Evaluating a Quality Model for Software Product Assessments – A Case Study

Michael Kläs[1], Klaus Lochmann[2], Lars Heinemann[2]

[1] Fraunhofer IESE
67663 Kaiserslautern
michael.klaes@iese.fraunhofer.de

[2] Technische Universität München,
85748 Garching, Germany
{lochmann, heineman}@in.tum.de

Abstract. *Background*: Software quality models have been proposed as a means for describing the concept of quality. Most quality models take an abstract view on quality characteristics. Therefore, they are not able to integrate measurement tools and metrics for conducting quality assessments of real software systems. To solve this problem, we developed a quality meta-model defining the structure of quality models that are detailed enough to specify quality characteristics and their links to metrics and measurement tools. *Aim*: In this paper, we present our evaluation of this meta-model in terms of its usability for constructing quality models that are suitable for quality assessments of real software systems. *Method*: For conducting the study, we developed an initial "proof-of-concept" quality model on the basis of static code analysis tools. This quality model was used for conducting quality assessments of Java-based software systems. The results were analyzed regarding two criteria: (1) the diversification provided by the results and (2) the congruence of the results with an independently conducted expert-based evaluation of the systems. *Results*: While the difference in the assessment results between the various systems is rather small, a correlation with the expert evaluation could be proven. Furthermore, the study provided useful insights for further work and improvements. *Conclusions*: We conclude that quality models based on the Quamoco meta-model are, in principle, capable of being operationalized for the automated quality assessment of software systems.

Keywords: Quality Model, Software Quality Evaluation, Empirical Study

1 Introduction

Software quality is a crucial factor for the sustainable success of a software product. To understand and manage the complex and multi-faceted concept of software quality, a number of quality models have been proposed, such as [1][2][3][6]. Unfortunately, most models take an abstract view on quality characteristics and are not detailed enough to enable an operationalization in terms of a quality assessment of real software systems. Besides the approaches for quality modeling, a variety of largely isolated software tools exist for measuring specific metrics related to quality.

The Quamoco[1] project aims at closing the gap between the abstract definitions in existing quality models on the one hand and analysis tools on the other hand by providing a quality meta-model that allows creating quality models that are operationalized to assess the quality of software products. A quality meta-model defines the principal structure of quality models; this means it contains the knowledge about how quality can be modeled. The concrete quality models using this structure contain specific knowledge about what constitutes quality in a certain context (e.g., project, company) and are therefore specific for the environment where they are employed. For instance, thresholds for certain quality metrics can be different depending on the respective application domain. Providing an appropriate quality meta-model is considered as important for defining consistent models that are useful for their defined application purpose [2][4].

Problem. The Quamoco quality meta-model should allow modeling concrete quality models that are detailed enough to perform product quality assessments. However, from discussions with industry representatives and form reviewing existing work, we are aware of a number of expectations regarding quality models appropriateness for performing quality assessments. The key criteria we identified are:

- The model supports *reliable* assessments. This means that if an assessment for a specific product is repeated, we obtain the same or at least a similar result.
- The model provides *valid* assessment results. This means the assessment results are in concordance with the results obtained by other (independent) quality evaluations of the assessed products.
- The assessments based on the model help to answer relevant questions by decision makers, in particular *'Which product is better with respect to quality in general or with respect to a certain quality aspect?'* This means that the results provided by the model have to *differentiate* between products of different quality.
- The model allows performing assessments in a *cost-efficient* manner.

Contribution. As a 'proof of concept' for checking whether quality models based on the proposed meta-model can fulfill these criteria, we created a first concrete quality model incorporating code measures that can be automatically determined by existing tools. This model and its structure are briefly described in Section 3. Since the model contains only measures automatically collected by tools and their evaluation is done in a fully automated manner using evaluation rules predefined in the model, we obtain repeatable and, accordingly, reliable assessment results. The high degree of automation also leads to a minimal amount of manual activities required to perform an assessment for a specific product and thus results in high cost efficiency. The two remaining criteria that should be fulfilled by a model in order to be useful for quality assessments – the *validity* of the model-based assessments (Goal 1) and the model's *ability to differentiate* products of different quality (Goal 2) – are evaluated in an empirical study presented in Section 4.

