Software quality assessment model: a systematic mapping study

Meng YAN[1], Xin XIA[2,*], Xiaohong ZHANG[3], Ling XU[3], Dan YANG[3] and Shanping LI[1]

Citation: SCIENCE CHINA Information Sciences **62**, 191101 (2019); doi: 10.1007/s11432-018-9608-3

View online: https://engine.scichina.com/doi/10.1007/s11432-018-9608-3

View Table of Contents: https://engine.scichina.com/publisher/scp/journal/SCIS/62/9

Published by the Science China Press

SCIENCE CHINA
Information Sciences

• REVIEW •

September 2019, Vol. 62 191101:1–191101:18
https://doi.org/10.1007/s11432-018-9608-3

Software quality assessment model: a systematic mapping study

Meng YAN[1], Xin XIA[2*], Xiaohong ZHANG[3], Ling XU[3], Dan YANG[3] & Shanping LI[1]

[1]*College of Computer Science and Technology, Zhejiang University, Hangzhou* 310027*, China;*
[2]*Faculty of Information Technology, Monash University, Melbourne* 3800*, Australia;*
[3]*School of Software Engineering, Chongqing University, Chongqing* 401331*, China*

Received 8 April 2018/Accepted 7 September 2018/Published online 26 July 2019

Abstract Quality model is regarded as a well-accepted approach for assessing, managing and improving software product quality. There are three categories of quality models for software products, i.e., definition model, assessment model, and prediction model. Quality assessment model (QAM) is a metric-based approach to assess the software quality. It is typically regarded as of high importance for its clear method on how to assess a system. However, the current state-of-the-art in QAM research is under limited investigation. To address this gap, the paper provides an organized and synthesized summary of the current QAMs. In detail, we conduct a systematic mapping study (SMS) for structuring the relevant articles. We obtain a total of 716 papers from the five databases, and 31 papers are selected as relevant studies at last. In summary, our work focuses on QAMs from the following aspects: software metrics, quality factors, aggregation methods, evaluation methods and tool support. According to the analysis results, our work discovers five needs that researchers in this area should continue to address: (1) new method and criteria to tailor a quality framework (i.e., structure of software metrics and quality factors) according to different specifics, (2) systematic investigations on the effectiveness, strength and weakness of different aggregation methods to guide the method selection in different context, (3) more investigations on evaluating QAMs in the context of industrial cases, (4) further investigations or real-world case studies on the QAMs related tools, and (5) building a public and diverse software benchmark which can be adopted in different application context.

Keywords software quality, systematic mapping study, quality assessment model, aggregation method

Citation Yan M, Xia X, Zhang X H, et al. Software quality assessment model: a systematic mapping study. Sci China Inf Sci, 2019, 62(9): 191101, https://doi.org/10.1007/s11432-018-9608-3

1 Introduction

Quality model is a well-accepted mean to describe and control the software quality. According to ISO/IEC 14598-1 [1], a quality model is a set of characteristics and the relationships between them which provide the basis for specifying requirements and evaluating quality. It has become a significant way for providing adequate confidence information that software products conform to requirements. The information is mainly used for quality assurance, decision making, costs estimating, and risk evaluation in software development and maintenance [2].

Along with the quality model provided by Boehm et al. [3], a multitude of diverse models for software products were proposed. Among them, several models have been developed or standardized, e.g., ISO 9126 [4] and ISO 25010 [5]. Some of them have been adopted or developed to evaluate the quality of industrial software projects and to predict project defects [6, 7]. Based on their different purposes, Deissenboeck et al. [8] classified these quality models into three categories, i.e., definition model, assessment

* Corresponding author (email: xin.xia@monash.edu)

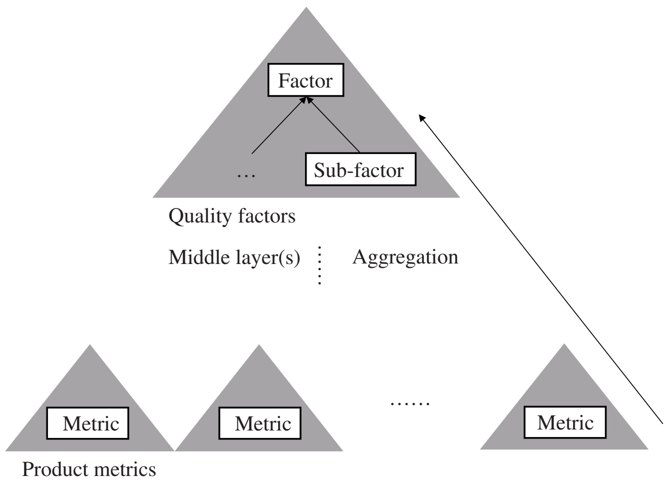

Figure 1 The general concept of a QAM. A QAM depends on an aggregation step to aggregate metrics to quality factors. Similar to ISO 9126, the factors may be decomposed into sub-factors. In addition, there may be several number of middle layers between metrics and quality factors to make the model more comprehensive, such as the practice layer and criteria layer in Squale model [13].

model, and prediction model. A definition model is mainly used to define or describe quality [4, 9]. A general shortcoming in most definition models is that the given definitions are mostly too abstract to perform constructive quality assurance. Many of them are often unclear as to how practitioners conduct the model operations [8]. An assessment model contains quality criteria with clear methods to assess each quality criterion. The assessment method is often a mathematical model which aggregates product metrics (identical with measures in this work) to quality factors. Under this way, an assessment model determines the value of quality factors. It is noted that a quality factor is a management-oriented attribute of software that contributes to its quality. It has many synonyms in this line of research, such as quality characteristic, quality aspect, quality attributes and qualities [10, 11]. Moreover, the requirements used in assessment models hold in prediction models as well. Additionally, a prediction model can support predictions to aid further activities, such as defect prediction.

Among the three kinds of quality models, software assessment models are typically regarded as of high importance for their clear guidance on how to assess a system [8]. A software product quality assessment model (QAM) helps bridge the gap between software metrics and software product quality factors. The common features of QAMs are listed below. First, a QAM contains a set of factors and metrics according its purpose and usage. The factors in many of the QAMs are often derived from the same international standard, such as ISO 9126 or ISO 25010 [5]. Since different QAMs possess different purposes and context, the metrics adopted in the different QAMs vary. Second, a QAM is a hierarchical model. This describes a decomposition of the general product quality into sub-qualities to make them easier to be understood and controlled [11]. They usually depend on an aggregation method to aggregate software metrics to quality factors [12] as Figure 1 shows. Third, a QAM is an automatic or semi-automatic process. A tool which implements the QAM can assist users to adopt and popularize the model. However, many of the QAMs have not been implemented into a tool. In addition, among the existing tools, some of which still stayed in an academic usage and did not meet the expectations of practitioners.

There are several studies that review software quality models, factors or tools. For example, Klas et al. [14] presented a comprehensive criteria to classify quality models which is named as CQML. Montagud et al. [15] analyzed existing quality factors and attributes for software product lines (SPL) in a systematic review. Riaz et al. [16] analyzed the maintainability models in their systematic review. Febrero et al. [17] studied software reliability models in their mapping study. Kitchenham [18] focused on the software metrics and aimed at identifying the trends in commonly used metrics (e.g., OO metrics and web-metrics). Tomas et al. [19] presented a review study that focused on open source tools to automatically collect software metrics in Java. However, most of the above-mentioned studies focused on general quality model, the current state-of-the-art in QAM research is under limited investigation. To address this gap, our work aims at providing an organized and synthesized summary of the published QAMs.

The goal of this work is to concentrate on QAMs with particular respect to provide an organized and synthesized summary. To accomplish this goal, we performed a systematic mapping study (SMS). A SMS is a methodology to systematically analyze a research topic in order to provide an overview of a research area through classification and counting contributions in relation to the classified categories [17, 20]. In detail, the specific goals of this paper are as follows.

