Model Quality in the Context of Model-Driven Development

Ida Solheim and Tor Neple

SINTEF ICT, P.O. Box 124 Blindern, N-0314 Oslo, Norway

Abstract. Model-Driven Development (MDD) poses new quality requirements to models. This paper presents these requirements by specializing a generic framework for model quality. Of particular interest are *transformability* and *maintainability*, two main quality criteria for models to be used in MDD. These two are decomposed into quality criteria that can be measured and evaluated. Another pertinent discussion item is the positive implication of MDD-related tools, both on the models in particular and on the success of the MDD process.

1 Introduction

1.1 Characteristics of Model-Driven Development

Model-driven development (MDD) has been around for some years, helping system engineers to analyze and document the systems to be created and maintained, and to generate parts of the program code automatically. In MDD, models are the prime artefacts. That means, models are in use throughout the whole production chain, from the early capture of user requirements to the production of executable code. Model transformations are essential, and these should preferably be automated. Indeed, tool support is by many considered a prerequisite for successful MDD (e.g. [1]).

Although MDD has been practiced for years, it did not gain ground until the Object Management Group (OMG) launched its Model-Driven Architecture (MDA™) initiative. Being "an approach to using models in software development" [2], MDA has boosted the development of tools and thereby (semi)automation of program development and maintenance. MDA motivates system development with the following characteristics:

- Many activities have models as input, or output, or both.
- Several of these activities are model transformations (while others are model analysis, model verification etc.).
- A transformation takes one or several models as input and produces a model (or models), or text, as output. During transformation, output models are supplied with domain-related information not present in the input model. An example of such a domain is the *platform* concept, often used for "implementation platform".

1.2 Model Quality – A Less Mentioned Concern

The authors of this paper believe that successful adoption of MDD depends on high-quality models, high-quality transformations, and high-quality transformation languages and tools.

While other authors have contributed to the understanding of quality related to transformations (e.g. [3]) and transformation languages (e.g. [4]), the quality of models in MDD has so far been a less mentioned concern.

According to Selic [5], *accuracy* has been the greatest problem for successful adoption of MDD. Lack of accuracy means imprecise models or modelling languages, paired with unclear rules for mapping to underlying implementation technologies.

The authors of this paper agree that Selic has a good point. However, in [5] the term accuracy is used for a collection of several undefined quality criteria. The purpose of this paper is to define more precise quality criteria for models to be used in MDD, and suggest how these criteria may be measured and evaluated.

1.3 The Structure of this Paper

The starting point for this work is a generic quality framework (chapter 2), which is specialized to a quality framework for MDD models and their environments (chapter 3). The implications of tools are discussed in chapter 4, and a conclusive summary is given in chapter 5.

2 A Generic Quality Framework

Krogstie and Sølvberg [6] presents a generic framework for discussing the quality of models. This framework will be used as a reference frame for discussing model quality in an MDD context, and will be refined for this purpose. Figure 1 depicts the framework's building blocks and their interrelationships, as described by Krogstie [7]. The explanation of the building blocks is rendered from [7] (mostly quoted):

- ***G,*** the (normally organizationally motivated) goals of the modelling task
- ***L***, the language extension, i.e., the set of all statements that are possible to make according to the graphemes, vocabulary, and syntax of the modelling languages used
- ***M***, the externalized model, i.e., the set of all statements in someone's model of part of the perceived reality written in a language