[1] *http://www.quamoco.de*

2 Related Work

A large number of quality models have been proposed in the literature, for example, [1][2][5][6]. These quality models define the term *quality* by decomposing it into more concrete quality attributes. However, they typically remain on a high level of abstraction and do not define how an actual quality assessment can be conducted using them. There is work on trying to establish a connection between those high-level quality models and measurement tools. For example, [9] and [10] developed an experimental quality model that specifies aggregation formulas needed for aggregating concrete measurement results. A more comprehensive approach for using quality models for quality assessments is being developed by the research project Squale[2], where researchers are developing an explicit quality model and a tool for evaluating software products. The main difference to our approach is that Squale uses a fixed quality model, whereas in Quamoco, the quality model can be edited, with the explicit meta-model guaranteeing that the structure of the created models is interpretable by the assessment tool chain. Moreover, Squale is limited to automated measures while Quamoco allows the seamless integration of the results of manual analysis activities like inspections and reviews.

Most existing work regarding quality models focuses on defining quality on a high level of abstraction. Work on using quality models for assessing the quality of software products is much more limited. Moreover, empirical evidence on quality assessments using these quality models is largely missing.

3 The Quamoco Quality Model

The quality assessment approach relies on a quality model that defines elements for specifying and measuring quality and for evaluating and aggregating the measurement results. The quality model is based on an explicit meta-model, whose main parts are summarized in the following; for further details, please see [4]. The quality model defines a product model of the software as suggested in similar forms in the literature [3][5]. The product model describes **Entities** and **part-of** and **is-a** relationships between them. When describing the quality of source code, typical entities in the model include *Class* and *Expression*, where *Expression* is further refined by *Relational expression* and *Mathematical expression*, which are in an *is-a* relation with *Expression*. The entities are characterized by **Attributes**, resulting in **Factors**. A factor is the central part of the quality model and describes a property of the software product with an influence on quality. A typical factor in the quality model is, for example, *Correctness* of *Relational expression*, which describes that a relational expression is correct if its operands have compatible types, units, scales, etc.

While factors describe properties of the software product, **Quality aspects** in the quality model describe the quality characteristics that are in the focus of the analysis, like the "-ilities" of the ISO 9126 [6]. The influence of factors on quality aspects is modeled as **Impacts**. For each impact an explicit justification and direction must be

[2] *http:// www.squale.org*

provided in prose text. An impact may, for example, be “The *Correctness* of *Relational expression*s has a positive impact on *Reliability*, because incorrect comparisons of data may cause arbitrary failures at runtime”. For conducting a quality assessment, the factors specified in the model must be quantified by **Measures**. A measure specifies which data have to be provided either by a tool or by manual inspection in order to provide an assessment of a factor.

An important purpose of operationalized quality models is to specify how the collected measurement values are aggregated, normalized, and transformed in evaluations in order to yield a quality assessment in terms of quality aspects. These aggregations/normalizations and mappings to an evaluation scale can be specified in a domain-specific language named **QIESL** (quality impact evaluation specification language), which can be used to specify rules for **Impact evaluation** and **Quality aspect evaluation** elements in the model. This language has a Java-like syntax and provides predefined functions for special purposes, e.g. a function for calculating a linear distribution, or a function for calculating the proportion of methods affected by a certain type of defect detected by a measure. The structure of this meta-model should make it possible to express the contents of existing quality models. In a previous study [4], we analyzed the generality of the quality meta-model and concluded that it is able to express the concepts of a large range of different quality models applied in practice.

Tooling. We developed tool support for creating and editing quality models using Eclipse/EMF. The quality model editor supports the user in building quality models consistent with the Quamoco meta-model by automatically checking corresponding modeling constraints and providing different views.

