

# Improving the Quality of UML Models in Practice

Christian F.J. Lange  
Eindhoven University of Technology  
P.O. Box 513  
5600 MB Eindhoven, The Netherlands  
C.F.J.Lange@tue.nl

## ABSTRACT

The importance of UML models in software engineering is increasing. Inherent to the UML is its lack of a formal semantics, its risk for inconsistency and completeness defects and the absence of modeling norms. These properties are sources for poor model quality and defects. To find out to which extent defects occur and what types of defects occur in practice we empirically investigate the state-of-the-practice of quality in UML models using a practitioners survey and a series of industrial case studies. Additionally we analyze the effects of defects in UML models experimentally. Based on this experiment we present an objective classification of UML defects which allows for prioritizing defects and thus allocate resources for defect removal. We aim at building a rule-set, metrics and visualization techniques to improve the quality of UML models during development. We propose a quality model that is specific for UML models. Finally, we propose modeling conventions, similar to coding conventions, to prevent for defects and to assure uniformity of modeling within an organization. We aim at empirically validating our techniques to provide pragmatic technology that can be transferred to industrial practice.

## Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tools and Techniques—*object-oriented design methods, computer-aided software engineering*

## General Terms

Experimentation, Design, Human Factors, Measurement

## Keywords

UML, Quality, Completeness, Consistency, Defect Detection

## 1. PROBLEM INTRODUCTION

The Unified Modeling Language (UML) [19] has gained significantly popularity in the recent years and is the de facto language for modeling software architecture and design. UML models are used at several stages during software engineering, such as the early analysis of the problem domain, documenting the architecture or specifying the detailed design of a system. The abstraction level of the used

models varies for the different stages. A UML model serves one or more specific purposes depending on the development stage and the stakeholders who are using the model. Purposes of UML models are amongst others: communication between people, documenting a system for use during maintenance, generation of test cases, prediction of the actual system's quality attributes and code generation. The goal of Model Driven Architecture (MDA) [18] is to transform (UML) models automatically into the implementation. Hence, we expect that the importance of UML models in software engineering will increase even more.

As described above the UML is a multi-purpose modeling language. It is a graphical notation that describes different views, such as the structural view or the behavioral view, of a system using a set of diagrams. In its current version 2.0 the UML offers 13 diagram types. A risk for inconsistency between views and incompleteness of views is inherent to multi-view modeling languages [5] such as the UML.

The UML lacks a formal semantics, i.e. the meaning of the elements of a UML model is not formally defined and may depend on the interpretation of individuals who are using the UML. Additionally the UML is only a notation and not a development method that prescribes how to model. Therefore the UML provides its users with a large degree of freedom on how to use it (which might be one of the reasons for the UML's popularity). The absence of norms on how to build, read and maintain UML models is a risk for the quality of UML models.

The purpose of this research is to identify quality problems due to the aforementioned risks in industrial UML models and to provide methods that aid in building UML models of a better quality than the current state-of-the-practice.

## 2. RELATED WORK

There exist several approaches that aim at solving the problems of inconsistency and incompleteness defects in UML models and the risk caused by the absence of a formal semantics. The Precise UML group [1] aims at defining a formal semantics for the UML. Küster [8] has a different approach: he defines a mapping from UML to a *semantic domain* which is a language with a formal semantics. Consistency defects can be identified and resolved in the semantic domain. Van der Straeten [21] proposes a similar approach: a mapping from UML to description logics to resolve consistency problems. These approaches are theoretically validated but the acceptance in practice is rather low.

The UML consistency workshop [9] and [16] investigate which types of defects occur in UML models. However these

studies investigate models from students or developers with a limited amount of experience, which is a threat to the validity of the presented results.

Inspection techniques (Fagan [4], Gilb [6]) are a means to detect flaws in software artefacts. Most existing research aims at inspecting source code, only little research has been done on UML inspection [10].

Some case tools such as ArgoUML provide the user with *critics* to pinpoint defects or problems in a UML model. However, the list of critics is relatively large even for small UML models. Therefore it is difficult for the user to decide where to start resolving defects. Additionally the large list of critics might seem unfeasible to solve for the user. Users need to be guided which defects are most severe and how specific defects relate to specific quality attributes.

### 3. RESEARCH HYPOTHESIS

In section 1 we described the risks in the use of the UML that are based on

- the fact that the UML is a multi-view notation,
- the absence of a formal semantics, and
- the large degree of freedom and the absence of norms provided by the UML.

