A SQuaRE-based Software Quality Evaluation Framework and its Case Study

Hidenori Nakai*, Naohiko Tsuda*, Kiyoshi Honda*, Hironori Washizaki*†‡, and Yoshiaki Fukazawa*
*Global Software Engineering Laboratory, Dept. Computer Science and Engineering, Waseda University, Tokyo, Japan
Email: {hide-and-seek, 821821}@toki.waseda.jp, khonda@ruri.waseda.jp, {washizaki, fukazawa}@waseda.jp
†National Institute of Informatics, Tokyo, Japan
‡SYSTEM INFORMATION CO.,LTD., Tokyo, Japan

***Abstract*—Software stakeholders, including developers, managers, and end users, require high quality software products. Several works have aimed to identify software quality, but the quality of software products is often not comprehensively, specifically, or effectively defined because previous approaches have focused on certain quality aspects. Moreover, the evaluation results of quality metrics often depend on software stakeholders so that it is often hard to compare quality evaluation results across software products. ISO/IEC has tried to define evaluation methods for the quality of software products and provide common standards, called the SQuaRE (Systems and software Quality Requirements and Evaluation) series including ISO/IEC 25022:2016 and ISO/IEC 25023:2016. However, the SQuaRE series include ambiguous metrics so that it is not always easy to apply the series to products and compare results. In this paper, we propose a SQuaRE-based software quality evaluation framework, which successfully concretized many product metrics and quality in use metrics originally defined in the SQuaRE series[1]. Through a case study targeting a commercial software product, we confirmed that our framework is concretely applicable to the software package/service product.**

***Index Terms*—Software quality evaluation, software measurement, software metric**

I. Introduction

Software stakeholders, including developers, managers, and end users, require high quality software products. Several works have aimed to identify software quality (e.g., [27] and [26]), but the quality of software products is not comprehensively, specifically, or effectively defined because previous approaches have focused on certain quality aspects. Therefore, software project stakeholders are unable to identify and understand all aspects of software quality.

Moreover, issues with definitions are obstacles to control and understand the quality of software products [31]. Since the software product quality has a subjective component [3], the evaluation results of quality metrics depend on software stakeholders.

On the other hand, ISO/IEC has tried to define evaluation methods for the quality of software products and provide common standards, called the Systems and software Quality Requirements and Evaluation (SQuaRE) series [16] including ISO/IEC 25022:2016 [17] and ISO/IEC 25023:2016 [18] . This series includes a comprehensive quality model [15], software product quality characteristics, and quality in use characteristics. Additionally, this series includes several metrics for each quality characteristic.

Measurement issues and ambiguities limit evaluation methods [13]. Especially, ambiguities in the evaluation methods make the definition of the software product quality difficult. [32] indicated that only 28% of companies apply the ISO/IEC standards to their software products since these standards are not so practical; the ISO/IEC standards are too general and have ambiguous inputs, outputs and metrics [13] [1].

More than 70% of companies are applying their own quality models [32] to their own developments. Additionally, other quality frameworks such as [36], [37], [38], [39], [40] and various product metrics such as [35], [30] have been proposed for software quality evaluations. However, organization-specific non-standard quality frameworks, metrics and models are hard to be compared since these focus on only the quality characteristics of interest.

To mitigate this situation, we propose a SQuaRE-based software quality evaluation framework, which successfully concretized many product metrics and quality in use metrics originally defined in the standards ISO/IEC 25022:2016 and ISO/IEC 25023:2016 in the SQuaRE series.

The contributions of this paper include:

- A SQuaRE-based software quality evaluation framework[2].
- A case study to evaluate the usefulness of our framework.

The remainder of this paper is organized as follows. Section II details our proposed framework. Section III shows a case study using our framework. Section IV describes related works. Finally, Section V concludes the paper.

II. Proposed Framework

Our framework is aimed to reduce ambiguities in metrics as well as clearly define inputs and outputs for quality metrics. Our framework supports software development project stakeholders specify necessary concrete product metrics based on international standards, evaluate the sufficiency of the quality

[1]This paper is an extended version of a poster "Initial Framework for Software Quality Evaluation based on ISO/IEC 25022 and ISO/IEC 25023"[28] presented at The 2016 IEEE International Conference on Software Quality, Reliability & Security (QRS 2016).