For conducting automatic quality assessments, we developed tool support based on the framework ConQAT[3]. It loads a quality model, takes the data of external static code analysis tools like FindBugs[4], and evaluates and aggregates the measurement results based on the QIESL formulas specified in the quality model.

Base Model. In order to evaluate the meta-model regarding its operationalizability, we developed a concrete quality model that is compliant to the meta-model. This quality model, though limited in size, shows how the meta-model concepts are applied in a meaningful way. The quality model describes 24 factors for the programming languages C/C++ and Java. In this paper, we focus on Java; therefore, only 10 factors are relevant. These factors have 11 impacts on quality aspects. For each impact, a QIESL formula was specified in the corresponding impact evaluation. The evaluation results produced by these formulas are used as a basis for the further discussions in this paper. Due to reasons of brevity, the contents of the base model are not discussed in detail here. However, a browsable representation of the base model can be found at the Quamoco Web Portal[5].

Figure 1 illustrates an excerpt of the quality model. The factor *Structuredness* of *Class* is measured based on a rule of the tool FindBugs. The impact describes that the factor has an influence on the aspect *Analyzability*. The QIESL formula connected with the impact specifies how the measurement results for the factor are interpreted with respect to their impact on *Analyzability*.

---

[3] *http://www.conqat.org*
[4] *http://findbugs.sourceforge.net*
[5] *https://quamoco.in.tum.de/webportal/version1.php*

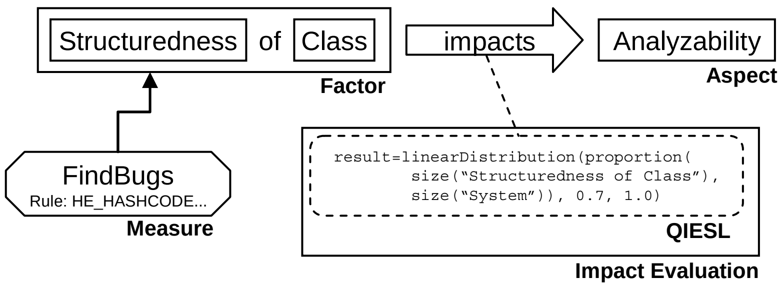

Figure 1: Excerpt of the Quality Model

4 Empirical Evaluation

This section describes the empirical evaluation of the appropriateness of the Base Model with respect to its application purpose, namely *assessing software quality*. We focus our investigation on the two aspects discussed in the introduction that have to be empirically evaluated. We first present the two corresponding study goals. In the following sub-sections, we derive criteria (and hypotheses) from these goals that will be check for the Base Model. For each criterion, we describe the evaluation procedure used and present and discuss the results obtained.

Goal 1 (Diversification). Evaluate whether the assessment results obtained by applying the Base Model provide a sufficient level of diversification between products with different quality levels to answer questions such as ‘Which product is better with respect to quality in general or with respect to a certain quality aspect?’.

Goal2 (Validity). Evaluate whether the Base Model provides *valid* assessment results, meaning that the assessment results are in concordance with the results obtained by another independent and valid approach for assessing product quality.

4.1 Evaluation of Diversification

To the best of our knowledge, there are no commonly accepted criteria for an appropriate level of diversification in the area of software quality assessment. The diversification provided on the bottom level of the model, where the collected measurement values are mapped onto values on the evaluation scale (in our case a number from 1 to 6), strongly depends on the approach used to define the responsible mapping function.

Benchmarking-based approaches typically define specific evaluation functions by analyzing a sample of products and calculating some statistics for the resulting measurement values. If the sample is large enough and representative of the population of assessed products, these approaches can give any form to the resulting distribution and therefore can also provide a defined level of diversification (as far as the measurement results differ between the products in the considered population). In many cases, such approaches create a distribution that is similar to an equal or uniform distribution, by mapping the quantiles of the measurement data distribution onto the values of the evaluation scale (e.g., [11]).

