A Metamodel and Supporting Process and Tool for Specifying Quality Models in Model-Based Software Development

Parastoo Mohagheghi, Vegard Dehlen, Tor Neple

SINTEF, P.O.Box 124 Blindern
N-0314 Oslo, Norway
{Parastoo.Mohagheghi, Vegard.Dehlen, Tor Neple}@sintef.no

Abstract. Modelling is applied increasingly more in software development; from developing sketches to blueprints of design and approaches that use models in all phases of software development such as the model-driven engineering approach. Consequently, developers need tools and techniques that allow them to reflect upon the quality of the models, as well as the environment used for developing these models such as modelling languages, modelling processes and tools. This article describes work on developing quality models in model-based software development by identifying stakeholders and their purposes of modelling, specifying quality goals based on these purposes, identifying means or practices required to achieve quality goals and selecting proper evaluation methods. These are steps in developing quality models that include rationale for selecting quality goals and is supported by a process, a metamodel and a tool developed in Eclipse. The contributions of the approach are firstly providing a framework for developing quality models that is tailored to model-based software development and secondly providing example quality models that may be reused by different projects, thus facilitating work on quality issues in software development.

ACM CCS Categories and Subject Descriptions: D.2.8 [Software Engineering]: Metrics- *product metrics*; D.2.6 [Software Engineering]: Programming Environments- *graphical environments*; D.2.2 [Design Tools and Techniques]: *Computer-aided software engineering (CASE)*; D.2.9 [Management]: *software quality assurance*.

Keywords: Quality model, modelling, model-driven engineering, metrics, metamodel.

1 Introduction

Model-Driven Engineering (MDE) is an approach to software development that emphasizes using models when specifying, developing, analyzing, verifying, maintaining and managing software systems[1]. The promises of MDE are many; among them better communication between stakeholders by raising the abstraction level and providing models from different views, increased portability of systems to different platforms, traceability between artefacts to prevent and detect defects, and reducing error-prone and costly manual work. MDE also provides the possibility to define modelling languages and development environments tailored to the specific needs of domains and organizations that will reduce the complexity of software development and increase developers' productivity. These promises cover several quality goals identified in various quality models and researchers have also started work on specific quality issues in MDE such as identifying characteristics of models that are required to achieve high-quality software.

Modelling is applied in varying degrees in software companies and MDE is an ambitious goal of using models. However, quality of models has also gained extensive attention in software development approaches where models act as blueprints of design or source code is only partly generated from models. In this article, we use the term "Model-Based Software Development (MBSD)" to cover MDE and other software development approaches where models play a central role in software development.

The work described in this article aims at developing a framework in the context of MBSD with the purpose of developing quality models for evaluating and improving the quality of models as well as the modelling environment, including modelling languages, modelling processes, tools and even transformations performed on models. With a "quality model" we mean a set of quality goals (also called quality attributes or quality characteristics in literature) and their relations defined by some stakeholders based on the purposes of modelling, accompanied by a set of practices or means to achieve the quality goals, evaluation methods and link to related literature.

There exist different quality models with their definitions of quality goals or attributes; most generic and a few with focus on models and modelling languages. We have discussed these in our previous publications [13] [14]. Some shortcomings of generic quality models are the lack of rationale behind selecting quality goals and often the lack of identifying means to improve software quality. Existing work on the quality of models is not extensive either. We have analyzed existing quality models and identified the main constructs required to define quality models with models and modelling environment in mind. These constructs and the relations between them are described in a metamodel that defines a common language for specifying quality models. Applying the practice of MDE, the metamodel is supported by a tool developed in Eclipse that allows specifying quality models visually. Developing quality models also requires a process that is presented here. We provide some

[1] We use the term MDE in the remainder of this article to cover approaches where models are the primary artifacts in software development and transformations the primary operations on models, covering for example the OMG's Model Driven Architecture (MDA) and Model-Driven software Development (MDD).

example quality models developed by using the framework based on a review of literature on the quality of models.

This article is an extended and modified version of our paper [14] presented at the 2008 Nordic Workshop on Model-Driven Engineering (NW-MoDE'08) held in Reykjavik- Iceland, 20-22 August 2008[2]. The focus of this article is still on quality models in MBSD but the motivation is discussed in more depth, a process for developing quality models is presented and new examples of quality models are added.

The remainder of this article is organized as follows. Section 2 presents related work on quality models and discusses shortcomings and the motivation behind developing a quality model for MBSD. Section 3 presents our framework for defining quality models which includes the metamodel, a process that defines steps in developing quality models and the supporting tool. Section 4 presents examples developed by reviewing the literature and using the framework. The article is concluded in Section 5 and future work is proposed.

2 Background and motivation

In this section we present related work on quality models and the motivation behind defining a quality model for MBSD.

2.1 Related work on quality models

Research on quality models in software engineering has been going on for decades and different quality models for processes and products have emerged. Some of the best known product quality models are:

- McCall's hierarchical quality model which focuses on product quality, dividing it into the *external view* as seen by users (quality factors to specify) and the *internal view* as seen by the developers (quality criteria to build) [11]. By answering "yes" and "no" to questions related to quality criteria, one may measure to what extent a quality criteria is achieved.
- Boehm's hierarchical quality model with three levels of quality characteristics: *high-level characteristics* from the users' perspective, *intermediate characteristics* which are software characteristics needed to achieve the high-level characteristics, and *primitive characteristics* which are foundation for evaluation and defining metrics [2].
- ISO standards, especially the ISO-9126 series [6] (recently updated in the SQuaRE series of standards [3]) with the hierarchical model of six quality factors and sub-characteristics related to each of them. The standard divides metrics into *internal*, *external* and *quality-in-use*.

