Early Validation of Software Quality Models with respect to Minimality and Completeness: An Empirical Analysis

Constanza Lampasona, Michael Kläs

Fraunhofer IESE

constanza.lampasona|michael.klaes @iese.fraunhofer.de

Alois Mayr

Johannes Kepler University, Linz

alois.mayr1@jku.at

Andreas Göb

SAP AG, Darmstadt

andreas.goeb@sap.com

Matthias Saft

Siemens AG, Corporate Technology, Munich,

matthias.saft@siemens.com

Abstract:

Validation of quality models is an important and necessary task in order to assure that the quality of a software product is being assessed using the appropriate model. Capture recapture models are employed to provide a quantification of the completeness characteristic of the model content. This article presents an approach for simultaneously validating software quality models with respect to their completeness and minimality. The validation process can be performed without the need of applying the model to a set of software products. The approach is an adaptation of capture-recapture models for estimating software defect content based on largely formalized reviews with defined instruments (questionnaires and templates). We discuss the feasibility of the approach and illustrate its application on four quality models. In conclusion, the application of this static approach has led to valuable information regarding enhancements for the validated quality models, although its statistical power is limited due to the low number of participants.

Keywords

Software quality models; validation; assessment; capture-recapture

1 Introduction

Many software product quality models have been proposed over the years. Early general purpose models include [18], [8], and [13]. In recent years, ISO25010 [10], the successor of ISO9126 [11], was published. It purpose is general as well, and it structures quality in a hierarchy of quality attributes and sub-attributes.

Uncountable quality models were proposed for specific contexts (e.g., [14]) or for specific quality foci (e.g., [1]) and purposes (e.g., [12]). Moreover, quality models can also be developed according to the specific needs of an organization [3].

As no general standardization for software quality models exists, most companies use their own or customized quality models [22][24].

A typical iteration during the development of a quality model consists of three major steps:

1) Specification: A general structure is specified that expresses what quality means in a specific context and how it can be quantified;
2) Operationalization: The model is operationalized by linking the defined measures with appropriate measurement instruments. Moreover, assessment rules (such as thresholds, target values, etc.) and aggregation rules (such as weights in a weighted sum aggregation approach) can be defined; and
3) Application: The model is validated by applying it to software products.

Validating the model after its application implies the collection of data using the defined instruments and applying the assessment and aggregation rules to obtain an assessment result. The quality of the quality model itself is validated by analyzing the assessment results for one or more software products. This is done by comparing the results to results from other assessments such as independent expert judgments, post-release defects, maintenance effort, etc. Although this validation, based on the application of the model, is essential to assure the model's further applicability in its defined context, we consider it as an acceptance validation (i.e., it is performed very late during the model development cycle). Issues regarding the content or structure of the quality model found at this stage can result in large rework effort since the changes affect not only the specification of the model but also its operationalization, and often require collecting data for a product that should be reassessed with the reworked model.

To reduce rework effort and improve quality early in the development, we propose to early validate the models after specification, additionally to their final validation after application. We propose validating quality models with respect to their completeness (it addresses all relevant aspects) and to their minimality (a model only contains aspects relevant for its application context). In this way, the model would cover the important aspects excluding irrelevant content, which may has remained in the model as a hangover after re-using an existing quality model.

The proposed approach consists of a process for quality model reviews/inspections by domain experts. The reviews are implemented as interviews were the experts are guided by a coordinator. During the interviews, data about

model completeness and minimality are collected. For data analysis, we propose using capture-recapture models for estimating model completeness.

As an initial validation of the proposed approach, we applied it within an industrial context on four quality models: on two consecutive versions of a quality model for embedded systems software (ESQM) [17], on one quality model for service oriented architectures software (SOA) [7] and on one quality model for standard software.

In this contribution, we aim at answering following research questions:

RQ1: Is the proposed approach valid? Do the values estimated with capture-recapture models correlate with the experts' subjective judgments?

RQ2: Is the proposed approach useful for early validating quality models?