• **To identify the categories of software metrics and quality factors used in QAMs.** Software metrics and quality factors constitute the framework of a QAM. The framework describes the model objective and the input elements. We identify the categories of the software metrics and quality factors to provide an overview.

• **To summarize the aggregation methods used in QAMs.** The aggregation method is the most important difference between an assessment model and a definition model. We summarize the aggregation methods to provide an overview for the aggregation step in QAMs. In particular, we summarize their ideas, advantages, disadvantages, and open issues of the aggregation methods.

• **To summarize the evaluation methods used for validating QAMs.** Model evaluation is an important aspect for indicating the effectiveness of QAM. We summarize the evaluation methods to provide an overview and analysis for the validation step in QAMs.

• **To identify the current tools that implement QAMs.** Tool implementation can help users for using and popularizing a QAM. We identify the current tools that implement the QAMs to provide an overview and their usage (i.e., industrial or academic).

• **To identify the research challenges for further improvement of QAMs.** The goal of this mapping study is to identify further needs that researchers in this area should continue to address. Therefore, we identify the most challenging aspects for further improvement of QAMs.

This paper extends our preliminary study [21] published in a conference. In summary, the main extensions are as follows.

• We add the summarization of the aggregation methods used in QAMs (see Subsection 3.2). In detail, we identify seven aggregation methods in selected studies. For each aggregation method, we identify which studies use this method and summarize the basic idea, advantages and disadvantages of this method. At last, we summarize two open issues in terms of aggregation method.

• We add the identification of the current challenges for further improvement of QAMs (see Subsection 3.5). In detail, we identify two challenges for the improvement, i.e., enhancing model diversity and building software benchmark.

• We restructure the research questions for more comprehensive. For example, we merge the software metrics and factors (i.e., RQ1 and RQ2 in our conference paper respectively) into one research question (RQ1 in this paper). In terms of the extended analysis, we insert the summarization of aggregation methods as the second research question in this paper (RQ2) and insert the research challenges as the fifth research question. Additionally, we add more details of the selected studies, i.e., the publication venues.

The following parts of this paper are structured as follows. We provide our research questions, mapping study process and an overview of selected studies in Section 2. We report the answers for each research question in Section 3. We report the threats to validity in Section 4. We describe the related studies of reviewing software quality models in Section 5. We draw the conclusion and provide our future plans in Section 6.

2 Systematic mapping process

Our systematic mapping study aims to identify, structure, and classify software quality assessment models according to five research questions. This section reports the research questions and details of the steps that we perform in this systematic mapping study according to the guidelines provided by [20].

Table 1 Research questions and motivations

Research question	Motivation
RQ1. What metrics and factors are commonly used by QAMs?	Identify the categories and trends of metrics and quality factors used in QAMs.
RQ2. What are the current aggregation methods used by QAMs?	Summarize current aggregation methods in QAMs.
RQ3. What are the current validation methods used for evaluating QAMs?	Summarize the current validation methods in QAMs.
RQ4. What are the current usage of tools based on QAMs?	Identify the current usage of the related tools.
RQ5. What are the challenges for the improvement of QAMs?	Identify the challenges for the further improvement.

Table 2 Selected databases

Database	Location
ISI web of knowledge	isiknowledge.com
Scopus	www.scopus.com
IEEE Xplore	www.ieeexplore.ieee.org
ACM digital library	www.portal.acm.org
Springer	link.springer.com

2.1 Research questions

Raising appropriate research questions is considered as of high importance for a systematic survey. It helps to provide structured and insightful findings in a specific field [22]. Table 1 presents the five research questions and related motivations in our study. First, RQ1 is raised to identify the adopted metrics and factors. Second, we analyze the aggregation methods for QAMs to answer RQ2. Third, RQ3 is raised to answer what kinds of validation methods are currently used for evaluating QAMs. Fourth, based on the studied QAMs, several tools have been proposed. RQ4 identifies the current state of these tools. The objective of the question is to describe the usage states of current tools based on these QAMs. The final research question (RQ5) addresses the current challenges for further improvement directions and opportunities.

2.2 Search strategy

The searching step is directly conducted through searching on the publication databases online by using a set of tailored strings. The databases utilized in this work are chosen using the following criteria: (1) the database contains publications that are relevant to the software quality model area; (2) the database is adopted or suggested in previous software engineering related reviews. Five of the largest and most complete scientific databases are selected as the search databases (see Table 2). IEEE Xplore, ACM digital library and Springer are widely recognized as being an efficient means to perform reviews [23]. The ISI web of knowledge is suggested by Chernyi [24] and Scopus is suggested by Kitchenham [18] in conducting review.

The search strings we used in this paper are created by using the following steps under the guideline [20]:

(1) Identify main search words from the research questions;

(2) Identify the keywords in relevant papers;

(3) Refine the keywords by identifying alternative synonyms for the search words in a thesaurus;

(4) Construct search strings by concatenating semantically similar words with Boolean OR;

(5) Construct search strings by concatenating the restricted words with Boolean AND;

(6) Generate advanced search strings for the different databases.

At last, the resulting search strings in different databases are shown in Table 3.

Table 3 Search strings in different databases

Database	Search string and settings
ISI web of knowledge	TS=(((“software quality model*”) OR (“software quality” AND “quality model*”)) AND (metric* OR measure*))
Scopus	TITLE-ABS((“software quality model*”) OR (“software quality” AND “quality model*”) AND (metric* OR measure*))
IEEE Xplore	((((“Document title”: “software quality model*” OR (“software quality” AND “quality model*”)) OR (Abstract: “software quality model*” OR (“software quality” AND “quality model*”))))) AND ((“Document title”: “metric*” OR “measure*”) OR (“Abstract”: “metric*” OR “measure*”))
ACM digital library	(((Title: “software quality model*”) OR ((Title: “software quality”) and (Title: “quality model*”))) OR ((Abstract: “software quality model*”) OR ((Abstract: “software quality”) and (Abstract: “quality model*”)))) AND ((Title: “metric*”) OR (Title: “measure*”) OR (Abstract: “metric*”) OR (Abstract: “measure*”))
Springer	‘ “software quality model” or (“software quality” and “quality model”) and (“measure*” or “metric*”)’

Table 4 Overview of search result

Stage	Papers	Added papers	Total papers
Stage 1: by search strings	716	0	716
Stage 2: by title and abstract	128	0	128
Stage 3: by content	28	3	31

Table 5 Inclusion criteria (IC) and exclusion criteria (EC)

IC	Description
1	The paper proposes a software quality assessment model.
2	The paper is based on software product metrics.
3	The paper focuses on software product quality rather than process quality.
4	The paper presents a hierarchical mapping model which aggregates metrics to factors.
EC	Description
1	The paper focuses on software process quality.
2	The paper focuses on a prediction model without an assessment model.
3	The paper only provides a definition quality model without an assessment method.
4	The paper is not accessible.
5	The document is not a paper, such as a conference cover, poster, etc.
6	The paper is not written in English.

2.3 Study selection

The selection of studies in this review is divided into three stages. In the first stage, the initial selection of studies is based on the search strings. As a result, there are 716 papers in our first stage as shown in Table 4.

