A Systematic Mapping Study of Quality Assessment Models for Software Products

Meng Yan*‡, Xin Xia†✓, Xiaohong Zhang‡✓, Ling Xu‡, and Dan Yang‡
*College of Computer Science and Technology, Zhejiang University, Hangzhou, China
†Department of Computer Science, University of British Columbia, Canada
‡School of Software Engineering, Chongqing University, Chongqing, China
Email: mengy@zju.edu.cn, xxia02@cs.ubc.ca, {xhongz, xuling, dyang}@cqu.edu.cn

***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, evaluation methods and tool support.**

***Index Terms*—Software quality, Quality assessment model, Systematic mapping study**

I. 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]–[5].

Along with the quality model provided by Boehm et al. [6], a multitude of diverse models for software products were proposed. Among them, several models have been developed or standardized, e.g., ISO 9126 [7] and ISO 25010 [8]. Some of them have been adopted or developed to evaluate the quality of industrial software projects and to predict project defects [9], [10]. Based on their different purposes, Deissenboeck et al. classified these quality models into three categories, i.e., definition model, assessment model, and prediction model [11]. A definition model is mainly used to define or describe quality [7], [12]. 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 [11]. 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 [13]. 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 [11]. However, 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.

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 [8]. 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 [14]. They usually depend on an aggregation method to aggregate software metrics to quality factors [15] as Figure 1 shows. Third, a QAM is an automatic or semi-automatic process. A tool which implements the QAM can assist users in adopting and popularizing 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.

The goal of this work is to concentrate on QAMs with particular respect to provide an organized and synthesized summary in terms of above-mentioned features. To accomplish this goal, we performed a systematic mapping study (SMS).

✓Corresponding authors.

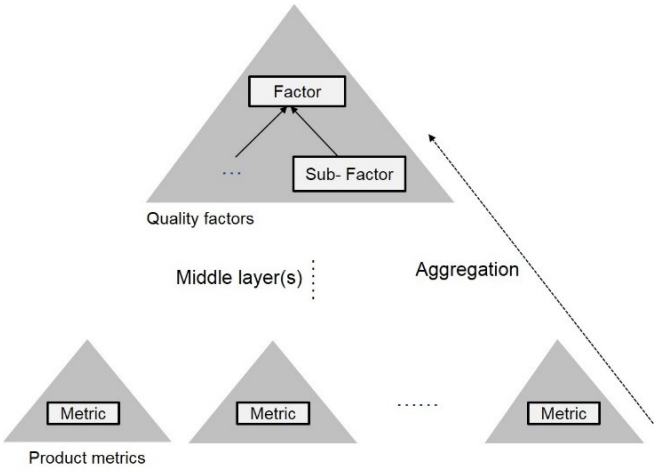

Fig. 1. The general concept of a QAM. A QAM depends on an aggregation step to aggregate metrics to quality factors. Similar to ISO 9126, the factors may be decomposed into sub-factors. In addition, there may be several number of middle layers between metrics and quality factors to make the model more comprehensive, such as the practice layer and criteria layer in Squale model [18].

TABLE I
RESEARCH QUESTIONS AND MOTIVATIONS

Research question	Motivation
RQ1. What software metrics are commonly used by QAMs?	Identify the categories and trends of metrics used in QAMs
RQ2. What quality factors are commonly used by QAMs?	Identify the categories and trends of quality factors used in QAMs
RQ3. What are the current validation methods used for evaluating QAMs?	Identify 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

TABLE II
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

A SMS is a methodogy 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 [16], [17]. In detail, the specific goals of this article are as follows: (1) To identify the categories of software metrics and quality factors used in QAMs. (2) To synthesize which means are currently used to evaluation QAMs. (3) To organize the current tools which implement these QAMs.

This article is 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 describe the related works reviewing software quality models in Section 5. We draw the conclusions and provide our future plans in Section 6.

II. SYSTEMATIC MAPPING PROCESS

Our systematic mapping study aims to identify, structure, and classify software quality assessment models according to four 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 [17].

