An Outline of Conceptual Framework for Certification of MDA Tools

Antons Cernickins[1], Oksana Nikiforova[1], Kristaps Ozols[2] and Janis Sejans[1]

[1]Riga Technical University, Institute of Applied Computer Systems
Kalku Street 1, Riga, LV-1658, Latvia

[2]ErpPro (Enterprise Resource Planning Professionals) Ltd.
K. Ulmana gatve 86F, Riga, Latvia

Abstract. Nowadays, the certification of applications and software systems is considered as a task of growing importance. In fact, software development life cycle itself is also considered as a matter of certification concern, especially with the emergence of new approaches. The proposal of Model Driven Architecture (MDA) shifted the software development towards modeling. However, MDA as such still lacks the appropriate implementation framework. The original article outlines a conceptual framework, which may be used as a foundation for certification of MDA tools.

1 Introduction

Since the introduction of OMG's Model Driven Architecture (MDA) and the principles of model-driven development in 2001, the way the software is developed nowadays have not changed dramatically as various experts predicted (e.g., the forecasts proclaimed at the European Conference of MDA in 2006). Despite the improvements offered to the software development community, which includes the attempt to raise the level of abstraction, as well as the attempt to increase the level of reuse [1], the actual impact of MDA on the process of software development remained the same [2]. [3] expects MDA tools, standards, and best practices evolve only over the next 10–15 years before MDA ultimately comes to be perceived as a "commodity."

In fact, the development of next-generation integrated development environments (IDEs) and CASE tools, which would utilize the latest trends in software development, as well as provide a platform for further improvements, is a matter of high interest from the industry (e.g., Eclipse platform [4]). As the proposal of MDA approach shifted the process of software development towards modeling, it also forced the next-generation tools to incorporate modeling, model transformation, and code generation capabilities. However, while the actual implementation of MDA as an approach still lacks the appropriate implementation framework, this results in incompliance among software development tools. Due to incompliance between software tools, a model of a software system created in one environment cannot be ade-

Cernickins A., Nikiforova O., Ozols K. and Sejans J. (2010).
An Outline of Conceptual Framework for Certification of MDA Tools.
In *Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development*, pages 60-69
DOI: 10.5220/0003044400600069

quately used in another environment. Thus, in order to approve that any specific product meets specific requirements or conforms to particular standards, a certification procedure hold by an independent third party is carried out.

Possible benefit from the certification procedure is that it offers more certainty about or confidence in developed software systems. It also helps in software sales, giving more confidence for prospective clients. Furthermore, certification is valuable, because the developer can be sure that the developed system will operate in predictable way as specified in the standards.

In case of Model Driven Architecture, the use of certification procedure would help in classifying various tools (tool suites) in terms of functionality (i.e., help customers to choose the right tools for their specific problem domain), fix compatibility issues among tools (i.e., provide interoperability for tools from various tool vendors), as well as provide additional pointers in understanding the whole MDA-oriented development process. The reason why the acceptance of MDA in software development community is still weak enough is because of overall complexity of the approach to the end users (too much problems with tools, no strict guides on development process, etc.). [5] states that OMG is in the early process of defining a tool certification process, but no official document exists on this topic at the time. When it comes to the analysis of the comparable experience in other areas, e.g., certification is useful in proofing compilers (e.g., compiler validation process and policy for ADA language [6]).

The original article contains a research on Model Driven Architecture, reviewing the possibility of MDA tool certification similarly to the standardization processes in other areas of activity—i.e., to assess the compliance with standards. The goal of the article is to outline a conceptual framework, which may be used as a foundation for certification of MDA tools.

The article is organized as follows. Section 2 overviews the background, as well as provides a brief review of related work. Section 3 outlines the vision on conceptual framework to be used for certification of MDA tools. Section 4 outlines the state of the art on how certain MDA tools correspond with the proposed framework. Section 5 concludes the article and provides pointers to further work.

2 Background and Related Works

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 [7]. In fact, the conformance statement for CORBA provided by The Open Group [8] is done in particular way. The compliance itself is nothing else but the satisfaction of software implementation to the standard specification [7]. [7] 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 foremost task is to define the object of certification [9]. According to [9], 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).

[9] 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.

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 [10]).

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, [11] is considered. [11] reviews the MDA approach within the variety of the CASE tools, which are proposed as supporting for MDA activities. The goal of the following research is to investigate the variety of the CASE tools, which are proposed as "MDA compliant," in order to classify them in accordance with the previously defined MDA tool specification. The provided specification of MDA tools consists of seven categories, specified in a hierarchical way flattened in the table (categories are divided into subcategories, subcategories—into groups, and groups—into single entries, accordingly) [11]:

- 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;
- Functional capabilities—the functional capabilities of a tool in such fields as environment, modeling, implementation, testing, documenting, project management, configuration management;

  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;

- Usability—usage efforts and individual assessments of such usage, including user interface, licensing and localization options, ease of use, quality of documentation etc.;
- 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;
- Maintainability—efforts needed to make specified modifications;
- Portability—ability of a tool to be transferred to another environment.