The discussion of the related work in the previous section shows that there are already approaches to define a formal semantics for the UML or at least to map the UML to a formal domain. We are interested in the two remaining risks and formulate our research hypotheses as follows:

- **Hypothesis 1.** What is the extent of quality issues i.e. *defects* that are caused by potential risks such as the UML's variety of diagrams, the provided degree of freedom and the absence of norms on how to use the notation?
- **Hypothesis 2.** The defects in UML models often remain undetected and cause misinterpretation between different readers of the UML models.
- **Hypothesis 3.** Techniques can be proposed that aid in developing UML models of a higher quality than existing development techniques.
- **Hypothesis 4.** The developed techniques lead to a better quality of UML models, i.e. decrease the number of defects.

### 4. PROPOSED SOLUTIONS

In this section we describe our proposed solutions for each of the hypotheses, we describe the methods we use and we present initial results.

#### 4.1 Approach for Hypothesis 1

To investigate hypothesis 1 we are interested in the current state-of-the-practice of quality in UML models. The term quality is used for several different notions. We define quality for the scope of our research as *the level of absence of defects in a UML model caused by a conflict between different views or the absence of norms for using the UML*.

We use a literature survey as a starting point to identify the problems that are known for UML models. We investigate the quality of UML models by two approaches: first

we conduct a survey to obtain a subjective self-assessment of UML practitioners to find out how practitioners use the UML, how they perceive the quality of UML models in practice, and which problems they encounter. In a second step, we objectively investigate the quality of industrial UML models using case studies.

##### 4.1.1 Practitioners Survey

To obtain a subjective self-assessment of UML practitioners we have conducted a survey using an online-questionnaire and we obtained 80 useful responses from industrial practitioners. These results [15] show that the UML is used rather loosely in practice, model defects such as incompleteness and inconsistency cause miscommunication and misinterpretations, and there is the need for quality improvement.

##### 4.1.2 Industrial Case Studies

To find out whether the practitioners' subjective perceptions correctly reflect the quality of UML models, we conduct industrial case studies to assess the quality of UML models objectively. Therefore we need a method to detect defects. Based on the survey we collected a set of rules specifying defects. These rules are implemented in our *Software Architecture Analysis Tool (SAAT)* to find rule violations. We measured for example the number of classes that do not occur in a sequence diagram, and the number of messages in sequence diagrams that do not correspond to a method defined in a class diagram.

During the case studies the set of rules was extended and refined based on the gained experience.

**Results.** [11, 15] So far we have conducted 14 industrial case studies in different application domains. UML models were built using different case tools and for various of the aforementioned purposes. The size of the models ranges from a few classes to 700 classes. The results show that the UML models contain a relatively large amount of defects, the type and distribution of defects depends on the purpose of the model. Different people create UML models in different ways (leading to different types of defects).

#### 4.2 Approach for Hypothesis 2

Despite existing inspection methods [10], the results of our case studies show that in practice UML models still contain a large amount of defects. Defects in UML models do not necessarily imply a (negative) effect when using the models. To investigate hypothesis 2 we analyze the effect of defects with respect to detection and interpretation.

In a controlled experiment we ask the subjects of the experiment to either indicate how they would implement a (part of) a system given a UML model or to indicate whether they detect a defect. Half of the UML models used in the experiment contain a defect to allow comparison between the realistic situation (defected) and the perfect situation (no defect). Due to a limited amount of available time it is not feasible to conduct the described experiment for all defect types that we encountered in our investigations for hypothesis 1. Therefore we must select a subset of common defects for this experiment. However, the experiment should be replicated to investigate a larger amount of defect types.

**Results.** 110 master students and 48 practitioners participated in this experiment. We analyzed the data obtained from the answers using statistical hypothesis testing. The results [14] show that defects often remain undetected

and defected models cause a variety of different interpretations opposed to models without a defect. As the UML is a means for communication between people it is desirable that a model is interpreted the same way by different readers. Hence, there is the need for techniques to solve this problem.

### 4.3 Approach for Hypothesis 3

Finding techniques to improve UML model quality is rather a *design problem* than a *knowledge problem*. In the following we propose techniques to improve the aforementioned quality issues, we describe how we expect to realize the ideas and how they are expected to contribute to the improvement of UML model quality. The proposed techniques augment each other rather than they are (competing) alternatives.

#### 4.3.1 Rule Set

Removing or preventing defects requires identification of the defects. Therefore we propose a set of rules to specify the occurrence of a defect. In the discussion of hypothesis 1 we already stated that the rule set is based on existing literature, the results of the practitioners survey and that it will be extended and refined during our ongoing industrial case studies.

