Quality Attributes for Software Metamodels

Manuel F. Bertoa and Antonio Vallecillo

GISUM/Atenea Research Group. Universidad de Málaga (Spain).
{bertoa,av}@lcc.uma.es

Abstract. As Model-Based Engineering (MBE) starts to be effectively used, some of its constituent disciplines such as Domain Specific Modeling and Metamodeling are becoming quite popular. Metamodels play a cornerstone role in these approaches, not only for defining Domain Specific Languages but also for specifying all kinds of artifacts involved in MBE. However, there is a lack of appropriate quality models that allow an effective assessment of Metamodels. Besides, the international standards that address the software products' quality issues (in particular, those from ISO and IEEE) have shown to be too general for dealing with the specific characteristics of Metamodels because of their dual nature of being models on the one hand, and representing modeling languages on the other. In this paper we propose a quality model for Metamodels, that defines a set of quality attributes for evaluating Metamodels.

1 Introduction

Model-Based Engineering (MBE) advocates the use of models as the key artifacts in all phases of development, from system specification and analysis, to design and implementation. Each model usually addresses one concern, independently from the rest of the issues involved in the construction of the system. A model is written in the language of its metamodel. A metamodel describes the concepts of the language, the relationships between them, and the structuring rules that constraint the model elements and their combinations in order to respect the domain rules.

Metamodels have become cornerstone elements in MBE, playing key roles not only for defining Domain Specific Languages (DSLs) but also for specifying all kinds of artifacts involved in MBE: models, transformations, types, etc.

Metamodels are also specially relevant in the context of Global Model Management [1, 2], in which megamodels are kinds of registries for resources available from a given model-driven platform, recording all accessible entities like models, metamodels, transformations, tools, and the various relations between them. All these entities are defined by models, which conform to metamodels, and described by languages and notations whose abstract syntax is specified by metamodels, too.

As the number of metamodels and its importance grow, the need to measure and evaluate the quality of metamodels becomes increasingly relevant. However, there is a lack of appropriate quality models that allow an effective assessment

of Metamodels. Besides, the international standards that address the software products' quality issues (in particular, those from ISO and IEEE) have shown to be too general for dealing with the specific characteristics of Metamodels because of their dual nature of being models on the one hand, and representing modeling languages on the other.

We need to understand what is the meaning of quality for a metamodel in the context of MBE, which are the attributes of metamodels that affect their quality, how can they be measured, and which is their degree of influence on the overall quality of a metamodel. This is precisely the goal of this work, in which we identify a set of quality attributes for Metamodels, and propose a Quality Model for them that defines a set of characteristics and sub-characteristics. This Quality Model, that is based on ISO/IEC 9126, provides all the information required to evaluate metamodels according to different criteria, and can be further customized depending on the contexts of use of specific modelers.

The structure of this paper is as follows. After this introduction, Section 2 briefly presents some basic metamodels concepts. Section 3 introduces the terminology used, as well as a short reference to ISO/IEC 9126 international standard [3]. Afer that, our quality model (QM4MM) is described in Section 4, in which a refinement of the ISO/IEC 9126 quality model is defined and quality attributes are related to each subcharacteristic. Finally, Section 5 draws some conclusions and outlines some future research activities.

2 Models and Metamodels

We normally say that a model *conforms to* its metamodel [4]. Metamodels are also models, and therefore they need to be written in another language, which is described by its meta-metamodel. This recursive definition normally ends at the meta-metalevel, since meta-metamodels conform to themselves. So, we are going to find some definition on what is a model and a metamodel.

A *Model* is:

- A description of (part of) a system written in a well-defined language. [5]
- A representation of part of the function, structure and/or behavior of a system. [6]
- A formal specification of the function, structure and/or behavior of an application or system. [7]
- A description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language. [7]
- A set of statements about some system under study. Here, statement means some expression about the system that can be considered true or false. A modeling language lets us express the statements in models of some class of system. [8]

And, a *Metamodel* is:

- A model of a well-defined language [5]

- A model of models [6]
- A model that defines the language for expressing a model. A meta-metamodel is a model that defines the language for expressing a metamodel. The relationship between a meta-metamodel and a metamodel is analogous to the relationship between a metamodel and a model. [9]
- A specification model for a class of system where each system in the class is itself a valid model expressed in a certain modeling language. That is, a metamodel makes statements about what can be expressed in the valid models of a certain modeling language. [8]

In our opinion, examining the above definitions, the metamodels can be analyzed from two points of view to assess their quality, treating them as a language (more precisely, as the abstract syntax for a language) and, also, as a model.

