Software Quality Metamodel for Requirement, Evaluation and Assessment

Iwona Dubielewicz*
Iwona.Dubielewicz@pwr.wroc.pl

Bogumiła Hnatkowska*,
Bogumila.Hnatkowska@pwr.wroc.pl

Zbigniew Huzar*
Zbigniew.Huzar@pwr.wroc.pl

Lech Tuzinkiewicz*
Lech.Tuzinkiewicz@pwr.wroc.pl

Abstract: Adequate quality of software is often crucial. Quality assurance as inseparable aspect of the software lifecycle demands for a precise quality model. The paper proposes a quality metamodel, which may be used to derive specific quality models necessary in different activities of software development. Quality requirement, evaluation and assessment models are considered in the paper. The metamodel expressed in the UML is based on ISO quality standards.

Key Words: Software quality, quality requirement, quality evaluation, quality assessment

1 Introduction

Developing high quality software is of prime importance. Comprehensive and possibly precise software specification is the first necessary condition to ensure adequate quality. Next, the quality should be controlled within the software development process, and eventually the quality of final software product should be evaluated and assessed.

Software quality is a subject of many software engineering recommendations, e.g., [11], and especially of ISO standard series [1] – [8].

The aim of the paper is to give precise quality metamodel for software product, called SQMREA. Instances of this metamodel are quality models that are to be applied in different stages of software development. The metamodel is based on ISO standards and is expressed in UML.

The reason of SQMREA is to enable assessment of quality levels of different artifacts produced during software development, i.e. code, models, specifications, and the resulting software product. The assessment can be done from different perspectives, i.e. from end-user point of view (external perspective), and from developer point of view (internal perspective) and can be partial, e.g. a single characteristic of the software product can be evaluated.

* Institute of Applied Informatics, Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

The paper is organized as follows. Section 2 gives an overview of ISO quality model. In Section 3 the software quality metamodel SQMREA is presented. Section 4, on the base of simple examples, explains how a concrete quality models may be derived from the SQMREA metamodel, and how the models may be used for evaluation, and assessment of software product. Finally, Section 5 discusses presented proposals and compares them to current literature.

2 Overview of quality models

Plato (427-347 BC) explained the notion of quality as an extent of perfection. The notion may relate to a product or to a production process. Now, the ISO standard [7] defines quality of a product as a *totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs.* This definition is different from that given in [9], where quality refers to the satisfaction of requirements.

In the paper, we concentrate on the quality of a software product. The product may be defined as a set of different artifacts, e.g., computer programs, procedures, documentation, data etc.

Comprehensive specification of software product quality is a starting point to quality evaluation. The specification is based on user needs, usually informally expressed. The needs serve as the basis for the formulation of *requirements.* A requirement is defined as *a condition or feature required by user to solve a problem or to reach specific goal* [11]. The requirements are expressed quantitatively through referring to values of software *attributes*, i.e. *measurable physical or abstract properties of the product.*

The quality is categorized into three interrelated perspectives reflecting different stages in the software lifecycle:

- *quality in use* represents a user point of view, and is derived from user needs,
- *external quality* represents a software developer point of view, and is based on properties of a whole software product,
- *internal quality* also represents a software developer point of view, but is based on properties of software product components or intermediate software products.

According to the perspectives, there are three quality models. Quality in use model is used for software product validation. External and internal quality models are used for software product verification. Internal quality model is used in the software development process, to assess intermediate products, while external quality model is used to assess a final product.

A quality model consists of characteristics and relationships between them. Standard [1] defines the following characteristics for internal and external quality models: *functionality*, *reliability*, *usability*, *efficiency*, *maintainability* and *portability*. These characteristics may be subdivided into multiple levels of sub-characteristics. For example, for usability, there are the following normative sub-characteristics: *understandability*, *learnability*, *operability*, *efficiency*, *attractiveness* and *usability compliance.*

Eventually, characteristics or sub-characteristics refer to software quality attributes. Some attributes may contribute to more than one characteristic or sub-characteristic. There are two kinds of attributes: structural and behavioral ones. Structural attributes are mainly used for internal quality model while behavioral attributes are mainly used for external quality model. The correlation between attributes used for internal and external models is usually determined by experience and depends on the particular context in which the software is used.

For an agreed sub-characteristic a set of metrics as functions on attributes is given. Acceptable ranges of the metrics specify recommended values of attributes.