In the second stage, we focused on the study inclusion and exclusion criteria. Regarding our research questions, the inclusion and exclusion criteria are shown in Table 5. There are two aspects to guide this criteria. First, there are many studies contain the keyword “software quality”, such as software product quality, software process quality and software defect prediction. In our work, we focus on software product quality which is a counterpart to process quality. Second, as stated in the introduction, we focus on the software quality assessment model, which includes software metrics, factors and aggregation methods. Those studies which only define a quality model or focus on software defect prediction are out of the scope of this paper. In this stage, we closely examined the title and abstract of each paper according to the inclusion and exclusion criteria. As a result, there are 128 papers which have the relevant titles and abstracts. These papers denote that they may be useful for the motivation of this review through the titles and abstracts. However, more verification efforts are required to examine them by reading the

Table 6 Detailed information of selected studies

Study ID	Title	Year
S1 [29]	Operationalised product quality models and assessment: the Quamoco approach	2015
S2 [30]	CLOUDQUAL: a quality model for cloud services	2014
S3 [28]	Efficiency measurement of Java Android code	2014
S4 [31]	Objective safety compliance checks for source code	2014
S5 [32]	SCQAM: a scalable structured code quality assessment method for industrial software	2014
S6 [33]	Test code quality and its relation to issue handling performance	2014
S7 [34]	Indirect method to measure software quality using CK-OO suite	2013
S8 [35]	MIDAS: a design quality assessment method for industrial software	2013
S9 [36]	Objective measurement of safety in the context of IEC 61508-3	2013
S10 [37]	A comprehensive code-based quality model for embedded systems	2012
S11 [38]	Standardized code quality benchmarking for improving software maintainability	2012
S12 [39]	The Quamoco product quality modelling and assessment approach	2012
S13 [2]	A probabilistic software quality model	2011
S14 [40]	Integrated software quality evaluation: a fuzzy multi-criteria approach	2011
S15 [41]	Evaluate the quality of foundational software platform by Bayesian network	2010
S16 [42]	Quality models for free/libre open source software – towards the “silver bullet” ?	2010
S17 [43]	The consortium for IT software quality	2010
S18 [44]	The SQALE analysis model: an analysis model compliant with the representation condition for assessing the quality of software source code	2010
S19 [45]	OQMw: An OO quality model for web applications	2009
S20 [13]	The Squale model – a practice-based industrial quality model	2009
S21 [46]	DEQUALITE: building design-based software quality models	2008
S22 [47]	2-D software quality model and case study in software flexibility research	2008
S23 [48]	The EMISQ method and its tool support-expert-based evaluation of internal software quality	2008
S24 [49]	The SQO-OSS quality model: measurement based open source software evaluation	2008
S25 [26]	Legacy system exorcism by Pareto’s principle	2005
S26 [50]	Construction of a systemic quality model for evaluating a software product	2003
S27 [27]	Software product and process assessment through profile-based evaluation	2003
S28 [51]	Using quality models in software package selection	2003
S29 [52]	A hierarchical model for object-oriented design quality assessment	2002
S30 [53]	Multi-criteria methodology contribution to the software quality evaluation	2001
S31 [54]	Software quality measurement: concepts and fuzzy neural relational model	1998

contents.

In the third stage, it is necessary to examine the contents of the selected papers from the second stage which have relevant titles or abstracts. According to the inclusion and exclusion criteria, 28 papers are selected in this stage. Finally, an additional search process is necessary to enhance the completeness of the selected studies. As a result, we considered two clues when conducting the additional search: (1) examining satisfied (i.e., satisfy the inclusion criteria) papers in the stage 3 by reviewing the references in the selected studies. (2) examining satisfied papers by reviewing citations in the selected studies [25]. Under this way, three more studies [26–28] which satisfy our inclusion criteria were selected in the additional search. Specially, if the paper does not present the whole description of the QAM, we obtain the detail information from other related sources, such as the technical reports and the model’s homepage. The summary of all the selected studies is shown in Table 6 in chronological order.

Additionally, we identify the publication venues of selected studies as shown in Table 7. Since quality model is a research topic which multiple areas, including software quality, software evolution, software testing and machine learning, the publication venues for the selected papers are diverse. The most frequent venues are *Software Quality Journal* (SQJ), International Conference on Software Engineering (ICSE), IEEE International Conference on Software Maintenance and Evolution (ICSME, also konwn as ICSM before 2014), *IEEE Transactions on Software Engineering* (TSE) and Euromicro Conference on

Table 7 Publication venues of selected studies

Publication venue	Type	# of papers
ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications	Conference	1
International Conference on Computational Intelligence for Modelling Control & Automation	Conference	1
IEEE International Conference on Fuzzy Systems at the IEEE World Congress on Computational Intelligence	Conference	1
IEEE International Conference on Software Maintenance and Evolution (ICSME)	Conference	2
International Conference on Data and Software Engineering (ICODSE)	Conference	1
International Conference on Intelligent Systems and Signal Processing (ISSP)	Conference	1
International Conference on Software Engineering (ICSE)	Conference	3
International Conference on Program Comprehension (ICPC)	Conference	1
International Symposium on Software Reliability Engineering (ISSRE)	Conference	1
International Conference on Advances in System Testing and Validation Lifecycle	Conference	1
Euromicro Conference on Software Engineering and Advanced Applications (SEAA)	Conference	2
Innovations in Systems and Software Engineering	Journal	1
International Journal of Software Engineering and Knowledge Engineering	Journal	1
Information and Software Technology (IST)	Journal	1
IEEE Transactions on Software Engineering (TSE)	Journal	2
IEEE Transactions on Industrial Informatics	Journal	1
IEEE Software	Journal	1
Journal of Information Processing Systems	Journal	1
Lecture Notes in Computer Science	Journal	1
Open Source Development, Communities and Quality	Journal	1
Pattern Languages of Programs	Journal	1
Software Quality Journal (SQJ)	Journal	3
Tamkang Journal of Science and Engineering	Journal	1
Software Engineering Approaches for Offshore and Outsourced Development	Book	1

Software Engineering and Advanced Applications (SEAA).

3 Results

3.1 RQ1: what metrics and factors are commonly used by QAMs?

This subsection provides the details of the metrics and factors currently used in the selected studies.

3.1.1 *What kinds of metrics are commonly used by QAMs?*

This subsection provides the details of the metrics currently used in the selected studies. We generalized the software metrics used in QAMs into categories based on [19, 55], and we also extended them based on the extra categories found in the selected studies. These eleven categories are listed as follows:

• Complexity metrics. They are derived from McCabe complexity [56] and Halstead complexity [57].

• Design metrics. This category captures the design related metrics, such as OO metric [58], modularization, design pattern, and dependencies metric [59].

• Code entity size metrics. Code entity size metrics are often used in a normalized way combined with other metrics, such as lines of code, number of classes, lines per method and Non-comment lines of code.

• Comment size metrics. They are often measured in order to quantify documentation and understandability, such as density of comment lines and ratio of comment lines to code.

• Coding conventions violations. The number of coding conventions violations is usually used as a quality determinant for readability and maintainability. For Java projects, the Sun Code Conventions are the most well-known coding conventions.