We use a relational metamodel which covers a subset of the UML metamodel as a basis to specify the rules. We focus on class diagram, sequence diagram, state chart and use case diagram. Besides using the rules for automated inspection in our SAAT tool we expect the rules being useful as guidance for manual inspection.

#### 4.3.2 Metrics at aggregate levels

The rules are a means to detect single instances of defects in a model. However, for purposes such as monitoring the progress of quality improvement, comparing the quality of several models or identifying low quality parts of a model, it is desirable to have information on an aggregate level. We aim to develop metrics that aggregate the information from metrics at the level of model element (e.g. class level), part of a model (e.g. package) or the entire model.

#### 4.3.3 Visualization

Metrics data and rule-checking results are often represented in tabular form. In these tables metrics values are attached to element names or element identifiers. An example row of such a table is

```
class LightController: 17.3, true, 7.0, 73.2.
```

Result tables for real-world systems often grow up to thousands of rows and are difficult to analyze. To understand such data, tables are not enough. Readers need to correlate their values with already familiar, understood model information, such as contained in the UML diagrams.

We propose an approach that combines metric data in an integrated view that is mapped on the existing UML diagrams. As a UML model represents a designer's *mental map* of a system, we expect the integrated view leads to a more intuitive understanding of the metrics data and a minimization of the cognitive disruption. Our proposed approach [12] enriches the existing UML diagrams with visualizations representing the metrics data. The proposed visualization techniques for this purpose include color, bar and line charts. These methods are proven to be useful in visualization in the area of geographical information systems

(GIS), where locational data (e.g. maps) is combined with non-locational data (e.g. population density).

**Ongoing Work.** We have implemented the proposed techniques in our visualization tool MetricView [20] and apply it to industrial UML models for validation.

#### 4.3.4 Quality Model

The aforementioned techniques aim at identifying defects in UML models and can be used to express the system's quality in terms of numbers for several types of defects. However, the techniques are not yet enough to assess the quality of a UML model in terms of quality attributes such as maintainability or understandability. This decomposed view of quality is necessary to evaluate a model's suitability for a specific purpose. A quality model is needed to relate the defect types and metrics to quality attributes.

We claim that existing quality models such as ISO9126 [7] or proposed by Boehm [3] are not suitable for UML quality evaluation, as they make no distinction between the quality of a software system and its description. Source code is in fact the implementation, therefore this distinction is not needed in source-code-based development. For model-based development the gap between description and the actual implementation is much larger. Therefore we propose an initial quality model [13] that makes the distinction between the system and its description (i.e. the model).

#### 4.3.5 Modeling Conventions

Our industrial case studies have shown that a large number of defects occur and that the UML is used in several different ways. In programming there exist conventions (or *coding standards*) to assure uniformity in source code. These conventions are often tailored for a specific community or organization. There are attempts [2] for establishing similar conventions for UML modeling to assure uniformity of modeling and prevent from creating defects, but these *modelling conventions* are mainly based on anecdotal evidence. We aim at improving the set of modeling conventions by using our empirically validated rule set as a basis.

### 4.4 Approach for Hypothesis 4

The purpose of hypothesis 4 is to validate the techniques that are designed for hypothesis 3. We use empirical techniques for the validation as detailed below.

**Rule-set, Metrics, Visualization, Quality Model:** To validate whether these techniques are useful in practice it has to be shown that they help improving the quality of UML models at a reasonable cost. We aim at doing this by case studies, where practitioners apply the techniques on real projects. The model quality after the techniques have been applied must be compared to a model from a similar context (skill of developers, project size,...) where the techniques were not applied. This could be a model from the same project before the techniques were applied or a similar model from our previous case studies (described for hypothesis 1). Additionally the effort needed to apply the techniques must be measured to investigate the cost of the techniques. However, this validation involves training of the developers and the commitment of industrial partners to participate in such a case study. Because such an intervention is associated with cost and risk for the project, it might be difficult to find industrial partners for participation in the case study. In that case we plan to conduct the case studies

with student projects. So far, the rule-set and the metrics have been applied to one smaller and one mid-size industrial case study and the number of defects has decreased. More case studies have to be conducted

**Modeling Conventions:** We currently conduct a controlled experiment with 108 students to investigate the usefulness of modeling conventions. The students' task is to model an information system according to a given textual description. We distribute the students over three treatment groups: (i) using modeling conventions, (ii) using modeling conventions and a rule-checking tool and (iii) no treatment. The set of modeling conventions is based on our previous work. After a six week design period we analyze the delivered models with respect to number of defects and an assessment using Lindland's quality framework [17]. The subjects collect the time for the design task to enable us to compare the development cost for the three treatment groups.