2 Related Work

2.1 Quality Model Validation Approaches

Theoretical and empirical validations of metrics are nowadays widely applied. Schneidewind [20] defined six criteria for metric validation (association, consistency, discriminative power, tracking, predictability, and repeatability) and showed that statistical methods are important for evaluating metrics' validity.

Xu et al. [23] used machine learning methods combined with statistical methods for metric validation. Theoretical as well as empirical validations were used also by Marchetto and Trentini [16] to analyze the metrics they developed for web applications. Empirical validation is also performed for prediction models with respect to their accuracy. Beaver [4], for example, empirically validated their quality model by comparing the values predicted during design with actual values obtained after software product delivery. Lincke et al. [15] validated their software quality models based in ISO9126 by comparing the results of applying the model to real world projects to experts' opinions and information from bug and test databases.

2.2 Capture-Recapture Models

Capture-recapture models have been used in many fields. In biology, for example, to estimate population size, in social sciences to estimate illegal immigration, and in medicine to estimate the size of a hidden disease burden in a population [5].

Petersen and Lincoln independently developed the first capture-recapture estimator, which is now known as the Lincoln-Peterson estimator [5]. The method consists in capturing, marking and releasing animals; and later capturing them again. In the second capture, marked and unmarked animals are considered. The first time n_j animals are caught, marked, and released. Afterwards a sample of n_z ani-

mals is captured. From them m_z have been marked. So, one can derive an estimator of the population size (N) assuming that the ratio of marked to total animals in the second sample reflects the same ratio in the population. With this, an estimator ($\hat{N}$) for population size can be derived:

$$\hat{N} = \frac{n_1 n_2}{m_2}$$

The model assumes a closed population (without births, immigrants, deaths, or emigrants), that all animals are equally likely to be captured in each sample, and that marks are neither lost, nor overlooked by the observer.

Later, Chapman modified the estimator to allow less bias:

$$\hat{N}_c = \frac{(n_1+1)(n_2+1)}{m_2+1} - 1$$

Since then, many capture-recapture models have been created and modified according to several variables which influence the variation on estimates, such as capture probability, number of samples, and other characteristics of the population (closed, open, combined closed+open).

Capture-recapture methods have been transferred and widely applied in different practical estimation problems in computer science; for example, for estimating the evolution of web pages in directories [2]. In this case the web pages are considered as animals and they can be submitted to or removed from directory indexes, which resembles the migration phenomenon in nature. Gupta and Thakur [9] applied the capture-recapture method for estimating the key sets probabilities in a semantic search technique. Many authors used the capture-recapture approach to estimate residual defects in software products (e.g., [21][6]). Capture-recapture models were adopted in software engineering mainly for inspections to predict the number of faults remaining in an artifact such as a requirements specification or source code documents [19].

3 Quality Model Early Validation Procedure

The process for validating quality models after specification consists of three steps: (1) Selection of model parts for review, (2) Reviews, and (3) Data analysis.

Experts with respect to quality assurance in the specific domain give their opinion about which quality attributes must be covered by a quality model based on the quality model goal provided (i.e., mandatory elements). Then, the person coordinating the study collects all answers by experts and based on them selects the excerpt of the quality model that should be reviewed. Alternatively, the excerpt can be randomly selected.

During the reviews, the reviewers independently inspect the model and are asked to identify irrelevant elements that do not contribute to the models objective, and answer questions with respect to model minimality. Then, the reviewers are asked to identify missing elements.

For data analysis, all answers are collected and capture-recapture models are used to estimate the completeness of the quality model after specification considering the model's context (e.g., domain, type of product, type of system).

4 Studies Design

We used the procedure described above as a part of four studies for the validation of quality models. Here, we describe the common design of the four studies; afterwards, we describe the four study runs where specific information which varies from study to study is given.

This section contains a description of study goals, study object in general, and of study conduction. Our study plan includes information about how we want to measure the completeness and minimality of a quality model. The section also encompasses reviewers' profiles, experimental material, and instruments.