[2] http://nw-mode2008.hi.is/

- Dromey's model, which has three main elements: quality attributes, product properties that are important for achieving quality attributes, and links between product properties to quality attributes [4].

Each product quality model includes a set of desired properties of the product (we call them for "quality goals" in our framework); often defined in a hierarchy starting either from external to internal quality goals or from high-level quality goals to more tangible and measurable ones. The McCall, Boehm and ISO models share some quality goals and differ in others, and it is often difficult to find the rationale behind selecting some quality goals and leaving others out and how sub-goals are related to high-level goals; see for example [1] on problems with ISO standards. And finally being generic, they do not include proper means to achieve the desired quality goals for a specific development approach or environment. Dromey's approach distinguishes itself since it requires identifying tangible quality-carrying properties of components that are important for achieving quality goals. For example, reliability of code can be achieved by expressions that are computable and free of side-effects. It also emphasizes establishing links between quality goals and quality-carrying properties required to achieve them, something which is missing in the other models.

The above were examples of product quality models developed before the era of MBSD, while there are also examples of quality models with models in mind as the one developed by Lindland et al. [8] and depicted in Fig. 1.

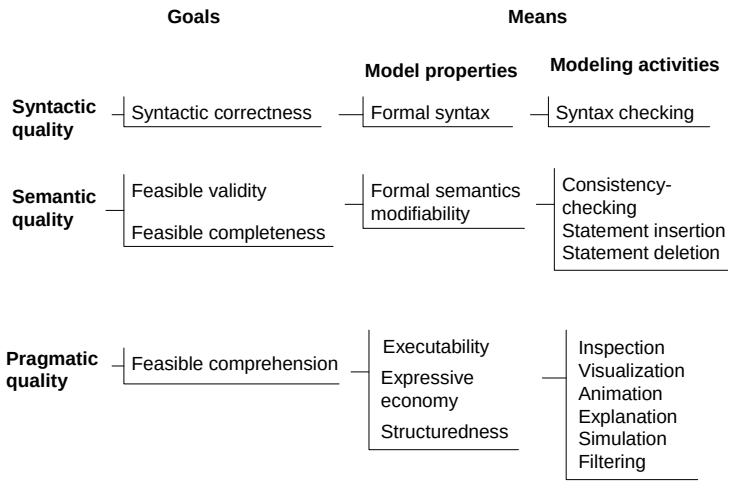

Fig. 1. Proposed framework by Lindland et al. for distinguishing quality goals and means to achieve them [8]

Lindland et al. identify three types of quality goals for conceptual models which are *syntactic quality* (adhering to modelling language syntax), semantic quality (correct elements and relations of the domain) and pragmatic quality (the

interpretation of the audience of the model). Others have extended this quality model by adding other quality goals, such as in [7]. Unhelkar has also used Lindland et al.'s framework to identify quality goals for UML models but replaces pragmatic quality with *aesthetics* [19]. Besides, his definition of syntax also covers documentation, packaging and other issues related to understandability of models that Lindland et al. have defined as pragmatic quality.

Lange and Chaudron have discussed different purposes of modelling and identified quality characteristics for each purpose, as depicted in Fig. 2, together with some metrics. The identified metrics are mostly size metrics (such as the number of elements in a diagram or relations between the numbers of elements in different diagrams) and object-oriented metrics with focus on design (such ad the depth of inheritance tree), in addition to a few more model-specific metrics such as the number of crossing lines in a diagram. Although the approach is relevant for MBSD with models developed in different stages of development and with different purposes, the quality model is for evaluation and does not have a constructive view to quality by including "means". Also the relations between purpose and characteristics and characteristics to metrics are not well justified.

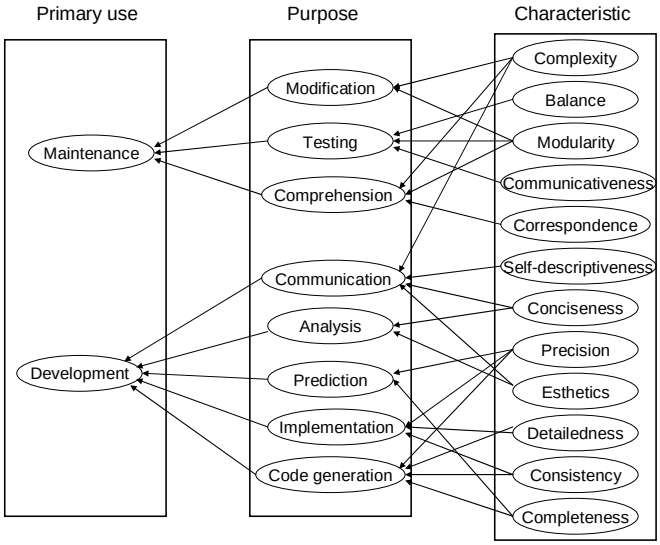

Fig. 2. Proposed quality model in [9] with different purposes of modelling and required quality characteristics

Some argue that focusing on product quality alone may not guarantee that an organization will deliver products of good quality. Products are created by processes and therefore the quality of processes should also be improved [16]. Some best known examples of standards or models for process quality are ISO:9001[3] series (which is a general international "quality management system" standard), ISO/IEC 12207[3] with

[3] http://www.iso.org/iso/home.htm

focus on software lifecycle processes, and the SEI CMMI model[4] with focus on process improvement. However, the relationship between process quality and product quality is far from clear [16]. MDE may also be viewed as a process for developing models and generating artefacts from them that includes a modelling language, modelling tools, transformations performed on models and a set of activities that should be performed to develop the necessary artefacts. Therefore several authors discuss the advantages of developing a model-driven development process or adapting the existing ones to MDE in order to improve the quality of models and the generated assets, for example [5] [17].