A. 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 [19]. Table I presents the three research questions and related motivations in our study. First, RQ1 is raised to identify the adopted software metrics. Second, RQ2 is raised to identify the adopted quality factors. Third, RQ3 is raised to answer what 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.

B. 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 II). IEEE Xplore, ACM Digital Library and Springer are widely recognized as being an efficient means to perform reviews [20]. The ISI Web of Knowledge is suggested by Chernyi [21] and Scopus is suggested by Barbara Kitchenham [22] in conducting review.

The search strings we used in this work are created by using the following steps under the guideline [17]: (1) Obtain main search words from the research questions. (2) Obtain the keywords in relevant papers. (3) Refined the words 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) Form the advanced search strings for the different databases. At last, the resulting search strings in different databases are shown in Table III.

C. 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. This initial search was performed

TABLE III
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*")'

between May 2015 and June 2015. As a result, there are 716 papers in our first stage as shown in Table VI.

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 IV and Table V respectively. There are two aspects to guide this criteria. First, there are many works 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 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 [23]. Under this way, three more studies [24]–[26] 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 VII in chronological order.

TABLE VI
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

III. RESULTS

A. RQ1: What software metrics are commonly used by QAMs?

This section provides the details of the metrics currently used in the selected studies. We generalize the software metrics used in QAMs into categories based on the literatures [53], [54], and we also extended them based on the extra categories found in the selected studies. These 11 categories are listed as follows:

Complexity metrics. They are derived from McCabe complexity [55] and Halstead complexity [56].

Design metrics. This category captures the design related metrics, such as OO metric [57], modularization, design pattern, and dependencies metric [58], [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.

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

TABLE IV
INCLUSION CRITERIA

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.

TABLE V
EXCLUSION CRITERIA

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.

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 [43].

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 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.

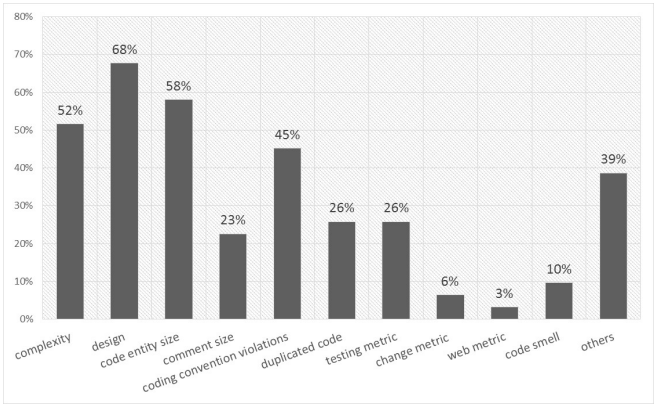

Fig. 2. Distribution of papers by metric category

B. RQ2: What quality factors are commonly used by QAMs?

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 happened in a few studies, such as compatibility in S1 and S12. To identify the commonly used factors, Figure 3 provides the graphic representation of the top ten factors which are mostly happened in the selected studies,the value of each bar represents the proportion of the selected studies which adopted the factor. It is seen that the most commonly focused factors are maintainability (58%, used by