4.1 Study Goals

The objectives for each application of this approach are:

(G1) Quality Model Minimality: Evaluate the minimality of the quality model from the perspective of quality managers in the specific context of the model. This evaluates whether the quality model contains only relevant elements.

(G2) Quality Model Completeness: Evaluate the completeness of the quality model from the perspective of quality managers in the specific context of the model. This evaluates whether the quality model contains all elements relevant for its specific context and if there is something important missing.

4.2 Study Object

The objects that are investigated using the proposed approach are *quality models*. The contents of a quality model depend on the object it considers, its application purposes, its context, and on the specific stakeholder using it.

The quality models we consider have a hierarchical structure like the one provided by the Quamoco meta-model [23]. The basic structures are acyclic graphs where sub-nodes are a-part-of or refine their super-node. The elements of the graph are called *factors* and represent some quality of the product or of a part of the product. Factors are the core quality concepts and are connected to *measures*. Factors can be of type *quality aspect* or *product factors*. The first is used to describe more abstract attributes and the latter rather concrete attributes.

The meta-model specifies further elements which were not explicitly considered for the design of the study; however they are explained to the reviewers in order to let them have the complete picture. They are *entities*, which are the constituting parts of the software; *impacts*, which represent influences between factors; *evaluations*, which capture the function for assessment; and *instruments*, which are the tools for performing measurement. A factor is associated to an entity, which are also structured in a hierarchy and has an associated evaluation.

For the application of the proposed validation approach a quality model do not have to contain all model elements proposed by the Quamoco meta-model. The only requirement on the structure of the models to be validated is a decomposition of elements into sub elements, that is, an is-part-of relationship or decomposition or specialization relationship.

4.3 Operationalization

This validation approach focuses on the *minimality* and the *completeness* of a quality model. It has to be evaluated whether a quality model contains elements of no relevance for its context and whether the quality model covers quality aspects typical for the respective domain.

Irrelevant elements are elements completely unsuitable for the model according to the model's context. Relevant elements are elements of interest for the context. They can be mandatory (a model is not acceptable unless these elements are provided), desirable (these elements would enhance the model, but would not make it unacceptable if they are absent), and optional (these elements may or may not be worthwhile).

A quality model is suitable for its specific context if it covers mandatory elements (completeness), and if it only contains relevant elements (minimality).

4.3.1 Model Minimality

We define that a model is minimal if it only contains relevant elements. On one hand, we measure this by individually asking the reviewers to mention irrelevant elements, they find in the model. On the other hand, we ask the reviewers to assess minimality by answering to the statement "I consider the model minimal". The answers are based on a 5-point Likert scale: 1: strongly agree, 2: agree, 3: neither agree nor disagree, 4: disagree, and 5: strongly disagree.

4.3.2 Model Completeness

We understand completeness as the property of a quality model that reflects to which degree the quality model contains relevant elements. Roughly, we want to find out if there is something important missing in the quality model.

Inspired on the capture-recapture approach, we can define a measure of completeness based on two base measures: the actual number of relevant elements contained in the model, including elements found during review (re_{actual}) and the total number of elements relevant for the model (re_{total}). The latter equals the sum of the actual number of relevant elements contained in the model and the number of relevant elements which are missing ($re_{total} = re_{actual} + re_{missing}$). With this, completeness is operationalized as:

$$\text{Completeness} = \frac{re_{actual}}{re_{total}} * 100\% = \frac{re_{actual}}{re_{actual} + re_{missing}} * 100\%$$

To predict $re_{missing}$ we use capture-recapture estimators. We use estimation because even if several independent reviews are performed, we cannot be certain that we find all relevant elements that are missing in the model. Thus, we need to define an estimator for the number of remaining missing elements.

Different capture-recapture models are based on different assumptions about heterogeneity (h) and time (t). Heterogeneity describes that there are faults that are more difficult to be found than other faults. Time describes that some inspectors are able to find more faults than other inspectors, according to their experience. According to these two variables, heterogeneity and time, the appropriate model can be selected for estimating missing elements. Table 1 shows the different combinations of heterogeneity and time for closed populations that have been used in software engineering. It is adapted from the table summarizing models used for software inspections presented in [19]. The selection of the most appropriate model can be done according to the probability detection for missing elements and the reviewers' abilities.

