswal B.K., Patton P.C. Design for Trustworthy Software: Tools, Techniques, and Methodology of Developing Robust Software. First Edition. Prentice Hall, 2007. 840 p. http://ptgmedia.pearsoncmg.com/images/0131872508/samplechapter/0131872508_ch01.pdf

[KHA 2004] Balbo S., Khan A. A Tale of two Methodologies: Heavyweight versus Agile // Department of Information Systems, The University of Melbourne. AusWeb04. The Tenth Australian World Wide Web Conference. – 2004 / Internet: http://ausweb.scu.edu.au/aw04/papers/edited/balbo/

[KLE 2003] Kleppe A., Warmer J., Bast W. MDA Explained: The Model Driven Architecture: Practice and Promise. Addison Wesley – April 2003. – 167 p.

[KRU 2000] Kruchten P. The Rational Unified Process An Introduction, Second Edition, Addison Wesley – 2000. – 320 p.

[LAN 2004] Langlois B., Exertier D. MDSoFa: a Model-Driven Software Factory. Thales Research & Technology France. – October 2004. / Internet: www.softmetaware.com/oopsla2004/langlois.pdf

[MAN 2006] Mansell J., Bediaga A., Vogel R. et. al. Process Framework for the Successful Adoption of Model Driven Development. A. Rensink and J. Warmer (Eds.): ECMDA-FA 2006, LNCS 4066. – 2006. – pp. 90–100.

[MEL 2002] Mellor S. J., Balcer J. M. Executable UML: A Foundation for Model-Driven Architecture. Addison Wesley. – 2002. 416. p.

[MEL 2004] Mellor S.J., Scott K., Uhl A. et. al. MDA Distilled: Principles of Model-Driven Architecture. Addison Wesley – 2004. - 176 p.

[MIC 2002] Microsoft. Microsoft Solutions Framework: MSF Process Model v. 3.1. White Paper, Microsoft Corporation – June 2002.

[MIC 2003] 2710B: Analyzing Reuirements and Defining Microsoft .NET Solution Architectures. Microsoft Official Course. – April 2003.

[MOF 2008] Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.0, 2008 / Internet: http://www.omg.org/docs/formal/08-04-03.pdf

[NAU 1969] Naur P., Randell B. (Eds.) Software Engineering. Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11th of October 1968. / Internet: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.pdf

[OMG 2006] Object Constraint Language Specification, version 2.0. Prepared by OMG: OMG – 2006 / Internet: http://www.omg.org/cgi-bin/apps/doc?formal/06-05-01.pdf

[OMG] OMG Model Driven Architecture. / Internet: http://www.omg.org/mda

[OSI 2010] Osis J., Asnina E. Model-Driven Domain Analysis and Software Development: Architectures and Functions. IGI Publishing. – 2010. – 487 p.

[PAL 2006] Palomäki J., Keto H. A Process-Ontological Model for Software Engineering // Philisophiocal Foundations on Information Systems Engineering. Tampere University of Technology/Pori. Department of Information Technology / Internet: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-240/paper3.pdf

[PEI 2010] Peixoto D. C. C., Batista V. A., Resende R. F. et. al. How to Welcome Software Process Improvement and Avoid Resistance to Change // Federal University of Minas Gerais, Brazil – 2010. / Internet: http://homepages.dcc.ufmg.br/~vitor/artigos/icsp2010_paper.pdf

[PIL 2005] Pilone D., Pitman N. UML 2.0 in a Nutshell. First Edition. – Sebastopol, USA. O'Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. – 2005. – 240 p.

[RAT 1998] Rational Software Corporation. Rational Unified Process. Best Practices for Software Development Teams // Rational Software White Paper. – 1998. – 21 p.

[RED 2006] Reddy S., Venkatesh R., Ansari Z. A relational approach to model transformation using QVT Relations // Tata Research Development and Design Centre, Pune, India. – 2006. / Internet: http://www.iist.unu.edu/~vs/wiki-files/QVT-TRDCC.pdf

[RIO 2006] Rios E., Bozheva T., Bediaga A. et. al. MDD Maturity Model: A Roadmap for Introducing Model-Driven Development // European Conference on Model Driven Architecture – Foundations and Applications, Rensink A., Warmer J. (Eds.), ECMDA-FA 2006 – Bilbao, Spain: Springer – 2006. – pp. 78-89.

[ROY 1970] Royce W. Managing the Development of Large Software Systems. IEEE WESCON, 1970. / Internet: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

[SCA 2001] Scacchi W. Process Models in Software Engineering. Institute for Software Research, University of California, Irvine – 2001. – 24 p. / Internet: http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf

[SER 2005] Serour M.K., Henderson-Sellers B. Resistance to Adoption of an OO Software Engineering Process: An Empirical Study // University of Technology, Sydney, Australia. European and Mediterranean Conference on Information Systems (EMCIS) –2005.

[SIE 2001] Siegel J., OMG Staff Strategy Group. Developing in OMG's Model-Driven Architecture. Object Management Group White Paper. – 2001.

[SPE 2008] Software Process Engineering Metamodel Specification (SPEM), Version 2.0, OMG specification. – 2008. / Internet. – http://www.omg.org/cgi-bin/doc?formal/2008-04-01

[STA 2006] Stahl T., Voelter M. Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, Ltd., 2006. 428 p.

[STE 1999] Stelzer D., Mellis W. Success Factors of Organizational Change in Software Process Improvement // Software Process Improvement and Practice, Volume 4, Issue 4. John Wiley & Sons Ltd –1999. / Internet: http://informationsmanagement.wirtschaft.tu-ilmenau.de/forschung/documents/successf.pdf

[STE 2003] Steinhau R. Guidelines for the Application of MDA and the Technologies covered by it. Deliverable 3.2. MODA-TEL. Interactive Objects Software GmbH – 2003. / Internet: http://www.modatel.org/~Modatel/pub/deliverables/D3.2-final.pdf

[SWE 2004] Abran A., Moore J.W., Bourque P. et. al. (Eds.) Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE. – Los Alamitos, California: IEEE Computer Society Press – 2004. – 302 p.

[THA 2005] Thayer R.H., Christensen M.J. (Eds.) Software Engineering, Volume 1: The Development Process, Third Edition – 11 River Street, Hoboken, NJ 07030, ASV: John Wiley & Sons. – 2005. – 540 p.

[TRA 2004] Tratt L. Model transformations and tool integration. – London, UK: Department of Computer Science, King's College London, Springer-Verlag – 2004. –12 p.

[TSU 2011] Tsui F., Karam O. Essentials of Software Engineering, Second Edition. Southern Polytechnic State University, Marietta, Georgia, USA. John and Bartlett Publishers, 40 Tall Pine Drive, Sudbury, MA 01776. – 2011. – 409 p.

[VLI 2008] Vliet H. Software Engineering: Principles and Practice, Third Edition. The Atrium, Southern Gate, Chester, West Sussex PO19 8SQ, England – John Wiley & Sons. – 2008. – 713 p.

[VOG 2006] Vogel R., Mantell K. MDA adoption for a SME: evolution, not revolution Phase II. // The Second European Conference on Model Driven Architecture (ECMDA 2006). Foundations and Applications. Bilbao, Spain. – 2006. – 13 p.

[WAK 2001] Wake W. C. Extreme Programming Explored. The XP series, Addison-Wesley Professional – 2001. – 192 p.