1. Metamodels as models.
   A key aspect of the metamodels is that they are obviously models. Therefore, to assess their quality it will be necessary to use and apply quality attributes that are meaningful to the models.
2. Metamodels as lenguages.
   Another way to approach the metamodels is analyzing them as the specification of the abstract syntax of languages. Therefore, we look for quality attributes that are applicable to languages, adapting these attributes to metamodels when necessary.

D. Moody propose in [10] to adopt a common vocabulary of concepts and terms based on ISO/IEC 9126 for conceptual model quality. His conclusion are valid for metamodels quality, too. That, metamodels quality frameworks (as conceptual model quality) should be structured using some principles based on ISO/IEC 9126:

1. Metamodel quality should be decomposed into a hierarchy of quality characteristics, subcharacteristics and measures.
2. Single-word labels should be used for each quality characteristic and subcharacteristic, using commonly-understood terms
3. Each quality characteristic and subcharacteristic should be defined using a single, concise sentence.
4. Measures should be defined for measuring each subcharacteristic.
5. Evaluation process: detailed procedures should be defined for conducting quality evaluations.

Although there are other approaches, such as Dromey's framework for quality models [11], ISO/IEC 9126 is a preferable option because it is widely accepted and used in practice and represents a good consensus.

3 Quality Models and software measurement concepts

In general, there is a lack of consensus about how to define and categorize software quality characteristics. Here we will follow the terminology defined in [12]. At the highest level, the main goal of a measurement process is to satisfy certain *information needs* by identifying the *entities* and the *attributes* of these *entities* (which are the targets of the measurement process). *Attributes* and *information needs* are related through *measurable concepts* (which belong to a *Quality Model*).

As *quality model* we have selected an adaptation of ISO/IEC 9126. Since that model is a generic quality model for any software product (or artifact), it requires some customization for the particular case of metamodels. The customized model, denominated QM4MM, is presented below. As ISO/IEC 1926, QM4MM defines a set of *characteristics* and *sub-characteristics*, together with the relationships between them. They provide the basis for specifying quality attributes and some measures for evaluating them.

Attributes are evaluated using *measures*. A *measure* relates a defined *measurement approach* and a *measurement scale*. A measurement approach is the logical sequence of operations, described generically, used in quantifying an attribute with respect to a specified scale [13]. A *measure* is expressed in *units*, and can be defined for more than one *attribute*.

The international standard ISO/IEC 9126 provides a quality model for software products, inheriting from the quality factors of McCall et al.[14] and the model of Boehm et al. [15]. This quality model propose decompose the quality of a software product in six characteristics:

- **Functionality**, the services offered by the product.
- **Reliability**, the confidence in a continuous and precise operation of the product.
- **Usability**, if users found the software product easy and efficient.
- **Efficiency**, evaluation (according to some defined criteria) whether the software performs appropriate use of resources.
- **Maintainability**, if it is easy to update and modify the software product.
- **Portability**, if the software product can be used (or not) in others environments.

The QM4MM quality model can classify most proposed attributes without much difficulty and it can be adapted to metamodels from the original software product environment. The main adaptation is to remove some subcharacteristics irrelevant to in the field of metamodeling. Then, we need to assign each attribute to the appropriate subcharacteristic. In general, the attributes have been gathered from various authors and we have used the original name and definition used by the authors, when it has been possible. In this process, some problems have emerged, notably the use of different names for the same concept (synonymy) or, contrarily, using the same name to represent different concepts (homonymy).

4 A Quality Model for Metamodels (QM4MM)

In this section, we present our proposed model which is an adaptation of the ISO/IEC 9126 quality model as mentioned above. All six characteristics are present but with its definitions adapted to the metamodel field and some subcharacteristics have been eliminated, the ones not relevant to the metamodels. All the characteristics have at least one subcharacteristic that is relevant to metamodels environment; this six characteristics can be found in the model that we are proposing.

We decided not to include the six compliance subcharacteristics, as we have no notice about metamodels standards for functionality, reliability, efficiency, usability, maintainability or portability. Furthermore, the specific subcharacteristics of software products do not make sense for static and non-executable artifacts as the metamodels (e.g. Time Behavior). Specifically, for the *functionality* we have focused on the suitability, accuracy, and interoperability, removing the security subcharacteristic. For the *reliability* characteristic, we only consider the maturity because it is difficult to think of a fault-tolerant metamodel or one with the ability to recover itself. The *usability* and *maintainability* keep all the subcharacteristics proposed by ISO/IEC 9126. Finally, the *portability* feature retains the Adaptability and Replaceability, eliminating the Instalability and Co-existence.