In initial phases of software development user quality needs are specified. On the base of them characteristics and metrics for internal and external quality models are selected. It is recommended that metrics for internal quality model should be selected in such a way that they enable prediction of values of metrics selected for external quality model. The problem how to select a set of quality characteristics, and especially metrics, is outside the scope of the paper.

3 SQMREA Metamodel

Our proposal for software quality metamodel for requirement, evaluation, and assessment is shown on Fig. 1. The metamodel is presented as a UML class diagram with a set of OCL constraints. The elements of the metamodel are divided into three parts placing in the center, and left and right sides of the diagram. The central part relates to a quality model, the right side relates to software quality requirement, and the left side relates to a quality of software product evaluation and assessment.

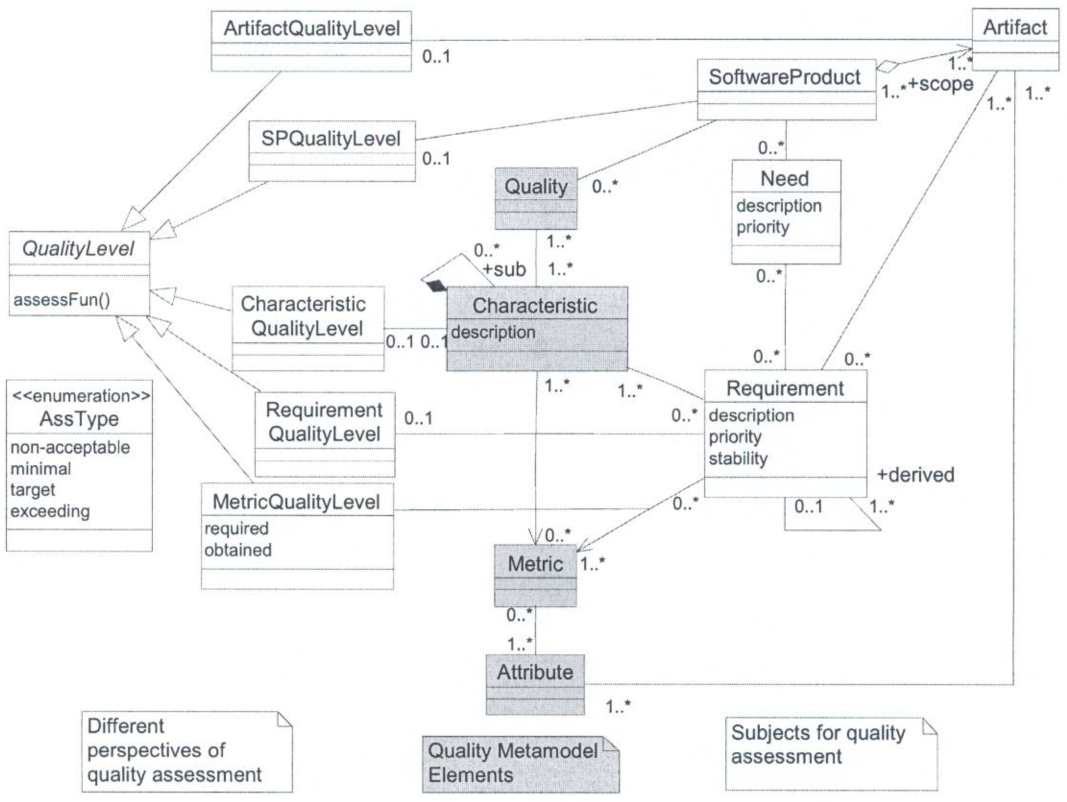

Fig. 1. SQMREA Metamodel

The central part of the diagram consists of *Quality, Characteristic, Metric,* and *Attribute* metaclasses. The metaclasses and respective associations represent notions described in Section 2.

Metrics are defined only for leaves of characteristic's tree, which is expressed by the OCL expression no 1:

```
context Characteristic inv:         // no 1
not self.sub->isEmpty() implies self.metric->isEmpty()
```

There are two additional associations related to *Metric* metaclass presented in Fig. 2, but not presented in Fig. 1. They say that one metric can be defined in terms of others (role: *reffered to*), and that one metric can be traced from others (role: *elicitated*).

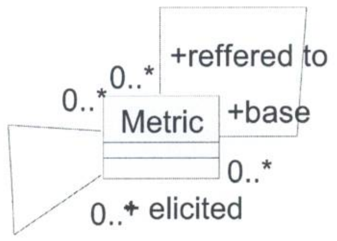

Fig. 2. Relationships between metrics

The middle part of SQMREA metamodel may be instantiated for a given purpose. Once instantiated may serve as the base for many projects.