Table 1 The different models for capture-recapture estimators [19]

Model	Prerequisites	Estimators
M_0	All missing elements have equal detection probability	M_0-ML: maximum likelihood
	All reviewers have equal detection ability	
M_t	All missing elements have equal detection probability	M_t-ML: maximum likelihood
	Reviewers have different detection abilities	M_t-Ch: Chao's estimator
M_h	Missing elements may have different detection probabilities	M_h-JK: Jackknife
	All reviewers have equal detection ability	M_h-Ch: Chao's estimator
M_{th}	Missing elements may have different detection probabilities	Mth-Ch: Chao's estimator
	Reviewers have different detection abilities	

Besides those estimations performed using the capture-recapture estimators, two questions were asked to the reviewers in order to collect their estimations with respect to model completeness. The first question consisted on asking them to estimate how many elements are still missing. The second question was an answer to a statement: “The model can be considered as complete after integrating the found elements”. This was to be answered using a 5-point Likert scale: 1: strongly agree, 2: agree, 3: neither agree nor disagree, 4: disagree, and 5: strongly disagree.

4.3.3 Model Understandability

We ask about the quality model understandability to consider it as a variation factor. The reviewers answer to the statement “I was able to understand the quality model (with reasonable amount of effort)” using a 5-point Likert scale: 1: strongly agree, 2: agree, 3: neither agree nor disagree, 4: disagree, and 5: strongly disagree.

4.4 Resources

We refer to the person guiding the interviews as *coordinator.* On the other hand, the *reviewers* are chosen according to the requirements of the study: They should be quality experts in the domain of the quality model being examined. We asked the reviewers two questions about their experience: 1) What is your experience with the specific quality model? and 2) What is your experience with other quality models in this domain? Both questions could be answered using a 5-point Likert scale: 1: No experience, 2: A little experience, 3: Some experience, 4: Much experience, and 5: A lot of experience.

4.5 Procedure

Tasks: The reviewers receive the selected model part for inspection as well as a short description of the model’s structure and objectives. Each reviewer independently inspects the model’s excerpt and generates a list documenting each missing element with a short description and its relevance. Moreover, the reviewers answer the question about their experience. Additionally, the reviewers are asked to subjectively evaluate the completeness and minimality of the model and if they would consider the model as complete if the missing element were integrated. Afterwards, all lists of missing elements are collected and compared, and the overlapping/shared missing elements serve as a basis for estimating the number of not yet detected missing elements. The coordinator identifies which missing elements noted by the individual reviewers may actually be regarded as the same. Then, the estimators are calculated.

Instruments: The instruments used in this study are:

1) A template for collecting identified missing elements, their subjective classification by the reviewer with respect to the difficulty each missing element, and elements of no relevance.

2) A questionnaire documenting the reviewers' experience and their subjective opinion on completeness and minimality of the model.
3) A template for documenting all missing elements and which reviewers found them to be used by the coordinator.[1]

4.6 Analysis Procedure

For data analysis, we use capture-recapture estimators for predicting the number of missing elements not found. As the missing elements are actually unknown and we cannot "inject" them in the models, like it could be done with defects in code, we assume that their detection probabilities are heterogeneous. Reviewers may have equal or different detection capabilities.

With this, we decided to use two estimators for which missing elements may have different detection probabilities: M_h-JK (all reviewers have equal detection ability) and M_{th}-Ch (reviewers have different detection abilities) and compare the results with the experts' estimations by means of correlations. With this we could see which estimator is more robust for estimating the number of missing elements in a quality model.

5 Studies Runs

We applied the proposed approach on four quality models: on two consecutive versions of a quality model for embedded systems software, on one version of a quality model for SOA and on a one quality model for standard software.