Also, this section classifies the attributes, related to models and languages quality, with each subcharacteristic of the proposed model. As previously mentioned, we are going to leave the attribute definitions proposed by each author when they are clearly defined in the original source or authors comments when there is not a definition. Obviously, not all authors refer to the quality concepts they use as quality attributes but, from our point of view, all can be treated this way.

4.1 Functionality

Definition: The capability of the metamodel to provide functions which meet stated and implied needs when the metamodel is used under specified conditions.

4.1.1 Suitability subcharacteristic. The capability of the metamodel to provide an appropriate set of functions for specified tasks and user objectives.

Relevance The model contains only the statements necessary for a particular transformation. [16]

Completeness There are four definitions related to completeness:

- (Completeness) Having all the necessary information and being detailed enough; according to the goals of modeling. Completeness is a semantic quality. [17]
- (Completeness)[1] $D \setminus M = \varnothing$, there are not statements that are correct and relevant about the domain, but are not included in the model. [18]

[1] According to [18], the quality is defined with respect to four components: the model (M); the language (L); the domain (D), and the interpretation (I).

- (Sufficient): Do the set of quality categories include all aspects of quality of information models? Are there any aspects that have been left out? [18]
- (Completeness) The model contains all statements that are correct and relevant about the domain. This can be checked against the ontological metamodel. [16]

Completeness/Comprehension For each stakeholder j, $M \setminus I_j = \varnothing$ and $I_j \setminus M = \varnothing$. There are no statements in the model which are not in the stakeholder's interpretation of the model, and vice versa. [18]

Conciseness Three definitions are related to this term:

- (Conciseness) The extent that the system is described to the point and not unnecessarily extensive. [19]
- (Expressive economy) How effectively can things be expressed in the language? This is also related to participant interpretation.[20]
- (Space economy) Concise models are produced. [21]

Consistency Language features cooperate to meet language design goals [21].

Detailedness The extent to which a model describes relevant details of the system (Appropriate abstraction level). [19]

Confinement Being in agreement with the purpose of modeling and the type of system, and being restricted to the modeling goals, such as being at the right abstraction level and not having information not required (for example including implementation details in analysis models). Confinement is related to semantic quality. [17]

Cohesiveness The extent to which the content of a metamodel (or ontology) is focused on one topic. [22]

4.1.2 Accuracy subcharacteristic.

The capability of the metamodel to provide the right or agreed results or effects with the needed degree of precision.

Precision The model is sufficiently accurate and detailed for a particular automatic transformation. [16]

Precision/completeness Average granularity or precision of the ontologys class model.[22]

Correctness There is several definitions related to this term:

- (Correctness) Including correct elements and correct relations between them and not violating rules and conventions. Thus it covers both syntactic correctness (right syntax or well-formedness) and semantic correctness (right meaning and relations relative to the knowledge about the domain).[17]
- (Syntactic Correctness) $M \setminus L = \varnothing$, there are no statements included in the model that are not part of the language. [18]
- (Validity) $M \setminus D = \varnothing$, there are no statements included in the model that are not correct and relevant about the domain. [18]
- (Well-formedness) The model complies with its language definition. This can be checked using the linguistic metamodel. [16]

Consistency Several definitions are related to consistency:

- (Consistency) The extent to which no conflicting information is contained. [19]
- (Consistency) No contradiction exists in the model, related to semantic quality. It covers consistency between views that belong to the same level of abstraction or development phase (horizontal consistency), and between views that model the same aspect, but at different levels of abstraction or in different development phases (vertical consistency). The model should also be unambiguous; i.e. not allowing multiple interpretations. [17]
- (Semantic Consistency) The extent of consistency in using the same values (vocabulary control) or elements to convey the same concepts and meanings in a metamodel (or ontology). [22]
- (Structural consistency) The extent to which similar elements (classes, properties) of a metamodel (or ontology) are represented with the same structure, format, and precision. [22]

Redundancy There are some definitions related to redundancy:

- (Conceptual redundancy) There should be one well defined mechanism to express each concept that we want to teach. [23]
- (Uniqueness) There are no redundant or overlapping features. [21]
- (Redundancy) The amount of noninformative content. [22]

Accuracy/Validity The extent to which information is legitimate or valid according to some stable reference source, such as a dictionary, or set of domain constraints and norms (soundness), or both. [22]

4.1.3 Interoperability subcharacteristic. The capability of the metamodel to interact with one or more specified metamodels.