TABLE VII
DETAILED INFORMATION OF SELECTED STUDIES

Study ID	Titile	Year
S1 [27]	Operationalised product quality models and assessment: The Quamoco approach	2015
S2 [28]	CLOUDQUAL: A Quality Model for Cloud Services	2014
S3 [26]	Efficiency Measurement of Java Android Code	2014
S4 [29]	Objective safety compliance checks for source code	2014
S5 [30]	SCQAM: A Scalable Structured Code Quality Assessment Method for Industrial Software	2014
S6 [31]	Test code quality and its relation to issue handling performance	2014
S7 [32]	Indirect Method to Measure Software Quality using CK-OO suite	2013
S8 [33]	MIDAS: A Design Quality Assessment Method for Industrial Software	2013
S9 [34]	Objective Measurement of Safety in the Context of IEC 61508-3	2013
S10 [35]	A Comprehensive Code-based Quality Model for Embedded Systems	2012
S11 [36]	Standardized code quality benchmarking for improving software maintainability	2012
S12 [37]	The Quamoco Product Quality Modelling and Assessment Approach	2012
S13 [2]	A probabilistic software quality model	2011
S14 [38]	Integrated Software Quality Evaluation: A Fuzzy Multi-Criteria Approach	2011
S15 [39]	Evaluate the Quality of Foundational Software Platform by Bayesian Network	2010
S16 [40]	Quality models for Free/Libre Open Source Software - towards the "Silver Bullet"?	2010
S17 [41]	The Consortium for IT Software Quality	2010
S18 [42]	The SQALE Analysis Model: An analysis model compliant with the representation condition for assessing the Quality of Software Source Code	2010
S19 [43]	OQMw: An OO Quality Model for Web Applications	2009
S20 [18]	The Squale Model - A Practice-Based Industrial Quality Model	2009
S21 [44]	DEQUALITE: building design-based software quality models	2008
S22 [45]	2-D Software Quality Model and Case Study in Software Flexibility Research	2008
S23 [46]	The EMISQ method and its tool support-expert-based evaluation of internal software quality	2008
S24 [47]	The SQO-OSS quality model: Measurement based open source software evaluation	2008
S25 [24]	Legacy System Exorcism by Pareto's Principle	2005
S26 [48]	Construction of a Systemic Quality Model for Evaluating a Software Product	2003
S27 [25]	software product and process assessment through profile-based evaluation	2003
S28 [49]	Using quality models in software package selection	2003
S29 [50]	A hierarchical model for object-oriented design quality assessment	2002
S30 [51]	Multi-Criteria Methodology Contribution to the Software Quality Evaluation	2001
S31 [52]	Software quality measurement: Concepts and fuzzy neural relational model	1998

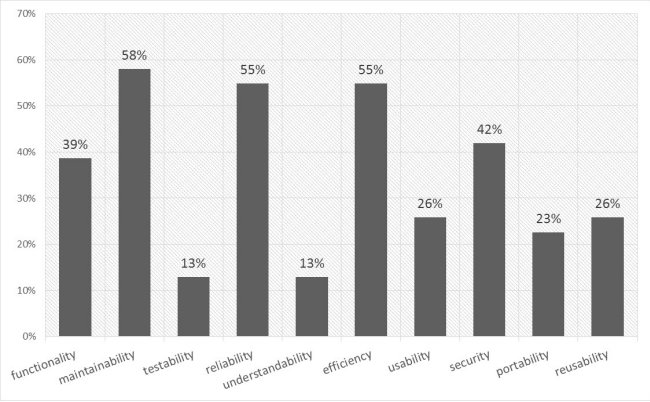

Fig. 3. Distribution of papers by factors

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 [41], [63]. We believe this fact is occurred because they are representative and significant quality aspects for software product.

C. 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 4 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 which 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 participantąŕs background.

The quality of software correlates with the performance in handling issues, such as fixing bugs and introducing features [31]. 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

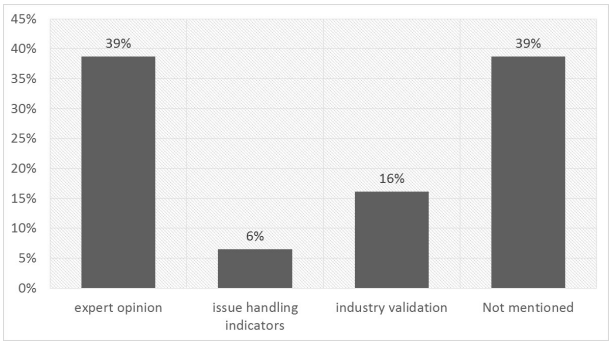

Fig. 4. Distribution of papers by evaluation methods

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 [18] 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.

D. 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 VIII 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 VIII 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 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.

IV. 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, etc. 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 [64]. 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. Though the definition model, assessment model and prediction model are defined for various purposes, they are not distinctly independent of each other [11]. For example, a prediction model cannot predict the quality without knowing how to assess it. Some prediction models many also contain an assessment model. Therefore, we cannot claim that we have collected all the software assessment models; however, we believe that we have captured a significant and typical set.

