Certification of MDA Tools: Vision and Application

Oksana Nikiforova, Natalja Pavlova, Antons Cernickins , Tatjana Jakona
Department of Applied Computer Science
Riga Technical University
Riga, Latvia
{oksana.nikiforova, natalja.pavlova, antons.cernickins, tatjana.jakona}@rtu.lv

***Abstract*—Currently software system development is under the conversion, where traditional code oriented software development is transformed into model driven approach. A lot of methods and tools are proposed to support main statements of the model driven software development. However, there do not exist mechanisms for notification of how much valuable are the Model Driven Architecture (MDA) support tools and how close they are to the main idea of model usage for software development. One of the possible solutions how to solve the problem of selection of appropriate MDA tool can be certification process similar to that in other industries. The paper proposes the framework for such certification and shows an application of it for several MDA support tools.**

Keywords-MDA; CASE-tools; certification; modeling.

I. INTRODUCTION

The complexity of software systems permanently increases. It requires from developers carefully select technologies and tools, which will be used during software development process. Researchers and developers try to automate software development process in order to minimize human and material resource costs. Therefore in one hand a big number of CASE-tools were created, each of which support some part of software lifecycle. Functionality of the CASE-tools may be duplicated. The concurrence occurs on this market. Unified opinion which tool is the best and how tools could be evaluated does not exist.

In other hand there are developed different processes, approaches and methods to software development. For example such processes as Rational Unified Process (RUP) [1], Microsoft Solutions Framework (MSF) [2], SCRUM [3], Extreme Programming (XP) [4], etc. exist. Model Driven Architecture (MDA) [5] proposed by Object Management Group (OMG) is popular approach to software development and can be applied within any software development process. Therefore a set of CASE tools, which support also several activities defined by MDA, also appear on the tool market. And it became much more complicated to examine CASE-tools, which also support model and transformation chain of MDA.

The area of the described here research is software development using CASE-tools in the framework of MDA. Software market is crowded with a variety of CASE-tools that automate stages of the development in the framework of MDA. Not developed any standardized procedures or criteria for how to assess compliance of CASE-tool to standards of MDA, to evaluate what part of the MDA chain considered CASE-tool supports. The goal of this paper is suggest the possibility of certification of CASE-tools based on a considered here evaluation criteria of compliance to the MDA.

The second section describes the difference between the term of Model Driven Software Development (MDSD) and principles of MDA applied for the software development. The third section describes existing researches in the area of MDA tool certification. The fourth section shows example of CASE-tool evaluation for functionality and portability aspects. In the last section the described research is concluded.

II. MODEL DRIVEN APPROACH IN SOFTWARE DEVELOPMENT

Requirements of customers and hence the software becomes more sophisticated and complex with the time. Therefore, developers should be more qualitative and should have tools to satisfy the needs of quality of the software on the level, required by clients, to respect deadlines and to deliver software that functions properly. According to Standish Group [6], only 29% of projects have been succeeded in 2004 (i.e. done in time, met client's expectations, while being not over budget). Paying attention to the development processes of the typical software development company, similar steps, tools and tests will be seen [6]. In order to optimize these activities, model-based approaches may be used, providing manipulations with models under meta-modeling process, as well as the usage of CASE tools for model transformation and generation. This approach to software development is realized with MDD [7].

Model Driven Development appears because there was a necessarily to decrease efforts, to create and use analysis and design models at each stage of the software development process and to automate the transformation of the models [6]. The separation of concerns is another foundation of MDD that provides the separation of high-level business logic from system's architecture and deployment platform. MDA initiative, the primary example of MDD, was introduced by OMG in 2001 to satisfy the needs of the modern software industry [5], [8].

