3rd International Workshop on Quality in Modeling

Jean-Louis Sourrouille[1,2], Ludwik Kuzniarz[3], Lars Pareto[4], Parastoo Mohagheghi[4]
Miroslaw Staron[5],

[1]Univ Lyon; [2]INSA-Lyon, LIESP, F-69621, Villeurbanne, France
[3]Blekinge Institute of Technology, Ronneby, Sweden
[4]SINTEF ICT, Oslo, Norway
[5]IT University of Göteborg, Göteborg, Sweden

Abstract. Software quality management is widely researched within Model Driven Software Development (MDD), from both industry practices and academic research viewpoints. The goal of this workshop was to gather researchers and practitioners interested in the emerging issues of quality in the context of MDD. During the first part of the workshop, selected papers were presented and discussed. The second part was divided into two working sessions. The first session was devoted to the introduction of model quality into the software development process by drawing a parallel with quality of code. An invited practitioner introduced issues related to quality of code, followed by a guided discussion based on a list of predefined questions. The second session was dealing with future work and research interests of the participants.

1. Introduction

Quality is an important issue in software engineering, and stakeholders involved in the development of software systems definitely are aware of the impact of the quality of both development process and produced artifacts on the quality of final system. The recent introduction of Model Driven Software Development (MDD) raises new challenges related to ensuring proper quality of the software produced when using this approach. Software quality management within MDD is widely researched from multiple perspectives. Furthermore, in software engineering, the issues of model quality need to be approached from the viewpoints of both industry practices and academic research in order to arrive at sound and industrially applicable results.

The Quality in Modeling series of workshop (QiM) aim to provide a forum for presenting and discussing emerging issues related to software quality in MDD. The intended result is to increase consensus in understanding quality of models and issues that influence this quality. In the previous QiM workshop, a common quality model was established [1]. The intention of this year's workshop was to discuss model quality issues related to software development processes. Within "usual" software development, code quality seems to be under-exploited. However, all the concepts and theory about code quality have been widely described. Therefore, a special attention is to be paid to practical issues such as the introduction of model quality into the software development process in a convenient and accepted way.

2. Summary of the paper contributions

The presentation of papers consisted of two sessions, each one with three topics addressed by the papers [4]:

- Towards model quality
  - Definition of a measurement procedure to accurately quantify the size of software developed with a Model-Driven Development (MDD) [4],
  - Definition of a proactive and process-driven approach based on a meta-approach to be instantiated in every sub-process producing a UML model [5],
  - Description and implementation of a customized style guide for UML [6].
- Frameworks for model quality
  - Definition of an operational framework to address database schema quality through both global and analytical views of quality [7],
  - Empirical validation of measures for UML class diagrams through a meta-analysis study from five controlled experiments [8],
  - Definition of a metamodel to precisely define quality elements and their relationships in a quality model [9].

3. Introducing model quality in the development process

The starting point of the discussion was the introduction, presented by an industrial expert, Marc Rambert from *Kalistick*, on how code quality management is approached in an industrial context [2]. Obviously, there are common points between model quality and code quality, but the main reason for this discussion was the actual use of quality in practice. Despite the fact that theoretical and practical aspects of code quality are well-known, and that a number of market tools for assessing code quality exist for a long time, the quality of code is not used as much as it could be. To analyze deeply theoretical aspects of quality is not enough. We have certainly to deal with practical and/or human aspects for model quality to enforce its usage by software developing teams. The aim of this discussion was to take lessons from experience related to code quality to increase our chances to introduce successfully model quality into the software development process. First, the industrial expert recalled issues related to code quality:

- Getting control over the technical quality of a software development, which requires clear objectives closely linked with project management, and aid to teams to achieve project goals,
- Improving the technical quality by providing teams with a set of good practices, detecting gaps between actual code and objectives, and focusing on key issues for improvement,
- Providing a balance between additional costs and returns on investment. Quality is not an absolute value; it is related to objectives that depend on the needs of the target application. For instance the requirements for an embedded system are not the same than for a text editor.

The expert also showed examples of metrics related to code quality, how quality can be assessed and visually reported to assist teams in their daily work and taking decisions for release. After this introduction, a discussion guided by a set of questions sent to the participants before the workshop was carried on. For each of the questions the course was as follows: the question was recalled followed by the moderator's comments; then the question was answered from the code quality perspective, and finally an open discussion was carried out aiming to find a "common" answer from a model quality perspective. In the sequel, a *quality problem* means a quality level lower than the expected one for a criterion, for instance the number of dependency cycles is too large. A *quality goal* is the expected value for some aggregate of assessments and metrics. Quality goals depend on the application requirements and should be defined by stakeholders before starting the development. In the following we list the questions and a summary of the discussion on each question.