[2]We have a plan to open our framework to the public through our web site at http://www.washi.cs.waseda.ac.jp/.

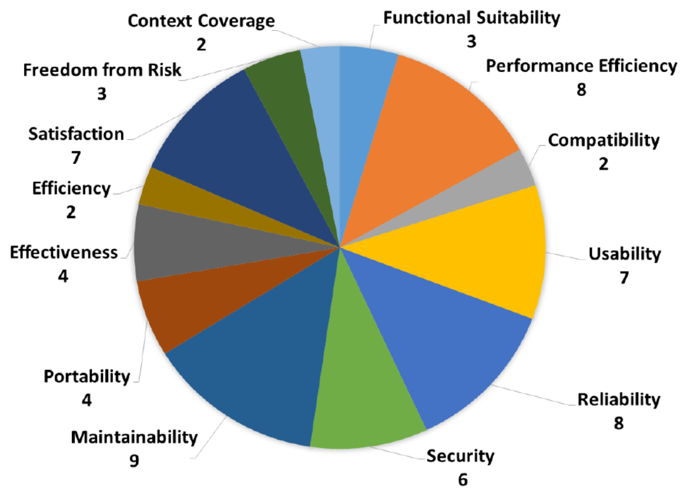

Fig. 1. Quality characteristics and number of metrics

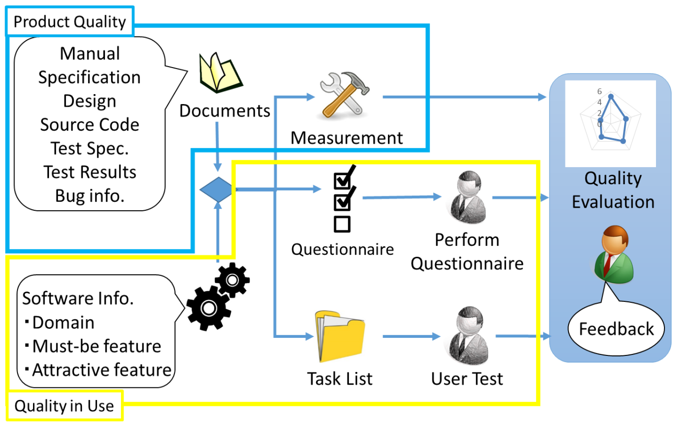

Fig. 2. Framework overview

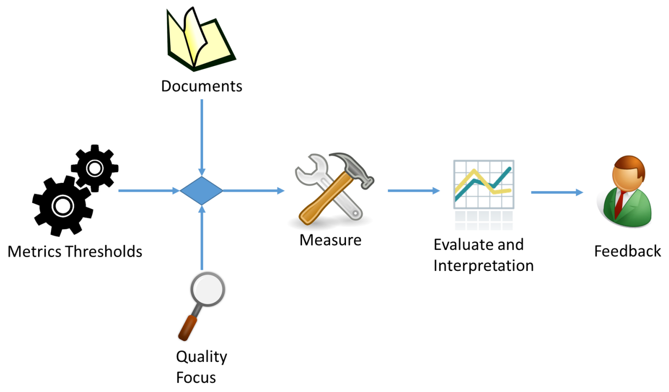

Fig. 3. Procedure to measure product quality

(sub-)characteristics based on the metrics, and identify areas for improvement according to the evaluation results.

Our framework is composed by two parts: "Product Quality" and "Quality in Use". The former contains internal and external product quality characteristics and metrics based on ISO/IEC 25023:2016, whereas the latter has quality characteristics and metrics of quality in use based on ISO/IEC 25022:2016. The product quality is expected to influence quality in use so that it is preferable to measure and ensure both quality.

In relation to PSQ Certification System [8], we selected and concretized 47 product metrics and 18 quality in use metrics that are concretely applicable to almost any software package/service products. These metrics cover more than 50% of the metrics originally defined in the SQuaRE series. The number of metrics for each quality characteristic is shown in Figure 1.

The overview of the procedure for employing our framework is shown in Figure 2. Our framework requires manual specifications, test specifications and bug information to measure the product quality. Moreover, our framework requires information collected via a questionnaire and a user test to measure quality in use metrics. Finally, entire software quality is assessed based on the results to clarify what quality characteristics are sufficient (or not).