Interoperability The ability to exchange. We need both to a) establish the mechanism to exchange (such as flow of information) and b) define how to extract or understand the information in order to process them. [24]

4.2 Reliability

Definition: The capability of the metamodel to maintain a specified level of performance when used under specified conditions.

4.2.1 Maturity subcharacteristic. The capability of the metamodel to avoid failure as a result of faults in the metamodel.

We have four attributes for maturity: volatility, currency, authority, and balance.

Volatility The amount of time the content of a (classes, properties) remains valid. [22]

Currency The currency of a metamodel (or ontology). [22]

Authority The degree of reputation of a metamodel (or ontology) in a given community or culture. [22]

Balance A model possesses balance to the extent that all parts of the system are described at an equal degree of all other model characteristics. [19]

4.3 Usability

Definition: The capability of the *metamodel* to be understood, learned, used and attractive to the user, when used under specified conditions.

For the Usability characteristic, we have some quality attributes for each subcharacteristic.

4.3.1 Understandability subcharacteristic. The capability of the metamodel to enable the user to understand whether the metamodel is suitable, and how it can be used for particular tasks and conditions of use.

Comprehensibility For human users, several aspects impact comprehensibility such as aesthetics of diagrams, organization of a model, model simplicity (or complexity which may be reduced by for example generalization and refactoring), conciseness (expressing much with little), and using domain concepts (Mitchell et al. call this continuity defined as carrying the structure and behavior of a problem domain into system and design models. [19]).

Self-Descriptiveness The extent that it contains enough information for a reader to determine its objectives, assumptions, constraints, inputs, outputs, components, and status. [19]

Complexity Measures the effort required to understand a model. [19]

Complexity The extent of cognitive complexity of a metamodel (or ontology) measured by some index or indices. [22]

Perceptibility How easy is it for persons to comprehend the language? This is related to the current and potential knowledge of the participants and their interpretation of models in the language. [20]

Naturalness The extent to which class and property names and definitions are expressed by conventional, typified terms and forms according to some general–purpose reference source. [22]

4.3.2 Learnability subcharacteristic. The capability of the metamodel to enable the user to learn its application (use, meaning, representation).

Clean Concepts The concepts we want to teach should be represented in the language in a way that directly reflects the theoretical model and is not compromised by secondary issues. [23]

4.3.3 Operability subcharacteristic. The capability of the metamodel to enable the user to operate and control it.

Expressive power What is it possible to express in the language? This is related to the domain. [20]

Simplicity No unnecessary complexity is included in the language [21]

Changeability Of importance in a dynamic world is the changeability of models when the domain or our understanding of it changes or the solution must evolve because of changing requirements. [17]

4.3.4 Attractiveness subcharacteristic. The capability of the metamodel to be attractive to the user.

Simplicity In order that a language be powerful and elegant it should not contain many concepts and it should not be defined with many words. [25]
Esthetics The extent that its graphical layout enables ease of understanding of the described system. [26]

4.4 Efficiency

Definition: The capability of the metamodel to provide appropriate performance, relative to the amount of resources used, under stated conditions.

4.4.1 Resource Utilization subcharacteristic. The capability of the metamodel to use appropriate amounts and types of resources when the metamodel performs its function under stated conditions.

Simplicity In order that a language be powerful and elegant it should not contain many concepts and it should not be defined with many words. [25]
Space Economy Concise models are produced. [21]

4.5 Maintainability

Definition: The capability of the metamodel to be modified.

Maintainability subcharacteristics are focused on assessing whether the metamodel allows changes or is ease to analyze. They see the metamodels as white boxes and assess more its internal properties than the external ones. Conversely, Portability subcharacteristics are aimed at assessing the possibility of exchanging metamodels or use them in other environments. This point of view treat metamodels as a whole or black boxes. Therefore, Modularity is a good candidate as quality attribute to assess maintainability.

4.5.1 Analyzability subcharacteristic. The capability of the metamodel to be diagnosed for deficiencies or causes of failures, or for the parts to be modified to be identified.

Modularity It is a general system concept. A continuum describing the degree to which a systems components may be separated and recombined. [27]
Modularity The extent that its parts are systematically structured and separated such that they can be understood in isolation. [19]

4.5.2 Changeability subcharacteristic. The capability of the metamodel to enable a specified modification to be implemented.