MDA proposes to use models on every stage of software development, specifying a set of tools that supports

 ISBN: 978-1-61208-165-6

construction of models with design and architectural patterns [8]. According to traditional software development life cycle, the application of Model Driven Development should consider the modeling approach as such. Unlike MDD, the MDA approach considers models as central part of the development process (assuming that model represents a set of diagrams, used to express the whole software system) [9]. MDA may be considered as a next stage in the evolution of software development process, which tends to bring some improvements into each step of the software development life cycle [8]. MDA is a framework, which contains technical standards developed in the supervision of OMG (in this case, OMG provides the guidelines of MDA application to software development) [8]. There are four principles that underlie the MDA approach [10]:

1. Models constructed with a well-defined notation are a milestones of system representation for enterprise-scale solutions;
2. System development is performed with construction of a set of models and execution of model transformations;
3. Models and transformations among them are described in a formal form with meta-models on MOF this description could serve as a basis for automation through different CASE tools;
4. The broad usage of model-based approaches requires standards to provide satisfaction of costumers and highest qualification of developers.

One of the milestones of MDA considers the text description of the models, formal descriptions of a system, models and code and possibility to apply the formal transformations on every model of a system, to refine it and obtain model, which is closest to user needs [10]. Considering the resources needed for software development, there is a need to reduce the overall production costs, making the software development process more profitable [11]. Here, the reuse of the existing models, patterns or code may be used (thought, it may be a way too complex or impossible). MDA proposes the following set of activities, which may improve the software development process and make an easier reuse of some components [11]:

1. Choose application model that corresponds with a problem domain;
2. Subset the model as necessary;
3. Choose models in accordance with the implementation technology platform;
4. Define the interconnection between models;
5. Generate the program code for software system.

In many cases, the necessity of introducing some changes into developed system (or system under development) appears. From this point, changes are introduced into the application model only (1) — changes will be automatically provided to the lower models. When the environment of system development should be changed, models for the new environment should be selected (3); program code should also be regenerated (5) [12]. Therefore, the application models are not changed, meaning that costs are lower, productivity is higher, as well as the maintenance of the system becomes much cheaper. With this approach each model, which is constructed in the framework of MDA guidelines, can be subsequently reused [11]. Fig. 1 shows the supporting component model of Model Driven Architecture. Components in Fig. 1 are depicted into the framework of MDA models and its transformations within the authors defined levels for system domain abstractions [9].

The MDA proposes to construct four basic models for developed system (Fig. 1):

1. Computation Independent Model (CIM) that reflects to business and its models— defined at problem domain level in Fig. 1;
2. Platform Independent Model (PIM) that reflects to analysis and design models of software system to be developed—defined at solution domain level in Fig. 1;
3. One or many Platform Specific Models (PSM) that reflect to detailed design models of software system under construction—defined at software domain level in Fig. 1;

