SCIENTIFIC PAPERS, UNIVERSITY OF LATVIA, 2010. Vol. 757
COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES 23–29 P.

Clarifying a vision on certification of MDA tools

Antons Cernickins
Riga Technical University, Institute of Applied Computer Systems,
Kalku Street 1, Riga, LV-1658, Latvia
antons.cernickins@rtu.lv

Abstract. As software systems tend to play a major role in many areas of activity, the certification of these systems or even small software applications becomes a task of growing importance [1]. Moreover, a software development life cycle itself is also considered as a matter of certification concern [2], especially with the emergence of new approaches. However, the main concern for a new approach is its acceptance from the side of the industry. In particular, the proposal of Model Driven Architecture shifted the process of software development towards modeling [2]. The actual implementation of MDA as an approach still lacks the appropriate implementation framework, resulting in incompliance among software development tools. This article proposes a vision on conceptual framework to be used as a foundation for certification of MDA tools. In particular, this includes the development of certification schemes to assess the compliance of the tools with incorporated standards from various perspectives.

Keywords: Model Driven Architecture, software development life cycle, certification, software development tools

1 Introduction

The continued integration of IT into various areas of activity facilitates the increase in dependence on the reliability, availability, and integrity of software systems [3]. Due to the overall complexity of modern hardware, software, and the ways of communication, the development of quality systems has become both a major scientific and engineering challenge [3].

However, the verification of developed system is also a matter of high importance, especially in safety-critical software systems (e.g., in domain of civil aviation), where system failure must be avoided [3]. Verification process done by a developer cannot guarantee the appropriate level of quality for such systems; usually, a third party is also incorporated into verification. Furthermore, the third party should be independent (i.e., should not represent the supplier or the acquirer side) to produce an objective and complete judgment of the system. 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 benefits from certification procedure are the following: it offers more certainty about or confidence in developed software systems [3]. It helps in software sales, giving more confidence for prospective clients. Certification is also valuable, because the developer of the software can be sure that their system will operate in predictable way as specified in the standards [3].

Software development standards provide complex methods and approaches for defining, specifying, visualizing, and documenting the artifacts of software systems [4]. Furthermore, software tools have been developed to support these artifacts. Since the most current trend in software development is shifted towards modeling [2], the development of software artifacts is being driven by models. In order to support the model-driven trend, as well as to make sure, the process of modeling is done equally (i.e., same rules, notation, etc.), a branch of new standards for software modeling and architecture specifications has been proposed [5]. Model Driven Architecture (MDA), introduced by OMG in 2001, covered a wide range of these specifications [5].

However, the implementation of MDA approach is still challenging enough: there is a lack of information on how the support for MDA approach should be provided [2]. This results in incompliance with proposed standards, because each tool vendor implements these standards in his own way. Due to incompliance between software tools, a model of a software system created in one environment cannot be adequately used in another environment.

The original article contains a research on Model Driven Architecture, reviewing the MDA-oriented software development life cycle in the context of developing a quantifiable certification scheme to assess the compliance of MDA support tools with OMG standards. The goal of the article is to propose a vision on conceptual framework to be used as a foundation for certification of MDA tools. This article represents the most current state of the art in author's doctoral studies.

In turn, the hypothesis intended to be proven in doctoral research is associated with the possibility to develop a quantifiable certification scheme along with a set of rules and guidelines on how to assess the compliance among MDA support tools. The practical value of the research is the development of the certification scheme itself, including the appropriate framework, rules, and guidelines.

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. Finally, Section 4 concludes the article and provides pointers to further work.

2 Background and Related Work

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 [4]. In particular, the conformance statement for CORBA provided by The Open Group [6] is done this way. In fact, the compliance itself is nothing else but the satisfaction of software implementation to the standard specification [4]. [4] comes with a 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 [1]. According to [1], 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).

[1] 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 [2]).

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, [7] is considered. [7] 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) [7]:

- 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 Vision

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 [2], an additional specification-based assessment of these tools is considered (discussed in [2]).

In short, the following four layers are used to describe the MDA-oriented software development life cycle [2], [5], [8], [9]:

- 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 [8]. 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 [10]: whereas software system is considered as a black box, the only thing to be analyzed is the output produced by specific input. Therefore, tester 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.