The right side part of the diagram consists of *SoftwareProduct*, *Artefact*, *Need*, and *Requirement* metaclasses. These elements are subjects for quality assessment, and at the same time they select characteristics and metrics for a quality model. Their instances are specific for a given software product.

Each requirement is associated to at least one artifact of the software product, and at least one characteristic of quality model. A requirement can be decomposed into other requirements. The leaves of the requirement's tree are associated with metrics by *MetricQualityLevel* association class. These metrics are chosen from among the metrics defining the characteristic the requirement belongs to, which is expressed by the OCL expression no 2:

```
context Characteristic inv:  // no 2
self.requirement.metric->forAll(m | self.metric->includes(m))
```

The elements of the left side part of the metamodel are used for quality evaluation and assessment. The root element of that part is abstract *QualityLevel* metaclass, which can be specialized for different elements:

- a requirement (*RequirementQualityLevel*),
- a characteristic (*CharacteristicQualityLevel*),
- an artifact (*ArtifactQualityLevel*),
- a whole software product (*SPQualityLevel*).

Each specialization of *QualityLevel* metaclass should provide its own assessment function (*assessFun*). The function yields values of *AssType*, i.e. non-acceptable, minimal, target or exceeding.

The elementary assessment (*MetricQualityLevel*) results from comparison of required and obtained values of a metric associated with a given requirement. The values of the assessment function for *MetricQualityLevel* are arguments for assessment functions of *RequirementQualityLevel*.

The values of the assessment function for *RequirementQualityLevel* are arguments for others assessment functions, i.e. *CharacteristicQualityLevel*, *ArtifactQualityLevel*, *SPQualityLevel*.

The non-elementary assessment functions may take for example a form of weighted sum of their arguments. In this case the metamodel may be extended to model different weights of importance for different requirements, characteristics, items, etc.

4 Exemplary Model

The SQMREA metamodel may be instantiated in many ways. The main part of it is a quality model. As it was explained in Sections 2, and 3, such models, once instantiated, may be reused for many projects.

In the presented example only external quality perspective is considered, and the appropriate quality model it is the same as in [1].

Let a text editor be the considered software product. First, we should express some needs, addressed to our product. We limits to two needs: N1: *Easy to use*; N2: *Effective*.

The needs are further refined by requirements. Each requirement is associated with quality model characteristics, and some metrics. For example, N1 need is specified by R1 requirement: *User friendly interface*. The interpretation of this requirement is given by two metrics, defined in [2], M1: *Ease of function learning*, and M2: *Help frequency*. This requirement belongs to usability characteristic – see Table 1 for details.

Table 1. Exemplary SQMREA metamodel instantiation

SP	Artifact	Need	Requirement/ Sub-requirement	Characteristic	Metric	Requirement Quality Level
Text editor	Executable program	N1: *Easy to use*	R1: *User friendly interface*	Usability	M1: *Ease of function learning*	Required: $x \leq 1.5$ min Obtained: $x = 1.5$ min
					M2: *Help frequency*	Required: $x < 3$ Obtained: $x = 4.2$
			R2: *Build-in help*		M3: *Help accessibility*	Required: $x \geq 80$ % Obtained: $x = 70$ %
		N2: *Effective*	R3: *Fast*	Efficiency	Not a leaf	
			/R3.1: *Fast document loading*		M4: *Response time*	Required: $x \leq 1$ min Obtained: $x = 0.5$ min

The last column in the table 1 contain instances of *MetricQualityLevel* metaclass. For each pair: *requirement – metric* two attributes should be defined. The first attribute – called *required* – specifies the required level of a given metric for a given requirement. The value of this attribute is usually defined as specification stage. The second attribute – called *obtained* – is the result of the measurement process. The value of this attribute shows the measure of the metric in the context of considered requirement.

To assess the quality of R1 requirement, first the quality of pairs R1-M1, and R1-M2 should be evaluated.

The exemplary definitions of quality assessment function for the pairs R1-M1, and R1-M2 are given in Table 2.

Table 2. Definition of assessment function for R1- M1, and R1-M2

Assessment function for R1-M1	
Range of obtained value (x)	Assessment result
$0 < x \leq 1.0$	exceeding
$1.0 < x \leq 1.5$	target
$1.5 < x \leq 2.0$	minimal
$2.0 < x$	non-acceptable

Assessment function for R1-M2	
Range of obtained value (x)	Assessment result
$0 < x < 0.1$	exceeding
$0.1 \leq x < 3$	target
$3 \leq x < 5$	minimal
$5 \leq x$	non-acceptable

On the base of Tables 1 and 2 it is easy to check that pair R1-M1 is assessed as target, while R1-M2 – as minimal.