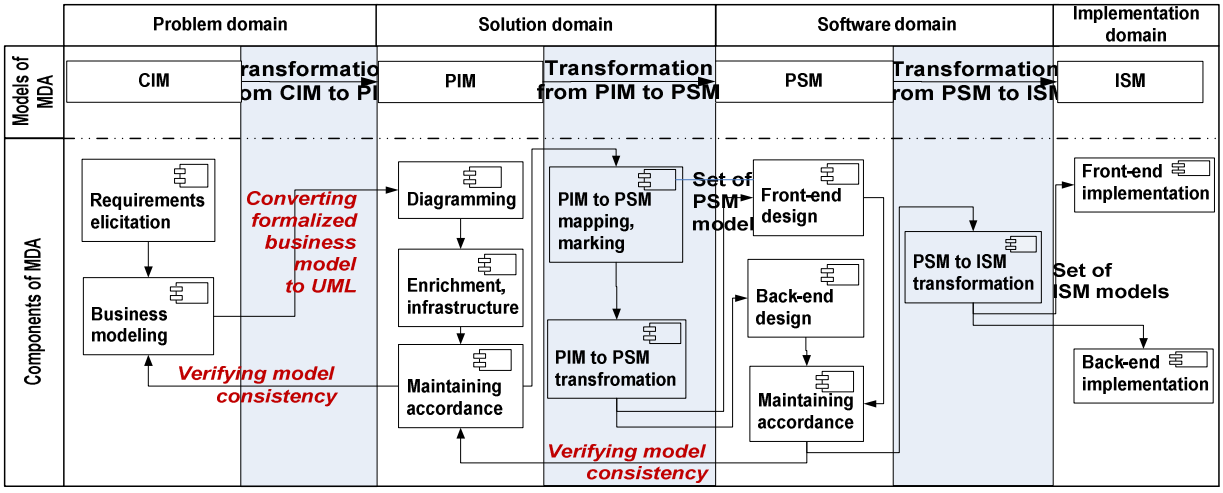

Figure 1. The component model of MDA (adopted from [13])

4. One or many Implementation Specific Models (ISM) that reflect to implementation and runtime models – defined at implementation domain level in Fig. 1.

Also, MDA components may be reflected to the main blocks of Model Driven Development. These blocks are the following [14]:

1. A model repository;
2. One or more domain modeling languages;
3. One or more workbench environments;
4. One or more modeling tools;
5. One or more transformation tools.

The components of MDA, shown on Fig. 1 are representing all of the activities included in the MDA-driven software development process. Dependence on information exchange, which is imported/exported from one component to another, is written on the arrows between components on Fig. 1.

Regular font on arrows between components means that the ability of import/export of models is possible. Transitions between components, which can be performed only by the human at this time, are expressed in italic. Authors' research [15] discusses different types of model transformations (e.g. formal, semi-formal, based on hints or manual) to satisfy the statements of MDA on model transformations. Up to the year 2006 the conclusion stated that there is no solution available to define the complete transformation CIM-to-PIM-to-PSM-to-Code. The weakest link here is exactly the construction of PIM or transformation from CIM to PIM. Solutions focused on construction of CIM and CIM-to-PIM transformations cannot insure that PIM is containing all the necessary information, as well as that the presentation of PIM is formal enough to be able to transform it into correct PSM [15]. Authors' efforts to find CASE tools up to date for CIM-to-PIM transformation and to state the component to support that activity carry to the point, that still there is no guarantee result of CIM-to-PIM transformations. Also, the verification of model consistency is under investigations and the role of model interchange and interchange standards become more and more important.

III. MAIN CONCEPTS OF MDA TOOLS CERTIFICATIONS

The idea lying behind the research is to provide a set of guidelines on the actual implementation of the MDA for the purpose of promoting it as a holistic approach for software development across the IT community. A branch of standards provided within MDA is defined in a form of specification, meaning that the specification-based testing may be used as a basis for compliance assessment [16]. In particular, the conformance statement for CORBA provided by The Open Group [17] is done this way. In fact, the compliance itself is nothing else but the satisfaction of software implementation to the standard specification [16]. [16] comes with an idea of considering the compliance test suite generation as a branch of constraint satisfaction problem, in which the first-order predicate is given and processed to find models that satisfy it. Following this work, instead of starting from a concrete set of constraints and trying to find the appropriate models, the construction (as well as the further classification) of all possible models is considered.

When it comes to development of a new certification scheme, the first and the foremost task is to define the object of certification [18]. According to [18], the following types of certification are possible:

- Product certification (accordance with particular technical standard);
- Process certification (accordance with ISO 9000 or similar standard);
- Personnel certification;
- Accreditation of certification bodies (the certification of certifiers).

[18] summarizes the study on various certification schemes and categorizes them into several groups, also providing a general structure of certification process itself, as well as presenting a new certification scheme used in space technology.

In fact, the type of certification procedure for current research can be determined as a combination of both the product and the process certification. Such a mixture of types will provide a more detailed outlook on various options to be considered in the certification scheme.

Basically, the former type of certification is considered, as software development tools (i.e., software products) are involved in the research. This may also include the specification of the most common features and options defined to clarify the accordance level of each tool from various perspectives (discussed in [19]).

As far as MDA-oriented software development life cycle represents the process, the latter type of certification should also be considered.

