Evaluating Quality in Model-Driven Engineering

Parastoo Mohagheghi, Jan Aagedal
SINTEF ICT- P .O. Box 124 Blindern, N-0314 Oslo, Norway
{parastoo.mohagheghi, jan.aagedal}@sintef.no

Abstract

In Model-Driven Engineering (MDE), models are the prime artifacts, and developing high-quality systems depends on developing high-quality models and performing transformations that preserve quality or even improve it. This paper presents quality goals in MDE and states that the quality of models is affected by the quality of modeling languages, tools, modeling processes, the knowledge and experience of modelers, and the quality assurance techniques applied. The paper further presents related work on these factors and identifies pertinent research challenges. Some quality goals such as well-formedness and precision are especially important in MDE. Research on quality in MDE can promote adoption of MDE for complex system engineering.

1. Introduction

A model is a representation of a system that hides some details to assist focusing on some aspects; for example structure of a system or its processes. Most Object-Oriented (OO) methods advocate the use of several models to describe the various aspects of the system under consideration, while there are also methods that combine several aspects in one model [10]. Generally a model need not be diagrammatic; it can combine graphical and textual information or be only textual.

Models are widely used in software development, but in current practice they are mainly used for communication between stakeholders, analyzing the problem, and documenting the system, while detailed design is code-centric. Model-Driven Engineering (MDE) is the term used for development processes that are model-centric as opposed to code-centric. In MDE, models are the prime artifacts and they may exist on multiple levels of abstractions and undergo transformations to other models and/or code.

A newly started project in SINTEF ICT has the goal to develop a framework for evaluating quality in MDE. MDE methods and tools are getting mature with time, and technology and tool providers have gained experience that is necessary to extend MDE from forward engineering in small-scale projects to round-trip engineering in complex systems, which also need model reuse, traceability, reverse engineering and composition of models. Thus quality issues also change scale and become more important. Research has already been done on various aspects of quality in modeling. This paper has weaved the threads of earlier research into an initial model showing their relations to each other and to the quality of models. It also describes research challenges and our plans for future work.

The remainder of the paper is organized as follows. Section 2 presents motivation of the work and the model that relates quality factors to each other. Section 3 presents some related work from our literature search, and Section 4 discusses specific quality challenges in MDE. Section 5 discusses future work and concludes the paper.

2. A model of quality in modeling

Models may be developed for the purpose of communication, documentation, analysis and design, test case generation and/or code generation. In MDE, models are refined progressively and transformed to new models or code. Our initial hypothesis is that there is a relation between the quality of the final software product and the quality of the models used to generate it, such as their consistency and completeness. But the quality of models in turn depends on:

- The quality of modeling language(s) used, such as their appropriateness for the domain and complexity;
- The quality of tools used for modeling and transformations, such as compliance with the

modeling languages and capability of combining information;

- The knowledge of developers of the problem in hand and their experience of modeling languages and tools in use;
- The quality of the modeling processes used;
- The quality assurance techniques applied to discover faults or weaknesses.

Figure 1 shows the above factors and their relations. Developers use the available modeling languages, tools and processes and develop models based on their knowledge of the problem and their experience. Some quality characteristics required from each of the factors are shown as well.

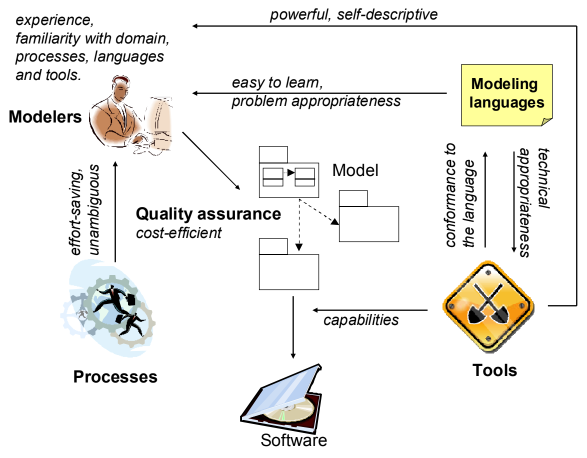

experience, familiarity with domain, processes, languages and tools.
powerful, self-descriptive
easy to learn, problem appropriateness
Modeling languages
Modelers
effort-saving, unambiguous
Model
Quality assurance
cost-efficient
conformance to the language
technical appropriateness
capabilities
Processes
Tools
Software