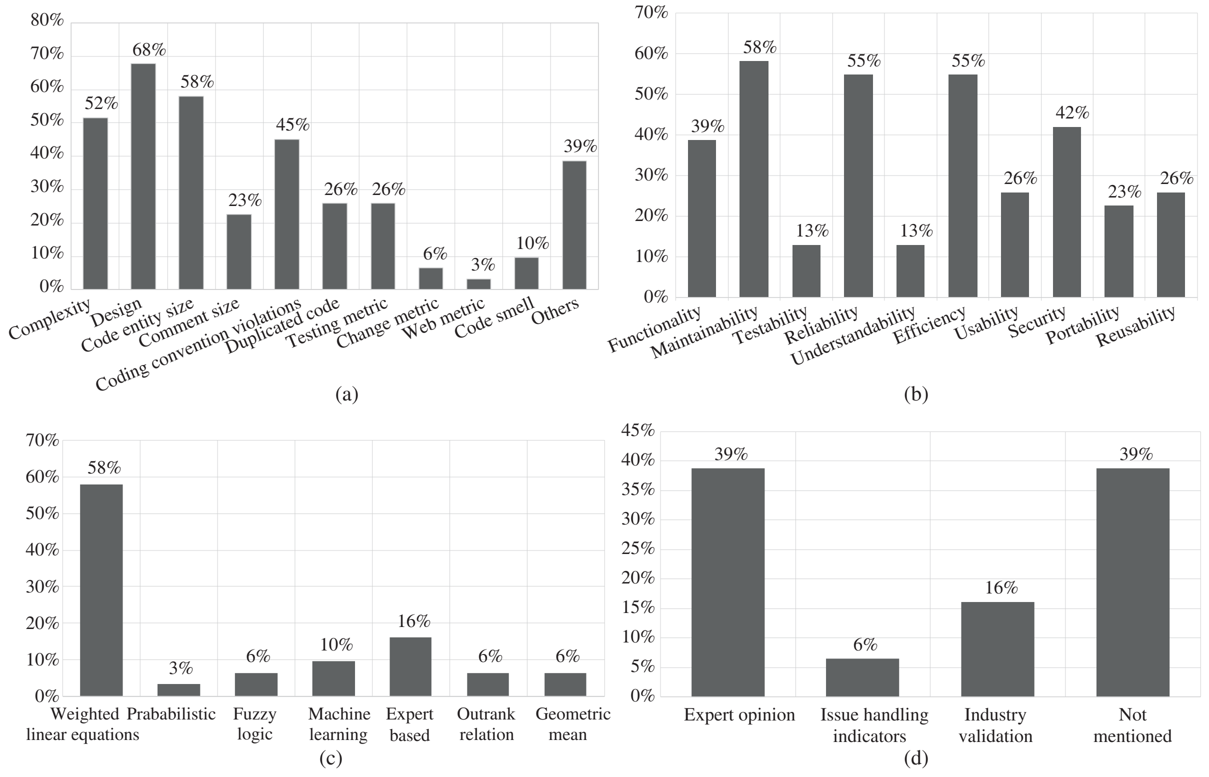

Figure 2 Distribution of papers by different features. (a) Metric category; (b) factors; (c) aggregation methods; (d) evaluation methods.

• Code smells. They are derived from the literature [60] and often used to define the possible refactoring because of the potential bugs.

• Duplicated code. A measure of the size of duplicated code, such as duplicated lines, duplicated blocks or duplicated tokens. It is often used as an indicator of maintainability and readability.

• Testing metrics. To measure what percentage of code has been tested by a test suite, such as function coverage and statement coverage.

• Change metrics. To measure what degree of change has been made in a revision, such as function change and mean change size.

• Web metrics. To measure the particular properties of web applications, such as navigation paths length and page click-stream distances [45].

• Others. Several of the selected QAMs contain both product metrics and non-product metrics. Others represent the metrics which are out of the design and product scope, such as defect metrics in issue tracking systems, requirement documentation and project community.

Figure 2(a) provides a graphic representation of the metric category proportion distribution, the size of each bar represents the proportion of the selected studies which adopted the metric category. For all the QAMs, since the decomposition principles used for factors usually dependent on the manual experience and application specifics as Deissenboeck et al. [61] stated, the adopted metrics are various according to the model objective and context. In summary, the most popular metrics used in QAMs are complexity (52%, used by S4, S5, S6, S11, S13, S14, S17, S18, S19, S20, S21, S22, S24, S25, S27 and S29), design (68%, used by S1, S4, S6, S7, S8, S9, S11, S12, S13, S14, S17, S18, S19, S20, S21, S22, S24, S25, S26, S29 and S31) and code entity size (58%, used by S1, S4, S6, S11, S12, S13, S14, S16, S17, S18, S19, S20, S21, S22, S24, S25, S27 and S29) metrics. Additionally, the code entity size metrics are traditional metrics which are often used in combination with other metrics [62]. Many studies also include the size metrics while using complexity and design metrics (S4, S6, S11, S13, S14, S17, S18, S19, S20, S21, S22, S24, S25, and S29). This evidence indicates that the three basic software metric categories are frequently used as quality determinants. The use of metrics like coding conventions violations, code smells and web metrics varies in different model contexts. For example, coding conventions violations and code smells vary in different program languages; web metrics are only suitable for web applications.

3.1.2 *What factors are commonly used by QAMs?*

Quality factor is a management-oriented attribute of software that contributes to its quality. It has many synonyms in this line of research, such as quality characteristic, quality aspect, quality attributes and qualities (e.g., maintainability, reliability) [10, 11]. We synthesize all the factors used in the selected studies. Most of the studied QAMs assess the software quality through multiple factors. Although we combine the synonyms, (e.g., functionality and functional, usability and utilization) there are still 42 factors occurred in all the selected studies. However, most of them only appeared in a few studies, such as compatibility in S1 and S12. To identify the commonly used factors, Figure 2(b) provides the graphic representation of the top ten factors which are mostly appeared in the selected studies, the value of each bar represents the proportion of the selected studies which adopted the factor. The most commonly focused factors are maintainability (58%, used by S1, S5, S6, S7, S10, S11, S12, S13, S14, S16, S17, S18, S20, S23, S24, S26, S28, S31), reliability (55%, used by S1, S2, S7, S12,S14, S15, S16, S17, S18, S19, S20, S23, S24, S26, S27, S28, S31) and efficiency (55%, used by S1, S3, S5, S6, S7, S8, S12, S14, S15, S16, S17, S18, S23, S26, S28, S29, S30) which are also stressed by the ISO 25010 and CISQ [43, 63]. This evidence indicates that maintainability, reliability and efficiency factors are frequently used as quality factors for assessment.

3.2 RQ2: what are the current aggregation methods used by QAMs?

Aggregation method in a QAM is used to aggregate software metrics into high-level factors. It is an important part of any assessment model and a reoccurring task. The choice of the appropriate aggregation method has a strong influence on the results. However, the choices of the aggregation methods are rarely justified in this line of research [11]. For example, only weighted linear equations are mentioned for aggregation in the IEEE standard 1061 [64]. This subsection identifies the current aggregation methods in the selected studies and analyzes the strength and weakness of each kind of method. Figure 2(c) shows the current aggregation methods distribution. One study may adopt two or more method categories. For example, S6 used both geometric mean and weighted linear equations. On the other hand, S14 used a fuzzy weighted average approach which contained two categories in Figure 2(c), namely weighted linear equations and fuzzy logic.

In the following sections, we will report each aggregation method listed in Figure 2(c). For each aggregation method, two aspects are considered in our discussion, i.e., motivations and difficulties. We try to address the following two questions: why the aggregation methods are proposed? And what the difficulties in the method operationalizing?

At the final subsection, we will discuss two typical open issues correlated to the aggregation method. These issues do not have a uniform standard for various models. (1) Normalization. Since the value ranges of metrics are different, normalization is a common way to calibrate the metric values before the aggregation step. This aspect reports what normalization methods are adopted in the studies. (2) Quality index categorization. This aspect reports what kinds of the quality index outputs in the studies (e.g., star index or numeric index in a certain range).

Weighted linear equations. This method aggregates metrics towards the factors by adopting a weighted linear combination. Weighted linear equations are the most commonly used aggregation method (58%, used by S1, S2, S3, S4, S6, S7, S9, S10, S11, S12, S14, S16, S17, S18, S19, S20, S25 and S29). The reason behind this phenomenon is that this method is simple to calculate and easy to interpret by practitioners. There are two issues in this aggregation method. First, the weighted linear equation method requires the metric values have interval scales. In real cases, many metrics are in an ordinal scales or judgments on ordinal scales (e.g., poor, average and excellent) [27]. If this method has to handle one or more ordinal scales, it should transform other scales into ordinal scales as well. This brings arbitrary information which may lead to a unfair assessment [27]. Second, how to decide the correct weights is a difficult task. A usually way is to adopt expert opinions. For example, S1 and S12 form the relevant rankings based on expert opinions and then calculate the weights from the relevant rankings by using the rank-order centroid method [65]. However, introducing the expert opinions may make the aggregation subjective [27]. A survey conducted among IT professionals has concluded that using subjective weights does not improve the model due to the lack of consensus among developers [66].