In order to provide a solid background for the certification scheme, as well as to clarify the means of the MDA tool as such, [20] is considered. [20] reviews the MDA approach within the variety of the CASE tools, which are proposed as supporting for MDA activities. The provided specification of MDA tools consists of seven categories, which definition and details are described in [20]:

1. Accordance with MDA-oriented life cycle—the accordance level of software development life cycle supported by a tool, which includes MDA-oriented activities combined into such subcategories as knowledge formalization (CIM), system model refinement (PIM), PIM-to-PSM mapping, system model implementation (PSM), and transformation support;
2. Functional capabilities—the functional capabilities of a tool in such fields as environment, modeling, implementation, testing, documenting, project management, configuration management;
3. Reliability—the capability of a tool to maintain the appropriate level of performance under certain conditions for a certain period of time, including repository management, automatic backup capabilities, data access management, error processing capabilities, as well as fault analysis capabilities;
4. Usability—usage efforts and individual assessments of such usage, including user interface, licensing and

 ISBN: 978-1-61208-165-6

localization options, ease of use, quality of documentation etc.;

5. Efficiency—the amount of resources needed to maintain the appropriate level of performance under certain conditions, including technical requirements, workload efficiency, as well as performance;
6. Maintainability—efforts needed to make specified modifications;
7. Portability—ability of a tool to be transferred to another environment.

The mentioned criteria involve several aspects of the features of the CASE-tool, such as usability and application. Current research is devoted to evaluation of CASE-tools regarding to modeling and implementation capabilities as it is important development property in the framework of MDA [21].

In order to clarify a vision on a certification scheme to assess the compliance of MDA tools, a conceptual framework is proposed. In fact, this framework should be used to verify the output produced by MDA tools. Whereas a wide variety of the tools intended for specific purposes (e.g., mapping definition) may be used [19], an additional specification-based assessment of these tools is considered (discussed in [19]).

In short, models defined by MDA are used to describe the MDA-oriented software development life cycle [11], [19], [5], namely they are CIM, PIM , PSM and ISM .

However, the only models to be specified and promoted by OMG (i.e., described in details) are PIM and PSM [11]. In fact, OMG does not provide any specific requirements for CIM (meaning that it is not "computational," not formal enough, etc.), as well as ISM itself — the actual source code generated from PSM—from modeling perspective looks out of scope. Despite this, all four layers are somehow covered by various software development tools.

The conceptual framework considers these four models as individual blocks, each of them having their own input and output. The origin of this idea has come from black box testing [22]: whereas software system is considered as a black box, the only thing to be analyzed is the output produced by specific input. Therefore, developer does not need to understand why the compiled code does what it does; here, the requirements are used to determine the correct output of black box testing.

In fact, the main artifacts for the conceptual framework are inputs and outputs. As far as CIM and ISM are out of scope from the perspective of OMG standards, the conceptual framework does not cover the according artifacts. The actual tool use in each block (i.e., what operations are performed) is also not the matter of high importance.

However, the main concern for each tool is the support of XMI standard [23]. In order to perform a transition from raw output to qualified input, the conceptual framework assesses the output from each tool. If tool conforms to OMG standards, then the output from this tool should be opened in other tool with no problems. If not, the conceptual framework would provide an appropriate suggestion on where the root of the problem lies.

While OMG does not provide any constraints (i.e., does not restrict) on the modeling language notation used with MDA (however, the use of UML is strongly recommended) [11] [5], the use of XMI for assessment of software development tools seems to be the only valuable option. This assessment is considered to be formal: a specification is said to be formal when it is based on a language that has a well-defined semantic meaning associated with each of its constructs [24]. It is this formalism, which allows the model to be expressed in a format such as XML, in accordance with a well-defined schema (XMI).

The specification of XMI standard as such is used to create the XML Schema of XMI standard [25], which provides a means by which the syntax and the semantics of an XMI document can be validated. XMI Schemas must be equivalent to those generated by the XMI Schema production rules specified in [23]. Equivalence means that XMI documents that are valid under the XMI Schema production rules would be valid in a conforming XMI Schema; in turn, those XMI documents that are not valid under the XMI Schema production rules are not valid in a conforming XMI Schema [23].