3 Concept of Certification Framework

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 [10], an additional specification-based assessment of these tools is considered (discussed in [10]).

In short, the following four layers are used to describe the MDA-oriented software development life cycle [1], [10], [12], [13]:

- Computation Independent Model (CIM)—represents the high-level specification of what the system is expected to do (i.e., describes the domain and requirements of the system). It might consist of a model from the informational viewpoint, which captures information about the data of a system;
- Platform Independent Model (PIM)—specifies the functionality of a system. It might consist of a model from the informational viewpoint, which captures information about the data of a system, and a model from the computational viewpoint, which captures information about the processing of a system;
- Platform Specific Model (PSM)—specifies the implementation of system's functionality on specific platform. It might consist of a model from the informational viewpoint, which captures information about the data of a system, and a model from the computational viewpoint, which captures information about the processing of a system, based on a specific platform;
- Implementation Specific Model (ISM) or source code—describes the implementation of a system in source code of specific platform.

However, the only layers to be specified and promoted by OMG (i.e., described in details) are PIM and PSM [1]. 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 layers as individual blocks, each of them having their own input and output (Fig. 1). The origin of this idea has come from black box testing [14]: 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.

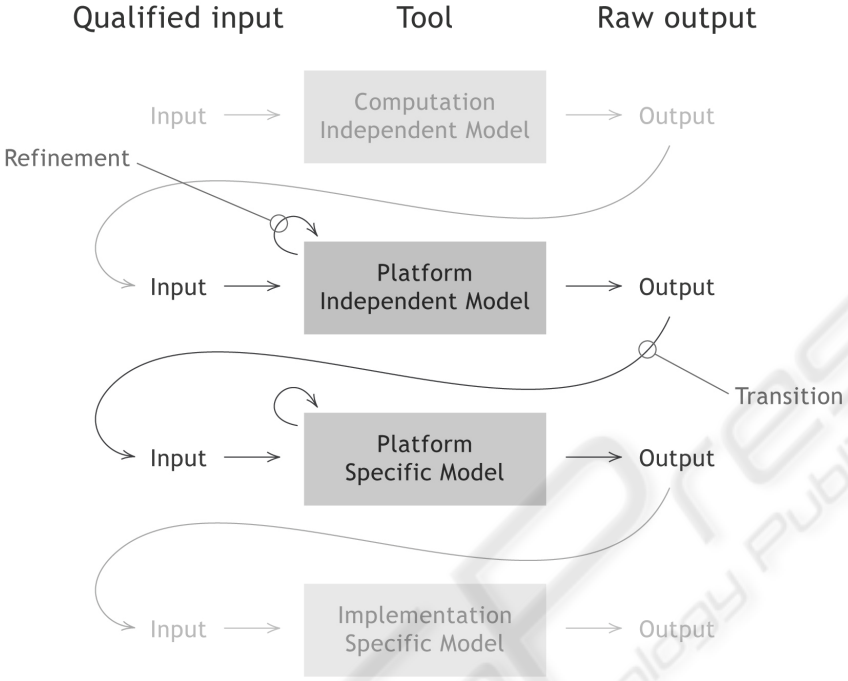

Fig. 1. Graphical representation of the conceptual framework.

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 (Fig. 1). 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 [15]. 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) [1] [12], 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 [5]. 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 [16], 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 [15]. 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 [15].

After the XML Schema of XMI standard is created, the developed tool creates a document data model, which consists of [16]:

- 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 [15] 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.

4 Correspondence from Tools

In order to examine the declared correspondence level of tools, a scope of correspondence should be defined first. Considering the information from previous Section, 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 2 should also be considered.

Based on [17], the following tools have been selected:

- ArgoUML 0.28;
- Altova UModel 2009;
- Sparx Systems Enterprise Architect 7.5.843;
- IBM Rational Rose Enterprise 7.0.0;
- MyEclipse Enterprise Workbench 7.1.1.

[17] 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. 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 [17]).

Table 1. Declared features of corresponding UML tools (based on [17]).

	ArgoUML	Altova UModel	Sparx Systems Enterprise Architect	IBM Rational Rose Enterprise	MyEclipse Enterprise Workbench
Common features					
UML	1.4	2.2	1.3, 1.4, 2.0, 2.1	1.4	2.1
UML Profiles	▫	▫	▫		▫
MOF/XMI	1.1, 1.2	2.1	1.1, 1.2, 2.1		1.0
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			▫		▫

Table 1. Follow-up.

	ArgoUML	Altova UModel	Sparx Systems Enterprise Architect	IBM Rational Rose Enterprise	MyEclipse Enterprise Workbench
Source code generation capabilities					
CORBA IDL			▫	▫	
Java	▫	▫	▫	▫	▫
C++	▫	▫	▫	▫	
C#	▫	▫	▫	▫	
VB.NET		▫	▫	▫	
PHP	▫		▫		
Other			Ada, Python, ActionScript		
Reverse engineering capabilities					
CORBA IDL	▫		▫	▫	
Java	▫	▫	▫	▫	▫
C++		▫	▫	▫	
C#		▫	▫	▫	
VB.NET		▫	▫	▫	
PHP			▫		
Other			C, Python, Visual Basic, ActionScript		

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.