Figure 1. Factors affecting model quality

What quality characteristics that are important depend also on the purpose of modeling. For example, if the purpose of modeling is communication between stakeholders and high-level system documentation, comprehensibility is more important than completeness.

3. Related work on quality in modeling

3.1. Quality of models

The EmpAnADa project (Empirical Analysis of Architecture and Design Quality) aims to develop techniques to improve the quality of UML models [3]. In a paper by project members (Chaudron and Lange), they describe their quality model relating primary use of models (either development or maintenance) to purposes and the required quality characteristics for each purpose [7]. For example, complexity is defined as the effort required for understanding a model and is important for communication, comprehension and modification. Figure 2 shows their quality model.

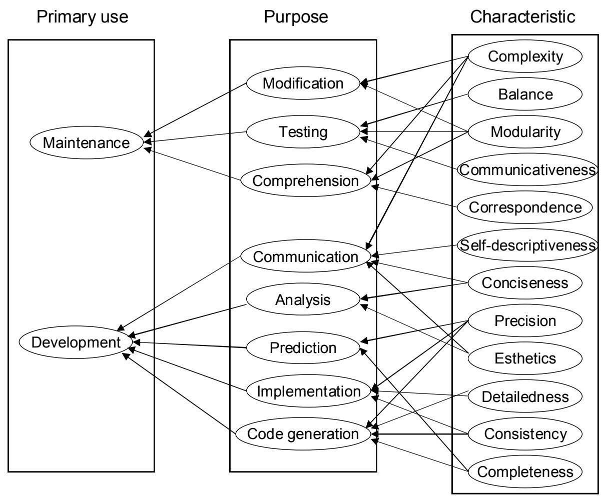


Figure 2. Quality model in [7]

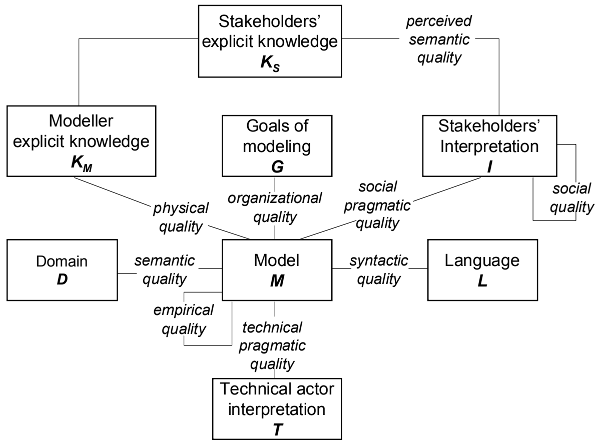
After selecting quality characteristics, a set of metrics are identified to measure the quality characteristics. For complexity, the proposed metrics are dynamicity (the complexity of a class internal behavior based on message calls and state transitions), DIT (the depth of inheritance tree), cohesion (which part of a class are needed to perform a single task), NCU (the number of classes per use case), and NUC (the number of use cases per class). Some of the metrics are traditional OO metrics such as coupling and cohesion, while a few are model-specific, such as the number of crossing lines in a diagram. One advantage of the approach is that it is metrics-based, but the metrics are mainly on the detailed design level and do not cover all the purposes of modeling.

One problem with the model is that relations between metrics and quality characteristics are often many-to-many. For example, all the identified metrics are important for "balance", which is defined as the extent that all parts of a system are described at an equal degree of all other model characteristics. Alternatively, should "balance" be defined as a characteristic composed of completeness, conciseness, modularity and self-descriptiveness? Another problem is that relations are often based on judgment. For example, ISO and IEEE have different hierarchies of quality attributes.