After the XML Schema of XMI standard is created, the developed tool creates a document data model, which consists of [25]:

- Vocabulary (element and attribute names);
- Content model (relationships and structure);
- Data types.

This model is used for further validation of XMI documents. Validation can determine whether the XML elements required by [23] are present in the XML document containing model data, whether XML attributes that are required in these XML elements have values for them, and whether some of the values are correct.

IV. EXAMPLE OF TOOL CERTIFICATION PROCESS

In order to examine the modeling and implementation capabilities of tools, a scope of correspondence should be defined first. Considering the information from previous Sections, the main concern is concentrated on PIM, its refinement, as well as further transition to PSM with similar concentration, accordingly. In addition, the specification of MDA tools provided in Section 3 should also be considered.

Based on [26], the following tools have been selected for evaluation:

- ArgoUML 0.28;
- Altova UModel 2009;
- Sparx Systems Enterprise Architect 7.5.843;
- IBM Rational Enterprise Architect 7.0.0;
- MyEclipse Enterprise Workbench 7.1.1.
- MS Visual Studio 2010

[26] considers these tools as UML tools, which provide source code generation capabilities from UML diagrams, as well as reverse engineering capabilities. However, the only use of UML does not guarantee that tool is "MDA complaint". That is why the most important features of UML tools should be mapped to the appropriate features of the MDA tools.

 ISBN: 978-1-61208-165-6

The compliance to MDA is defined in Section 3 with 7 view points: accordance with MDA-oriented life cycle; functional capabilities; reliability; usability; efficiency; maintainability and portability. Currently selected CASE-tools are evaluated according some of functional capabilities, namely, modeling and implementation, which are represented with UML support and programming languages support. As well as for portability, this is presented with supporting of different platforms, interchange format and programming languages in source code generation and reverse engineering. Therefore, a model defined in appropriate modeling notation (as was mentioned before, the use of UML is suggested), a model enrichment (transition) to meet the specifics of selected platform, generation of platform-specific source code, as well as support for MOF/XMI should be considered as the most important features of these tools.

Other features like configuration management, testing, project management, etc. are the matter of secondary importance.

These tools feature a source code generation approach based on template definition, meaning that a file (i.e., template) describing the use of meta-data information should be defined first. If several tasks are considered, it is possible to define a set of templates, where each template deals with an appropriate task (here, a nested hierarchy is considered, where main template contains information about complementary templates). Certain tools (such as UModel and Enterprise Architect, namely) provide an ability to redefine the set of supplied generation templates, whereas other tools are unable to provide such a feature.

Table 1 provides an outlook on several features declared by tool vendors that are important for correspondence with the proposed approach (based on [26]).

TABLE I Declared features of corresponding UML tools (based on [26])

	ArgoUML	Altova UModel	Sparx Systems Enterprise Architect	IBM Rational Rose Enterprise	MyEclipse Enterprise Workbench	MS Visual Studio 2010
Common features						
UML	1.4	2.2	1.3, 1.4, 2.0, 2.1	1.4	2.1	2.0
UML Profiles	•	•	•		•	•
MOF/XMI	1.1, 1.2	2.1	1.1, 1.2, 2.1		1.0	2.1
XMI import/export	•	•	•		•	•
UML Diagram support						
Class	•	•	•	•	•	•
Component	•	•	•	•	•	•
Composite structure		•	•		•	
Deployment	•	•	•	•	•	
Object	•	•	•	•	•	•
Package		•	•		•	•
Profile		•	•		•	•
Activity	•	•	•	•	•	•
State machine		•	•		•	
Statechart UML 1.x	•		•	•		
Use case	•	•	•	•	•	•
Communication			•		•	
Collaboration UML 1.x	•		•	•		
Interaction overview			•		•	
Sequence	•	•	•	•	•	
Timing			•		•	
Source code generation capabilities						
CORBA IDL			•	•		
Java	•	•	•	•	•	
C++	•	•	•	•		•
C#	•	•	•	•		•
VB.NET		•	•	•		•
PHP	•		•			
Other			Ada, Python, ActionScript			J#, JScript
Reverse engineering capabilities						
CORBA IDL	•		•	•		
Java	•	•	•	•	•	•
C++		•	•	•		•
C#		•	•	•		•
VB.NET		•	•	•		•
PHP			•			
Other			C, Python, Visual Basic, ActionScript			J#, JScript