Probabilistic. Bakota et al. [2] proposed a probabilistic aggregation method for calculating the factors. It aggregated metrics into factors relying on a probabilistic "goodness" function. The motivation is that this method has addressed two questions. First, most of the previous researches used simple weighting or linear combination to aggregate, while this method defines a probabilistic model in the aggregation. Second, the value or the category of the quality factor is represented by one number or a category, while this method uses a probabilistic distribution which integrates the ambiguity coming from the lack of consensus.

On the other hand, there are also two difficulties in applying this method. First, the method depends on a benchmark which was used to produce the probabilistic goodness function. In our selected studies, six studies (S1, S4, S6, S11, S12 and S13) used a benchmark in quality modeling process. There are two typical benchmarks. Namely, S6 and S11 used the same benchmark provided by SIG and S13 used the benchmark provided by University of Szeged. One difficulty that prevents the method from popularization is that these benchmarks are not publicly accessible. Second, this method is not simple enough for developers. Wagner et al. [29] stated that practitioners have difficulties in interpreting such probabilistic distributions in their experience.

Fuzzy logic. S14 and S31 adopted the fuzzy logic based approach in the aggregation step. This method adopts triangular fuzzy sets to denote the quality ratings and weights. The ratings and weights are represented as a fuzzy membership function. It indicates the degree of the specific inputs to describe the node [40]. In the aggregation, fuzzy operations [67] are employed to conduct the multiplication and addition operations. This method was proposed to address the fuzziness or uncertainty in estimating the parameters in quality modeling. For example, simple linear combination methods are not completely reliable because of the numeric weighting values assigned to different characteristics are changeable and inconsistent [40]. While in this method, fuzzy logic enables one to infer definite insights from highly imprecise, vague and ambiguous data. However, there are also several difficulties in this method. First, it is difficult to decide the triangular fuzzy sets and fuzzify the different metrics. There can be different criteria to fuzzify various metrics. This relies on the experiences of different experts using this method. Second, there is an assumption to perform this method. Namely, the factors and sub-factors have been prioritized appropriately. However, there is not a well-accepted criteria to perform this task in various environments.

Machine learning. S15, S21 and S22 provided a machine learning aggregation method which created construction of rules to calculate quality factors. A common feature of this method is that it needs to learn the patterns between the metrics and factors from prior data. In detail, S15 and S22 proposed an assessment model by using Bayesian networks to learn and evaluate a given system by using Bayesian network probabilistic to reason. S21 adopted the JRip and J48 methods to learn rule sets which can create the construction between design patterns and quality factors. This method is suitable for mining the hidden patterns and uncertain knowledge from prior data. However, the difficulties are also obvious. It needs to introduce domain knowledge and predefine some prior cause-effect relationships in the learning step through questionnaires or other kinds of surveys. For example, the learning data of S15 were collected by questionnaires form 50 domain experts. And the learning criteria of S22 was collected by questionnaires from 20 developers. This may suffer from the bias from the knowledge of the participants and bring the difficulty in the work reoccurring.

Expert based. S5, S8, S23, S26 and S28 provided an expert based assessing approach. A common feature of this method is that the assessment process is usually performed by combining the code analyzers and expert review. Usually, the aggregation step of the method is a semi-automatic process, since it substantially relied on the experts and the knowledge base such as pre-defined guidelines, checklists and templates [32]. Despite the insightful results in automatic assessment models, there are still some unsolved problems. This method is proposed to address the false positive problems in the automatic assessment models. For example, some metrics or rules which are detected by an automatic tool may be false positive. The falsely detected metrics or rules may limit the performance of automatic assessment approaches [48, 68]. On the other hand, an obvious weakness is that it is expensive to conduct an assessment. Additionally, similar to the difficulties in the machine learning method, the experience and background of the experts play a significant role in the assessment. This may introduce the bias which comes from the knowledge and experience of participants.

Outrank relation. S24 and S27 proposed the outrank relation based approach to build the quality assessment model. Outrank relation is often used to aid decision making in voting and social choice theory [69, 70]. In the operation, the basic thinking of this method is: accomplish a choice or a ranking task on a set of alternatives and each alternative is compared to all other alternatives in turn. The difference between this method and the weighted linear equations method is that this method can handle metric values with an ordinal scale (e.g., pool, fair, excellent). Since the aggregation operator plays a significant role in guaranteeing a correct result, it is necessary to consider the semantics of each operator, the correlated metrics and factors [27]. This method is proposed to address the semantic uncertainty in the aggregation. However, similar to the manual review step in expert based method, the difficult part in this method is that evaluators are necessary to establish the comparison. Besides, if there is a strong incomparability between the alternatives and the profiles, further discussion with the decision makers is demanded.

Geometric mean. S6 and S30 adopted the geometric mean method in the aggregation step. Compared to the arithmetic mean method, geometric mean is proposed to handle some specific conditions, such as the aggregation from length, height and depth to the volume of an object [27]. One suitable scene of using this method is that it requires all the son nodes have a high score since the global score deteriorates exponentially with respect to the importance of the son nodes. For example, S6 proposed a test code quality model. The global quality of the test code requires all three factors (completeness, effectiveness, maintainability) are of high quality. In addition, the three factors are not prior with each other. Therefore, they adopted the geometric mean method and the three factors are of the same importance. However, the geometric mean method is not suitable for all the conditions. The choice of the arithmetic mean and the geometric mean depends on the semantics between the son node and the parent node.

In addition, there are two open issues in aggregation methods.

(1) Normalization. In terms of the metric calibration, there is a difficulty in the aggregation. The ranges of the metrics are various. For example, SLOC and DIT take their values in different intervals. In this case, the aggregation has to consider not to dilute the results of one metric into the other [71]. In order to compose these metrics in a uniform interval, normalization is a common solution in the selected studies. There are several typical categories of the normalization methods in the selected studies. The first category is normalization based on size metrics. For example, S3, S17 and S25 adopted this normalization method. A general way is to divide the metric values or the quality scores by software size metrics, such as total lines of code, total number of classes and total number of packages. The second category is normalization based on comparing the values to the benchmark base. For example, S1, S4 and S10 adopted this method. Usually, the benchmarking base contains a large number of systems. Then, the normalization is conducted by comparing the metric values of the product under assessing to those values in the benchmarking base. The advantage is that this enables us to explore if the product is better or worse than other products. The third category is normalization based on a pre-defined threshold function for each metric. The function can be linear or non-linear according to the semantics of the metric. For example, the Squale model proposed in S20 adopted this method. Among the three categories of normalization method, the first category is the simplest one, because it does not require prior knowledge. The second and the third methods require prior knowledge or experience, such as benchmark and thresholds, which may prevent the model from generalization.

(2) Quality index categorization. This aspect reports what kinds of the quality index outputs in the studies. There are two typical kinds of the quality index outputs. The first kind is a numeric index in a range. For example, S1, S2, S3, S4, S5, S12 and S13 output a numeric quality index. Another kind

is the categorical quality index (e.g., a star index or school grades). For example, S6, S7, S10 and S11 output a categorical quality index. Comparing to the categorical quality index, the advantage of the numeric quality index is that there is no loss of information [2]. Comparing to the numeric quality index, the advantage of the categorical quality index output is that it is more comprehensive for the project managers at the first sight. We suggest that if the model is proposed for serious academic usage, a numeric output quality index is better. For example, the probabilistic quality model of S13 requires the numeric output to build a more precise benchmark. While if the model is proposed to be implemented as a product for various stakeholders, the categorical quality index is a better choice. For example, the SQALE [44] plugin for SonarQube adopts various color to represent the quality index of various code entities (green is fine and red is poor). Such that the mangers, testers and developers can locate the high-risk files or packages quickly.