Other approaches define the mapping between the measurement results and the resulting evaluations using the knowledge of experts or target values based on empirical studies or literature reviews as in the Base Model presented. In such cases, it is much more relevant to check the level of diversification provided because a sufficient level of diversification is not automatically assured by the approach.

In the area of operational research, where 'diversification' is a known concept, a series of measures are being discussed to determine diversification on a nominal or ordinal scale [8]. One of these measures commonly used is *entropy* (*E*), which originates from information theory and is defined as:

$E = -\sum p_i \ln(p_i)$ for $i=1\ldots m$, where p_i is the probability to obtain scale level i

E=0 means there is a probability of 100% to get the same assessment result for each product and, consequently, there is no diversification at all. On the other hand, a high value of E means the assessment results are well distributed across the scale levels. Since the maximum obtainable entropy depends on the number of levels offered by the scale, we can compute the *normalized entropy* (*e*) by dividing *E* with its maximum for a given number of scale levels (*m*), namely *ln(m)*, and obtain a value between 0 and 1:

$e = -\ln(m)^{-1} \sum p_i \ln(p_i)$ for $i=1\ldots m$

In order to use this equation to estimate the diversification provided by our evaluations, we have to approximate the probability values p_i for each scale level. We can do this by determining the ratio between the assessment results in the sample with level i (n_i) and the total number of results in the sample (n): $p_i = n_i / n$ for $i = 1 \ldots n$.

If we want to define a criterion for checking whether sufficient diversification is provided by the evaluations in the model, we first have to assume a certain kind of distribution of the assessment results. A maximal normalized entropy and thus diversification is provided by a perfect equal/uniform distribution of the results on the evaluation scale.

We consider a discretized normal distribution across the levels '1' to '6' with a mean of 3.5 and a variance of 1 (Figure 2) as the lower bound for an acceptable diversification. This means that around two-thirds of the assessments provide a '3' or '4' ($<1\sigma$ distance) and around 5 percent a '1' or a '6' ($>2\sigma$ distance). The corresponding normalized entropy is ~0.80 when measured for the total population. However, depending on the size of the sample used to estimate the p_i values, it would not be uncommon to obtain an *e* value of not more than 0.60 for this kind of distribution.

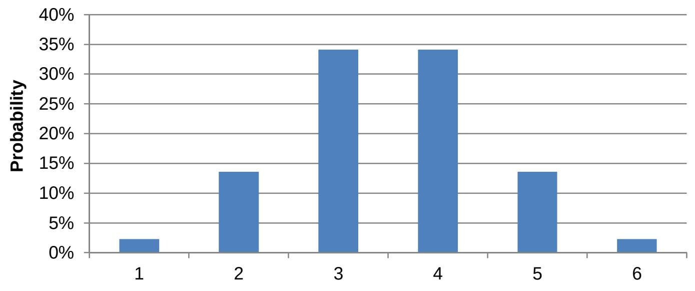

Figure 2: Normal distribution N(3.5,1) discretized on levels 1 to 6.

Hence, in the absence of other criteria, we use (as a rule of thumb) a threshold of ≤0.50 for samples between 10 and 15 assessed products as an indicator of inappropriate diversification. More accurate thresholds could be calculated by performing a simulation study using the actual sample size as input.

However, not only the entropy value but also the range of the evaluation results and their distribution over the evaluation scale should be considered, for instance by visualizing and checking them in a box plot chart.

Procedure: During the study, the Base Model was used to assess 13 software products written in Java. Since the model should be applicable for a broad range of software products, our test objects covered a range of open source projects different in type and size (JabRef 2.3, TV-Browser, RSSOwl, Log4j, Checkstyle, ConQAT, JabRef 2.5, Tomcat) as well as five closed source projects. Each product was assessed with respect to 8 factors influencing the product quality by determining the corresponding impact evaluation result using grades 1 to 6. For each impact evaluation, the distribution of the evaluation results was presented by a box plot and the normalized entropy (e) was calculated.

