3MQM: A Maturity Model for Model-based Quality Management

Michael Kläs, Adam Trendowicz, Jens Heidrich, Ove Armbrust

Fraunhofer Institute for Experimental Software Engineering,
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{michael.klaes, adam.trendowicz, jens.heidrich, ove.armbrust}@iese.fraunhofer.de

Abstract. Managing product quality during the development, operation, and maintenance of software-intensive systems is a challenging task. Although many organizations use quality models to define, control, measure, or improve different quality aspects of their development artifacts, only very little guidance is available on how to assess the maturity of an organization with respect to model-based quality management (MQM). Thus, it is difficult for an organization to improve its usage of quality models. Existing process maturity models such as CMMI or SPICE are too generic to provide specific guidance for the improvement of MQM. This paper presents a Maturity Model for Model-based Quality Management (3MQM) as a first step towards better support for determining the maturity of current MQM and for identifying improvement possibilities with respect to MQM. The model can be applied to provide an integrated overview of the maturity of an organization with respect to the usage of quality models.

Keywords: Software Quality, Quality Management, Quality Model, Maturity Model, Process Assessment Method.

1 Introduction & Related Work

Nowadays, software practitioners are faced with the challenging task of managing the quality of software-intensive systems in a predictable and controllable way. According to a study [9] conducted in the context of the German QuaMoCo project [8], many organizations rely on quality models for specifying, controlling, measuring, or improving different quality aspects of their development artifacts. The maturity of their quality management process is crucial for coming up with appropriate models that actually work in practice. This includes, for instance, the definition of quality models, their adaptation for specific projects, their maintenance over time, and their introduction into the respective organization. ISO/IEC 9126 [1] defines a quality model for software products that is widely used as a basis for developing company-specific quality models. Recently, the ISO started developing the ISO/IEC 25000 series [2] of standards to replace ISO/IEC 9126 and the corresponding quality evaluation standard ISO/IEC 14598 [3]. There are many aspects involved in setting up a successful model-based quality management system (MQM), some of which are addressed by these standards. However, very little guidance is available on how to assess the capability of an organization regarding all those aspects and evaluate its maturity with respect to MQM. Thus, it is difficult for an organization to establish a clear path for improving and optimizing its usage of quality models.

With respect to software processes in general, however, a number of improvement approaches exist. Model-based approaches such as ISO/IEC 15504 (SPICE) [4] or CMMI [5] compare an organization's processes or methods to a reference model containing well-known and widely accepted best practices. Typically, the elements of such a reference model are associated with different levels that are supposed to reflect an organization's different capability or maturity levels. Therefore, this type of model is called *capability* or *maturity model.* An assessment or appraisal determines to which degree an organization complies with the demands of the respective model and is typically performed by comparing the actually used processes against the requirements for these processes as stated in the reference model. Such assessments may serve to evaluate an organization with respect to its process maturity, or to identify improvement options for an organization's processes.

One well-known capability model is described in ISO/IEC 15504, often referred to as SPICE. It defines requirements for performing process assessments and provides an exemplar assessment model that complies with these requirements. In fact, SPICE is often used synonymously with this exemplar model, which we will also do throughout the remainder of this paper. SPICE defines process areas like *project management* or *software construction,* which encapsulate the most important activities *(base practices)* and expected outcomes for each process area. In addition to that, *generic practices* are instantiated for every process area, e.g., concerning management of the respective activities or unifying activities across the whole organization. A process area is evaluated on a scale from 0 to 5, reflecting the organization's capability with respect to, for example, project management.

The CMMI model provides similar features and adds an organizational maturity level, which is assumed to reflect the respective organization's process maturity in a single value. In order to reach a certain maturity level, an organization must hence reach minimum capability levels for defined process areas. Capability/maturity models such as SPICE and CMMI have continuously become more popular, and their concepts of which are hence being transferred to other, specialized models, such as TPI for test process improvement [6]. However, for model-based quality management, no such transfer exists.