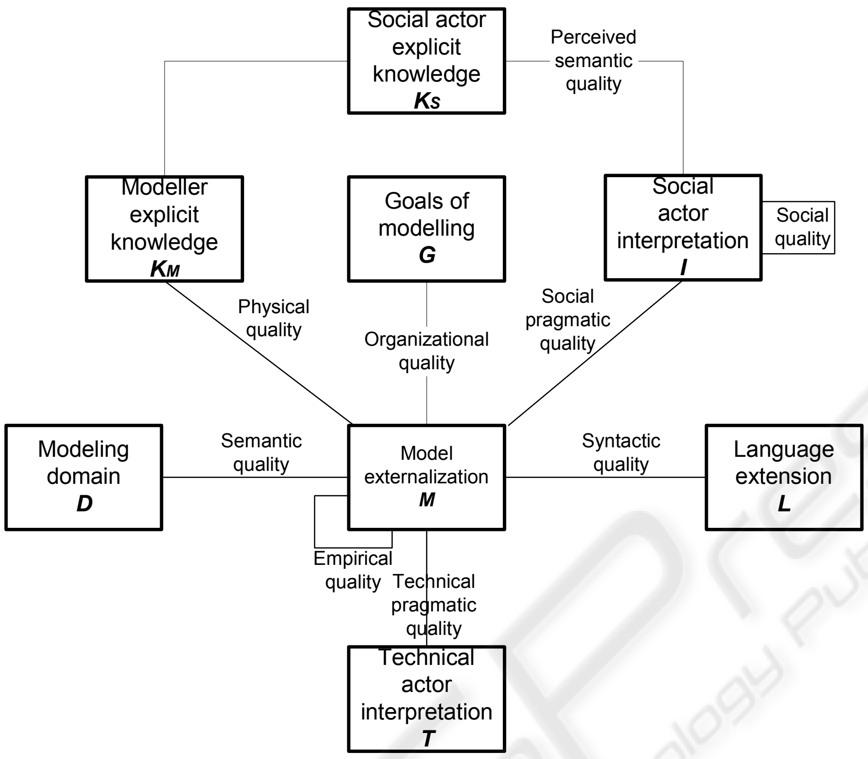

Fig. 1. Krogstie's generic framework for discussing the quality of models (rendered by courtesy of the author).

- ***D***, the domain, i.e., the set of all statements which can be stated about the situation at hand. Enterprise domains are socially constructed, and are more or less inter-subjectively agreed. That the world is socially constructed does not make it any less important to model that world.
- $\boldsymbol{K_s}$, the relevant explicit knowledge of the set of stakeholders involved in modelling
- $\boldsymbol{K_M}$, the relevant explicit knowledge of the set of stakeholders *actively* involved in modelling
- ***I***, the social actor interpretation, i.e., the set of all statements which the audience think that an externalized model consists of
- ***T***, the technical actor interpretation, i.e., the statements in the model as 'interpreted' by different model activators (e.g., modelling tools, transformation tools)

The various qualities are expressed as relations between pairs of these building blocks. The next chapter elaborates on model quality aspects related to MDD, refining the above framework accordingly.

3 Quality Criteria for MDD Models and their Environments

3.1 Overview

A quality framework specialized with respect to MDD is depicted in Figure 2. The authors of this paper want to emphasize *transformability* and *maintainability* as the two main quality criteria for models to be used in MDD. Models must have the ability to be transformed – to other models of greater detail (specialization), and at last to executable pieces of code for selected technical platforms. Transformability may be decomposed into:

- completeness (semantic quality)
- relevance (technical pragmatic quality)
- precision (technical pragmatic quality)
- well-formedness (syntactic quality)

Also, models for use in MDD need to be maintained during the system's lifetime. One of MDD's strengths is rapid iterations of the development cycle analysis—design—implementation—test, a feature that supports incremental development strategies. Given this setting, it is of paramount importance that changes made to the requirements are rendered correctly in the models and reflected in the code. A means to keep track of changes is to trace them, from the requirements through the necessary steps all the way to the code, and back. Therefore, maintainability of models may be decomposed into:

- traceability (technical pragmatic quality)
- well-designedness (syntactic quality)

Out of the six quality criteria listed above, only one (completeness) is explicitly mentioned in Krogstie and Sølvberg [6]. The remaining five may be considered refinements of generic relations shown in Fig. 1. The transformability and maintainability criteria are explained in the following subsections.

The *environments* of MDD models are here defined to be the change traces, the tools, and the MDD process itself. The change traces and the tools belong to the technical pragmatic quality.