While the above literature covers quality models, others have proposed approaches for defining quality models for a development approach. For example Dromey defines a five step process for building product-specific quality models [4]:

1. Identify a set of high-level quality attributes for the product.
2. Identify the product components.
3. Identify and classify the most significant, tangible, quality-carrying properties for each component.
4. Propose a set of axioms for linking product properties to quality attributes.
5. Evaluate the model, identify its weaknesses and refine it.

And Trendowicz and Punter have identified activities during development of a quality model for software product lines as depicted in Fig. 3.

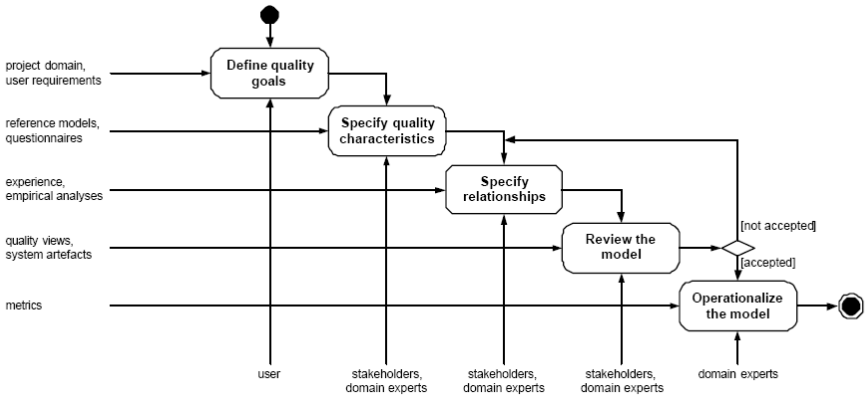

Fig. 3. Activities during development of quality models as defined in [18]

2.2 The need for a quality model in MBSD

Our work has focus on the quality goals of software models and the development environment around modelling in MBSD. This covers actually a spectrum of development approaches where modelling is applied for more than visualizing the code; from when models communicate the design, to when they are used to generate code while code and models co-exist, to when development is done solely using

[4] http://www.sei.cmu.edu/cmmi/general/index.html

models. Brown has discussed the spectrum of approaches to modelling as presented in [17] and depicted in Fig. 4. The "Model centric" approach is still based on code while the models are the main artefacts. Most (or all, if possible) of the code is generated from models; the developers, however, are given a possibility to add the code and synchronize it with models. "Models only" covers executable modelling techniques. We refer to these two approaches as MDE in this article, while MBSD covers approaches on the right side when models are used for more than code visualization. Fowler has also identified three modes for UML use which he calls for UMLAsSketch, UMLAsBlueprint and UMLAsProgrammingLangauge[5]. Thus models may be viewed either as intermediate or final products.

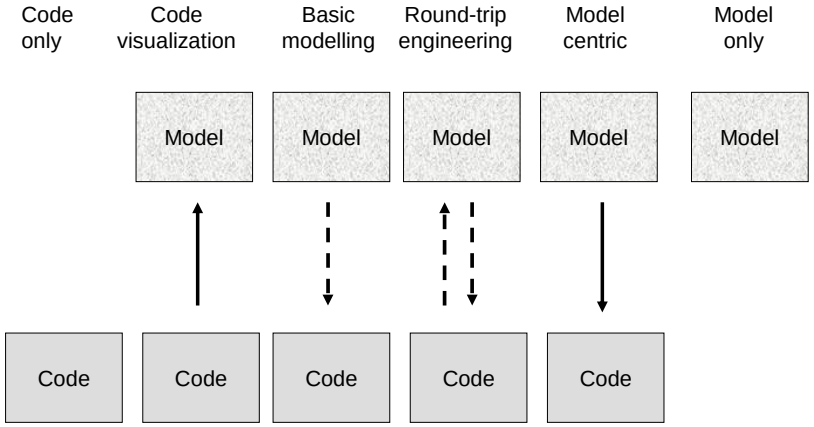

Fig. 4. Modelling spectrum defined by Brown and presented in from [17]

In MDE, models are subject of transformation to other models or text such as source code and by improving the quality of the models we will consequently improve the quality of the final product. The quality of the final product depends on other factors as well such as the appropriateness of modelling tools and processes, transformations performed, the target platform, quality assurance activities and the expertise of developers. Also in the other approaches where code is partly generated from models or models act as blueprint to communicate design, the quality of models should be considered.

The framework we are developing for MBSD has several purposes:

- It can be viewed as a kind of research programme to facilitate the understanding of the meaning of quality in this development approach;

[5] See his blog http://martinfowler.com/bliki/

- It includes necessary concepts for defining quality models that are needed in a MBSD approach; thus providing a tailored approach to the generic quality models;
- It provides a platform for collecting and analyzing state of the art and including results of empirical studies which may be reused by different industrial or research projects, thus saving effort by reuse of exiting models.

Furthermore, our quality framework has a *constructive view* to quality; i.e., the purpose is to achieve the desired quality goals by implementing proper means or practices; especially MDE practices. It is also essential to evaluate the models to be sure that the desired quality has been achieved. Therefore our approach includes identifying proper methods of evaluation. Finally, the quality model provides concepts for defining quality goals from multiple views and thus relating intentions of users to quality goals. We present the concepts of our quality framework in the next section.