Results: Figure 3 compares the discretized normal distribution that was used as a baseline with the results of a selection of impact evaluations. In total, five impact evaluations in the Base Model such as 'Technical name x Conformity', which is presented in Figure 3, rated all products with the best grade '1' resulting in e = 0. Despite not using grades '5' and '6', the evaluation for 'Class Comment x Consistency' provided good diversification across the remaining grades resulting in an acceptable entropy value (e=0.72). In general, we could observe that the results of most evaluations tend towards the lower half of the scale (i.e., grades '1' to '3').

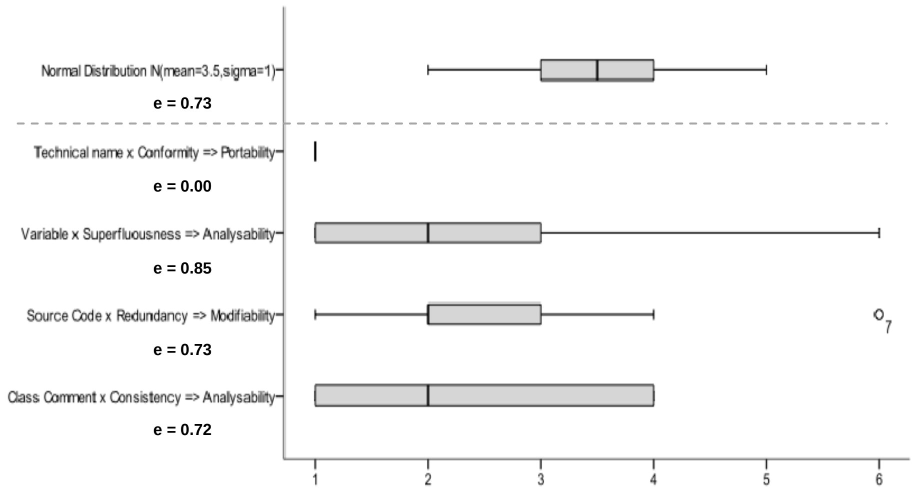

Figure 3: Discretized normal distribution N(3.5,1) compared to the results of selected impact evaluations for n=13 product assessments.

Interpretation: In general, we see three possible reasons for the lack of diversification provided by certain impact evaluations. (1) The assessed products may be all very similar (in our case excellent) with respect to a factor without diversification (e.g., 'Technical name x Conformity'). (2) The factor might not be sufficiently operationalized by the measures collected. This can mean that the measures do not cover all relevant aspects of the factor or that they measure only issues that rarely occur in practice. (3) The evaluation function used to map the measurement values to a specific grade is inappropriate (i.e., it does not differentiate sufficiently between good and bad results). Since we assessed a broad range of different products, it is likely that either the chosen measures are not sufficient for covering the factor or the mapping (including the normalization) does not differentiate sufficiently. To precisely identify the possible reasons, more tests are necessary. The missing diversification provided by certain factors seems not to be a general problem with the meta-model since other factors provide sufficient diversification. However, the results already indicate that the sensitivity of various impact evaluations has to be increased in order to provide better differentiation between products of varying quality on the factor level.

Threats to Validity: Since we could not determine the quality of the assessed products with respect to each factor independently of the evaluations provided by the model, the major validity threat is that the 13 assessed products might be too similar with respect to certain factors to provide good differentiation for each factor without making the assessment results instable due to oversensitive evaluation functions.

4.2 Evaluation of Assessment Validity

In order to evaluate the validity of the model-based quality assessments, we need an independently obtained criterion for product quality that we can compare with our assessment results. Since no measurement data were available that directly measure the quality or the quality aspects of interest for the assessed products, we used as the independent criterion an expert-based quality rating provided in the 'Linzer Software-Verkostung' [7] for a set of five open source products. The rating is a combination of ratings provided independently by nine experienced Java experts.