V. Related Work

There are several studies related to reviewing software quality models which support the establishment of this work. Deissenboeck et al. [11] 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. [65] 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 works 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

TABLE VIII
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

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. [66] focused on reviewing the existing quality measures and attributes for software product lines (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. [67] 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. [16] 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. Barbara Kitchenham [22] 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 Kitchenhamąś work which tried to synthesize the relevant published papers. Tomas et al. [54] 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 [18] and SQALE [42] which are also reviewed in this work.

VI. 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, 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) 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. (3) 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.

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.

Acknowledgment. This work was partially supported by NSFC Program (No. 61602403 and 61572426), National Key Technology R&D Program of the Ministry of Science and Technology of China (No. 2015BAH17F01), and Chongqing Research Program of Basic Science & Frontier Technology (No. cstc2017jcyjB0305).

REFERENCES

[1] I. ISO, "Iso/iec 14598-1," *Information Technology, Software Product Evaluation*, 1999.

[2] T. Bakota, P. Hegedűs, P. Körtvélyesi, R. Ferenc, and T. Gyimóthy, "A probabilistic software quality model," in *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*. IEEE, 2011, pp. 243–252.

[3] M. Yan, X. Zhang, C. Liu, J. Zou, L. Xu, and X. Xia, "Learning to aggregate: an automated aggregation method for software quality model," in *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE Press, 2017, pp. 268–270.

[4] M. Yan, X. Xia, X. Zhang, D. Yang, and L. Xu, "Automating aggregation for software quality modeling," in *ICSME*. IEEE, 2017, p. to appear.

[5] M. Yan, Y. Fu, X. Zhang, D. Yang, L. Xu, and J. D. Kymer, "Automatically classifying software changes via discriminative topic model: Supporting multi-category and cross-project," *Journal of Systems and Software*, vol. 113, pp. 296–308, 2016.

[6] B. W. Boehm, J. R. Brown, and H. Kaspar, "Characteristics of software quality," 1978.

[7] I. ISO, "Iec 9126-software engineering-product quality," *International Organization for Standardization*, 2001.

[8] ——, "Iec 25010," *Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models*, 2011.

[9] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7346–7354, 2009.

[10] M. M. Ozturk, U. Cavusoglu, and A. Zengin, "A novel defect prediction method for web pages using k-means++," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6496–6506, 2015.

[11] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, "Software quality models: Purposes, usage scenarios and requirements," in *ICSE Workshop on Software Quality*, 2009, Conference Proceedings, pp. 9–14.

[12] R. G. Dromey, "A model for software product quality," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146–162, 1995.

[13] I. ISO, "Ieee, systems and software engineering–vocabulary," ISO/IEC/IEEE 24765: 2010 (E)) Piscataway, NJ: IEEE computer society, Report, 2010.

[14] S. Wagner, *Software product quality control*. Springer, 2013.

[15] U. Chotjaratwanich and C. Arpnikanondt, "A visualization technique for metrics-based hierarchical quality models," in *Software Engineering Conference (APSEC), 2012 19th Asia-Pacific*, vol. 1. IEEE, 2012, pp. 733–736.

[16] F. Febrero, C. Calero, and M. A. Moraga, "A systematic mapping study of software reliability modeling," *Information and Software Technology*, vol. 56, no. 8, pp. 839–849, 2014.

[17] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *EASE*, 2008, Conference Proceedings.

[18] K. Mordal-Manet, F. Balmas, S. Denier, S. Ducasse, H. Wertz, J. Laval, F. Bellingard, and P. Vaillergues, "The squale modeląła practice-based industrial quality model," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*. IEEE, 2009, pp. 531–534.

[19] B. Kitchenham, *Guidelines for performing systematic literature reviews in software engineering*. Keele University, 2007.

[20] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of systems and software*, vol. 80, no. 4, pp. 571–583, 2007.

[21] A. I. Chernyi, "The isi web of knowledge, a modern system for the information support of scientific research: a review," *Scientific and Technical Information Processing*, vol. 36, no. 6, pp. 351–358, 2009.

[22] B. Kitchenham, "What's up with software metrics? - a preliminary mapping study," *Journal of Systems and Software*, vol. 83, no. 1, pp. 37–51, 2010.