3 A framework for defining quality models

The quality model is built around a metamodel that is introduced in Section 3.1. In Section 3.2 we present the process that supports developing quality models that adhere to this metamodel while the tool support is discussed in Section 3.3.

3.1 The metamodel

A *metamodel* is an explicit model of the constructs and rules needed to build specific models within a domain of interest; in our case quality models. Earlier approaches have focused on quality models with a set of quality goals as discussed in Section 2.1 while Wagner and Deissenboeck have identified the need for a metamodel that enables defining quality goals in a so-called *base model*, which may be extended later to application-specific *purpose models* [20]. They have identified some elements of this metamodel to be:

- Purpose of the quality model; as being constructive, predictive or assessing;
- View; as being either product, user, manufacturing or value-based;
- Quality attribute or goal such as those defined in the ISO standards;
- Technique; if a quality model focuses on a specific technique, for example inspections;
- Abstractness, which is the detail of a model, for example being general or product-specific.

The working session in the 2nd workshop on Quality in Modeling (QiM'07 held in conjunction with MoDELS 2007) put three questions for participants to answer[6]:

- What qualities of models and modelling matter?

[6] Proceedings at http://www.ipd.bth.se/lku/Quality%2Din%2DModeling%2D2007/

- How do they relate (similarity or dependence)?
- How can they be measured?

The contributions led to identifying several quality goals that can be included in a quality model for models. However, the issues of relations and measurement were not answered to the same extent, and as we discussed in relation with ISO and similar models, defining quality goals and classifying them per se is not enough without discussing how to achieve these goals and who are the intended users. The model proposed to organize the contributions includes high-level model quality characteristics, low-level quality attributes and metrics.

Comparing quality models show that they share the concepts of quality goals (either flat or hierarchical) and metrics for evaluation. The constructive view to quality also requires including "means". Finally since models are developed for different purposes, the concept of purpose is necessary. Fig. 5 depicts the main constructs of our metamodel and the relationships, while they are described in the remainder of this section.

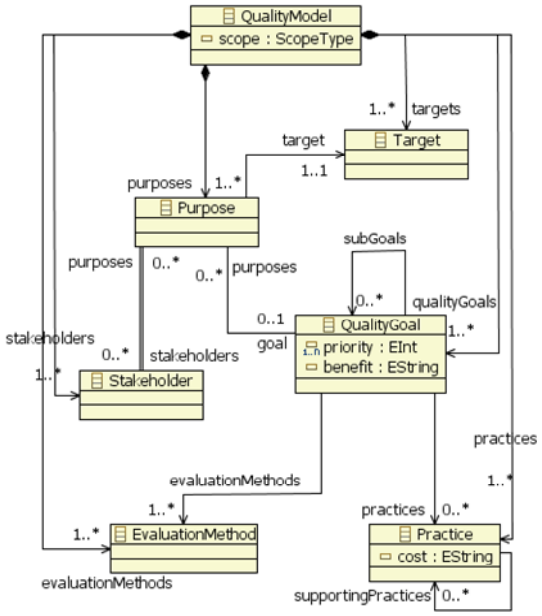

Fig. 5. Main constructs of the QiM metamodel for developing quality models

Not visible in the figure are attributes; all the metamodel elements have a "type", "definition" and "reference". The use of the "type" property varies for different elements and is proposed to used for classification purposes, "definition" is a textual description of the element, while "reference" is used to link references from literature to the element.

QualityModel
A quality model is a collection of quality entities and their relations. It has a scope type which may be used to indicate whether it is generic or related to a specific domain. The concept of "domain" may be used to refer to application domains such as telecom or business systems, or a domain of improvement such as modelling or communication with stakeholders. Quality model contains the constructs Stakeholder, Target, Purpose, QualityGoal, Practice and EvaluationMethod.

Stakeholder
Stakeholder is used to indicate stakeholders of a quality model such as model users, model developers or managers. By modelling stakeholders, we show who is defining quality goals and should also participate in evaluating them.

Target
A target is the artefact or activity that is the subject of quality improvement or assessment. Examples are models, metamodels, tools, modelling languages, transformations, and modelling process. Targets have type and for models, we can identify types such as:

- Computational Independent Model (CIM) / Platform Independent Model (PIM) / Platform Specific Model (PSM);
- Specification / analysis / implementation / documentation model;
- Structural versus behaviour model.

Target has an additional attribute called "phase" for indicating the development phase where the target is used.

Purpose
Purpose describes what expectations or interest stakeholders have in a given target of the quality model. For example, the purpose of developers for modelling may be generation of code and the purpose of system analyst may be communication with customers. Based on the purposes of stakeholders, quality goals are identified.

QualityGoal
We define a quality goal as a clear and understandable definition of what quality means to a stakeholder and for a defined purpose. The rationale behind having a quality goal is given by the purpose. For example, for generation of correct code from models, models should be correct. A good definition must let us measure quality in a meaningful way. Therefore high-level and less tangible quality goals should be refined into more tangible ones that may be evaluated; either quantitatively and based on metrics or based on some kind of user or expert judgment.

We have used the term "quality goal" and not "quality attribute" to avoid confusion with attributes of the constructs of the quality metamodel. One of the attributes of quality goals is "type", which is used for classifying goals; for example by:

- Using the classification of Lindland et al. as described in Section 2.1;
- Classifying quality goals into "hard goals" that can be achieved by an activity or "soft goals" that can be positively or negatively affected by an activity or activities;

- Classifying into product, project or process quality goals;
- Classifying into external, internal and quality-in-use.