3.3 RQ3: what are the current validation methods used for evaluating QAMs?

Model validation is significant because of its practical application. It is used to evaluate whether the QAMs provide valid and insightful assessment results. The difficulty lies in evaluating the performance on the same environment. In our selected studies, there are three categories of the validation methods: expert opinion, issue handling indicators, and industry validation. Among the three methods, expert opinion is the most frequently used evaluation method (39%, used by S1, S6, S9, S10, S12, S13, S17, S21, S23, S24, S26 and S29) as Figure 2(d) shows. It is an empirical method which is often used in this line of research, especially in problems which lack a public labeled dataset. The common features of the expert opinion evaluation method are listed below. First, developers who are regarded as participants or experts possess certain experience in the area. Second, a guide or checklist is often needed for the evaluation process. The weakness of this method lies in the bias coming from the diverse expertise of the participant's background.

The quality of software correlates with the performance in handling issues, such as fixing bugs and introducing features [33]. S2 and S11 evaluated the soundness of their models through issue handling metrics. It revealed that their quality model possessed a significant positive relation with issue handling performance. Another evaluation method is industry feedback which is adopted in S4, S5, S8, S16 and S20. For example, the quality assessment model Squale [13] in S20 was designed by Air France-KLM and Qualixo company at first and its evaluation relied on the practical feedback from PSA Peugeot-Citroen and Air France-KLM. They stated that the model was well accepted by managers and developers. The similarity between the method and expert opinion lies in that both of them need participants. The difference is that the industry feedback method based on a larger and more diverse set of industrial projects and developers.

3.4 RQ4: what are the current usage of tools based on QAMs?

A tool which is implemented based on the QAMs plays a significant role in popularizing a model. It is used to facilitate automatic quality evaluation. However, the results show that most of the selected studies did not provide a tool to support the automatic assessment. In total, only eight selected studies provide a tool to assist their evaluation as Table 8 shows. We classify the usage of the tools into two categories: industrial usage and academic usage. If the tool was proposed or currently used in an industrial environment, we classify it as industrial usage. If the tool was proposed in an academic institution and there is no further clue which indicating the usage in industrial environment, we classify it as academic usage.

Table 8 lists the overview of the eight tools. The results show that half of the tools are used in an industrial environment and half of the tools stay in academic usage. This may imply that these tools which stay in academic usage because they do not well satisfy industrial environment requirements. Why are these tools not widely used in the industrial environment? We suggest that further investigations or real-world case studies should be performed to address this question. With regard to the industrial usage

Table 8 Overview of the eight tools

Tool	Related study	Current usage	Publish year	Open source
SIG quality model	S6, S11	Industrial	2007	No
Quamoco	S1, S12	Industrial	2008	Yes
FSQQT	S14	Academic	2011	No
SQALE	S18	Industrial	2010	No
Squale	S20	Industrial	2008	Yes
SPQR	S23	Academic	2008	No
Alitheia	S24	Academic	2008	Yes
Xradar	S25	Academic	2004	Yes

tools, there are two open source tools, namely Squale provided in study S20 and Quamoco provided in study S1 and S12. The other two tools require purchase.

3.5 RQ5: what are the challenges for the improvement of QAMs?

We note two challenges for the improvement of QAMs.

Model diversity. This refers to that the QAM takes into account diverse application context. However, in the selected studies, 84% of them only consider particular model applicable context. There are three model context categories, namely program language type, code file type and application type. First, since several software metrics (e.g., coding conventions violations) depend on the languages, many studies are restricted to particular program languages. For example, the QAM proposed in S3 is only able to be adopted in Java and the model proposed in S10 is only able to be adopted in projects which are written in C/C++. Second, the metrics for different code file types (e.g., test code, function code) are different. For example, the Assertions-McCabe ratio metric proposed in S6 is only applicable for test code. Thus, the QAM proposed in S6 is only able to assess the test code quality. Third, different applications possess different features. For example, the QAM proposed in S10 is restricted to embedded software and the QAM proposed in S19 is restricted to web applications. Besides, some of the selected studies (S14, S22, S26, S27 and S31) did not mention the model context. Fortunately, several QAMs have begun to support diverse application context. For example, the Quamoco model proposed in S1 and the SQALE model proposed in S18 had been adopted to support diverse languages, such as C/C++, C#, Java, embedded Ada and COBOL. This aspect is the challenge and opportunity for the improvement of QAMs. In the future, more work is needed to enhance the diversity of QAMs.

Software benchmarks. Although many of the QAMs are derived from an international standard (e.g., ISO 9126 and ISO 25010), there is not a standard in calculating quality indices from source code metrics. A main challenge is that there is not a comparison with other systems [38]. A software benchmark is a repository to provide such information. It stores the results from a lot of standard software evaluations and it is usually used for learning thresholds or normalization. When each evaluation is performed, the benchmark will be updated. In the selected studies, several studies (S1, S4, S6, S11, S12 and S13) used a benchmark. S6 and S11 used the benchmark proposed by SIG. It holds evaluation results for around 200 systems including open source and industrial projects. The probabilistic model in S13 proposed a benchmark which consists of 100 systems written in Java, including both open source and industrial projects. However, most of the QAMs do not adopt a software benchmark. Although there are several existing benchmarks, they are not publicly accessible and only include particular kind of projects. Building a public benchmark which consists of more and diverse systems is a significant requirement for the future.

4 Threats to validity

Despite the fact that our work is performed by following the systematic mapping guide line, there may be several threats to the validity.

Lack of universal taxonomy. Obviously, software quality is a topic that is relevant to many software engineering fields, including software defect prediction, software requirement and process quality, software product quality. All these relevant studies may be termed as "software quality". There is not an appropriate and widely used taxonomy for the QAM. For example, they may not mention the "product" and "assessment" in the titles and abstracts. Thus, in order to include the correct studies as completely as possible, we use "software quality" in the search string to mitigate this problem. The primary search results may include the studies in all the above fields. Then, we make a manual selection by reading the titles and abstracts (reading contents if needed) according to the inclusion and exclusion criteria.

Study selection bias. We are not aware of biases we may have had for selecting studies in a survey [72]. Improperly selected search terms and inclusion/exclusion criteria may lead to attrition bias. Some relevant papers may not been found in the databases under our search and selection criteria. However, the search step relied on both criteria: the databases and the quality of the studies. The used databases cover the software engineering research well and we manually read the titles and abstracts (reading contents if needed) of each alternative to decide the selection. Therefore, we are reasonably confident that we are unlikely to have missed many significant relevant studies.

Study completeness. According to our inclusion and exclusion criteria, 31 papers are selected as relevant studies at last. We identified these 31 papers following the scientific guideline for performing systematic mapping study in software engineering [20]. Thus, we believe that the threat to find other relevant papers is limited.

Finding repeatability. The goal of this paper is to provide an organized and synthesized summary of software quality assessment models. Although we identified five research directions, we did not claim these directions are better than others. A different scholar may identify other research directions with the same set of papers. To mitigate this issue, we provided the detailed steps of conducting this survey to ensure the repeatability of the data collection and statistics, and two authors worked independently to identify the research directions. The five research directions are based on the discussions of the two authors.

5 Related work

There are several studies related to reviewing software quality models which support the establishment of this work. Deissenboeck et al. [8] classified existing quality models into definition models, assessment models and prediction models. According to this classification, they described the purpose and the scenario for the usage of each category. Similar to their work, Klas et al. [14] presented a comprehensive criteria for classifying quality models which is named as CQML. The classification scheme helps to obtain the summary and the relationship of existing quality models. It is organized by the following dimensions: object, purpose, quality focus and resource. The difference between the above-mentioned two studies and our work lies in two aspects. First, they aim to provide a classification scheme rather than a reviewing study, while in our work, we try to review the existing papers on one category (i.e., assessment models) according to Deissenboeck's definition by a mapping study. Second, they provide the guide of how to classify a quality model, we aim to provide a systematical state-of-the-art in QAM research by focusing on QAM specific aspects, such as metrics, factors, evaluation methods.