[23] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *Management Information Systems Quarterly*, vol. 26, no. 2, p. 3, 2002.

[24] K. Kvam, R. Lie, and D. Bakkelund, "Legacy system exorcism by pareto's principle," in *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. ACM, 2005, pp. 250–256.

[25] M. Morisio, I. Stamelos, and A. Tsoukias, "Software product and process assessment through profile-based evaluation," *International Journal of Software Engineering and Knowledge Engineering*, vol. 13, no. 05, pp. 495–512, 2003.

[26] N. Satrijandi and Y. Widyani, "Efficiency measurement of java android code," in *Data and Software Engineering (ICODSE), 2014 International Conference on*. IEEE, 2014, pp. 1–6.

[27] S. Wagner, A. Goeb, L. Heinemann, M. Kläs, C. Lampasona, K. Lochmann, A. Mayr, R. Plösch, A. Seidl, J. Streit *et al.*, "Operationalised product quality models and assessment: The quamoco approach," *Information and Software Technology*, vol. 62, pp. 101–123, 2015.

[28] Z. Xianrong, P. Martin, K. Brohman, and X. Li Da, "Cloudqual: A quality model for cloud services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1527–1536, 2014.

[29] A. Mayr, R. Plösch, and M. Saft, "Objective safety compliance checks for source code," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 115–124.

[30] S. Gupta, H. K. Singh, R. D. Venkatasubramanyam, and U. Uppili, "Scqam: a scalable structured code quality assessment method for industrial software," in *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 2014, pp. 244–252.

[31] D. Athanasiou, A. Nugroho, J. Visser, and A. Zaidman, "Test code quality and its relation to issue handling performance," *IEEE Transactions on Software Engineering*, vol. 40, no. 11, pp. 1100–1125, 2014.

[32] S. Srivastava and R. Kumar, "Indirect method to measure software quality using ck-oo suite," in *International Conference on Intelligent Systems and Signal Processing*, 2013, Conference Proceedings, pp. 47–51.

[33] G. Samarthyam, G. Suryanarayana, T. Sharma, and S. Gupta, "Midas: a design quality assessment method for industrial software," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 911–920.

[34] A. Mayr, R. Plösch, and M. Saft, "Objective measurement of safety in the context of iec 61508-3," in *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*. IEEE, 2013, pp. 45–52.

[35] A. Mayr, R. Plösch, M. Kläs, C. Lampasona, and M. Saft, "A comprehensive code-based quality model for embedded systems: systematic development and validation by industrial projects," in *Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on*. IEEE, 2012, pp. 281–290.

[36] R. Baggen, J. P. Correia, K. Schill, and J. Visser, "Standardized code quality benchmarking for improving software maintainability," *Software Quality Journal*, vol. 20, no. 2, pp. 287–307, 2012.

[37] S. Wagner, K. Lochmann, L. Heinemann, M. Kläs, A. Trendowicz, R. Plösch, A. Seidl, A. Goeb, and J. Streit, "The quamoco product quality modelling and assessment approach," in *Proceedings of the 34th international conference on software engineering*. IEEE Press, 2012, pp. 1133–1142.

[38] J. S. Challa, A. Paul, Y. Dada, V. Nerella, P. R. Srivastava, and A. P. Singh, "Integrated software quality evaluation: A fuzzy multi-criteria approach," *JIPS*, vol. 7, no. 3, pp. 473–518, 2011.

[39] Y. Lan, Y. Liu, and M. Kuang, *Evaluate the Quality of Foundational Software Platform by Bayesian Network*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6320, book section 43, pp. 342–349.

[40] R. Glott, A.-K. Groven, K. Haaland, and A. Tannenberg, "Quality models for free/libre open source software towards the ąřsilver bulletąś?" in *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*. IEEE, 2010, pp. 439–446.

[41] R. Soley and B. Curtis, *The Consortium for IT Software Quality*, ser. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2010, vol. 54, book section 2, pp. 2–5.