A. *Product Quality*

The product quality indicates the degree of how the required needs (e.g., software purpose, performance, usability of product, and easy maintainability) are satisfied. If this quality is insufficient, the software product may include incidents, high development or maintenance costs, and violations of user needs. Therefore, product quality should be identified.

The product quality involves internal/external quality characteristics and sub-characteristics (e.g., one quality characteristic is Functional Suitability and functional completeness is one of its sub-characteristics) and metrics based on ISO/IEC 25023:2016. There are 47 product quality metrics. Some of these metrics focus on the main functions of the software. The main functions are must-have capabilities that are described in catalogs.

Our framework requires information from the following elements to measure the product quality: manuals, specifications, design, source code, violations of the coding standard, test specifications, test results, and bug information. In addition, thresholds, which are defined based on metrics information from many domain software products, are needed for an objective quality interpretation. However, in this research, the initial thresholds are defined based on the prediction approach [14] and conventional work. Figure 3 overviews the procedure to measure the product quality.

The following steps are used to measure the product quality.

1) Define the quality to be considered by project stakeholders.
2) Define the thresholds for the quality metrics using the proposed framework.
3) List information for the measurement based on select documents (e.g., manual, specifications, etc.).
4) Measure and evaluate the quality metrics based on thresholds.

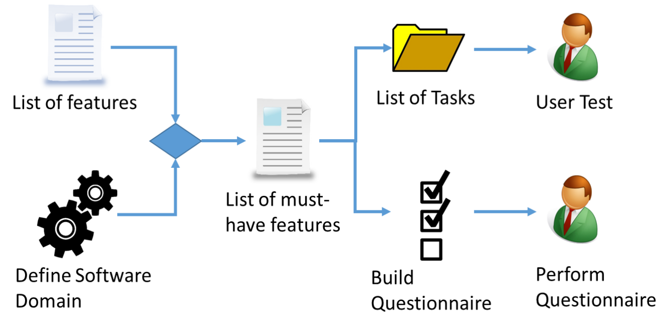

Fig. 4. Procedure to measure the quality in use

For example, "Functional Suitability" indicates whether the software functions satisfy users' needs. The lack of Functional Suitability means that the software does not meet the users' expectations or needs. One measure of this characteristic is as follows:

$$\begin{aligned} X &= 1 - A/B \\ A &= Number\ of\ functional\ requirements \\ &\quad not\ implemented \\ B &= Number\ of\ functional\ requirements \end{aligned}$$

B. Quality in Use

Quality in use indicates the degree that a software product satisfies a specific user's needs, effectively, efficiently, and satisfactorily to achieve a user's goals and mitigate risks in the context of use. If this quality is insufficient, users tend to be dissatisfied with a software product. Because an insufficient quality in use may reduce the number of users, this metric should be identified.

Quality in use involves certain quality characteristics and their sub-characteristics (e.g., the usefulness is one of the sub-charecteristics of the satisfaction), and metrics based on ISO/IEC 25022:2016. There are 18 metrics for quality in use.

Our framework requires experimental information (i.e., a user test and a questionnaire) to measure the quality in use. The user test evaluates the effectiveness, efficiency, and satisfaction. The questionnaire is related to satisfaction, freedom from risk, and context coverage. It should be noted that the questionnaire is developed according to [19] [41] and popular usability measurement scales such as SUS [6], SUMI [20], and NPS [29].

Figure 4 overviews the procedure to measure quality in use. The following steps are used to measure the quality in use:

1) Define the software product domain by vendor or third organization to evaluate the software.
2) Define must-have features in the software domain.
3) Create normal/abnormal tasks based on the must-have features for the user test and build a questionnaire considering the must-have features.
4) Prepare a test environment based on the desired system requirements and distribute the questionnaires to the actual users.
5) Implement a user test and questionnaire.

III. Case Study

To confirm the usefulness of our framework, we applied it to a commercial software product.

A. Design and Result

Due to lack of some necessary information about the software and its development project, the case study measured 30 product metrics and 6 quality in use metrics. All of these metrics can be measured, suggesting that project stakeholders can adapt these metrics and measurements to their own projects. The measurement results of most metrics reached 100%, indicating that the quality of the target product is fairly good.