To sum up, UModel and Enterprise Architect provide the richest set of functional features, with the latter being the most functional one in terms of source code generation and reverse engineering capabilities. However, when it comes down to interoperability among the tools—the main concern for the proposed conceptual framework—even those with same version of XMI standard fail. In theory, the project developed in ArgoUML should be operable in Enterprise Architect easily due to the same version of XMI standard used in both tools (and vice versa). Similar arguments are also exposed on such tools as UModel and Enterprise Architect for the same reason. The most common error relates to incorrect syntax in XMI files, which clearly outlines the problems with proper implementation of standards from the side of vendors.

Microsoft Visual Studio could be used as logical sequel of previously examined tool. This tool does not support modeling activities, but support different programming languages for software development.

V. CONCLUSIONS

The paper discusses possibility of certification of MDA CASE-tools, to find out some standard in existing assortment of tools. Basic principles of MDA were examined to achieve this goal. The paper defines components of MDA, and relationships among them. During this research basic

principles of certification and defined properties of certification corresponding to MDA were found. Tool certification principles are important milestone to understand how certification could be performed and which result we want to obtain.

During this research 6 tools were examined within the correspondence to modeling capabilities in the framework of MDA: ArgoUML, Altova UModel, Sparx System Enterprise Architect, IBM Rational Enterprise Architect, My Eclipes Enterprise Workbench and MS Visual Studio 2010. The first five tools are pure modeling tools, and the last one is MS Visual studio positioned as development tool with modeling capabilities.

The main contribution of the paper is the stressed necessity for CASE tools certification. The paper shows possibility of CASE tools certification in the context of existing concepts. Possibility of tool verification in accordance to proposed framework is shown in example of 6 CASE tools analysis.

With such an abundance of various CASE-tools, both commercial and open-source, their certification is required. Due to the fact that MDA is now the most widely used approach in software development, it makes sense to certify the CASE-tools in the framework of MDA. It is important to identify the main criteria to determine conformance of CASE-tool to the standards of MDA, and in the same time these criteria should display conformance of CASE-tool to the tasks facing the developer. The entire certification process as a whole will only improve the quality of CASE-tools and provide the ability to track information about new features in the developed CASE-tools.

ACKNOWLEDGMENT

The research reflected in the paper is supported by Grant of Latvian Council of Science No. 09.1245 "Methods, models and tools for developing and governance of agile information systems" and by ERAF project "Evolution of RTU international collaboration, projects and capacity in science and technologies".

REFERENCES

[1] I.Jacobson, G.Booch, J.Rumbaugh, "The Unified Software Development Process", Addison-Wesley, 1999.

[2] Microsoft Solution Framework: http://technet.microsoft.com/en-us/library/bb497059.aspx

[3] Introduction to Scrum - an Agile Process: http://www.mountaingoatsoftware.com/topics/scrum

[4] Extreme Programming: A gentle introduction: http://www.extremeprogramming.org/

[5] MDA Guide, version 1.0.1: http://www.omg.org/docs/omg/03-06-01.pdf

[6] R.Bendraou, P.Desfray, M.Gervais, A.Muller, "MDA Tool Components: a proposal for packaging know-how in model driven development," Software and Systems Modeling, Vol.7, No.3,. Springer, Berlin 2008, pp.329-343.