Changeability Of importance in a dynamic world is the changeability of models when the domain or our understanding of it changes or the solution must evolve because of changing requirements. [17]

Modularity It is a general system concept. A continuum describing the degree to which a systems components may be separated and recombined. [27]
Reversibility Implementation changes can be propagated into the model. [21]

4.5.3 Testability subcharacteristic. The capability of the metamodel to enable modified (meta)model to be validated.

Verifiability The extent to which the correctness of content of a metamodel (or ontology) is verifiable or provable in the context of a particular activity. [22]
Documentability The degree to which the metamodel is documented or self-explanatory.

4.6 Portability

Definition: The capability of the metamodel to be transferred from one environment to another.

Portability is assessed using the adaptability and replaceability subcharacteristic, having only a few attributes. As previously mentioned, the replaceability is related to the ability of the metamodel to be used in place of another, to be interchangeable as a whole.

4.6.1 Adaptability subcharacteristic. The capability of the metamodel to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.

Flexibility A quality model should be flexible because of the context dependency of software quality.
Scalability Large and small system can be modeled. [21]
Reducibility Meaning what features is provided by the language to deal with large and complex systems. This attribute is used in [20] citing Seltevit PhD thesis[28].

4.6.2 Replaceability subcharacteristic. The capability of the metamodel to be used in place of another specified metamodel for the same purpose in the same environment.

Interchangeability The capability of the metamodel to be exchanged or interchanged.
Specificity The extent to which the metamodel is specific rather than general.

Table 1 shows our proposed quality model QM4MM with a set of quality attributes for each selected subcharacteristic. The last column shows the context of the metamodel attribute, depending on whether it is considered as a model or as a language.

Characteristic	**Subcharacteristic**	**Attribute**	**Context**
Functionality	Suitability	Relevance	models
		Completeness	models
		Conciseness	languages models
		Consistency	languages models
		Detailedness	models
		Confinement	models
		Cohesiveness	models
	Accuracy	Precision	models
		Precision/completeness	models
		Correctness	languages models
		Consistency	languages
		Redundancy	languages models
		Accuracy/Validity	models
	Interoperability	Interoperability	models
Reliability	Maturity	Volatility	models
		Currency	models
		Authority	models
		Balance	models
Usability	Understandability	Comprehensibility	models
		Self-Descriptiveness	models
		Complexity	models
		Perceptibility	languages
		Naturalness	models
	Learnability	Changeability	models
		Clean Concepts	languages
	Operability	Expressive power	languages
		Simplicity	languages
		Changeability	models
	Attractiveness	Simplicity	languages
		Esthetics	models
Efficiency	Resource Utilization	Simplicity	languages
		Space Economy	languages
Maintainability	Analyzability	Modularity	models
	Changeability	Changeability	models
		Modularity	languages
		Reversibility	languages
	Testability	Verifiability	models
Portability	Adaptability	Flexibility	models
		Scalability	models
		Reducibility	languages
	Replaceability	Interchangeability	models
		Specificity	models

Table 1. Quality attributes for Metamodels

5 Discussion and Future Work

This position paper presents an ISO/IEC 9126 quality model particularization adapted to deal with the specific properties of Metamodels. A set of quality attributes has been identified through the analysis of several proposals for models and for programming language quality. We want to stress the importance of establishing the relationship between the QM4MM quality model that we are proposing here and these identified attributes, since this is something that most of the proposals and standards do not cover.

Our long-term objective is to assess metamodels by using an automatic process as well as its supporting tools. This is a very interesting goal to pursue, although we are well aware of the many problems involved. We think that it is necessary to be capable of measuring any artefact used in an engineering process, like MBE. So we are proposing a framework to help the measurement of metamodels in order to evaluate them. First of all we need agreements on specific and detailed quality models in order for these models to be applied. Our work provides a modest initial proposal in this direction. And secondly, to our way of thinking, the evaluation of the quality of metamodels has to be done through automatic and independent means.

There is a lot of work to be done in order to assess the quality of a metamodel. The next step, and the most difficult one, should be the definition of measures for quality attributes in the QM4MM model. Ideally measures should be objective and automatic but this is very difficult to achieve in the case of all the attributes that depend on the skills and expertise of a human user. In addition, once a measure is formally defined it should be validated i.e., we must verify that the given measure is actually measuring what it is meant to measure.

Acknowledgements. We would like to express our gratitude to the anonymous reviewers for their insightful comments and suggestions. This work has been partially supported by Spanish Research Projects TIN2008-03107, P07-TIC-03184 and PAC08-0024-5991.