The basic idea illustrated in this paper is to make use of the standard framework for process assessments as defined in ISO/IEC 15504 for creating a comprehensive capability model for selected MQM process areas and an overall maturity model for model-based quality management (3MQM) that introduces relevant quality practices for MQM process areas and capability levels. 3MQM will support companies in systematically improving their capability regarding selected MQM process areas and their overall MQM maturity.

The paper is structured as follows: Section 2 introduces the basic ideas for our capability and maturity model for MQM. Section 3 summarizes the basic approach and discusses future work in this area.

2 Maturity Model for Model-Based Quality Management

The model we propose defines process areas (PAs), which are used to evaluate an organization's capability with respect to MQM and to guide improvement activities. Capability level (CL) definitions based on quality practices (QPs) support this evaluation. We further propose a mapping of CL profiles to maturity levels (ML) describing an organization's overall maturity with respect to MQM.

2.1 Process Areas

Since we propose an MQM maturity model for software organizations, we focus in our proposal on primary software-related life cycle processes, including software verification & validation. In alignment to the SPICE standard, the engineering PAs are separated into *software requirements analysis*, *design*, and *construction*. We did not include software integration as a distinct area because usually no explicit MQM takes place in this PA. For the sake of simplicity, we further combined the SPICE PAs of software testing, joint reviews, verification, and validation into the single PA *verification and validation*. Our selection of PAs is currently driven by the artifacts to which MQM is applied. That is to say, we address the processes producing these artifacts. PAs such as product evaluation or quality assurance that benefit from or are in charge of specific MQM aspects, but do not produce such artifacts, are not included. In order to underline and assure the alignment of MQM in specific PAs with the global quality goals addressing the product as a whole, we included an additional PA named *global quality management*. In order to illustrate the need of such a PA, we can think, for example, about a global quality goal like "fast system responds time for queries" that may be broken down and supported by corresponding goals in the design PA ("high performance architecture") and the construction PA ("efficient algorithms"). Fig. 1 displays the proposed process areas. Although the initial MQM maturity model focuses on primary life cycle processes, it may also be applied to software during operation and maintenance phases. Operation is addressed by the global quality management PA, which focuses on the software in operation. The use of quality models in maintenance and development is similar; therefore, the developed PAs are expected to also be applicable to product maintenance phases.

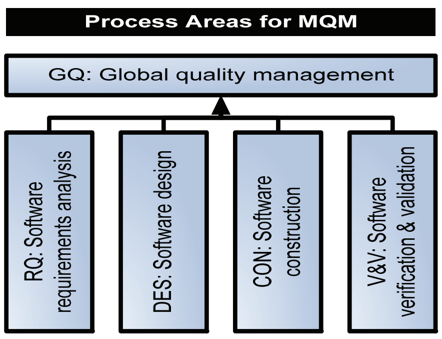
Process Areas for MQM
GQ: Global quality management
RQ: Software requirements analysis
DES: Software design
CON: Software construction
V&V: Software verification & validation

Fig. 1. Proposed process areas for model-based quality management

2.2 Capability Levels

We propose choosing the CLs based on a set of typical application purposes for quality models. Such purposes may be to specify, measure and monitor, evaluate and control, manage, or measure and predict the quality of a software-intensive product [7]. Since these purposes form a partial order with respect to an organization's quality modeling capabilities, they can be utilized to define corresponding CLs:

- CL0: *Incomplete* – Concepts of MQM are not implemented, or fail to achieve their purposes.
- CL1: *Specifying* – On a project level, a common understanding of quality is specified in the specific process area and linked to higher level goals. Guidelines describe strategies for archiving this quality.
- CL2: *Monitoring* – The understanding of quality is operationalized and measured in the process area at appropriate points in time. The trends in measurement data are analyzed.
- CL3: *Controlling* – A common understanding of quality is defined in the process area on an organizational level. Collected measurement data are evaluated in order to control quality.
- CL4: *Managing* – Impacts on quality are known for the specific area process and are actively managed in case of measured quality deviations.
- CL5: *Predicting* – The quality in the specific process area is predictable because the quantitative impact of the most important influence factors on quality is known. Therefore, quality in the specific process area can be planned quantitatively.