[7] J. Krogstie, "Integrating enterprise and IS development using a model driven approach," Proc. 13th International Conference on Information Systems Development—Advances in Theory, Practice and Education, Springer. New York, 2005, pp.43-53.

[8] M.Guttman, J.Parodi, "Real-Life MDA: Solving Business Problems with Model Driven Architecture," Morgan Kaufmann, San Francisco, 2007.

[9] O.Nikiforova, V.Nikulsins, U.Sukovskis, "Integration of MDA Framework into the Model of Traditional Software Development," Frontiers in Artificial Intelligence and Applications, Vol.187, IOS Press. Amsterdam, 2009, pp.229-239.

[10] A.Brown, J.Conallen, D.Tropeano, "Models, Modeling, and Model Driven Development," S.Beydeda, M.Book, V.Gruhn, (eds.) Model-Driven Software Development, Springer, Berlin, 2005, pp.1-17.

[11] S.Mellor, K.Scott, A.Uhl, D.Weise, "MDA Distilled: Principles of Model-Driven Architecture," Addison-Wesley, San Francisco, 2004.

[12] A.Cernickins, O.Nikiforova, "An Approach to Classification of MDA Tools," The 49th Scientific Conference of Riga Technical University, Computer Science, Applied Computer Systems. Riga, 2008, pp 72-83.

[13] O.Nikiforova, A.Cernickins, N.Pavlova, "Discussing the Difference between Model Driven Architecture and Model Driven Development in the Context of Supporting Tools," The 4th International Conference on Software Engineering Advances (ICSEA), International Workshop on Enterprise Information Systems (ENTISY), IEEE Computer Society, 2009, pp.1-6.

[14] A.Uhl, "Model-Driven Development in the Enterprise," IEEE Software, Vol.25, IEEE Press, Washington, 2008, pp.46-49.

[15] O.Nikiforova, M.Kuzmina, N.Pavlova, "Formal Development of PIM in the Framework of MDA: Myth or Reality," The 46th Scientific Conference of Riga Technical University, Computer Science, Applied Computer Systems, Riga, 2006, pp. 42-53.

[16] P.Bunyakiati, A.Finkelstein, D.Rosenblum, "The Certification of Software Tools with respect to Software Standards," IEEE International Conference on Information Reuse and Integration, 2007.

[17] CORBA 2.3 Conformance statement template: http://www.opengroup.org/csq/csqdata/blanks/OB1.html

[18] H.Schäbe, "A Comparison of Different Software Certification Schemes": http://www.sipi61508.com/ciks/schabe1.pdf

[19] A.Cernickins, O.Nikiforova, "On Foundation for Certification of MDA Tools: Defining a Specification," RTU 50th International Scientific Conference, Computer Science, Applied Computer Systems, 2010, pp.45-51.

[20] A.Cernickins, "An analytical review of Model Driven Architecture (MDA) tools," Master's thesis. Riga, 2009.

[21] A.Cernickins, "Clarifying a Vision on Certification of MDA Tools," Scientific Papers, University of Latvia. Vol.757. Computer Science and Information Technologies, Latvia, Riga, 5.-7. July, 2010, pp 23-29.

[22] I.Sommerville, "Software Engineering" (8th edition), Addison-Wesley, Wokingham, 2006.

[23] MOF 2.0/XMI Mapping, Version 2.1.1: http://www.omg.org/spec/XMI/2.1.1/PDF

[24] Implementing Model Driven Architecture using Enterprise Architect. Mapping MDA Concepts to EA Features: http://www.sparxsystems.com/downloads/whitepapers/EA4MDA_White_Paper_Features.pdf

[25] XML Schema: http://www.w3.org/XML/Schema

[26] A.Cernickins, O.Nikiforova, K.Ozols, J.Sejans, "An Outline of Conceptual Framework for Certification of MDA Tools," Model-Driven Architecture and Modeling Theory-Driven Development, Greece, Athens, 22.-24. July, 2010. - pp 60-69.

 ISBN: 978-1-61208-165-6