Common studies context for the four runs: In the publicly funded project Quamoco[2], software quality models were developed, and the biggest challenge was to make them sound and applicable in practice. The quality models were created in cooperation with industry partners, driven by the needs and requirements of different application domains. The development of these models involved the identification of relevant quality elements and their specification and operationalization in a way that makes them measurable and assessable.

1st Study Object - Embedded Systems Quality Model v2: The study object for the initial application was a quality model developed for embedded software (ESQMv2). This model adheres to the Quamoco meta-model and consists of two hierarchies of quality factors.

First, technical quality requirements are structured along programming issues in order to classify them. The technical quality requirements are largely provided

1 Templates and questionnaires can be provided by request.

2 http://www.quamoco.de

with proper rules of the static code analyzer PC-lint[1] to operationalize them. Moreover, some manual review instructions serve as measures for coarse-grained requirements. One example of a technical quality requirement is *MEM1 –Avoid wrong and invalid references* and which is measured using 19 PC-lint rules.

Second, a hierarchy of goals provides a general management view on quality for embedded systems and is inspired by the ISO9126 quality attributes. None of the goals is measured directly (like the technical quality requirements); however, they can be determined via impact relations emerging from the technical quality requirements. An example of a goal is *G. Code minimality*, which is impacted by six technical quality requirements. One of them is *COR5 –Avoidance of unnecessary constructs*.

Altogether, ESQMv2 consists of 25 goals, 61 technical quality requirements, and 304 measures.

Three goals were selected for review: *G. Error handling*, *G. Robustness*, and *G. Code minimality*. Additionally, five technical quality requirements were selected: *PROT10 –Ensure proper handling of unavoidable exceptions*, *DES4 –Usage of strong type systems*, *MEM3 –Predictable usage of memory*, *PROC2 –Proper usage of switch statements*, and *DEC3 –Avoidance of function like macros*.

2nd Study Object - Embedded Systems Quality Model v3: The embedded quality model in version 3 (ESQMv3) has been improved based on the validation results of version 2 and was the study object of this run.

The model was enhanced by a hierarchical layer of programming language independent product factors. A product factor describes a property of a product entity, e.g., *Reference Validity of Assignment Statements*. Additionally, the hierarchy of goals was replaced by the ISO25010 quality characteristics. In contrast to ESQMv2, the requirements are now impacted by the technical product factors and are not quantified directly by means of measures.

Altogether, ESQMv3 consists of the 38 ISO25010 quality-characteristics (*-ilities*), 60 requirements, 194 product factors (32 of them are only for structuring purpose) and 336 measures.

The model excerpt for review was randomly selected and includes three quality aspects (*reusability*, *time behaviour*, and *reliability*) and five technical requirements (called *DEC4*, *PROC5*, *PROT3*, *CON1*, *DES4* – the reviewers were familiar with these abbreviations).

3rd Study Object - Quality Model for SOA: The quality model for SOA (SOAQM) focuses on properties that are specific for service-oriented architec-

1 http://www.gimpel.com/html/index.htm

tures. The model is divided into two layers: The SOA *conformance* layer describes basic architectural and technical principles (e.g., reuse by composition of services) that every SOA-based product should fulfill. These principles contribute to the product's quality to a certain degree.

Further aspects of quality are described in the SOA *additions* quality layer. It contains product factors (e.g., statelessness of services), which influence product quality beyond those concepts described in the conformance layer.

The model contains 175 elements, including 6 SOA-specific entities, 37 product factors, 35 impacts, 59 evaluations, and 38 measures.

A set of four SOA principles was selected and their completeness with respect to measures was evaluated. The selected SOA principles were: *Abstraction level @Service*, *Functional granularity @Service*, *Standardization*, and *Composition*.

4th Study Object - Quality Model for Standard Software: The quality model for standard software (StQM) describes quality characteristics and properties of a software product that are especially important for standard software products. As opposed to custom software development, standard software products are delivered to a large number of customers. Because of this, specific quality requirements have been developed and grouped into the categories administration and servicing, business continuity, business integration, user facilitation, and environmental requirements. Typical requirements of the first category include *ease of customization* and *easy application of patches*.