Montagud et al. [15] focused on reviewing the existing quality measures and attributes for SPL in a systematic review. They found 165 measures and 91 different quality attributes. The similarity with our work is that both of us focus on the product quality. The difference is that they aimed to classify general measures and attributes, while this work focuses on the assessment models.

In addition, there exist similar related systematic reviews which focus on a particular factor of software quality. For example, Riaz et al. [16] focused on the maintainability predicting methods and metrics in their systematic review. They selected 15 relevant studies to synthesize the forecasting methods, metrics and factors, validation methods in maintainability forecasting. The similarity with this work is that both of us focus on the methods, metrics and factors. Febrero et al. [17] presents a mapping study to analyze and structure the literature on software reliability modeling. They investigated the overview of relevant literature, research topics and adopted models. Different from the above two reviews; our work aims to review the integrated quality model rather than a particular quality factor.

There are several related systematic reviews or mapping studies which focus on metrics and tools in software quality modeling. Kitchenham [18] focused on the software metrics and aimed at identifying the trends in commonly used metrics (e.g., OO metrics and web-metrics). A preliminary mapping study was presented in Kitchenham's work which tried to synthesize the relevant published papers. Tomas et al. [19] presents a review study of the currently used open source software tools that automate the collection of software metrics in Java. Many of the tools in their work implemented a software quality model, such as Squale [13] and SQALE [44] which are also reviewed in this work.

6 Conclusion and future work

The main contribution of this work is to provide a systematic mapping study of quality assessment models for software products. Five databases are searched and a total of 716 studies are obtained. Finally, according to our inclusion and exclusion criteria, 31 studies are selected as relevant studies which are taken into consideration for addressing our five research questions. These studies are organized according to their publication attributes and our research questions. The synthesized data extracted from them allow us to observe the development of QAMs from the following aspects: software metrics, quality factors, aggregation methods, evaluation methods, tool support, model context and benchmark.

In summary, the conclusions are drawn as follows. (1) QAMs are dependent on the application context, the structure of quality factors and metrics adopted in different QAMs are various in different model context. One problem is that few studies propose a guideline in how to construct the quality framework from metrics to factors in different application context. Researchers in this area should continue to investigate the guideline and criteria to tailor a quality framework (i.e., structure of quality metrics and factors) according to different specifics. (2) There are seven aggregation methods in the selected studies. Each method has its advantages and difficulties. One issue remains unanswered is that how to select the appropriate aggregation method in different context. Few studies have been proposed to conclude the effectiveness, strength and weakness of different aggregation methods to guide the method selection in different context. Further investigations are required to explore which aggregation method is the appropriate choice in different environments. (3) We observe that model evaluation is a difficult task due to the lack of standard data. It needs to be noted that only a few systematic industrial case studies have been published to evaluate the quality assessment model. Therefore, more research is required to investigate the benefits and problems of applying QAMs in the context of industrial cases. (4) Only a small portion of the selected studies provide a tool to implement the automatic assessment. Among these tools, many of them are not widely used in the industrial environment. This may imply that these tools do not well satisfy industrial environment requirement. Further investigations or real-world case studies should be performed to address this question. (5) In terms of the future challenges and needs, we suggest to improve the model diversity and build a public and diverse software benchmark which consists of different kinds of both open source and industrial projects.

In the future, we plan to enhance the existing QAMs by addressing the current challenge and needs. For example, we plan to organize a public and diverse benchmark for model construction and evaluation. It will be beneficial to enhance the diversity of QAMs in different application context. In an addition, we plan to perform case studies on diverse categories of real-word industrial systems to track the benefits and problems of the existing QAMs.

Acknowledgements This work was partially supported by National Key Research and Development Program of China (Grant No. 2018YFB1003904), National Natural Science Foundation of China (Grant No. 61602403), and China Postdoctoral Science Foundation (Grant No. 2017M621931).

References

1 ISO. Information technology, software product evaluation. ISO/IEC 14598-1. 1999. https://www.iso.org/standard/24902.html

2 Bakota T, Hegedus P, Kortvelyesi P, et al. A probabilistic software quality model. In: Proceedings of the 27th IEEE International Conference on Software Maintenance, 2011. 243–252

3 Boehm B W, Brown J R, Kaspar H. Characteristics of Software Quality. Amsterdam: North-Holland Publishing, 1978

4 ISO. Software engineering-product quality. ISO/IEC 9126. 2001. https://www.iso.org/standard/22749.html

5 ISO. Systems and software engineering-systems and software quality requirements and evaluation (SQuaRE)-system and software quality models. ISO/IEC 25010. 2011. https://www.iso.org/standard/35733.html

6 Catal C, Diri B. A systematic review of software fault prediction studies. Expert Syst Appl, 2009, 36: 7346–7354

7 Öztürk M M, Cavusoglu U, Zengin A. A novel defect prediction method for web pages using k-means++. Expert Syst Appl, 2015, 42: 6496–6506

8 Deissenboeck F, Juergens E, Lochmann K, et al. Software quality models: purposes, usage scenarios and requirements. In: Proceedings of the ICSE Workshop on Software Quality, 2009. 9–14

9 Dromey R G. A model for software product quality. IEEE Trans Softw Eng, 1995, 21: 146–162

10 ISO. Systems and software engineering – vocabulary. ISO/IEC/IEEE 24765. 2010. https://www.iso.org/standard/50518.html

11 Wagner S. Software Product Quality Control. Berlin: Springer, 2013

12 Chotjaratwanich U, Arpnikanondt C. A visualization technique for metrics-based hierarchical quality models. In: Proceedings of the 19th Asia-Pacific Software Engineering Conference (APSEC), 2012. 733–736

13 Mordal-Manet K, Balmas F, Denier S, et al. The Squale model — a practice-based industrial quality model. In: Proceedings of the IEEE International Conference on Software Maintenance, 2009. 531–534

14 Klas M, Heidrich J, Munch J, et al. CQML scheme: a classification scheme for comprehensive quality model landscapes. In: Proceedings of the EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2009. 243–250

15 Montagud S, Abrahão S, Insfran E. A systematic review of quality attributes and measures for software product lines. Softw Qual J, 2012, 20: 425–486

16 Riaz M, Mendes E, Tempero E. A systematic review of software maintainability prediction and metrics. In: Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, 2009. 367–377

17 Febrero F, Calero C, Moraga M. A systematic mapping study of software reliability modeling. Inf Softw Tech, 2014, 56: 839–849

18 Kitchenham B. What's up with software metrics? — a preliminary mapping study. J Syst Softw, 2010, 83: 37–51

19 Tomas P, Escalona M J, Mejias M. Open source tools for measuring the internal quality of Java software products. A survey. Comput Stand Inter, 2013, 36: 244–255

20 Petersen K, Feldt R, Mujtaba S. Systematic mapping studies in software engineering. In: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008. 68–77

21 Yan M, Xia X, Zhang X H, et al. A systematic mapping study of quality assessment models for software products. In: Proceedings of the International Conference on Software Analysis, Testing and Evolution (SATE), 2017. 63–71

22 Kitchenham B. Guidelines for Performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report EBSE-2007-01. 2007

23 Brereton P, Kitchenham B A, Budgen D, et al. Lessons from applying the systematic literature review process within the software engineering domain. J Syst Softw, 2007, 80: 571–583

24 Chernyi A I. The ISI web of knowledge, a modern system for the information support of scientific research: a review. Sci Tech Inf Proc, 2009, 36: 351–358