## 5. EXPECTED CONTRIBUTIONS

We summarize the expected contributions of our research and indicate where we already achieved results by giving the reference to our publications.

**State-of-the-practice of the quality in UML models,** assessed using a practitioners survey and industrial case studies [15, 11]. Most results in literature [16, 9] are based on subjects with little or no industrial experience and are therefore less generalizable.

**Effects of defects.** We investigate experimentally the effects of defects in UML models [14].

**Objective defect classification.** As a result of the previously mentioned experiment we obtain an objective classification of defects with respect to the *risk for not being detected* and the *risk for misinterpretation*. Other classifications of UML defects found in literature are subjective.

**Rule-Set.** To identify and find defects in UML models.

**Metrics.** To aggregate the rule-checking results on the level of model element or entire model.

**Visualization.** To improve metrics-based analysis of UML models by combining metrics results and UML diagrams in one view. [12, 20]

**Quality Model.** To relate defects and metrics to quality attributes [13]. A quality model is necessary to define sufficient quality goals, as it is economically not feasible to achieve '100% quality'. Existing quality models do not distinguish between described system and its description.

**Modeling Conventions.** To prevent for defects and to ensure uniformity amongst different modelers. In addition to existing literature we conduct a controlled experiment for validation.

## 6. REFERENCES

- [1] The Precise UML Group.  
<http://www.cs.york.ac.uk/puml/>.
- [2] S. W. Ambler. *The Elements of UML Style*. Ronin International, 2003.
- [3] B. Boehm. *Software Engineering Economics*. Prentice Hall, London, 1981.
- [4] M. E. Fagan. Advances in software inspections. *IEEE Tr. on Software Engineering*, 12(7):744–751, 1986.
- [5] A. C. W. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multi-perspective specifications. *IEEE Transactions on Software Engineering*, 20(8):569–578, August 1994.
- [6] T. Gilb and D. Graham. *Software Inspection*. Addison Wesley Publishing Co., 1993.
- [7] ISO/IEC FCD 9126-1.2. *Information Technology - Software Product Quality*, part i: quality model edition, 1998.
- [8] J. M. Küster. *Consistency Management of Object-Oriented Behavioral Models*. Phd thesis, Universität Paderborn, Paderborn, March 2004.
- [9] L. Kuzniarz, Z. Huzar, G. Reggio, J.-L. Sourrouille, and M. Staron. *2nd Workshop on Consistency Problems in UML-based Software Development at the UML2003*. Blekinge Institute of Technology, 2003.
- [10] O. Laitenberger and J.-M. DeBaud. An encompassing life-cycle centric survey of software inspection. *Journal of Systems and Software*, 2000.
- [11] C. F. J. Lange and M. R. V. Chaudron. An empirical assessment of completeness in UML designs. In *Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE'04)*, pages 111–121, 2004.
- [12] C. F. J. Lange and M. R. V. Chaudron. Combining metrics data and the structure of UML models using GIS visualization approaches. In *Proceedings of the IEEE International Conference on Information Technology 2005*, volume 2, pages 322–326, April 2005.
- [13] C. F. J. Lange and M. R. V. Chaudron. Managing model quality in UML-based software development. In *Preproceedings of STEP05*, 2005.
- [14] C. F. J. Lange and M. R. V. Chaudron. Effects of defects in UML models - an experimental investigation. In *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*. ACM, May 2006.
- [15] C. F. J. Lange, M. R. V. Chaudron, and J. Muskens. In practice: UML software architecture and design description. *IEEE Software*, 23(2), March 2006.
- [16] F. Leung and N. Bolloju. Analyzing the quality of domain models developed by novice systems analysts. In *Proceedings of the 38th Hawaii International Conference on Systems Sciences*. IEEE, 2005.
- [17] O. I. Lindland, G. Sindre, and A. Sølvberg. Understanding quality in conceptual modeling. *IEEE Software*, 11(2):42–49, March 1994.
- [18] Object Management Group. *MDA Guide, Version 1.0.1*, omg/03-06-01 edition, June 2003.
- [19] Object Management Group. *Unified Modeling Language, Adopted Final Specification, Version 2.0*, ptc/03-09-15 edition, December 2003.
- [20] M. A. Termeer, C. F. J. Lange, A. Telea, and M. R. V. Chaudron. Visual exploration of combined architectural and metric information. In *Proceedings of VISSOFT'05*. IEEE CS Press, 2005.
- [21] R. van der Straeten. *Inconsistentiebeheer in modelgebaseerde ontwikkeling*. Phd thesis, Vrije Universiteit Brussel, Brussel, Belgium, 2005.