The above quality model also includes rules that can be checked by appropriate tools. An example of such a rule is that "an abstract class should have a subclass". Rules are also proposed by Berenbach, together with some metrics that can be automatically collected by a tool called DesignAdvisor [1]. Examples of metrics are "Unused class" or "Incorrectly defined interface". Berenbach means that metrics have been proposed for quite some time but what is "good" or "bad" is not well researched.

The quality of UML 2.0 models is the subject of a book published in 2005 [16]. Building on earlier work, Unhelkar defines quality in three dimensions: *syntactical correctness* (adhering to the rules of UML 2.0), *semantic correctness* (representing intended meanings) *and consistency*, and *aesthetics* (symmetric, complete and pleasant models). The author provides checklists for quality in the above dimensions for each UML diagram.

Finally, Krogstie et al. have developed a framework for quality of models and modeling languages [4, 5, and 6]. Figure 3 shows a view of their framework.



Stakeholders' explicit knowledge K_S
perceived semantic quality
Modeller explicit knowledge K_M
Goals of modeling G
Stakeholders' Interpretation I
physical quality
organizational quality
social pragmatic quality
social quality
Domain D
semantic quality
Model M
syntactic quality
Language L
empirical quality
technical pragmatic quality
Technical actor interpretation T

Figure 3. Krogstie et al.'s quality framework

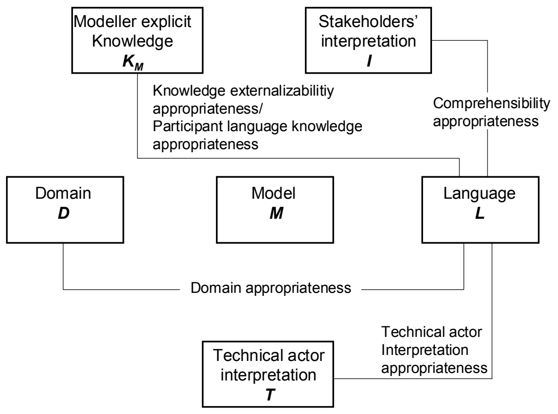
In Figure 3, quality goals are defined as relations between blocks. For example, *syntactic quality* has the goal that all statements in the model are according to the syntax of the modeling language, *empirical quality* comprises comprehensibility matters such as layout and readability, *organizational quality* covers that the model fulfills the goal of modeling, and *social quality* is the agreement among stakeholders' interpretations. From the above quality goals, some can be objectively measured such as syntactic quality and social quality (if a modeling language has a formal semantics), while others such as domain or modelers' knowledge are not measurable. We have not found an empirical evaluation of this framework on models.

3.2. Quality of modeling languages

Like programming languages, modeling languages have a concrete syntax, abstract syntax and semantics. Therefore, quality requirements for language design in general also apply for modeling languages. Previously, modeling languages tended to focus on specification whilst programming languages emphasized implementation [2]. This distinction is blurred now since models can also be executable.

Krogstie et al. have extended their quality framework presented in Figure 3 to identify characteristics of modeling languages, as depicted in Figure 4. These are:

- *Domain appropriateness*: Ideally, the conceptual basis must be powerful enough to express anything in the domain, and not more.
- *Participant language knowledge appropriateness*: It is best to base a language on experience with languages previously used.
- *Knowledge externalizability appropriateness*: There should be no statements in the explicit knowledge of the participant that cannot be expressed in the language. This quality is highly dependent on the knowledge of participants.
- *Comprehensibility appropriateness:* There is a list of characteristics for this aspect such as that the number of phenomena should be reasonable, and they should be distinguishable from each other.
- *Technical actor interpretation appropriateness*: For the technical actors (modeling tools), it is especially important that the language lends itself to automatic reasoning. This requires formality; i.e., both formal syntax and semantics.