Quality goals may also have relations between them that are defined as HELPS, BREAKS and DEPENDS. One may also assign priority to quality goals and identify benefits.

Practice
We define a practice as the means required to achieve a quality goal. For example, using modelling style and naming conventions are practices that can help developing correct and consistent models. Practices may be supported by other practices which allows refining practices and specify them in more details. For example we can define "modelling style" as a practice to improve the layout of models and relate different types of styles as supporting practices. Practices may also rely on other practices using the relationship DEPENDS or having relations HELPS and BREAKS. Practice has an additional attribute called "cost" used to estimate or document the cost of implementing a practice.

EvaluationMethod
Every quality goal should be evaluated as defined by the evaluation method. Evaluation method includes metrics and other appropriate ways of evaluation such as expert judgment, interviews or surveys. For example, using a domain specific code generator (a practice in our model) will result in less error-prone code (a quality goal) that can be evaluated by the reduction in the number of defects (a metric). From a value-based viewpoint with emphasize on costs and savings, it is also important to estimate how much effort is saved (less defects require less detection and correction activities) compared to effort spent on developing the code generator.

The above constructs are the main constructs of the QiM quality metamodel and other elements inherit from them or extend them. For example, models and metamodels inherit from target, and metrics inherit from evaluation method.

The QiM quality metamodel shares concepts with earlier work and extends them as described below:

- Defining quality goals depends on the phase of software development and the purpose of using models, as also emphasized in the quality model of Lange and Chaudron depicted in Fig. 2. Using the concept of "Purpose" together with "Stakeholder" allows defining the rationale behind quality goals.
- Lindland et al. have emphasized identifying means that help achieving quality goals [8]. We have used the concept of "Practice" with the same purpose although named differently. However, their quality model does not include purposes and evaluation methods, but a hierarchy of means where some are equal to evaluation methods.
- We have not included the concept "quality-carrying property" as in Dromey's model [4] but these are identified by defining sub-goals for specific targets[7].

[7] The first release of the metamodel included quality-carrying property but in practice it was difficult to separate these from quality goals.

- Our quality model has a wider scope that earlier work on the quality of models. We allow defining quality goals for other targets such as modelling languages and tools.

3.2 The supporting process

In addition to necessary concepts, developing a quality model requires a process that shows the steps and the required inputs and outputs as discussed in Section 2.1. Fig. 6 shows the process we have defined related to our metamodel. The process allows both *quality engineering* and *quality assessment*, the first with focus on improving quality by identifying relevant practices, and the second with focus on evaluating or assessing quality.

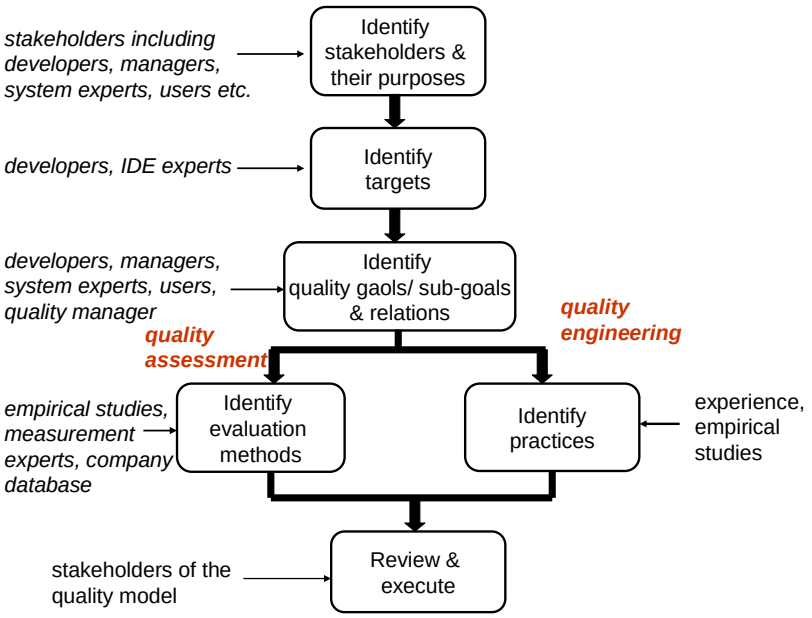

Fig. 6. Steps in developing quality models; specifying both quality engineering and quality evaluation

The steps in defining a quality model are:

1. **Identify stakeholders and their purposes**: stakeholders are usually defined as everyone who is affected by a project. For example it may be everyone who is interested to apply the MDE approach such as project managers, test team, development team, maintenance team, or non-technical experts. After identifying the stakeholders, their expectation or purposes should be identified such as using models for documentation, using a language and developed models for generating source code etc.