1. Not all metrics can be measured automatically. Some quality properties such as architecture design value can be assessed manually.
   Should we keep only automatic measurement?
   To what extent can we keep manual assessment?

   The overall answer is that it should be as automatic as possible depending on the required effort. If the semantics is to be considered then it should be manual. Even when the assessment is automatic, a manual interpretation is necessary in order to assess where we are and to undertake recovery actions.
2. Software quality is relative to the requirements of an application: there is no absolute level of quality, and "over-quality" is just non-quality.
   How to define/express the quality requirements of an application from the end user point of view?
   How to avoid overestimated quality requirements, which implies over-costs?

   The schema Fig. 1 expresses a basic principle: increasing the development costs to reduce maintenance costs is limited by the increase in total costs. To define the suitable quality level, the industrial expert draws a quality profile (Fig. 2) from replies to a questionnaire directly related to project requirements but only indirectly related to quality. Stakeholders have different requirements and they should reach an agreement. The quality requirements for models seem to be closer to system quality requirements than quality requirements for code, which might result in some differences in the description.

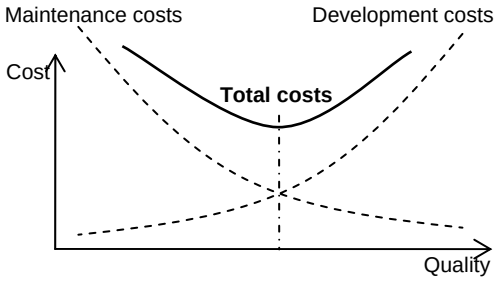

Fig. 1. Optimal cost

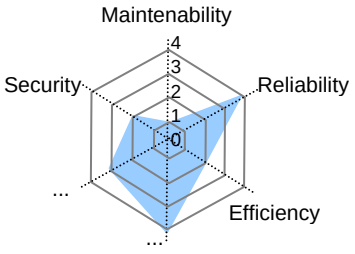

Fig. 2. Quality profile

3. When actual quality is different from the expected one, some quality goals may have a greater importance for the end user.
   How to express the importance of application's quality goals?

   In the industrial context, a dashboard shows the gap between actual and expected profiles, and improvement plans are proposed to reduce the gaps. All quality aspects have not the same importance for the stakeholders, leading to two kinds of conflicts: (i) internal, between quality attributes, for instance to increase maintainability may reduce performance; (ii) external, the same attribute being more important for one stakeholder that for another one.

4. Continuous quality assessment would be a burden for the developer and would probably result in some reject. On the other hand, to find quality problems as early as possible is a well-known need in software engineering.
   What could be the frequency or suitable moment for quality assessment?

   The frequency may depend on cost of checks. In practice, weekly assessment is enough for code. When no gap between actual values and goals is detected, and when the trend of evolution does not deviate, the human cost is quite null; otherwise a quick reaction is possible. Regarding models, instead of regular checks, quality gates or check points could be introduced in the development process to ensure quality levels.

5. When quality problems are detected, there are numerous possible actions. These actions generally depend on the application status: to correct existing software might be risky (non-regression) and of limited interest, while to correct software under construction will have a better return.
   Should we help the user to define the actions to achieve?
   Should we prioritize the quality problems to deal with?
   Finally, should the developer deal with all the quality problems?

   Another factor affects the decision: how fast the developer can find the fault? Actions to undertake depend on project management decisions, and to help managers to make their decisions is necessary.

6. There is a link between metrics and quality goals.
   To what extent does the developer need to know this relationship?
   When this relationship is known, is it acceptable for a developer to limit quality problems to stay just below thresholds that triggers quality problems?

   Example: if the metric for readability is that a sequence diagram includes less than 20 connected elements, the quality goal can specify that at most 5 sequence diagrams may break the rule. Below 5 rule violations, no quality problem is detected, therefore the developer is not warned and the situation is acceptable.

7. To find a way to deal with a quality problem is not always easy.
   Do we need to aid correcting quality problems?
   Do we need to show the place of the problem in the model (see tool below)?

   Tools detect the symptoms but not the causes: the tool may detect that there are too many dependencies, which does not explain why and how to reduce them. Similarly, patterns and anti-patterns explain why and where the problem is, but not how to correct it. Since errors may come from a combination of several causes, explanations should be precise. Regarding model quality, to help developers

implies embedding quality checks in a tool, which is a great difference with code.

8. Developer training could be an important aspect to get better results.
   What kind of training would be useful: before development? When errors occur? Should developers training focus on actual metrics? Or only on principles?

   The experience shows that teaching rules has no interest. The best way seems to train on principles, and when a problem occurs to train on this problem. Moreover, training professionals has psychological issues to take into account.
9. Many stakeholders will look at quality results, each one with a different point of view.
   How to present the results in a suitable way for each stakeholder?
   What are the suitable ways for developers?