Table 3 introduces exemplary definition of *assessFun*() for R1 requirement. The definitions refer to assessment results of metrics associated with R1 requirement. The quality of R1 is evaluated as minimal.

Table 3. Definition of assessment function for R1

Assessment result for R1-M1	**Assessment result for R1-M2**	**Assessment result for R1**
Non-acceptable	any	non-acceptable
any	non-acceptable	non-acceptable
minimal	≥ minimal	minimal
≥ minimal	minimal	minimal
≥ target	target	target
target	≥ target	target
exceeding	exceeding	exceeding

The assessment process may be continued for other elements, i.e. characteristics, artifacts and so on.

5 Conclusions and related works

Several models of software quality have been suggested over the years [10]. All of them are based on a *quality factor* notion. The quality factors are recognized as those software attributes, which deal with fulfillment of software requirements.

The first quality model proposed by McCall from 1977 [10] consisted of 11 quality factors grouped into three categories: product operation (which deals with daily operation of the software), product revision (which deals with software maintenance) and product transition (which refers to the capability of software adaptation to other environments).

Subsequent models consisting of 12 to 15 factors were suggested in 1988 by Deutsch &Willis and in 1987 by Evans & Marciniak [10]. These models differ from McCall model in removing one factor and different categorization of other quality factors.

These three models contributed to quality models which were elaborated by ISO. The ISO standards are the base for construction of quality metamodel SQMREA presented in the paper. This metamodel grasps not only quality aspects but also its requirements, evaluation, and assessment. The SQMREA metamodel enables:

- tracing requirements from needs, and requirements from other requirements,

- disciplined association of requirements with sub-characteristics and metrics during software requirements specification,
- assessment of quality for separate artifacts as well for a whole software product.

Instances of SQMRE metamodel – quality models – may be reused for many projects. They also may be a convenient form for the experience gathering within different software life-cycles.

The following quality levels are taking into account when we consider software quality [14]:

1) data quality,
2) code quality,
3) model/architecture quality,
4) process quality,
5) management quality,
6) the quality environment.

In the paper we concentrated on models for product evaluation (levels 1-3 from the classification given above). There are many other standards, e.g. the Capability Maturity Model [13], which provide accepted guidelines for improving the software (development) process itself (levels 4-6). Our considerations relate to Quality Control process which deals with ensuring the quality of whatever is produced within the process, while CMM relates to Quality Assurance process which deals with the quality of process that produces artifacts [14].

According to our knowledge, there are no programming tools supporting requirements, evaluation and assessment of quality in the sense of [6]. A project of such a tool based on SQMRE is under elaboration.

There are not many papers presenting quality metamodels. The only one found by us is [12]. The metamodel described in this work is partially similar with our – it includes such elements as *QualityModel* (*Quality* in our metamodel), *QualityAttribute* (*Characteristic*), *Metric*, *BusinessGoal* (*Need*). But that metamodel can't be used for requirements (quality) assessment and evaluation.

References

1. ISO/IEC 9621-1:2000, Software engineering – Product quality - Part 1: Quality model
2. ISO/IEC TR 9621-2:2002, Software engineering – Product quality - Part 2: External metrics
3. ISO/IEC TR 9621-3:2002, Software engineering – Product quality - Part 3: Internal metrics
4. ISO/IEC TR 9621-4:2002, Software engineering – Product quality - Part 4: Quality in use metrics
5. ISO/IEC 14598-3:2000, Software engineering – Product evaluation – Part 3: Process for developers
6. ISO/IEC TR 15904:2000, Information technology – Software process assessment
7. ISO/IEC 15939:2002, Software engineering – Software measurement process
8. ISO/IEC 25000:2005, Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide for SQuaRE
9. ISO/IEC 9000:2000,Quality management system –Fundamentals and Vocabulary,
10. Galin D., Software Quality Assurance, Pearson Education Limit, 2004
11. SWEBOK, Guide to the Software Engineering Body of Knowledge, 2002
12. Bayer J., Design for Quality, IESE-Report No. 084.03/E, September 2003, available at: www.iese.fraunhofer.de/pdf_files/iese-084_03.pdf

13. Jalote P., CMM in Practice: Process for Executing Software Projects at Infosys, Boston, Addison-Wesley, 2000
14. Unhelkar B., Process Quality Assurance for UML-Based Projects, Addison-Wesley, 2002