Modeller explicit Knowledge K_M
Stakeholders' interpretation I
Knowledge externalizabilitiy appropriateness/ Participant language knowledge appropriateness
Comprehensibility appropriateness
Domain D
Model M
Language L
Domain appropriateness
Technical actor Interpretation appropriateness
Technical actor interpretation T

Figure 4. The language quality framework defined by Krogstie et al.

Krogstie et al. have evaluated UML 1.4 using the above framework with the conclusion that the language is difficult to comprehend, the concepts involved in different phases are fundamentally different and it has low formality, while its strength is in creation of design models for OO systems [5].

A model conforms to a meta-model that defines constructs and rules to build models. An essential requirement of a meta-modeling language is therefore its ability to concisely capture all aspects of a modeling language, including its syntax and semantics [2]. Rossi et al. write that that there exists an intrinsic dependency between the metamodel and the learnability of a language since a language's meta-model serves as an indication of its functional complexity [12]. A metamodel's conceptual complexity should lead to greater expressive power, and thus smaller models in size. For example, modeling languages developed for a specific domain have more expressive power and are closer to the experts' knowledge of the domain than general-purpose modeling languages, but may be more complex to learn for a novice.

Rossi et al. therefore propose measuring the complexity of a modeling language by looking at its meta-model. A modeling language is viewed as a set of techniques such as class diagrams and state machine diagrams. For each technique, metrics of the count of object types, relationships and property types in the metamodel are calculated. The conceptual complexity of a technique is a sum vector of the above metrics, and the complexity of a modeling language is a sum vector of complexity of all its techniques. The techniques are assumed not to be interrelated. Rossi et al. have compared several OO methods using the above metrics and concluded that OO methods get more complex with time. Another evaluation done by Siau and Cao (presented in [5]) based on the above metrics concluded that UML is 2 to 11 times more complex than other OO methods.

As an example, we can mention that the OMG Systems Modeling Language (OMG SysML™) defines correctness, precision, conciseness, consistency and understandability as its quality goals [9], without further discussion on what these terms mean and how they are achieved.

3.3. Quality of tools

Few have evaluated modeling tools and their relation to modeling languages and the quality of models. We present here the few studies found so far.

Østbø has compared Select Enterprise Edition version 6.0.53 with Rational Rose 2000 Enterprise Edition for a set of requirements in an industrial case such as documentation appropriateness and multi-user support [17]. Purchase et al. discuss the problem of differences in UML graphical notations and have performed an experiment to compare a few notations for understanding diagrams and finding errors in them [11]. Two examples of notations and the related results are presented in Figure 5.

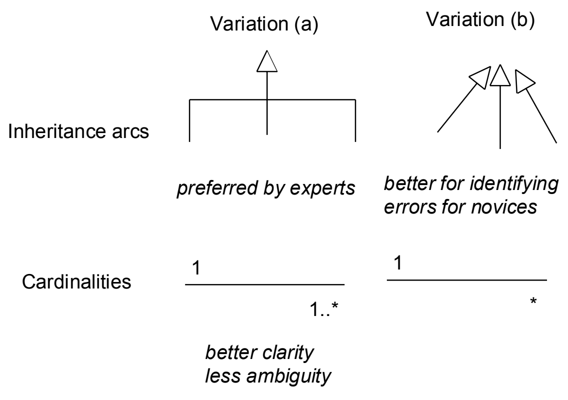


Figure 5. Some of the notational variations in the experiment of [11]

The importance of the experiment is that it confirms the role of notations for comprehension and error avoidance, and thus tool developers should make choices between equivalent notations that improve the quality of models.

Unhelkar provides a checklist for evaluating UML-based case tools, such as "Compliance with UML" and "Ability to suit a process" [16], but the checklist is not used to evaluate the tools presented in the book. Such a list may be developed for MDE tools related to the quality goals presented in Section 4.