2. **Identify targets**: in this step we focus on identifying the targets of quality work that are related to the identified purposes. The question to answer is what should be in place to achieve the purposes. This may be models, modelling environment etc.
3. **Identify quality goals / sub-goals and their relations**: Identifying quality goals should involve all stakeholders and reflect the purposes of modelling and the priorities of the project. While quality goals show the required characteristics at a level that is easy to communicate, they may still be at a level too high to measure. It may necessary to decompose each quality goal into more specific quality goals which are called sub-goals. Some questions to answer are the role of the target in the software development life cycle (for example models can be for communication or generation), its future, compatibility with the environment, ease of use, or satisfying some constraints. Relationships, priorities and benefits (to be used in a Return-On-Investment analysis) should also be identified at this stage.
4. **Identify practices**: Practices show how we plan to achieve a quality goal. For example mandating the use of a certain development process will reduce the use of "non-normative" techniques, and should make it easier to develop consistent set of artefacts. Practices may be combined or support one another to achieve a desired quality goal, and they have some cost.
5. **Identify evaluation methods**: Specify how to evaluate quality goals or sub-goals; e.g., measuring quantitatively by metrics or subjective evaluation, inspections using checklists or interviewing the users.
6. **Review and execute**: The quality model should be reviewed for characteristics such as completeness, flexibility, transparency, relevance and possibility to be adopted. Execution is implementing practices or performing evaluation, and reporting the results.

Our metamodel and the related process for defining quality models based on it are generic enough to be used for any domain or related to any development approach. However, we have adapted the approach by adding targets and developing example quality models in MBSD.

3.3 Tool support

We started modelling our quality models with StarUML[8]. The problem with using a general modelling tool is that models get complicated and difficult to comprehend with the increasing number of elements and links, and the possibility for extension of elements and generation of other artefacts from models is limited. To support developers in specifying quality models for different domains or purposes and gaining experience with our metamodel, we provide tool support for the metamodel described in Section 3. An early version of this tool called *QualityMode* has been implemented on the Eclipse platform using the Graphical Model Framework (GMF). Eclipse is widely used as a tool and development platform within academia and consequently provides several benefits; (1) people are experienced in using the environment, (2)

[8] http://staruml.sourceforge.net/en/

using it promotes interoperability and allows our models to be used by other EMF-based tools, and (3) many plug-ins exist for possible reuse. The GMF plug-in, for example, allows one to rather quickly create a concrete syntax in a graphical editor. To support our concrete syntax, the metamodel from Fig. 5 has been extended and detailed with additional concepts.

The concrete syntax is currently simple, mostly using rectangles with different colours and annotations to distinguish different model elements, but it is easy to add suitable icons to help understandability of models. One example is the icon selected in the current version for "Stakeholder". Some basic constructs are provided as nodes (such as "QualityGoal") while others are shown as relations (such as "Purpose"), attributes (such as "Reference"), or an item list to choose from (for example for building quality models in MDE, the target such as "Model" or "Language" is selected from a list). This syntax is considered temporary, and our intention is to increase the use of graphics to differentiate concepts. The flexibility of GMF in defining graphics and icons is also a key reason for choosing GMF over UML-profiling for our tool solution. However, having an early implementation of the tool, our plan is to test it in a use case in order to gain experience with its usability and ability to model quality models. These experiences will also be the basis of the following tool iterations. We show an example model developed with the tool in the next section.

4 Example: A quality model for improving the quality of models

We have performed a review of literature concerning definitions of model quality and the proposed approaches to improve it[9]. Different publications have identified different quality goals for models while we have integrated these into a model that shows six quality goals of models- called the *6C goals*- that are required by various stakeholders. Fig. 7 shows these quality goals and when in the development process they are important. A detailed definition of quality goals is out of the scope of this article. We defined them shortly as:

- *Correctness*; as 1) including right elements and correct relations between them, and including correct statements about the domain. This is related to our understanding of the domain and is called semantic validity in the framework of Lindland et al. [8]; b) not violating rules and conventions; for example adhering to language syntax (well-formedness), style rules, naming rules or other rules or conventions.
- *Completeness*; as having all the necessary information and being detailed enough according to the purpose of modelling.
- *Consistency*; as no contradictions in the model. It covers consistency between views or models that belong to the same level of abstraction or development phase (horizontal consistency), and between views or models that represent the same aspect, but at different levels of abstraction or in different development

[9] The results are submitted for publication while a short overview of model quality goals is given in [15].

phases (vertical consistency). It also covers semantic consistency between models; i.e., the same element does not have multiple meanings in different models.

- *Comprehensibility*; as being understandable by the intended users; either human users or tools. For human users, several aspects impact comprehensibility such as aesthetics of diagrams, organization of a model, model simplicity or complexity, conciseness (expressing much with little), and using concepts familiar for the users or selected from the domain ontology. For tools, having a precise or formal syntax and semantic helps analysis and generation.
- *Confinement*; as being in agreement with the purpose of modelling and the type of system and being restricted to the modelling purpose. A model is a description from which detail has been removed intentionally. A confined model does not have unnecessary information and is not more complex or detailed than necessary.
- *Changeability*; as supporting changes or improvements so that models can be improved or evolved rapidly and continuously. Changeability is required since both the domain and our understanding of it or requirements of the system evolve with time.

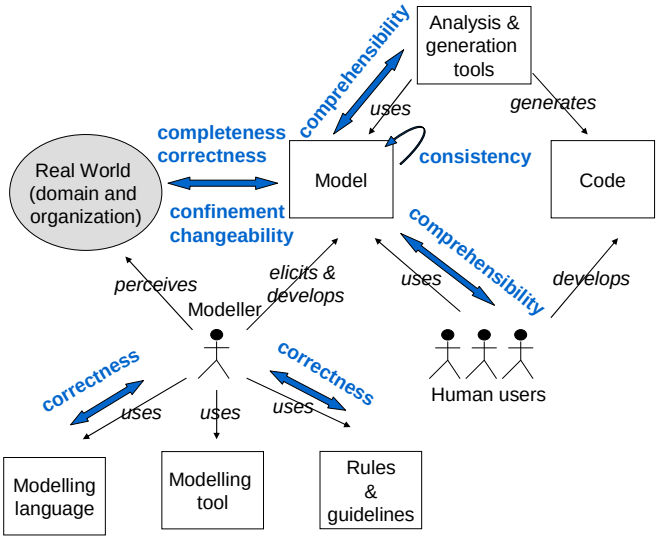

Fig. 7. Model quality goals in MBSD with transformation of real world to running software

We have further identified practices proposed in literature for improving the quality of models and related evaluation methods. By using our framework, we have developed a quality model that is partly presented here.

