J. Inf. Technol. Appl. Manag. 24(3): 73~92, September 2017
https://doi.org/10.21219/jitam.2017.24.3.073

ISSN 1598-6284 (Print)
ISSN 2508-1209 (Online)

What Should Using a Software Product and Usability of the Software Product Be?

Seokha Koh* · Jialei Jiang**

Abstract

Usability is one of the most important concepts regarding software quality. It can be interpreted as the goodness associated with using the software product. This paper distinguishes the goodness of an individual using experience and the goodness of a product for using. This paper proposes a software quality view model which classifies software quality views into two broad categories of end view and means view. End view includes long-term view and short-term view which is classified further into performer's view on software activity and third party's view on software activity. Means view includes intrinsic view and contingency view. The analysis of ISO 25000 Series SQuaRE demonstrates the necessity to decompose product quality model and quality in use model into five models corresponding to the software quality views respectively. The analysis on playability shows that the universal definition of usability may be an illusion. The results provide the theoretical basis to build a comprehensive and consistent body of knowledge regarding software quality, which is consisted with the set of quality models and the theories explaining the relationships among the elements of the models.

Keywords : Software Quality, Usability, Quality in Use, Playability, ISO/IEC 25000 Series SQuaRE (The International Standard Organization 25000 Series Systems and software Quality Requirement and Evaluation)

Received : 2017. 07. 03. Revised : 2017. 09. 05. Final Acceptance : 2017. 09. 18.

※ We appreciate Sangsan Brick Co. supporting this research by the development fund of the Chungbuk National University.

* Corresponding Author, Professor, Department of Management Information Systems, Chungbuk National University, 1 Chungdae-ro Seowon-gu Cheongju, Chungbuk 28644, Korea, Tel : +82-43-261-2356, Fax : +82-43-273-2355, e-mail : shkoh@cbnu.ac.kr

** Doctoral Student, Department of Management Information Systems, Chungbuk National University, e-mail : 365678785@qq.com

1. Introduction

Using is the ultimate goal of all kinds of product including software. A software product should be good to use. In this paper, the usability of software is defined as the goodness for using. This generic definition will be elaborated later in this paper.

ISO's software quality models generally regard usability as the property of the software product. Specifically, ISO 25000 Series SQuaRE defines usability as "*the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*"[1] [ISO/IEC 25010:2011, p. 12]. SQuaRE defines quality in use almost the same as usability except that it "*includes freedom from risk and context coverage*" [ISO/IEC 25010:2011, p. 8, 12, 18; ISO/IEC 25022:2016]. It emphasizes that "*usability can either be specified or measured as a product characteristic in terms of its sub-characteristics, or specified or measured directly by measures that are subset of quality in use*" [ISO/IEC 25010:2011, p. 9, 12]. However, their sub-characteristics and measures are quite different. So, they represent different things in fact.

Gonzalez Sanchez et al. [2009a, 2009b] define playability as "*a set of properties that describe the player experience using a specific game system whose main objective is to provide enjoyment and entertainment.*" ISO 9241-11 is the reference model of Gonzalez Sanchez et al. [2009a]. They say that playability is the extension of ISO's usability or quality in use for the player centered video game. They, however, specify playability as the property of using experience.

This paper distinguishes instance usability which represents the goodness as the property of using experience and product usability which presents the goodness as the property of software product. This paper also critically reviews SQuaRE, which is the extended version of ISO 9241-11.

SQuaRE is the most extensive software quality model ever existed, which contains product quality model and quality in use model which contain 13 characteristics, 40 sub-characteristics, and 123 measures for systems and software quality in total. The chief goal of SQuaRE is to assist software engineers to develop software products with high quality. It is an enormous and elaborated body of knowledge about software quality which deserves appreciation. It, however, suffers from ambiguity, inconsistency, and contradictions making it un-suitable for ordinary software engineers to measure the design quality of software product [Al-Kilidar, 2005; Haboush et al., 2014; Kitchenham and Pfleeger, 1996; Koh, 2016, 2017a, 2017b; Koh and Whang, 2016].

It is the purpose of this paper to provide theoretical basis to resolve the conceptual vagueness and inconsistency associated with software quality models including SQuaRE. It is beyond the scope of this paper, however, to present precise and rigorous definitions of associated concepts.

1) In this paper, italic font denotes that corresponding part is quoted with no or only slight changes from the cited literature.

2. Usability Instance and Product Usability

Koh [2017a] defines software activity as the activity which is performed on the software product by a person or a group of persons. In this paper, using is defined as the interaction between the software product and a person or a group of persons through the user interface of the product, which is distinguished from the following types of software activity:

- **Studying the product**: A type of software activity to increase the user's knowledge or expertise about the product itself. It can be performed on other materials too.
- **Testing**: A type of software activity to find out whether the product performs as intended or required.
- **Customizing user interface**: A type of software activity to make the product better to use.

According to above definitions, using can be performed only on the execution of a software product in operation. As the result, maintaining and porting (transferring) are excluded from using.