In the IEEE standard [12], several *validity criteria* are proposed for validating software quality metrics. Most of them assume that the collected measures and the independent criterion both use an interval or ratio scale. However, while the results of the Base Model assessments are provided as a value characterizing the product quality between 1 (best possible) and 6 (worst possible), the assessment results of the Linzer Software-Verkostung are provided on an ordinal scale as a ranking from best (1) to worst (5) product. Consequently, we had to limit our investigation to the validity criterion '*consistency*' [12], which can be applied on interval scale data. In our case, it will characterize the concordance between a product ranking based on the assessments provided by our model and the ranking provided independently by a group of experts. This means that ***we determine whether the Base Model can accurately rank the set of assessed products with respect to their quality*** (as perceived by experts).

Following [12], we measure consistency by computing the *Spearman's rank correlation coefficient* (r) between both rankings, where a high positive correlation means high consistency between the two rankings. Since we want to check whether a potentially observed positive correlation is just due to chance or is a result of using an appropriate quality model, we state the (alternative) hypothesis H_A with alpha = 0.05:

H_A: There is a positive correlation between the ranking provided by the Base Model (BM) and the ranking provided by the "Linzer Software-Verkostung" (LSV).

$$\mathbf{r\ (\ ranking_{BM},\ ranking_{LSV}\ ) > 0} \quad [\text{ i.e., } H_0\text{: r (ranking}_{BM}\text{, ranking}_{LSV}\text{)} \leq 0\ ]$$

Procedure: During the study, the Base Model was used to assess the quality of five open source products for which results of the Linzer Software-Verkostung were available: JabRef 2.3, TV-Browser, RSSOwl, Log4j, and Checkstyle. We had to limit our evaluation to these five products since the remaining seven products used for the evaluation of diversification were not part of the Linzer Software-Verkostung. Furthermore, there was no other independent validity criterion that we could use.

For the sake of simplicity, we assumed in the Base Model that each factor has the same relevance for the overall perceived product quality. Thus, the aggregated assessment result for each product corresponds to a weighted sum with an equal weight for each factor. Taking the sum, we implicitly assume the same distance between the different grades (1 to 6). This means that the difference in quality between a product with grade 1 and a product with grade 2 is assumed to be equal to the difference in quality between a product with grade 2 and a product with the grade 3.

In a final step, the assessed products were ordered by the results for their overall quality provided by the Base Model and compared with the ranking provided by the Linzer Software-Verkostung.

Results: Figure 4 shows the assessment results using the Base Model and the resulting product ranking as well as the ranking of the Linzer Software-Verkostung. The calculated Spearman's rho correlation is ***r = 0.975***, which is close to a perfect correlation of 1. Hypothesis ***H_A*** can also be ***accepted*** on a high level of significance (p=0.002) meaning that there is a significant positive correlation between the ranking provided by the Base Model and the ranking provided by the Linzer Software-Verkostung.

Assessed Product	Result using BM	Ranking by BM	Ranking by LSV
Checkstyle	1.00	1	1
Log4j	1.22	2	2
RSSOwl	1.44	3	3
TV-Browser	1.44	3	4
Jab-Ref 2.3	1.89	4	5

Figure 4: Comparison of the assessment results and 'Linzer Software-Verkostung'

Interpretation: Despite the partly missing differentiation of the assessment results on a lower level of granularity (i.e., with respect to specific factors), the assessments of the overall product quality turn out to be consistent and thus valid when compared to an independent criterion for quality, in this case provided in the form of an expert-based assessment. This indicates that the Quamoco meta-model can be used to specify quality models that provide valid automated quality assessments of software systems. Although this conclusion is supported by a very high and statistically significant correlation, there are some threats to validity that need to be considered.

Threats to Validity: The most relevant threats we see are (1) we cannot guarantee that the criterion chosen for the validation, namely the expert-based quality rating (of 9 experts reviewing each product), adequately represents the quality of the products, (2) the generalizability of our results is limited by the fact that the number of assessed products (5 systems), their type (Java, Open Source), and the factors considered in the assessment are limited.