There are three types of stakeholders when discussing the quality of models:

- Developers who need to understand models for the purpose of implementation and modification. For implementation, models should be defect-free (correct,

complete and consistent) in the first place while for modification models should also be understandable and changeable.

- Tools that should interpret and analyze models or generate other artefacts from them. For them, models should be technically comprehensible.
- Others who use models for the sake of communication such as discussing design and communication with customers. They need models that are understandable and do not include unnecessary implementation details. Of course models should also be correct, complete and consistent, however to a varying degree. For example in a multi-diagram approach such as UML, consistency between diagrams may be more important than completeness for correct interpretation of models; as confirmed by an experiment presented in [10].

Fig. 8 shows model quality goals from the viewpoint of these stakeholders based on their purposes of modelling. Stakeholders are shown on the top of the figure, purposes are shown as relations between target ("Application model" in this case) and quality goals, while stakeholders interested in a quality goal are shown in {} related to a purpose.

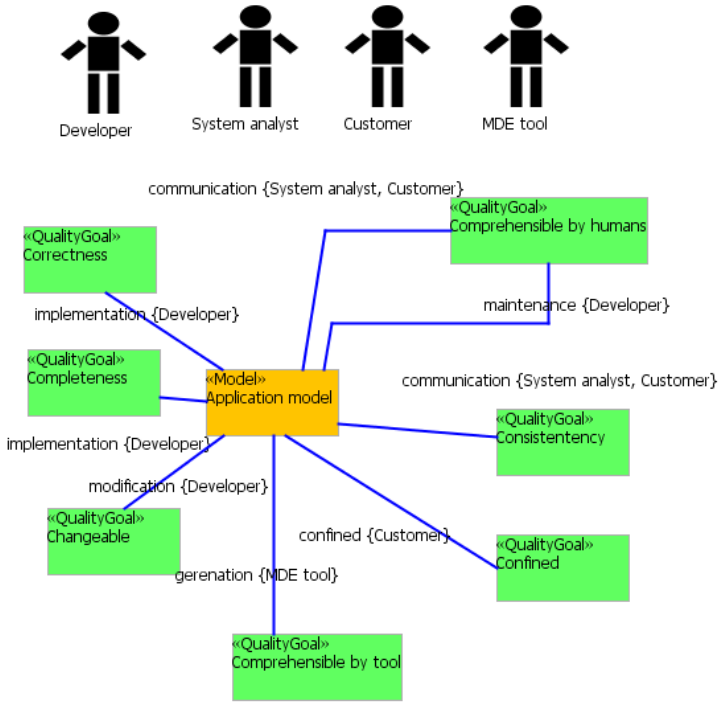

Fig. 8. Stakeholders, purposes of modelling and related quality goals for models

Fig. 9 shows that correctness can be further refined in sub-goals:

- “Language Syntax correctness” refers to adhering to language syntax. It can be achieved by integrating syntax rules in modelling tools and preventing syntax errors.
- “Correct names” refers to naming of elements in a model and may be achieved by using naming conventions during modelling.
- “Correct meanings and relations” is a type of semantic quality and may be achieved by having a formal (or precise) semantics and using semantic conventions. There are various ways to develop models with formal semantics: transforming informal models to formal ones, using formal languages, using constraints such as adding OCL (the Object Constraint Language) constraints to models, and including semantic in Domain-Specific Modelling Languages (DSML). These may be modelled as supporting practices of “Formal semantics”.
- “Domain validity” refers to being valid for the relevant domain where a domain ontology may be used to achieve this validity or domain experts may be involved during developing domain models.

Proper evaluation methods are identified as well. Some evaluation methods such as “Inspection” may be selected from a list while it is also possible to add others such as “Analysis by tool”.

Fig. 9. Identified sub-goals related to model correctness together with practices and evaluation methods

In [14] we presented another quality model for the quality of DSMLs which shows that our approach is flexible to be used for different targets. The quality model for DSMLs is developed using the previous version of the tool and the graphics is therefore slightly different while the constructs are the same.

5 Conclusions and future work

This article started by giving an overview over selected quality models in software engineering, which uncovered both weaknesses and strengths of existing approaches. Quality models should give a sound rationale for the selected set of quality goals and explain the relationships between concepts. One way of avoiding a static set of quality goals, which should rather be selected dynamically depending on the stakeholders' needs, is to have a dynamic and flexible framework that allows developers to define a quality model for the context; whether it is a general purpose quality model or one tailored for a specific domain. In model-based software development, selecting quality goals also depends on the purposes of modelling and the practices organizations choose to implement such as developing domain-specific modelling languages and using transformations for generating artefacts from models. A way of achieving this is to provide a framework that is flexible while adapted to model-based development needs. The QiM metamodel includes a set of important concepts that needs to be considered when defining quality models, and takes into consideration the previous work done in the field of quality in software engineering and quality of models. Metamodeling is one of the main practices of model-driven engineering and we take advantage of it to provide a language for developing quality models. Although we focus on the quality of models and the modelling environment, the metamodel is generic and may be used for defining quality models in any area. We also have a first implementation of a tool that offers graphical modelling of quality models, allowing quick development by reuse of existing concepts and quality models.