In addition, a user test has been conducted by the following way. Firstly, the product's vendor developed a list of normal user tasks based on their scenario test. After that, we developed a set of abnormal tasks based on the normal task list. Finally, two students belonging to our laboratory conducted the user test. During the user test, the vendor's developers helped them perform the tasks since the subjects were not so familiar with the target software product. It took several hours to complete the user test. In the user test, all tasks were completed, but the results revealed some problems such as "There may be some potential bugs".

B. Assessment

The vendor of the target product assessed the evaluation results objectively, which revealed the following:

Internal Quality How the target of the number of bugs is defined should be revised and refined.

External Quality The measurement results indicate useful suggestions to improve the quality such as refining the product testing process.

Quality in Use The result of the user test can help improve the quality of software products and user satisfaction.

C. Discussion

Our framework collects metrics information based on documents such as specifications, test designs, and manuals. Because the format of these documents depends on the organization or project, information about some metrics could not be collected in the case study as it does not exist.

Our framework may be time consuming for project stakeholders to implement. Because project stakeholders have difficulty introducing all the metrics and measurements defined in our framework due to time limitations, some metrics and measurements should be revised to improve the feasibility of evaluating quality.

IV. Related Work

There is an SQuaRE-based Software Product Quality Certification [2] evaluating only maintainability and functional suitability. In contrast, our research not only examines maintainability and functional suitability but also investigates other quality characteristics of the SQuaRE series.

There is a quality meta model to define specific operationalized quality models [33]. However, it could be hard to compare their measurement/evaluation results to other software product's quality.

[22] presented a scheme to identify a suitable quality model based on the existing quality model's purpose (e.g., quality specification, quality measure, monitoring quality, and quality improvement), object (e.g., product, process, and resources), and quality focus (general or specific). Unlike our framework, this scheme cannot be used as a quality measurement.

There are various quality models and metrics such as HDCE [21] and COQUALMO [7]; however, these models/approaches require subjective expert assessments.

V. Conclusion and Future Work

In this paper, we developed a comprehensive SQuaRE-based software quality evaluation framework. It consists of 47 quality metrics, 18 quality in use metrics, and corresponding measurement methods involving documents, user tests and questionnaires. Our contributions are (1) defining the framework based on international standards ISO/IEC 25022:2016 and ISO/IEC 25023:2016, (2) defining a procedure to implement our framework to assess quality, (3) incorporating feasible metrics into our framework, and (4) demonstrating the usefulness of our framework by conducting a case study.

As our future work, we plan to apply our framework to various domains to confirm its general applicability. During such application, we will refine the metrics to improve feasibility and usefulness. Additionally, we will clearly relate the quality characteristics and metrics by building Goal-Question-Metric (GQM) [5] models and validating these relationships through additional case studies. Finally, we will consider possible quality-specific patterns (such as security patterns [10], [11], [12], [9], [4]) to utilize our framework by identifying individual patterns as well as pattern combinations and relationships [23], [24], [34], [25].

Acknowledgment

This work has been conducted as a part of "Research Initiative on Advanced Software Engineering in 2015" supported by Software Reliability Enhancement Center (SEC), Information Technology Promotion Agency Japan (IPA). Moreover, this work was partially supported by JSPS KAKENHI Grant Numbers 25330091 and 16H02804.

References

[1] A. Abran et al. Usability meanings and interpretations in ISO standards. *Software Quality Journal*, 11(4):325–338, Nov. 2003.

[2] AENOR. Iso 25000 software product quality certification. http://www.en.aenor.es/aenor/certificacion/ (last visit: 2016 Dec 29).

[3] H. Al-Kilidar et al. The use and usefulness of the ISO/IEC 9126 quality standard. In *ISESE*, 2005.

[4] A. Bandara et al. Security patterns: Comparing modeling approaches. In *Software Engineering for Secure Systems*, 2010.

[5] V. Basili et al. Goal, question, metric paradigm. In *Encycloperia of Software Engineering*, 1994.

[6] J. Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[7] S. Chulani et al. Modeling software defect introduction and removal: COQUALMO. Technical report, USC-CSSE, 1999.

[8] CSAJ. Psq certification system. http://www.psq-japan.com/english/ (last visit: 2016 Dec 29).