These capability levels define a natural order of improvement: the more sophisticated the purpose, the higher the associated CL. For instance, we cannot measure and monitor quality without first specifying it; we cannot control quality (i.e., provide targets or thresholds) without quantifying it; and so on.

2.3 Maturity Levels

In order to evaluate an organization's overall maturity with respect to model-based quality management, we propose a mapping of the CLs to an ML. The major benefit of such a mapping is that it shows directions for improvement during the introduction and refinement of MQM in a company. Similar to CMMI, we propose five MLs:

- ML1: *Initial* – Every organization is on ML 1.
- ML2: *Monitoring* – In the organization, projects are actively monitoring product quality in the process areas software requirements analysis and software construction.
- ML3: *Controlling* – The organization is consistently controlling quality for software requirements analysis, software construction, and the complete product (global quality).

- ML4: *Managing* – The organization is consistently controlling quality for the complete product (global quality) and for every process area for MQM, and, in addition, considers influence factors for actively managing the most important process areas
- ML5: *Predicting* – The organization is consistently managing quality for the complete product (global quality) and for every process area, and ,in addition, quantitatively controls influence factors for the most important process areas.

Table 1 depicts the construction of organizational MLs based on the CLs: In order to reach ML2, an organization must achieve CL2 at least for the process areas *software requirements analysis* and *software construction*. Likewise, in order to reach ML3, an organization must achieve CL3 at least for the process areas *software requirements analysis, software construction,* and *global quality.*

Table 1. Maturity level mapping to process area capability levels

		Process Area				
		Requirements	Design	Construction	V&V	Global Quality
Maturity Level	1	-	-	-	-	-
	2	2	-	2	-	2
	3	3	-	3	-	3
	4	3*	3*	3*	3*	3*
	5	4^+	4^+	4^+	4^+	4^+

*: All process areas must achieve at least CL 3, plus at least one process area must reach CL 4.
$^+$: All process areas must achieve at least CL 4, plus at least one process area must reach CL 5.

2.4 Example: Software Construction Quality Practices

In this paragraph, we provide an example definition of quality practices (QP) for MQM capability levels for the *software construction* process area (i.e., we focus on the quality of the software code). In this initial version of our maturity model, we do not (yet) distinguish base and generic practices as, for example, SPICE does. Please note that the term "establish" within a QP implies the existence of four sub-practices, which state that the process outputs must be (1) *defined*, (2) *introduced*, i.e., communicated to all relevant stakeholders, (3) *applied*, i.e., application should be ensured by appropriate control mechanisms and trainings, and (4) *maintained*, i.e., necessary changes must be implemented and their effects on other process outputs must be traced and resolved. We will not repeat these sub-practices for the remainder of this paper.

1 – Specifying

QP1.1: Establish quality goals with respect to software code.
Code quality goals are goals related to the quality of software code, for example, low complexity or little code cloning.

QP1.2: Establish an integrated code quality model that is derived from code quality goals by qualitatively refining them (horizontal goal alignment).
Integrated means that all quality goals are considered in one model. However, the model may be comprised of several sub-models.

QP1.3: Establish links between code quality goals and global quality goals (vertical goal alignment).
Global quality goals can be goals for the overall software-intensive system.

QP1.4: Establish coding guidelines based on the code quality model.
Coding guidelines provide constructive strategies for reaching the code quality goals.

2 – Monitoring

QP2.1: Establish measures for quantifying quality goals defined in the code quality model.
Goal-oriented measurement assures that collected measurement data contributes to at least one quality goal.

3 – Controlling

QP3.1: Establish quantitative evaluation criteria for each code quality goal.
An evaluation criterion consists of a defined procedure that maps measurement values to an evaluation scale. Usually, for this purpose a baseline or threshold values are used to evaluate goal achievement.

QP3.2: Establish rules for aggregating evaluation results.
Aggregation rules may allow for defining an overall evaluation for code quality comprising all related aspects and goals.

QP3.3: Establish an organization-wide code quality model.
All code quality goals across the organization should be integrated into a unified model.

QP3.4: Establish a standard method for adapting organization-wide code quality models.
An adaptation method is required for adapting the organizational code quality model to the particular context, e.g., a particular software development project.