Five StQM factors was selected to evaluate their completeness with respect to measures. The selected factors were: *ease of learning @product*, *hardware independence @source code layer*, *ease of installation @product*, *multiple language and culture support @product*, and *providing and easy application @product*.

Table 2 Summary of results

Quality model		**Minimality**[a*]	**Mandatory elements coverage**[b*]	**Understandability**[c*]	**#Missing elements estimation by M_h-JK ($re_{missing}$) → Completeness (%)**	**#Missing elements estimation by M_{th}-Ch ($re_{missing}$)→ Completeness (%)**	**#Missing elements subjective estimation by reviewers (min-max)**[d] **→ Completeness (%)**	**Completeness agreement**[e*]
EQ Mv2	*Goals*	μ=2.50 σ=1.00	μ=2.11 σ=0.60	μ=2.00 σ=0.00	24→49%	30→43%	2-8→74-92%	$\tilde{x} \leq 2$
	Requirements				15→64%	12→69%	4-7→79-87%	$\tilde{x} \leq 2$
EQ Mv3	*Quality characteristics*	μ=2.40 σ=0.89	μ=2.00 σ=1.25	μ=1.80 σ=0.45	37→56%	60→44%	3-10→83-94%	$\tilde{x} \leq 2$
	Requirements				46→63%	57→58%	1-14→85-99%	$\tilde{x} \leq 2$
SOAQM - *SOA principles*		μ=1.00 σ=0.00	μ=1.20 σ=0.45	μ=1.60 σ=0.55	10→53%	18→54%	0→100%	$\tilde{x} \leq 2$
StQM - *Factors*		μ=1.40 σ=0.55	μ=1.58 σ=0.79	μ=1.40 σ=0.89	15→53%	12→59%	0→100%	$\tilde{x} \leq 2$

a: *I consider the model minimal*.

b: *I consider that the quality model is appropriate regarding this characteristic.*

c: *I was able to understand the quality model (with reasonable amount of effort)*

d: *How many elements do you think are still missing? (for the inspected factors after the missing elements you detected are included)*

e: *The model can be considered as complete after integrating the found elements.*

*: Scale: 1: strongly agree, 2: agree, 3: neither agree nor disagree, 4: disagree, and 5: strongly disagree.

6 Studies Results

Here we report the results obtained for all studies. When we refer to the participants, those are meant that reviewed the respective model. Numbers are reported in Table 2.

Mentioned mandatory elements: When we asked about mandatory elements, the reviewers believed that the models cover the mentioned characteristics appropriately. The reviewers mentioned as mandatory elements for ESQMv2, efficiency, reliability, and timeliness, and for ESQMv3, resource efficiency, real time and failure tolerance. Abstraction and granularity were mentioned as mandatory elements of the SOAQM. Interoperability/integration and portability/platform independency were mentioned for the StQM.

Reviewers' profile: The reviewers' self-assessment of experience with regard to the reviewed model were: ESQMv2: μ=3.50, σ=1.29, with number of reviewers n=4; ESQMv3: μ=3.00, σ=1.26, n=6; SOAQM: μ=3.40, σ=1.51, n=5; StQM: μ=3.4, σ=0.54, n=5. Their self-assessment of experience with regard to quality models in the respective context were: ESQMv2: μ=3.50 σ=1.29; ESQMv3: μ=2.00 σ=0.89; SOAQM: μ=2.40 σ=1.51; StQM: μ=2.20 σ=1.30.

7 Analysis

RQ1: Is the proposed approach valid? Do the values estimated with capture-recapture models correlate with the experts' subjective judgments?

The subjective assessment of completeness by the experts differs substantially from the estimated completeness values. They appraise the quality models as more complete than when the estimators are used.