Studying the product and customizing user interface are frequently regarded as a sub-type of using. As the result, study-ability (or learnability) or customizability are frequently classified as sub-characteristics or measures of usability [Gonzalez Sanchez et al., 2009a, 2009b; ISO 9126-1:2001, ISO 25010:2011, ISO 25023:2017, Microsoft Corporation, 2010; Nielson, 2016]. They, however, can occur independently with using [Koh, 2017b]. Accessing is another example of software activity which can occur independently with using: A person may access a software product, for example, to test or study it, or customize its interface.

- **Instance of using**: The sequence of using actions to accomplish specific goals. It includes the immediate results or effects of the actions too.
- **Instance of playing**: The continuation of using a game software product to play. It is a sub-type of the using instance.
- **Usability function**: The function by which the usability of the using instance is determined. It is typically defined in terms of a set of evaluation criteria such as, for example, effectiveness, efficiency, and satisfaction, by which the goodness of using instance is defined. Multiple usability function can be defined simultaneously.
- **Instance of usability**: The value of usability function of a using instance. If multiple usability function is defined, then multiple distinctive usability instances may exist for a using instance. The usability instance is assumed to be determined regardless of measurement.
- **Usability of the software product**: The mean of all usability instances associated with a software product. It is not determined until the product is retired. So, it can be only estimated during its lifetime.

Instance of using, instance of playing, instance of usability, and usability of software product

will be called using instance, playing instance, usability instance, and product usability respectively, in short. The typical example of using instance is completing a user task ("*activities performed by the user (using software product) towards a specified goal*") of SQuaRE [ISO/IEC 25023:2016 pp. 4, 37]. The user task includes activities either physical or cognitive [ISO/IEC 25023: 2016 p. 4]. As a matter of course, every physical activity requires cognitive activities. The task which is composed with only cognitive activities, however, is not regarded as a using instance. SQuaRE also defines a task as "*a function that needs to be accomplished within a defined period of time, where the function mentioned here is the software itself*" [ISO/IEC 25021:2012, p. 22]. In this definition, the term task is amount to task type, in fact. In this paper, the task type corresponds to the type of using instance.

It is noticeable that the term task is especially appropriate for the business application software which is developed to support the work in an organization. It can be very unnatural or irrelevant to define tasks for some type of software such as, for example, game software. In that regard, using instance is distinguished from the performance of task.

Many existing sub-characteristics of usability correspond to the evaluation criteria of usability instance. Effectiveness, efficiency, and satisfaction are typical examples of such sub-characteristics. They correspond to the criteria by which the using instance is evaluated. A software product is evaluated to be good for using if the instances of using it are generally effective, efficient, or satisfactory. In this paper, the set of evaluation criteria is used as the synonym of usability function.

Beside specification of the evaluation criteria, the usability function should include operational definitions to assign a specific value of the criteria to each using instance. Both evaluation criteria and their operational definitions constitute the usability function. Multiple usability functions which shares common evaluation criteria can be defined. Moreover, various usability functions which do not share even the evaluation criteria can be defined. What matters is which one is more useful. The usefulness will vary case by case.

An individual usability instance can be hardly accepted to represent the characteristic of the software product since it is generally affected by the system, context, and the performer of using instances. The product usability can be accepted as the characteristic of the software product since the influence of the systems, contexts, and users is factored out.

3. What Can be Included in Usability: Information that Can be Obtained from Each Individual Using Instance

3.1 Effectiveness and Efficiency: Third party's View

According to Oxford Living Dictionaries, effective means "*successful in producing a desired or intended result*". Software quality literature typically defines effectiveness in terms of "achievement of goals." Software effectiveness is typically defined to be concerned with only

whether and how well goals are achieved but not to be concerned with how the goals are achieved. As the result, for example, SQuaRE emphasizes that "*effectiveness measures do not take account of how the goals were achieved, only the extent to which they were achieved*" [ISO/IEC 25022:2016, p. 10]. SQuaRE also specifies the goals to be specified (refer <Table 1>).

SQuaRE define measure as "*variable to which a value is assigned as the result of measurement*' as the noun or '*make a measurement*" as the verb, where measurement is defined as "*a set of operations having the object of determining a value of a measure*" [ISO/IEC 25010:2011, p. 19]. Metric is typically defined as the ratio or interval measure which generates the data that can be added [Cooper and Schidler, 2008; p. 545]. Every quality model requires quality data subject to summation. In this paper, measure and metric will be used inter-changeably. Measure, however, will be used when it is necessary to denote that it is the metric of SQuaRE.

Among SQuaRE's five measures of effectiveness, 'errors in a task,' 'task with errors,' and 'task error intensity' focus on errors (refer Table 2). However, users can complete their tasks or achieve their objectives despite of errors. Since, these measures do not conform to the definition of effectiveness.