4 – Managing

QP4.1: Establish a qualitative link between variation factors and their influences on code quality goals.
Variation factors include environmental characteristics (e.g., related to project context, resources, personnel, or processes) that have an impact on the achievement of the code quality goals. Qualitative refers to the type of impact, that is, whether it has a positive or negative impact on achieving a specific quality goal.

5 – Predicting

QP5.1: Establish a quantitative link between variation factors and the quality goals they have an impact on.
Quantitative relationship refers to a functional relationship between quality goals and the values of related variation and context factors, for instance, related to project context, resources, personnel, or processes.

3 Summary and Future Work

Improving the maturity of an organization regarding its quality management activities is crucial for staying competitive in a highly dynamic business environment in which product quality is one central differentiator between competitors. Model-based quality management (MQM) is becoming more and more popular because it supports an organization in making product quality measureable and therewith controllable. The intention of the maturity model for model-based quality management (3MQM) proposed in this paper is to help organizations in systematically building up a clear path towards improving their MQM capability of selected process areas and their overall MQM maturity.

The basic idea of 3MQM is to create an ISO/IEC 15504-compatible model for determining the capability of MQM process areas, and to define a corresponding maturity model. Initially, we distinguish four process areas addressing different phases of the software development life cycle and one process area integrating quality management on a global level, i.e., encompassing the complete product. A number of quality practices are used to describe capability levels for the process areas, covering typical application purposes of quality models. Based on the capability levels, an organization-specific maturity level can be determined. The capability levels together with the maturity level show an organization how and where to make systematic improvements with respect to model-based quality management.

Our next step will be to complete the 3MQM approach. This especially includes a more detailed definition of all quality practices for all capability levels and process areas. It is also important to note that the selection of process areas is probably not final and may be extended in the future. One possible candidate from SPICE's supporting life cycle processes that may be included in the final model is, for instance, "documentation". Furthermore, the approach will be evaluated in industrial case studies as part of the public German research project QuaMoCo. The goal of the QuaMoCo project is to define an operationalized tailor-made product quality model for different domains and organizations. The evaluation will monitor the capability levels of selected process areas and the overall MQM maturity of an organization after different stages of the QuaMoCo quality model have been introduced. Another interesting research topic will be to explicitly align the quality goals within quality models with an organization's overall business goals and strategies.

Acknowledgments

Parts of this work have been funded by the BMBF project "QuaMoCo – Software-Qualität: Flexible Modellierung und integriertes Controlling" (grant 01 IS 08 023 C).

References

1. International Organization for Standardization, "ISO/IEC 9126 International Standard, Software engineering – Product quality, Part 1: Quality model," ISO/IEC, Geneva, Switzerland 2001.
2. International Organization for Standardization, "ISO/IEC 25000 Software product Quality Requirements and Evaluation (SQuaRE): Guide to SQuaRE," ISO/IEC, Geneva, Switzerland 2005
3. International Organization for Standardization, "ISO/IEC 14598 International Standard, Software product evaluation - Part 1: General overview", ISO/IEC, Geneva, Switzerland 1999.
4. International Organization for Standardization, "ISO/IEC 15504:2004, Information technology - Process assessment," ISO/IEC, Geneva, Switzerland 2006.
5. "CMMI for Development, Version 1.2", CMU/SEI-2006-TR-008, Carnegie Mellon University, 2006.
6. Sogeti, "Test Process Improvement," http://www.sogeti.de/tpi-test-process-improvement.html, last visited 2009-11-11.
7. Klaes, M., Heidrich, J., Muench, J., Trendowicz, A., "CQML Scheme: A Classification Scheme for Comprehensive Quality Model Landscapes," Proceedings of the 35th EUROMICRO Conference Software Engineering and Advanced Applications, IEEE Computer Society, pp. 243-250, 2009.
8. "Software-Qualität: Flexible Modellierung und integriertes Controlling (QuaMoCo)", Website: http://www.quamoco.de, last visited 2009-11-14.
9. Wagner, S., Lochmann, K., Winter, S., Goeb, A., Klaes, M., "Quality Models in Practice. A Preliminary Analysis," Proceedings of 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, 2009.