5 Conclusions and Future Work

In this paper, we investigated the question of whether the meta-model proposed by the Quamoco project can be used to define quality models that are appropriate for quality assessments of real software systems. In order to answer this question, we developed an initial quality model (Base Model) using the proposed meta-model and evaluated in a case study whether the quality model can diversify software products of different quality on the level of quality factors. Moreover, we compared the results of the assessment with an independent quality ranking of the products performed by experts.

In the model, we identified five factors for which we had to reject our assumption that the model is able to differentiate between products of different quality with respect to these factor. This leads us to the conclusion that the sensitivity of the affected impact evaluations needs to be increased. However, our results also indicate that the model provides a quality assessment in line with the findings of an independent expert group. We found a very high and also statistically significant correlation between the manual quality ranking and the results of the automated assessment based on the quality model.

We conclude that the Quamoco meta-model can be used to specify quality models that provide valid automated quality assessments of software systems. As future work, we plan to refine the impact evaluations in order to achieve better results with regard to diversification among software systems. Moreover, we plan to extend the quality model to include more quality characteristics and measurements.

Acknowledgments

This work was supported in part by the German Federal Ministry of Education and Research (Quamoco Project, No. 01 IS 08 023 C). We would also like to thank Sonnhild Namingha for reviewing a first version of this article.

References

[1] Boehm, B. W. (1978): Characteristics of Software Quality, North-Holland.

[2] Kitchenham, B.; Linkman, S. G.; Pasquini, A.; Nanni, V. (1997): The SQUID approach to defining a quality model. In: Software Quality Journal, Vol. 6, Issue 3, pp. 211–233.

[3] Deissenboeck, F.; Wagner, S.; Pizka, M.; Teuchert, S.; Girard, J.-F. (2007): An Activity-Based Quality Model for Maintainability. In: Proc. of the 23rd Int. Conf. on Software Maintenance (ICSM 2007).

[4] Kläs, M.; Lampasona, C.; Nunnenmacher, S.; Wagner, S.; Herrmannsdörfer, M.; Lochmann, K. (2010): How to Evaluate Meta-Models for Software Quality? In: Proc. of the joined Int. Conf. on Software Measurement. IWSM/MetriKon/Mensura 2010, Shaker, pp. 443-462.

[5] Dromey, R. G. (1995): A model for software product quality. In: Software Engineering, IEEE Transactions on, Vol. 21, Issue 2, pp. 146–162.

[6] ISO/IEC. 9126-1 (2001): Software engineering – Product quality – Part 1: Quality model

[7] Plösch, R. (2010): Software-Verkostung: Ein neuer Ansatz zur Validierung von Software Qualitätsmanagement-Werkzeugen. http://www.ipo.jku.at/dokumente/upload/Software%20Verkostung%20Impulsvortrag.pdf.

[8] Troutt, M. D.; Acar, W. (2005): A Lorenz-Pareto measure of pure diversification, European Journal of Operational Research, Vol. 167, Issue 2, pp. 543-549, ISSN 0377-2217, DOI: 10.1016/j.ejor.2004.02.022.

[9] Schackmann, H.; Jansen, M.; Lichter, H. (2009): Tool Support for User-Defined Quality Assessment Models. In: Proc. of MetriKon 2009, Shaker.

[10] Marinescu, C.; Marinescu, R.; Mihancea, R. F.; Ratiu, D.; Wettel, R. (2005): iPlasma: An Integrated Platform for Quality Assessment of Object-Oriented Design. In: Proc. of 21st IEEE Int. Conf. on Software Maintenance - Industrial and Tool volume (ICSM '05).

[11] Gruber, H.; Plösch, R. (2010): On the validity of benchmarking for evaluating code quality. In: Proc. of the joined Int. Conf. on Software Measurement. IWSM/MetriKon/Mensura 2010, Shaker, pp. 463-481.

[12] IEEE. 1061 (1998): Standard for a Software Quality Metrics Methodology.