[9] E. B. Fernandez et al. Using security patterns to build secure systems. In *SPAQu*, 2007.

[10] E. B. Fernandez et al. Abstract security patterns. In *PLoP*, 2008.

[11] E. B. Fernandez et al. Classifying security patterns. In *APWeb*, 2008.

[12] E. B. Fernandez et al. Modeling misuse patterns. In *ARES*, 2009.

[13] J. Heidrich et al. Model-based quality management of software development projects. In *Software Project Management in a Changing World*, pages 125–156. Springer, 2014.

[14] K. Honda et al. A generalized software reliability model considering uncertainty and dynamics in development. In *PROFES*. 2013.

[15] ISO/IEC. *ISO/IEC 25010:2011 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. 2011.

[16] ISO/IEC. *ISO/IEC 25000:2014 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE*. 2014.

[17] ISO/IEC. *ISO/IEC 25022:2016 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of quality in use*. 2015.

[18] ISO/IEC. *ISO/IEC 25023:2016 Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of system and software product quality*. 2015.

[19] J.-Y. Jian et al. Foundations for an empirically determined scale of trust in automated systems. *IJCE*, 4(1):53–71, 2010.

[20] J. Kirakowski et al. SUMI: the software usability measurement inventory. *British Journal of Educational Technology*, 24(3):210–212, 1993.

[21] M. Kläs et al. Managing software quality through a hybrid defect content and effectiveness model. In *ESEM '08*, pages 321–323. ACM, 2008.

[22] M. Kläs et al. CQML scheme: A classification scheme for comprehensive quality model landscapes. In *SEAA*, pages 243–250, 2009.

[23] A. Kubo et al. Analyzing relations among software patterns based on document similarity. In *ITCC*, pages 298–303, 2005.

[24] A. Kubo et al. Extracting relations among embedded software design patterns. *Journal of Integrated Design and Process Science*, 9(3), 2005.

[25] A. Kubo et al. Extracting relations among security patterns. In *SPAQu*, pages 1–6, 2007.

[26] J. Münch et al. Software project control centers: concepts and approaches. *Journal of Systems and Software*, 70(1):3–19, 2004.

[27] H. Nakai et al. Continuous product-focused project monitoring with trend patterns and GQM. In *APSEC '14*, volume 2, pages 69–74, 2014.

[28] H. Nakai et al. Initial framework for software quality evaluation based on iso/iec 25022 and iso/iec 25023. In *QRS, Poster*, 2016.

[29] F. F. Reichheld. The one number you need to grow. *Harvard business review*, 81(12):46–55, 2003.

[30] K. Sakamoto et al. Open code coverage framework: A consistent and flexible framework for measuring test coverage supporting multiple programming languages. In *QSIC*, pages 262–269, 2010.

[31] A. Trendowicz et al. Model-based product quality evaluation with multi-criteria decision analysis. *CoRR*, abs/1401.1913, 2014.

[32] S. Wagner et al. Software quality models in practice - survey results-. https://mediatum.ub.tum.de/doc/1110601/1110601.pdf, 2010.

[33] S. Wagner et al. The quamoco product quality modelling and assessment approach. In *ICSE '12*, pages 1133–1142, 2012.

[34] H. Washizaki et al. Relation analysis among patterns on software development process. In *PROFES*, pages 299–313, 2005.

[35] H. Washizaki et al. A coupling-based complexity metric for remote component-based software systems toward maintainability estimation. In *APSEC*, pages 79–86, 2006.

[36] H. Washizaki et al. Experiments on quality evaluation of embedded software in japan robot software design contest. In *ICSE*, 2006.

[37] H. Washizaki et al. A framework for measuring and evaluating program source code quality. In *PROFES*, pages 284–299, 2007.

[38] H. Washizaki et al. Quality evaluation of embedded software in robot software design contest. *Progress in Informatics*, (4), 2007.

[39] H. Washizaki et al. A metrics suite for measuring quality characteristics of javabeans components. In *PROFES*, pages 45–60, 2008.

[40] H. Washizaki et al. Reusability metrics for program source code written in C language and their evaluation. In *PROFES*, pages 89–103, 2012.

[41] D. Watson et al. Development and validation of brief measures of positive and negative affect: the PANAS scales. *JPSP*, 54(6), 1988.