References

1. Allilaire, F., Bézivin, J., Brunelière, H., Jouault, F.: Global Model Management in Eclipse GMT/AM3. In: Proc. of the Eclipse Technology eXchange (eTX) workshop at ECOOP 2006, Nantes, France (2006) `http://www.sciences.univ-nantes.fr/lina/atl/AMMAROOT/AM3/`.
2. Bézivin, J., Jouault, F., Rosenthal, P., Valduriez, P.: Modeling in the large and modeling in the small. In Aßmann, U., Aksit, M., Rensink, A., eds.: Model Driven Architecture, European MDA Workshops: Foundations and Applications (MDAFA 2003/2004). Volume 3599 of LNCS., Springer (2005) 33–46
3. ISO/IEC 9126: Software Engineering – Product Quality – Part 1: Quality model. International Standards Organization, Geneva, Switzerland. (2001)
4. Bézivin, J.: On the unification power of models. Journal on Software and Systems Modeling **4**(2) (2005) 171–188

5. Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture Practice and Promise. Addison Wesley (2003)
6. OMG: Model Driven Architecture (MDA). Object Management Group. (2001) OMG document ormsc/2001-07-01.
7. OMG: MDA Guide Version 1.0.1. Object Management Group. (2003) OMG document omg/2003-06-01.
8. Seidewitz, E.: What models mean. Software, IEEE **20**(5) (2003) 26 – 32
9. OMG: Meta Object Facility (MOF) Specification Version 1.4. Object Management Group. (2002) OMG document formal/02-04-03.
10. Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data & Knowledge Engineering **55** (2005) 243–276
11. Dromey, R.: A model for software product quality. Software Engineering, IEEE Transactions on **21**(2) (1995) 146–162
12. García, F., Bertoa, M.F., et al.: Towards a consistent terminology for software measurement (2005) Submitted for Publication.
13. ISO/IEC 15939: Software engineering – Software measurement process. International Standards Organization, Geneva, Switzerland. (2002)
14. McCall, J.A., Richards, P.K., Walters, G.F.: Factors in software quality, volume III: Preliminary handbook on software quality for an acquisition manager. Technical Report RADC-TR-77-369, vol. III, Hanscom AFB, MA 01731 (1977)
15. Boehm, B., Brown, J.R., Kaspar, J., et al.: Characteristics of Software Quality. North Holland, Amsterdam (1978)
16. Solheim, I., Neple, T.: Model Quality in the Context of Model-Driven Development. In: Proc. of MDEIS'06. (2006) 27–35
17. Mohagheghi, P., andTor Neple, V.D.: Towards a tool-supported quality model for model-driven engineering. In: Proc. of the 3rd Workshop on Quality in Modeling, Co-located with MoDELS 2008. (2008)
18. Moody, D.L., Sindre, G., Brasethvik, T., Sølvberg, A.: Evaluating the quality of information models: Empirical testing of a conceptual model quality framework. In: Proc. of ICSE'03, IEEE Computer Society (2003) 295–305
19. Lange, C.F.J., Chaudron, M.R.V.: Managing model quality in UML-based software development. In: Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice. (2005) 7–16
20. Krogstie, J.: Conceptual Modeling for Computerized Information Systems Support in Organizations. PhD thesis, University of Trondheim, Norway (1995)
21. Paige, R.F., Ostroff, J.S., Brooke, P.J.: Principles for modeling language design. Information and Software Technology **42** (2000) 665–675
22. Stvilia, B.: A model for ontology quality evaluation. First Monday [Online] **12**(12) (2007)
23. Kolling, M., Rosenberg, J.: Blue - a language for teaching object-oriented programming. In: Proceedings ACM SIGCSE Symposium. (1996) 190–194
24. Staikopoulos, A., Bordbar, B.: A comparative study of metamodel integration and interoperability in uml and web services. In: Model Driven Architecture Foundations and Applications, Springer-Verlag / Heidelberg (2005) 145–159
25. Wirth, N.: On the design of programming languages. Information Processing **74** (1974) 386–393
26. Purchase, H.C., Colpoys, L., amd D. Carrington, M.M., Britton., C.: Uml class diagram syntax: an empirical study of comprehension. In: In Australian symposium on Information visualisation. Volume 9. (2001) 113–120

27. Schilling, M.: Towards a general modular systems theory and its application to inter-firm product modularity. Academy of Management Review **25** (2000) 312–334
28. Seltveit, A.H.: Complexity Reduction in Information Systems Modelling. PhD thesis, University of Trondheim, Norway (1994)