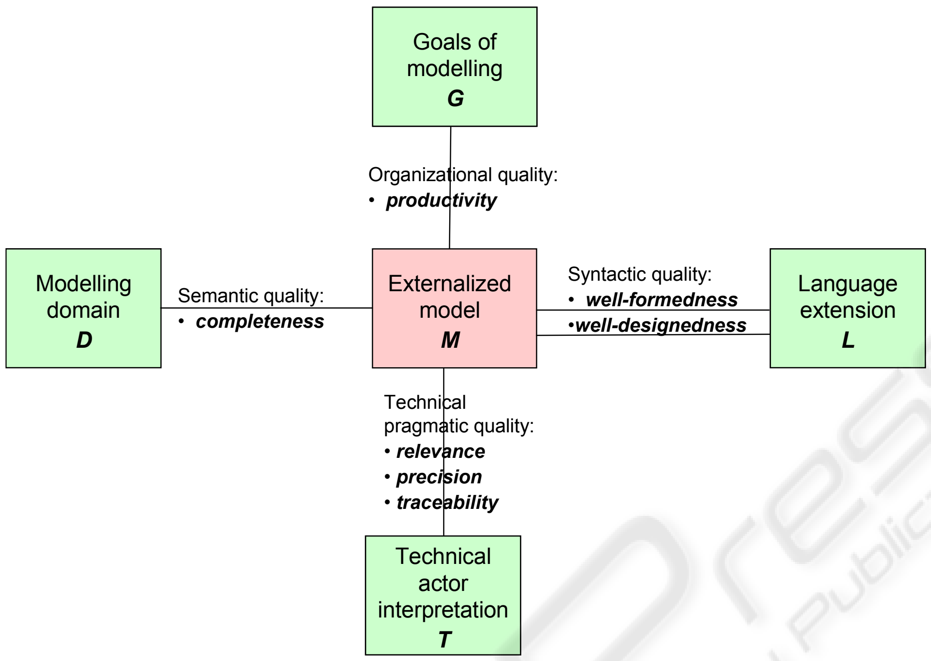

Fig. 2. A specialized framework for model quality in MDD.

Concerning the MDD process as such, we may identify a primary goal of achieving higher productivity in the development process. Hence,

- productivity (organizational quality)

may be considered a quality criterion. In accordance with [6], this is in Fig. 2 expressed as a relation between the goals of modelling (G) and the externalized model (M). However, for MDD, productivity should rather appear as a quality of M, L, T and D in combination. Productivity is hard to measure, and results cannot easily be generalized. The MODELWARE project [8] of the EU IST programme aims at measuring the productivity of MDD in industrial trials, based on approaches described in [9].

3.2 Transformability

Completeness is pointed out by Krogstie and Sølvberg [6] as an essential for the semantic quality of models. Completeness assures that the model contains all statements that are correct and relevant about the domain, and can be measured by a percentage as prescribed in [6].

Whereas Krogstie and Sølvberg [op. cit.] consider *relevance* to be a property of completeness, the authors of this paper would like to emphasize relevance as a distinct quality criterion. However, the relevance of a model used in MDD depends on both the model itself (M) and its transformation as specified by the technical actor interpretation (T). High relevance means that no more statements are included in the model than those which are going to be transformed. Relevance can be measured as the percentage of model elements actually used in a particular transformation. Making a larger model than necessary has a negative consequence in MDD; one has to drag along unused model elements (or code), which may complicate documentation, blur comprehension and hamper maintenance.

Precision reflects the level of detail and accuracy required for a model to be transformed successfully. The result of the transformation may be another model, which in case must be well-formed. Or, the result may be program code which can be compiled without errors and which constitutes some meaningful result, e.g. a component, a class structure or an interface. It may be possible to measure precision on a scale (ordinal or interval). However, these authors prefer to evaluate model precision as yes/no. This means, either the model is sufficiently precise for transformation, or it is not.