[42] J.-L. Letouzey and T. Coq, "The sqale analysis model: An analysis model compliant with the representation condition for assessing the quality of software source code," in *International Conference on Advances in System Testing and Validation Lifecycle*. IEEE, 2010, Conference Proceedings, pp. 43–48.

[43] A. Marchetto, "Oqmw: An oo quality model for web applications," *Tamkang Journal of Science and Engineering*, vol. 12, no. 4, pp. 459–470, 2009.

[44] F. Khomh and Y.-G. Guéhéneuc, "Dequalite: building design-based software quality models," in *Proceedings of the 15th Conference on Pattern Languages of Programs*. ACM, 2008, p. 2.

[45] Z. Li, L. Lin, and G. Hui, "2-d software quality model and case study in software flexibility research," in *International Conference on Intelligence for Modelling Control & Automation*, 2008, Conference Proceedings, pp. 1147–1152.

[46] R. Plosch, H. Gruber, A. Hentschel, C. Korner, G. Pomberger, S. Schiffer, M. Saft, and S. Storck, "The emisq method and its tool support-expert-based evaluation of internal software quality," *Innovations in Systems and Software Engineering*, vol. 4, no. 1, pp. 3–15, 2008.

[47] I. Samoladas, G. Gousios, D. Spinellis, and I. Stamelos, *The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation*, ser. IFIP ÍC The International Federation for Information Processing. Springer US, 2008, vol. 275, book section 19, pp. 237–248.

[48] M. Ortega, M. Perez, and T. Rojas, "Construction of a systemic quality model for evaluating a software product," *Software Quality Journal*, vol. 11, no. 3, pp. 219–242, 2003.

[49] X. Franch and J. P. Carvallo, "Using quality models in software package selection," *IEEE Software*, vol. 20, no. 1, pp. 34–41, 2003.

[50] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, 2002.

[51] M.-J. Blin and A. Tsoukias, "Multi-criteria methodology contribution to the software quality evaluation," *Software Quality Journal*, vol. 9, no. 2, pp. 113–132, 2001.

[52] W. Pedrycz, J. F. Peters, and S. Ramanna, "Software quality measurement: concepts and fuzzy neural relational model," in *IEEE International Conference on Fuzzy Systems Proceedings . IEEE World Congress on Computational Intelligence*, vol. 2, 1998, Conference Proceedings, pp. 1026–1031 vol.2.

[53] O. Gomez, H. Oktaba, M. Piattini, and F. Garcia, *A Systematic Review Measurement in Software Engineering: State-of-the-Art in Measures*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2008, vol. 10, book section 14, pp. 165–176.

[54] P. Tomas, M. J. Escalona, and M. Mejias, "Open source tools for measuring the internal quality of java software products. a survey," *Computer Standards & Interfaces*, vol. 36, no. 1, pp. 244–255, 2013.

[55] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, no. 4, pp. 308–320, 1976.

[56] M. H. Halstead, *Elements of Software Science*. Elsevier Science Inc., 1977.

[57] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[58] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

[59] M. Yan, Y. Fang, D. Lo, X. Xia, and X. Zhang, "File-level defect prediction: Unsupervised vs. supervised models," in *ESEM*. ACM, 2017, p. to appear.

[60] M. Fowler, *Refactoring: improving the design of existing code*. Pearson Education India, 1999.

[61] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.-F. Girard, "An activity-based quality model for maintainability," in *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*. IEEE, 2007, pp. 184–193.

[62] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, vol. 27, pp. 504–518, 2015.

[63] R. Ploesch, S. Schuerz, and C. Koerner, "On the validity of the it-cisq quality model for automatic measurement of maintainability," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 2. IEEE, 2015, pp. 326–334.

[64] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 33–53, 2007.

[65] M. Kläs, J. Heidrich, J. Münch, and A. Trendowicz, "Cqml scheme: A classification scheme for comprehensive quality model landscapes," in *Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on*. IEEE, 2009, pp. 243–250.

[66] S. Montagud, S. Abrahão, and E. Insfran, "A systematic review of quality attributes and measures for software product lines," *Software Quality Journal*, vol. 20, no. 3-4, pp. 425–486, 2012.

[67] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 367–377.