3.4. Quality of modeling processes

General quality guidelines should be adapted to the domain, the task in hand, the organization practices and modeling goals, and the tools they plan to use. Berenbach writes that in his study, lack of process contributed to a large number of errors and the diagrams designed by the modelers were not uniform [1].

Siau and Tian propose using a method that evaluates goals and operations involved in performing a task (such as drawing a use case diagram) and measuring the time it takes to perform these [13]. The measures will give some idea of the complexity and usability of modeling techniques and processes. The MODELWARE project (MODELling solution for softWARE systems, IST-511731, ended in 2006) has done work on defining modeling maturity level of organizations based on the role of modeling in their software development process; for example from manual ad-hoc to full model-based. This work builds on some earlier work by IBM and others as described in [8]. The highest maturity level needs establishing a complete model-centric development process. The project has also identified some engineering and business metrics to be used in the process framework, such as "Code generation ratio" and "Model completeness". The proposed process framework may be basis for future work.

3.5. Quality assurance techniques

Without quality assurance, models can become complex, incomplete and inconsistent with each other, and difficult to maintain. Special inspection techniques are earlier proposed to detect faults in UML models, such as the Object-Oriented Reading Techniques (OORT) [15]. Further literature search may reveal other quality assurance techniques specifically applicable on models.

4. Quality aspects in MDE

In the previous section, we discussed work on the quality of models and factors that affect this. In this section, we discuss some special quality requirements for MDE.

Solheim and Neple write that MDE motivates system development with the following characteristics [14]:

- Many activities have models as input, output, or both.
- Several of these activities are model transformations (while others are model analysis, model verification, etc.), applied by tools.
- During a transformation, output models are supplied with domain-related information not present in the input model. An example is the *platform* concept. Models should therefore be complete but not include unnecessary or redundant information.

Solheim and Neple emphasize two quality criteria important in MDE:

- *Transformability:* Models must have the ability to be transformed to other models of greater detail, and to executable pieces of code. Transformability is decomposed into completeness (correct according to the domain), relevance (containing no extra elements), precision for transformation, and well-formedness or compliance to the model's metamodel.
- *Modifiability:* Changes made to the requirements are rendered correctly in the models and reflected in the code. Modifiability is decomposed into traceability of model elements, and well-designedness or not being too complex.

We add that models will be incrementally developed and composed with other models, which means that models should be consistent with each other and maintainable. Challenges of complex system engineering should also be solved in MDE. Such challenges include reuse of models (which needs decomposition and composition), reverse engineering from code to models, combining several modeling languages (either general purpose like UML or domain-specific), and version control of models. To solve these challenges, models should be modular, easily extensible and exchangeable between tools. Means should therefore be identified for these quality goals in tools, languages and modeling processes. For example, modeling languages should be extensible with profiles and packages, and tools should support export and import of sub-models.

MDE relies heavily on tools for all the activities. Tool reliance and automatic transformation can be an advantage since many quality considerations can be integrated into tools, like keeping track of changes and following the syntax or metamodel.

5. Conclusion and future work

Based on Figure 1 and discussions in Sections 3 and 4, we conclude that quality in modeling and specially in MDE is composed of several aspects that cover technical factors (such as complexity of languages and their metamodels, transformability of models and capabilities of tools), psychological factors (such as learnability, familiarity with the language and ease of interpretation), Human-Computer Interaction factors (such as usability and aesthetic aspects) and organizational factors (the domain and goals of modeling). For MDE, we should also focus on metamodeling and modeling languages, transformations, model-centric processes, tools used in different stages of software development, and quality assurance techniques adapted to MDE.

We have started a project on quality in MDE to answer questions like whether a model is complete or suitable for automation, or whether a modeling technique is appropriate for a certain purpose of modeling. To answer these questions, we plan to identify goals and quality requirements in different stages of the MDE, identify questions to evaluate each goal, and identify metrics to answer the questions. Earlier checklists and proposed metrics will be examined in this process. Quality goals, metrics and guidelines will give input to quality assurance techniques. Such goals and metrics must be assessed in real projects or experiments. It is impossible to cover all the identified factors, or identify metrics that are applicable to all modeling languages or techniques. Therefore, we should focus on some quality factors and possibly for specific domains or purposes of modeling, depending on the results of our literature review and the projects we are involved in.