Well-formedness is a syntactic quality of utmost importance to model transformation. According to OMG [2], a transformation from one model to another is dependent on a mapping between the two respective metamodels. Hence, any model to be transformed must comply with its metamodel. For example, a model written in UML must comply with UML's metamodel. Also, there may exist sub-languages with limitations on the vocabulary and/or grammar rules of the overall language. Examples of such sub-languages are UML profiles. A well-formed model complies not only to its metamodel, but also to its sub-language (profile) if appropriate. A measure of well-formedness should yield 100 % before transformation is started.

3.3 Maintainability

3.3.1 Traceability

Traceability has been pointed out as an important aspect of MDD. One of the purposes of maintaining traces between model elements is to check a model element's origin, e.g. in a requirement model, and to follow a model element through transformations. In the latter case, the trace can also tell what kind of transformation was used, and which transformation rule was applied. Albeit traceability doubtlessly may involve more than one model, and indeed may involve artefacts other than models, this section discusses traceability as a quality of a model. This means, to what degree the model is usable in a scenario where traceability is needed.

Traceability may be vital for the management of large MDD projects, and for the maintenance of systems built according to MDD. Tool-supported traceability may range from "enterprise-wide" traceability solutions to simple traces maintained by the modelling workbench. A model's traceability depends on unique identifiers for the different elements that constitute the model; otherwise no traces can be established. Unique identifiers are supported by some modelling tools, but not all. In addition to the identification of model elements, one will need a mechanism that logs and documents all transitions undergone by each model element. Such a mechanism is currently under development in the IST project MODELWARE [8].

A traceability metric for a model could be the model's *trace coverage*, defined as a percentage denoting the proportion of traceable model elements relative to the total number of model elements.

3.3.2 Well-designedness

The maintainability of object-oriented systems has been studied by several authors, e.g. Briand [10]. The main approach has been various combinations of measurements, obtained by counting properties of object-oriented structures found in class diagrams. Marinescu [11] introduced a quality model for object-oriented systems, applying well-known metrics for the purpose of revealing particular design flaws. Among the design flaws that can be revealed by his method, are flaws resulting from *not* using selected design patterns described by Gamma et al. [12].

Well-designed models are understandable and tidy. In MDD, well-designedness deserves much attention because the models are the prime artefacts. Maintenance should preferably start with the models resulting from the last development cycle. If changes are made directly to the generated code, they should be reflected in the models as soon as possible to ensure the correspondence between the models and the code. Bad model design may complicate the code, confuse the developers, ruin the model-code correspondence and impede the use of MDD.

4 The Implications of Tools

In MDD, tools are used to create models, to transform one model into another, to generate non-model software artefacts, to maintain traces, etc. In such a setting, the human model-creation steps can be heavily guided by the tools. This means that several quality parameters can be kept at sufficient levels through guides and constraints in the tooling. It is also probable that the modeller will put most work into those models that will be subject to usage further down the MDD transformation chain.

A modelling tool will typically not allow a model to violate its metamodel. At least, the model will be compliant with *the tool's interpretation* of the metamodel. This is a feature that has been observed in UML tools in the past, when tool vendors have added capabilities not compliant to the UML metamodel as defined by the standard. Such extensions may cause problems in an MDD tool chain if a common non-standard metamodel, shared between the tools, is required.

A positive feature of some UML modelling tools is a mechanism allowing the user to check whether a model is compliant with the applied profile. In MDD, this is essential as most UML model transformations use stereotypes and extra properties in the transformation process. While the profile provides explicit language constraints, the tool enforces these constraints on the models. The quality of tool support for profile adherence is thus shared between the profile itself (how explicit are the constraints) and the tool (how well are these constraints enforced). In these cases, the quality of the model at hand is therefore a combination of the quality of the model and the quality of the applied profile.

Modelling tools can also help ensure that the structure (e.g. package organisation) of a model is in accordance with the expectations of the down-chain tools. This is typically done by the use of model templates or more formally defined constraints on the model structure.