The framework includes a process for defining quality models which is applicable to our research as well: Stakeholders are in this case those who want to develop quality models in model-based software development with the purpose of improving software quality. Quality goals of the framework were defined in Section 2.2 such as being generic to be applicable in multiple contexts while integrating research on model-based software development, and allowing us to develop reusable models. The approach we selected for developing the framework (i.e., practices) has been performing state of the art analysis in a systematic review, identifying generic constructs in a metamodel and developing a process and tool around it that allows visual modelling of quality models. Evaluation of the framework is done by developing example models such as the one described in Section 4 and empirical work that we plan to perform in different research projects in order to evaluate the usefulness and ease of use of the framework.

The contributions of our work are developing a) the framework for developing quality models including the metamodel, process and tool; and b) applying the framework on the quality of models and domain-specific modelling languages so far. These models may be used in a library of customizable quality models in model-based software development.

For future work, we plan to complete the quality models for models and domain-specific modelling languages and detail them with more metrics and evidence from empirical studies. The intention is to develop base models that can be extended by users. The empirical evaluation of the usefulness of our framework is done currently

in the MODELPLEX project[10] where we develop quality models for domain-specific modelling languages. The results of this work will be published in future. In parallel and based on experiences from these cases, we also plan to iterate on the metamodel and the tool support, in order to provide an expressive and easy-to-use environment for modelling quality models.

Acknowledgments. This work has been co-funded by the Quality in Model-Driven Engineering project (cf. http://quality-mde.org/) at SINTEF ICT in Oslo, Norway and the European Commission within the 6th Framework Programme project MODELPLEX contract number 034081. We thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Al-Kilidar, H., Cox, K., Kitchenham, B.: The Use and Usefulness of the ISO/IEC 9126 Quality Standard. International Symposium on Empirical Software Engineering, 7 p. (2005)
2. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., Merritt, M.: Characteristics of Software Quality. North Holland (1978)
3. Bøegh, J.: A New Standard for Quality Requirements. IEEE Software 25(2), pp. 57--63 (2008)
4. Dromey, R.G.: Concerning the Chimera. IEEE Software 13 (1), pp. 33--43 (1996)
5. Gavras, A., Belaunde, M., Pires, L.F., Almeida, J.P.A.: Towards an MDA-based Development Methodology for Distributed Applications. Proc. EWSA'04, 1st European Workshop on Software Architecture, LNCS volume 3047, Springer Berlin / Heidelberg, pp. 230--240 (2004)
6. ISO, International Organization for Standardization: ISO 9126-1:2001, Software Engineering – Product Quality, Part 1: Quality model (2001)
7. Krogstie, J.: Evaluating UML Using a Generic Quality Framework. Chapter in *UML and the Unified Process*, Idea Group Publishing, pp. 1--22 (2003)
8. Lindland, O.I., Sindre, G., Solvberg, A.: Understanding Quality in Conceptual Modeling. IEEE Software 11(2), pp. 42--49 (1994)
9. Lange, C.F.J., Chaudron, M.R.V.: 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)
10. Lange, C.F.J.: Assessing and Improving the Quality of Modeling - A Series of Empirical Studies about the UML. PhD thesis, URL http://www.langomat.de/research/thesis/thesis.pdf (2007)
11. McCall, J. A., Richards, P. K., Walters, G. F.: Factors in Software Quality. Nat'l Tech. Information Service, Vol. 1, 2 and 3 (1977)
12. Mohagheghi, P., Aagedal, J. Ø.: Evaluating Quality in Model-Driven Engineering. In: Workshop on Modeling in Software Engineering (MISE'07), In: Proc. of ICSE'07, 6. p (2007)
13. Mohagheghi, P., Dehlen, V.: Developing a Quality Framework for Model-Driven Engineering. 2nd Workshop on Quality in Modeling at MoDELS 2007, 15 p., URL: http://www.ipd.bth.se/lku/Quality%2Din%2DModeling%2D2007/ (2007)

[10] http://www.modelplex.org

14. Mohagheghi, P., Dehlen, V.: A Metamodel for Specifying Quality Models in Model-Driven Engineering. Proceedings of the Nordic Workshop on Model Driven Engineering, pp. 51--65, URL http://www.quality-mde.org/publications.html (2008)
15. Mohagheghi, P., Dehlen, V., Neple, T.: Towards a Tool-Supported Quality Model for Model-Driven Engineering. Proceedings of Workshop on Quality in Modeling (QiM'08), 15 p., URL http://www.quality-mde.org/publications.html (2008)
16. Satpathy, M., Harrison, R. Snook, C., Butler. M.: A Generic Model for Assessing Process Quality. Proceedings of the 10th International Workshop on New Approaches in Software Measurement, pp. 94—110 (2000)
17. Staron, M.: Adopting Model Driven Software Development in Industry – A Case Study at Two Companies. Proc. MoDELS 2006, LNCS 4199, 2006, pp. 57-72.
18. Trendowicz, A., Punter, T.: Quality Modeling for Software Product Lines. In: 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'03), 7 p. (2003)
19. Unhelkar. Unhelkar, B.: Verification and Validation for Quality of UML 2.0 Models. Wiley-Interscience (2005)
20. Wagner, S., Deissenboeck, F.: An Integrated Approach to Quality Modeling. Fifth International Workshop on Software Quality, In: Proc. of ICSE'07, 6 p. (2007)