Moreover, the experts' estimations are not correlated with none of the estimations calculated using the Jackknife and Chao's estimators (Table 3) (For N=6, a value of at least 0.622 is required for correlation at a significance level of 0.05). We could not validate the approach with respect to experts' judgments. This reflects the problem of having different experts providing estimates for different models. Regarding completeness agreement, there seems to be no consistency, neither with the expert completeness estimate nor with the completeness estimates from the capture-recapture models. One further open question is whether the expert judgment is a valid validation criterion, since we do not know about its accuracy in general.

The validation of this approach is very difficult. On the one hand, we do not have a control group. On the other, we cannot "inject" missing elements. Moreover, an experiment with novices would be very difficult because of the experience re-

quired to review this kind of artifacts. So, the challenge of validating this approach is still open until other more precise quantitative data is available.

Table 3 Correlations between experts' judgments and capture-recapture estimators

			Completeness		
			Experts min	**Experts max**	**Experts mean**
Completeness	M_h-JK	Pearson Correlation	-0.199	-0.289	-0.336
		Sig. (2-tailed)	0.706	0.579	0.515
		N	6	6	6
	M_{th}-Ch	Pearson Correlation	0.225	-0.142	-0.189
		Sig. (2-tailed)	0.668	0.788	0.720
		N	6	6	6

RQ2: Is the proposed approach useful for early validating quality models?

The developers of the quality models told us that they find this kind of review, where missing elements are searched in a structured, more rigorous way very interesting and useful, because they could enhance their models in the specification phase. In the case of the SOAQM and StQM, the a priori selection of mandatory elements was much appreciated. So, the proposed approach can be used for reviewing models and get an additional piece of information to decide whether inspect the model again based on the number of elements still missing.

8 Threats to Validity

The contents of a quality models depend on the object/artifact they consider, their application purposes, their contexts, and the specific stakeholders using them. Because a quality model is too extensive to be investigated completely with respect to the presence of all relevant model elements, a part of the model was analyzed. Only (randomly) selected parts of the model were reviewed to reduce the effort for the participating domain experts. Therefore, when extrapolating the completeness results to judge the overall completeness of the model, there is a risk that the selection might not be representative.

The low number of reviewers resulted in a low degree of overlapping between the detected missing elements, which reduces the accuracy of the applied capture-recapture estimates.

The reviewer's experience may be a confounding factor influencing the study result. A lack of knowledge about the quality model may lead to potential problems being overlooked. On the other hand, misunderstanding the quality model may also lead to inappropriate or erroneously identified problems.

9 Conclusions and Lessons Learned

The applications of the static approach gave good feedback to the quality model developers. Using this information, they were able to improve the evaluated quality model.

We observed that even if a detailed questionnaire is available, an instructor guiding the domain experts during the review session improves the quality and comparability of the obtained review results.

Completeness values of more than 80% should not be expected or set as a target since capturing all aspects and providing all relevant measures for product quality is an illusory goal. In order to obtain reliable completeness estimates, a larger number of domain experts should be involved.

Using the capture-recapture models enables an early validation of the models and do not require historical data, though it remains an estimation.

In the future we will continue addressing the problem of validating quality models prior to their application.

Acknowledgments

Parts of this work have been funded by the German Federal Ministry for Education and research projects Quamoco (grant 01IS08023) and SINNODIUM (grant 01IC12S01F). We gratefully acknowledge all reviewers for their participation.

References

1. Abran A, Desharnais JM, St. Pierre D, and Symons C (2007): COSMIC Funtional size measurement method.
2. Anagnostopoulos I and Stavropoulos P (2006): Adopting wildlife experiments for web evolution estimations: the role of an AI web page classifier. In: IEEE/WIC/ACM International Conference on Web Intelligence, WI 2006, pp. 897-901.
3. Basili V, Caldiera G, and Rombach D (1994): Goal question metric paradigm. In: J. Marciniak, ed. Encyclopedia of Software Engineering. John Wiley and Sons. Inc., New York, pp. 528-532.
4. Beaver JM, Schiavone GA, and Berrios JS (2005): Predicting software suitability using Bayesian belief networks. In: Proceedings of the 4th International Conference on Machine Learning and Applications (ICMLA ’05), IEEE Computer Society, pp. 82-88.
5. Böhning D (2008): Recent developments in capture-recapture methods and their applications. Biometrical Journal, 50(6), pp. 954-956.