5 Conclusion and Future Work

The original article outlined a conceptual framework, which may be used as a foundation for certification of MDA tools. In fact, this framework provides a basis for the development of XMI validation utility. The practical implementation of XMI validation utility is based on the same principles used in HTML, XML, and other markup languages.

In order to develop the XMI validation utility, the XMI standard itself should be examined in details. In fact, the most current iteration on examining the XMI standard showed that it should be defined stricter: the case of preserving tool-specific information may be treated as the source of problems regarding tool interoperability. Meanwhile, the design principles of XML Schemas and documents provided in the standard are detailed enough for the actual implementation in XMI validation utility. However, a more comprehensive study on the use of XML Schemas should be also considered. All in all the XMI validation utility itself is yet another component in software certification scheme, which should be used in a combination with the others.

In extending this work, the next step is the development of several reference models to cover the whole MDA-oriented development life cycle. These will provide a step-by-step guide for the developers. As for modeling notation, the use of UML is considered. According to [18], a minimal set of UML diagrams is already defined. In turn, a recommended and complete set of UML diagrams should be proposed, as well as mutually compared.

A more comprehensive study on MDA features [11] may result in defining the levels of MDA maturity (just like maturity levels in CMMI [19]). The four-layer architecture of MDA could be selected as a basis for this: e.g., PSM together with ISM layers could be considered at the lowest levels (as there are dozens of tools with source code generation capabilities available on today's market), while features related to the development of PIM and CIM could be considered at the higher ones.

Acknowledgements

The research reflected in the paper partly is supported by Grant of Latvian Council of Science No. 09.1269 "Methods and Models Based on Distributed Artificial Intelligence and Web Technologies for Development of Intelligent Applied Software and Computer System Architecture." The research reflected in the paper partly is supported by Riga Technical University in cooperation with Microsoft under the project No. FLPP-2010/20 "Research of Principles of Model Driven Architecture in Software Development Tools."

References

1. Mellor, S., Scott, K., Uhl, A., Weise, D.: MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley, San Francisco (2004)

2. Nikiforova, O., Cernickins, A., Sejans, J.: On Automatic Transition from Initial System Models into Software Components. In: The 5th International Conference on Software Engineering Advances (ICSEA), IEEE Computer Society (submitted for publication) (2010)
3. Guttman, M., Parodi, J.: Real-Life MDA: Solving Business Problems with Model Driven Architecture. Morgan Kaufmann Publishers, San Francisco (2007)
4. Eclipse Platform, http://www.eclipse.org/platform/
5. Implementing Model Driven Architecture using Enterprise Architect. Mapping MDA Concepts to EA Features, http://www.sparxsystems.com/downloads/whitepapers/EA4MDA_White_Paper_Features.pdf
6. Ada 95 Compiler Validation Process and Policy, http://www.sigada.org/ada_95/validation/process.html
7. Bunyakiati, P., Finkelstein, A., and Rosenblum, D.: The Certification of Software Tools with respect to Software Standards. IEEE International Conference on Information Reuse and Integration (2007)
8. CORBA 2.3 Conformance statement template, http://www.opengroup.org/csq/csqdata/blanks/OB1.html
9. Schäbe, H.: A Comparison of Different Software Certification Schemes, In: Guiding Principles to the Implementation of IEC-61508 (2004) http://www.sipi61508.com/ciks/schabe1.pdf
10. Cernickins, A., Nikiforova, O.: On Foundation for Certification of MDA Tools: Defining a Specification. RTU 50th International Scientific Conference, Computer Science, Applied Computer Systems (in press) (2009)
11. Cernickins, A.: An analytical review of Model Driven Architecture (MDA) tools. Master's thesis. Riga (2009) / Čerņičkins, A.: Modelvadāmās arhitektūras rīku analītisks apskats. Maģistra darbs. Rīga (2009)
12. MDA Guide 1.0.1. Object Management Group, http://www.omg.org/docs/omg/03-06-01.pdf
13. Alhir, S.: Understanding the Model Driven Architecture, In: Methods & Tools 2003. Martinig & Associates (2003) 17–24
14. Sommerville, I.: Software Engineering (8th edition). Addison-Wesley, Wokingham, (2006)
15. MOF 2.0/XMI Mapping, Version 2.1.1, http://www.omg.org/spec/XMI/2.1.1/PDF
16. XML Schema, http://www.w3.org/XML/Schema
17. Ozols, K.: An application analysis of UML tools for generation of source code. Master's thesis. Riga (2009) / Ozols, K.: UML koda ģenerēšanas rīku lietošanas analīze programmu sagatavju izstrādei. Maģistra darbs. Rīga (2009)
18. Nikiforova, O.: General Framework for Object-Oriented Software Development Process. Scientific Proceedings of Riga Technical University. Series–Computer Science, Applied Computer Systems, 13 vol., RTU Riga (2002) 132–144
19. CMMI | Overview, http://www.sei.cmu.edu/cmmi/