SINTEF ICT has broad engagement in MDE and experience from several international projects with focus on modeling such as the MODELWARE and MODELPLEX projects (MODELing solution for comPLEX software systems, IST-34081, started in September 2006 for three years, http://www.modelplex.org/). We plan to use examples of models, languages and tools that are developed in these projects in our empirical work.

6. References

[1] B. Berenbach, "Evaluation of Large, Complex UML Analysis and Design Models", *Proc. 26th Int'l Conference on Software Engineering (ICSE'04)*, pp. 232- 241, 2004.

[2] Clark, T., A. Evans, P. Sammut, and J. Willans, *Applied Metamodelling- A Foundation for Language Driven Development*, Version 0.1, Accessible at http://www. xactium.com, 2004.

[3] EmpAnADa project, http://www.win.tue.nl/empanada/, last visited on 14 January 2007.

[4] J. Krogstie, O.I. Lindland, and G. Sindre, "Defining Quality Aspects for Conceptual Models", In E. D. Falkenberg, W. Hesse, & A. Olive (Eds.). *Proc. the IFIP8.1 working conference on Information Systems Concepts (ISCO3); Towards a Consolidation of Views,* pp. 216-231, 1995.

[5] J. Krogstie, "Evaluating UML Using a Generic Quality Framework", chapter in *UML and the Unified Process*, Idea Group Publishing, pp. 1-22, 2003.

[6] Krogstie, J., and A. Sølvberg, *Information System Engineering: Conceptual Modelling in a Quality Perspective*, Kompendiumforlaget, Trondheim, Norway, 2003.

[7] C.F.J. Lange, and M.R.V. Chaudron, "Managing Model Quality in UML-based Software Development", *Proc. 13th Int'l Workshop on Software Technology and Engineering Practice (STEP'05)*, pp. 7-16, 2005.

[8] MODELWARE reports "D2.3 MDD Maturity Levels Definition" and "D2.8 MDD Process Framework", accessible at http://www.modelware-ist.org/, 2006.

[9] OMG SysML™, http://www.sysml.org/, last visited on 17 January 2007.

[10] M. Peleg, and D. Dori, "The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods"; *IEEE Trans. Software Engineering*, 26(8), pp. 742-759, 2000.

[11] H.C. Purchase, L. Colpoys, M. McGill, D. Carrington, and C. Britton, "UML Class Diagram Syntax: An Empirical Study of Comprehension", *Proc. Australian Symposium on Information Visualization*, Volume 9, pp. 113-120, 2001.

[12] M. Rossi, and S. Brinkkemper, "Complexity Metrics for System Development Methods and Techniques", *Information Systems*, 21(2), pp. 209-227, 1996.

[13] K. Siau, and Y. Tian, "The Complexity of Unified Modeling Language: A GOMS Analysis", *Proc. Twenty-Second International Conference on Information Systems,* pp. 443-447, 2001.

[14] I. Solheim, and T. Neple, "Model Quality in the Context of Model-Driven Development", *Proc. 2nd International Workshop on Model-Driven Enterprise Information Systems (MDEIS'06)*, pp. 27-35, 2006.

[15] G.H. Travassos, F. Shull, M. Fredericks, and V.R. Basili, "Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality", *Proc. OOPSLA'99*, pp. 47-56, in *ACM SIGPLAN Notices, 34(10)*, Oct. 1999.

[16] Unhelkar, B., *Verification and Validation for Quality of UML 2.0 Models*, Wiley, 2005.

[17] Østbø, M., *Anvendelse av UML til Dokumentering av Generiske Systemer*, Master thesis in Norwegian with the title: Applying UML to the Documentation of Generic Systems. University of Stavanger, 2001.