   A dashboard for each category of stakeholder is useful in industrial context.
10. Code quality assessment does not depend on a tool.
    Is it desirable to assess model quality independently of any tool (for instance by analyzing XMI files)? Or should the IDE tool include quality assessment?

    Participants mainly think that tools should include quality assessment. This ideal solution requires customizing each tool.

Comments. Due to lack of time, but also because all the answers are not actually known, there is no precise answer for each question. The list was limited to 10 items, aiming to tackle different practical aspects about model quality. Anyway, answers will be needed to bring model quality into play. These aspects complement the quality model discussed during the previous workshop.

4. Working session on a road map for further research

The second session of the working part was devoted to development of a common research roadmap, by the following procedure. 1) The presentations were analyzed using the unified quality model of the QiM'07 workshop [10]; analysis result was a graph associating presentations to distinct qualities of concern. 2) This graph was presented for review: contributing sites were asked to check that associated qualities were appropriate and complete with respect to the research pursued at the site. 3) The graph was revised collectively. The outcome is given in the Table below. The topmost part shows the home base of the research groups behind the QiM'08 contributions. The lowermost part shows the white spots with respect to the common quality model: these qualities have no immediate connection to the presented talks, and should be fertile ground for research.

Acknowledgement. We would like to thank Marc Rambert, from Kalistick, provider of the 1rst SaaS platform for code quality, who kindly agreed to prepare the invited talk for the working session, and to moderate the discussion based on code quality.

		P1 Valencia	P2 Genova	P3 Lyon	P4 Namur	P5 Ciadud	P6 Oslo	P7 Ville-Urbaine
Model Quality	Correctness		x	x			x	
	Consistency		x	x			x	
	Completeness		x				x	
	Conciseness		x				x	
	Maintainability				x	x	x	x
	Complexity					x		
	Size	x				x		
	Understandability				x	x	x	
	Transferability							x
Infrastructure	Automatic	x		x				
Process Quality	Rigourously defined	x	x	x			x	
	Automateable							x
	Easily configureable							x
Project Quality	Rigourously managed							x
	Automatically Measurable							x
	Modeling Guidelines			x				

White Spots	**Model Quality**	Navigability, Traceability, Measurable, Stable, Precision Improving,, Detailedness
	Infrastructure Quality	Ubiquitous, Updateable, Seclusive, Flexible, Categorial, Archival, Cohesive, Efficient
	Process Quality	Predictability, Reuse of good practices, Roll backing, Quality Assurance for models, Explicit about modeling purpose, Voluntary, Incentive, Regulatory, Process Support, Measurability, Effectiveness, Productivity
	Project Quality	Qualified staff**,** Skill, Experience, Tools

References

1 Ludwik Kuzniarz, Lars Pareto, Jean-Louis Sourrouille, Miroslaw Staron, "The Third Workshop on Quality in Modeling", LNCS 5002, Springer, 2008, pp.271-274

2 Nicolas Blanc-dit-Grenadier, Marc Rambert, Jean-Louis Sourrouille, Régis Aubry, "Toward a real integration of quality in software development", ICSSEA'08, to appear, 2008

3 Jean-Louis Sourrouille, Miroslaw Staron, Ludwik Kuzniarz, ed, Proc. of the 3rd Workshop on Quality in Modeling, IT University of Göteborg RR 2008:02, ISSN 1654-4870, pp1-88

4 Beatriz Maron, Nelly Condori-Fernandez, Oscar PastorJean, “Design of a Functional Size Measurement Procedure for a Model-Driven Software Development Method”, in [3]

5 Gianna Reggio, Egidio Astesiano, Filippo Ricca, “A proactive process-driven approach in the quest for high quality UML models”, in [3]

6 Mohammed Hindawi, Lionel Morel, Régis Aubry, Jean-Louis Sourrouille, “Description and Implementation of a Style Guide for UML”, in [3]

7 Jonathan Lemaitre, , Jean-Luc Hainaut, “A Combined Global-Analytical Quality Framework for Data Models”, in [3]

8 M. Esperanza Manso, Jose A. Cruz-Lemus, Marcela Genero, Mario Piattini, “Empirical Validation of Measures for UML Class Diagrams: A Meta-Analysis Study”, in [3]

9 Parastoo Mohagheghi, Vegard Dehlen, Tor Neple, “Towards a Tool-Supported Quality Model for Model-Driven Engineering”, in [3]

10 L. Pareto, C. Lange, P. Mohagheghi, V. Dehlen, M. Staron, C. Bouhours, F. Weil, C. Bastarrica, S. Rivas, P.O. Rossel, L. Kuzniarz, Towards a Unified Quality Model for Models, 2nd QiM, RR in Soft. Eng. and Management 2008:01, Gothenburg University