6. Briand LC., El Emam K, Freimut BG, and Laitenberger O (2000): A comprehensive evaluation of capture-recapture models for estimating software defect content. In: IEEE Transactions on Software Engineering, 26(6), 518-540.
7. Göb A and Lochmann K (2011): A software quality model for SOA. In: Proceedings of the 8th international workshop on Software quality, pp.18-25.
8. Grady RB and Caswell DL (1987): Software metrics: establishing a company-wide program. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.
9. Gupta S and Thakur N (2010): Using capture-recapture method for web intelligence. In: IEEE International Conference Computational Intelligence and Computing Researh (ICCIC), pp.1-5.
10. ISO/IEC 25010 (2009): Software engineering - Software product quality requirements and evaluation (SQuaRE) - Quality model and guide.
11. ISO/IEC 9126-1 (2001): Software engineering - Software product quality - Part 1: Quality model.
12. Kläs M, Elberzhager F, Münch J, Hartjes K, and von Graevemeyer O (2010): Transparent combination of expert and measurement data for defect prediction - An industrial case study. In: Proceedings of the 32nd International Conference on Software Engineering (ICSE 2010), pp. 119-128.
13. Kitchenham B (1987): Towards a constructive quality model: Part1: Software quality modelling, measurement and prediction. Software Engineering Journal, 2(4), pp. 105-113.
14. Lampasona C, Heidrich J, Basili VR, and Ocampo A (2012): Software quality modeling experiences at an oil company. In: Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 243-246.
15. Lincke R, Gutzmann T, and Löwe W (2010): Software quality prediction models compared. In: Proceedongs of the 10th International Conference on Quality Software (QSIC), pp.82-91.
16. Marchetto A and Trentini A (2007): A framework to build quality models for web applications. Computer Journal of the International Arab Journal of Information Technology, 4(2), pp. 168-176.
17. Mayr A, Plösch R, Kläs M, Lampasona C, and Saft M (2012): A comprehensive code-based quality model for embedded systems - Systematic development and validation by industrial projects. In: Proceedings of the 23rd IEEE International Symposium on Software Reliability Engineering (ISSRE 2012), pp. 281-290.
18. McCall J, Richards P, and Walters G (1977): Factors in software quality, Volume-III.: US Rome Air Development Center Reports.
19. Petersson H, Thelin T, Runeson P, and Wohlin C (2004): Capture-recapture in software inspections after 10 years research. J Syst Softw, pp. 249-264.

20. Schneidewind N (1992): Methodology for validating software metrics. IEEE Transactions on Software Engineering, 18(5), pp. 410-422.
21. Vander Wiel SA and Votta LG (1993): Assessing software designs using capture-recapture methods. IEEE Transactions on Software Engineering, 19(11), pp. 1045-1054.
22. Wagner S, Lochmann K, Winter S, Goeb A, Klaes M, and Nunnenmacher S (2010): Software quality models in practice: survey results. Available online, last accessed 2 July 2013: http://wck.me/1zZ
23. Wagner S, Lochmann K, Heinemann L, Kläs M, Trendowicz A, Plösch R, Seidl A, Goeb A, and Streit J (2012): The Quamoco product quality modelling and assessment approach. In: IEEE Computer Society: 34th International Conference on Software Engineering. ICSE '2012, pp. 1133-1142.
24. Wagner S, Lochmann K, Winter S, Goeb A, and Kläs M (2009): Quality models in practice. A preliminary analysis. In: Proceedings of 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), pp. 464-467.
25. Xu J, Ho D, and Capretz LF (2010): An empirical study on the procedure to derive software quality estimation models. International Journal of Computer Science and Information Technology (IJCSIT), 2(4), pp. 1-16.