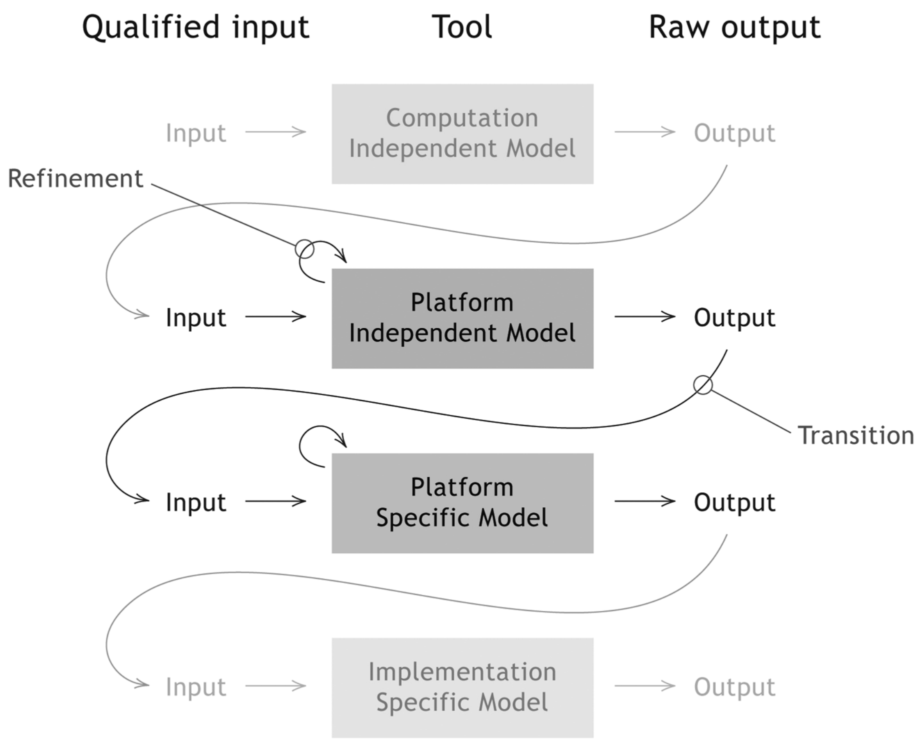

Fig. 1. Graphical representation of the conceptual framework

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 on 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 [11]. 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) [5], [8], the use of XMI for assessment of software development tools seems to be the only valuable option.

3.1 Implementation

In order to implement the conceptual framework, the development of a software tool is considered. The main concern of this tool is proper parsing and analysis of the XMI documents. As far as the technical standard of XMI like any other standard continue to evolve (the most current release is 2.1.1 [11]), a modular software design model is considered in the developed tool.

The specification of XMI standard as such is used to create the XML Schema of XMI standard [12], 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 [11]. 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 [11].

After the XML Schema of XMI standard is created, the developed tool creates a document data model, which consists of [12]:

- 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 [11] 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.

After the document has been successfully parsed and analyzed, the developed tool outlines elements that are incorrect from the specification viewpoint, providing a “clean representation” of XMI document.

4 Conclusion and Future Work

This article outlined a vision on conceptual framework to be used as a foundation for certification of MDA tools. The ideas stated in this article are intended to be proven in doctoral research, with an aim to develop a quantifiable certification scheme to assess the compliance of MDA support tools with OMG standards.

In prospective, the conceptual framework described in this article will be used as a basis for XMI validation utility. However, this 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 [13], 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.

References

1. Schäbe, H.: A Comparison of Different Software Certification Schemes, *http://www.sipi61508.com/ciks/schabe1.pdf*
2. Cernickins, A., Nikiforova, O.: On Foundation for Certification of MDA Tools: Defining a Specification. RTU 50th International Scientific Conference, Computer Science, Applied Computer Systems (2009)
3. Heck, P., Klabbers, M., van Eekelen, M.: A software product certification model. In: Software Quality Journal, vol. 18 (1), pp. 37–55. Springer Netherlands (2009)

4. 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)
5. MDA Guide 1.0.1. Object Management Group, *http://www.omg.org/docs/omg/03-06-01.pdf*
6. CORBA 2.3 Conformance statement template, *http://www.opengroup.org/csq/csqdata/blanks/OB1.html*
7. 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)
8. Mellor, S., Scott, K., Uhl, A., Weise, D.: MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley, San Francisco (2004)
9. Alhir, S.: Understanding the Model Driven Architecture, In: Methods & Tools 2003, pp.17–24. Martinig & Associates (2003)
10. Sommerville, I.: Software Engineering (8th edition). Addison-Wesley, Wokingham, (2006)
11. MOF 2.0/XMI Mapping, Version 2.1.1, *http://www.omg.org/spec/XMI/2.1.1/PDF*
12. XML Schema, *http://www.w3.org/XML/Schema*
13. Nikiforova, O.: Object-oriented System Analysis. Drukatava, Riga (2007) / Ņikoforova, O.: Objektorientētā sistēmanalīze. Drukātava, Rīga (2007)