5 Conclusion

Models have been used for years without direct influence on system implementation. However, the adoption of MDD forces system developers to spend more effort on making high-quality models. This paper has presented a framework for reasoning about model quality in the context of MDD. Since (automatic) model transformation is a crucial activity in MDD, several quality measures depend on both the model and the transformation (or transformation tool). Such dependency is indicated by the association line between M and T in Fig. 2. Although measures may be obtained on an ordinal or ratio scale, some quality criteria need to reach a sufficiently high level – a threshold – in order for transformations to succeed. The table below gives a summary of the quality criteria and suggestions of how to measure and evaluate them.

Quality Criterion	Type of Quality	Explanation
Transformability		
Completeness	semantic	The model contains all statements that are correct and relevant about the domain (from [6]). Suggested measurement unit: percentage.
Well-formedness	syntactic	The model complies with its metamodel, and also with its specified language profile, if appropriate. Suggested measurement unit: percentage. Suggested evaluation: yes/no.
Precision	technical pragmatic	The model is sufficiently accurate and detailed for a particular automatic transformation. Suggested evaluation: yes/no.
Relevance	technical pragmatic	The model contains only the statements necessary for a particular transformation. Suggested measurement unit: percentage.
Maintainability		
Traceability	technical pragmatic	The model's elements can be traced backward to their origin (requirements), and forward to their result (another model or program code). Suggested metric: *trace coverage,* the proportion of traceable model elements relative to the total number of model elements.
Well-designedness	syntactic	The model has a tidy design, making it understandable by humans and transformable to an understandable and tidy result. Suggested metric: The quality model of Marinescu [11], preferably extended with other diagrams than class diagrams.

The use of tools in MDD serves several purposes. In addition to facilitating the drawing, maintenance and transformation of models, tools also have some built-in quality controls. It is desirable that the quality controls performed by tools are extended to support as many as possible of the quality criteria listed above.

Future work will apply the presented quality framework to models used in MDD projects within industry or public administration. Such trials are expected to give valuable feedback to the appropriateness and further refinement of the framework.

Acknowledgements

This work has been conducted in the context of MODELWARE, a project co-funded by the European Commission under the "Information Society Technologies" Sixth Framework Programme (2002-2006). Information included in this document reflects only the author's views. The European Community is not liable for any use that may be made of the information contained therein.

References

1. Alanen, M., et al.: *Model Driven Engineering: A Position Paper*. 1st International Workshop on Model-Based Methodologies for Pervasive and Embedded Software, MOMPES'04 (2004)
2. Object Management Group: MDA Guide. Ver. 1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf (2003).
3. Bézivin, J., et al.: *The ATL Transformation-based Model Management Framework*. IRIN, Université de Nantes (2003)
4. Grønmo, R., et al.: *Evaluation of the Proposed QVTMerge Language for Model Transformations*. The First International Workshop on Model-Driven Enterprise Information Systems (MDEIS-2005). Miami (2005)
5. Selic, B., The Pragmatics of Model-Driven Development. In: *IEEE Software*, September/October 2003, http://computer.org/software
6. Krogstie, J. and Sølvberg, A.: *Information systems engineering - Conceptual modeling in a quality perspective*. Kompendiumforlaget. Trondheim, Norway (2003)
7. Krogstie, J. (2003). Evaluating UML Using a Generic Quality Framework. In: Favre, L. (ed.): UML and the Unified Process. IRM Press 1-22.
8. IST Project 511731 (2004-2006). *MODELWARE. Modeling solution for software systems,* http://www.modelware-ist.org/.
9. MODELWARE: *MDD Business Metrics (in prep.)*. Sixth Framework programme (2006)
10. Briand, L., Daly, J., and Wüst, J.: A Unified Framework for Coupling Measurement in Object-Oriented Systems. In: *IEEE Transactions on Software Engineering,* Vol. 25(1), No. January/February, 1999
11. Marinescu, R.: *Measurement and Quality in Object-Oriented Design*. University of Timisoara (2002)
12. Gamma, R., et al.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley (1994)