25 Webster J, Watson R T. Analyzing the past to prepare for the future: writing a literature review. Manag Inf Syst Quart, 2002, 26: 3

26 Kvam K, Lie R, Bakkelund D. Legacy system exorcism by Pareto's principle. In: Proceedings of the Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, 2005. 250–256

27 Morisio M, Stamelos I, Tsoukias A. Software product and process assessment through profile-based evaluation. Int J Soft Eng Knowl Eng, 2003, 13: 495–512

28 Satrijandi N, Widyani Y. Efficiency measurement of java Android code. In: Proceedings of the International Conference on Data and Software Engineering, 2014

29 Wagner S, Goeb A, Heinemann L, et al. Operationalised product quality models and assessment: the Quamoco approach. Inf Softw Tech, 2015, 62: 101–123

30 Zheng X R, Martin P, Brohman K, et al. CLOUDQUAL: a quality model for cloud services. IEEE Trans Ind Inf, 2014, 10: 1527–1536

31 Mayr A, Plosch R, Saft M. Objective safety compliance checks for source code. In: Proceedings of the 36th International

Conference on Software Engineering, 2014. 115–124
32 Gupta S, Singh H K, Venkatasubramanyam R D, et al. SCQAM: a scalable structured code quality assessment method for industrial software. In: Proceedings of the 22nd International Conference on Program Comprehension, 2014. 244–252
33 Athanasiou D, Nugroho A, Visser J, et al. Test code quality and its relation to issue handling performance. IEEE Trans Softw Eng, 2014, 40: 1100–1125
34 Srivastava S, Kumar R. Indirect method to measure software quality using CK-OO suite. In: Proceedings of the International Conference on Intelligent Systems and Signal Processing, 2013. 47–51
35 Samarthyam G, Suryanarayana G, Sharma T. Midas: a design quality assessment method for industrial software. In: Proceedings of the International Conference on Software Engineering, 2013. 911–920
36 Mayr A, Plosch R, Saft M. Objective measurement of safety in the context of IEC 61508-3. In: Proceedings of the EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2013. 45–52
37 Mayr A, Plosch R, Klas M, et al. A comprehensive code-based quality model for embedded systems: systematic development and validation by industrial projects. In: Proceedings of the 23rd International Symposium on Software Reliability Engineering (ISSRE), 2012. 281–290
38 Baggen R, Correia J P, Schill K, et al. Standardized code quality benchmarking for improving software maintainability. Softw Qual J, 2012, 20: 287–307
39 Wagner S, Lochmann K, Heinemann L, et al. The Quamoco product quality modelling and assessment approach. In: Proceedings of the International Conference on Software Engineering, 2012. 1133–1142
40 Challa J S, Paul A, Dada Y, et al. Integrated software quality evaluation: a fuzzy multi-criteria approach. J Inf Proc Syst, 2011, 7: 473–518
41 Lan Y Q, Liu Y F, Kuang M X. Evaluate the quality of foundational software platform by Bayesian network. In: Proceedings of International Conference on Artificial Intelligence and Computational Intelligence, 2010
42 Glott R, Groven A K, Haaland K, et al. Quality models for free/libre open source software towards the "silver bullet"? In: Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2010. 439–446
43 Soley R M, Curtis B. The consortium for IT software quality. In: Proceedings of International Conference on Software Quality, 2013
44 Letouzey J L, Coq T. The sqale analysis model: an analysis model compliant with the representation condition for assessing the quality of software source code. In: Proceedings of the International Conference on Advances in System Testing and Validation Lifecycle, 2010. 43–48
45 Marchetto A. OQMw: an OO quality model for web applications. Tamkang J Sci Eng, 2009, 12: 459–470
46 Khomh F, Gueheneuc Y G. DEQUALITE: building design-based software quality models. In: Proceedings of the 15th Conference on Pattern Languages of Programs, 2008
47 Li Z, Lin L, Hui G. 2-d software quality model and case study in software flexibility research. In: Proceedings of the International Conference on Intelligence for Modelling Control and Automation, 2008. 1147–1152
48 Plösch R, Gruber H, Hentschel A, et al. The EMISQ method and its tool support-expert-based evaluation of internal software quality. Innov Syst Softw Eng, 2008, 4: 3–15
49 Samoladas I, Gousios G, Spinellis D. The SQO-OSS quality model: measurement based open source software evaluation. In: Proceedings of IFIP International Conference on Open Source Systems, 2008. 237–248
50 Ortega M, Pérez M, Rojas T. Construction of a systemic quality model for evaluating a software product. Softw Qual J, 2003, 11: 219–242
51 Franch X, Carvallo J P. Using quality models in software package selection. IEEE Softw, 2003, 20: 34–41
52 Bansiya J, Davis C G. A hierarchical model for object-oriented design quality assessment. IEEE Trans Softw Eng, 2002, 28: 4–17
53 Blin M J, Tsoukiás A. Multi-criteria methodology contribution to the software quality evaluation. Softw Qual J, 2001, 9: 113–132
54 Pedrycz W, Peters J F, Ramanna S. Software quality measurement: concepts and fuzzy neural relational model. In: Proceedings of the IEEE International Conference on Fuzzy Systems Proceedings, 1998. 1026–1031
55 Gomez O, Oktaba H, Piattini M, et al. A systematic review measurement in software engineering: state-of-the-art in measures. In: Poroceedings of International Conference on Software and Data Technologies, 2008. 165–176
56 McCabe T J. A complexity measure. IEEE Trans Softw Eng, 1976, 2: 308–320
57 Halstead M H. Elements of Software Science. New York: Elsevier Science Inc., 1977
58 Chidamber S R, Kemerer C F. A metrics suite for object oriented design. IEEE Trans Softw Eng, 1994, 20: 476–493
59 Martin R C. Agile Software Development: Principles, Patterns, and Practices. Upper Saddle River: Prentice Hall PTR, 2003
60 Fowler M. Refactoring: Improving the Design of Existing Code. Boston: Addison-Wesley Longman Publishing, 1999
61 Deissenboeck F, Wagner S, Pizka M, et al. An activity-based quality model for maintainability. In: Proceedings of

the IEEE International Conference on Software Maintenance, 2007. 184–193

62 Malhotra R. A systematic review of machine learning techniques for software fault prediction. Appl Soft Comput, 2015, 27: 504–518

63 Ploesch R, Schuerz S, Koerner C. On the validity of the it-cisq quality model for automatic measurement of maintainability. In: Proceedings of the 39th Annual Computer Software and Applications Conference (COMPSAC), 2015. 326–334

64 IEEE. IEEE recommended practice for software requirements specifications. In: Institute of Electrical and Electronics Engineers, 1998

65 Barron F H, Barrett B E. Decision quality using ranked attribute weights. Manage Sci, 1996, 42: 1515–1523

66 Correia J P, Kanellopoulos Y, Visser J. A survey-based study of the mapping of system properties to ISO/IEC 9126 maintainability characteristics. In: Proceedings of the IEEE International Conference on Software Maintenance, 2009. 61–70

67 Ross T J. Fuzzy Logic with Engineering Applications. Hoboken: John Wiley and Sons, 2009

68 Heckman S, Williams L. A systematic literature review of actionable alert identification techniques for automated static code analysis. Inf Softw Tech, 2011, 53: 363–387

69 Köksalan M, Ulu C. An interactive approach for placing alternatives in preference classes. Eur J Oper Res, 2003, 144: 429–439

70 Larichev O. An approach to ordinal classification problems. Int Trans Oper Res, 1994, 1: 375–385

71 Mordal K, Anquetil N, Laval J, et al. Software quality metrics aggregation in industry. J Softw Evol Proc, 2013, 25: 1117–1135

72 Jorgensen M, Shepperd M. A systematic review of software development cost estimation studies. IEEE Trans Softw Eng, 2007, 33: 33–53