〈Table 1〉 Definitions of Effectiveness: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a, 2009b]

Quality Model	Definition	Remarks
Quality in Use	Accuracy and completeness with which users achieve specified goals.	Goals are pre-specified by third party
GS et al. [2009a]	Degree to which specific users (players) can achieve the proposed goals with precision and completeness in the context of use, the video game.	
GS et al. [2009b]	Time and resource necessary to offer players a fun and entertaining experience whilst they achieve the game's various objectives and reach the final goal.	Virtually identical to efficiency

〈Table 2〉 Metrics of Effectiveness: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a]

Quality Model	Metric	Description	Remarks
Quality in Use	Task completed	Proportion of the tasks that are completed correctly without assistance.	Under-estimate the true value. Focused on detecting bad design which causes tasks not to be completed or objectives not to be achived.
	Objective achieved	Proportion of the objectives that are achieved correctly without assistance.	
	Errors in a task	Number of errors made by the user during a task.	Errors are more closely related with efficiency. Effectiveness is not affected if goals are achieved even if errors are made. Focused on detecting faults which causes errors.
	Task with errors	Proportion of the tasks where errors were made by the users.	
	Task error intensity	Proportion of users making an error.	
GS et al. [2009a]	Goal effectiveness	What proportion of the goals is achieved correctly?	Virtually identical to those of Quality in Use. Represent the third party's view. Do not concern the intensions of individual users.
	Goal completion	What proportion of the goals are completed	
	Number of attempt	What is the frequency of attempts?	

'Tasks completed' and 'objectives achieved' ignore the tasks partially completed and the objectives partially achieved respectively to systematically underestimate the true value of effectiveness. Moreover, 'tasks completed' conforms to the definition of effectiveness only if 'achieving the goals' is specified as 'completing the tasks.' Rather, the focus of these measures is given on the uncompleted tasks and unachieved objectives which can be the symptoms of bad system design [ISO/IEC 25022:2016, p. 10].

All the effectiveness measures of SQuaRE focus on the symptoms of faults or bad design. They are designed to detect symptoms of faults or bad design in their measurement process. The information can be feedback to the development process to improve the quality of the product being developed. That is, they summarize the results of 'black-box' testing. They are designed to fulfill the need of people managing development, acquisition, evaluation, or maintenance of software and system [ISO/IEC 25022:2016, p. 1]. As the result, SQuaRE's effectiveness represents the view of the third party. They are not designed to concern the users' intensions of individual using instances. So, the goals can and should be specified in advance of using.

On the other hand, SQuaRE's efficiency can represent both the third party's view and the user's view (refer <Table 3> and <Table 4>).

〈Table 3〉 Definitions of Efficiency: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a]

Quality Model	Definition	Remarks
Quality in Use	Capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.	Defined objectively. Can represent both the third party's view and the user's view.
GS et al. [2009a]	The degree to which specific users (players) can achieve the goals proposed by investing an appropriate amount of resources in relation to the effectiveness achieved in a context of use, the video game.	Although it is defined in the term of achievement of goals, it metrics are practically the same as those of Quality in Use.

〈Table 4〉 Metrics of Efficiency: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a]

Quality Model	Metric	Description	Remarks
Quality in Use	Task time	Time taken to successfully complete a task.	'Task time' has universal and common meanings, and can be measured the most straightforwardly. The others represent the view of developers or maintainers: They can provide the information about faults or bad design.
	Time efficiency	Efficiency with which users achieve their objectives over time when using the system	
	Productive time ratio	Proportion of the time that the user is performing productive actions.	
	Unnecessary actions	Proportion of the actions performed by the user that were not necessary to achieve the task.	
	Consequence of fatigue	Decrease in human performance after continuous use.	
	Cost effectiveness	Cost-effectiveness of the user.	Regarding the user.
GS et al. [2009a]	Goal time	How long does it take to complete a goal?	The same as 'task time.'
	Goal efficiency	How efficient are the users?	Regarding the user.
	Relative user efficiency	How efficient is a player compared to an export?	

Among six SQuaRE's efficiency measures, 'task time' is purely objective. It has very universal and common meanings, interpretations, or implications, and can be measured the most straightforwardly.

The other measures require more information and can be affected by the measurer's judgment. Moreover, 'consequence of fatigue' does not measure efficiency itself, but measures the change rate of efficiency, 'Task time' is the default measure of efficiency which should be used whenever possible. It is even useless to combine 'time efficiency,' and 'cost effectiveness' in combination with 'task time' to increase reliability since they are virtually redundant with 'task time.' 'Productive time ratio' and 'unnecessary actions' produce proportions making it very hard to combine them with 'task time.' Rather, they are useful to detect faults or bad design which causes unproductive or unnecessary actions or fatigue. In that regards, those other than 'task time' represent the view of third parties other than the user: Developers or maintainers. Only 'task time' represents both the performer's view and the third party's view.

Effectiveness and efficiency of Gonzalez Sanchez et al.'s [2009a] playability are virtually the same as the corresponding parts of SQuaRE, although their wording is slightly different. Playing represents using the software product for fun or pleasure. On the other hand, SQuaRE concerns chiefly using the software product for work. It is the reason why the task is the key word and why efficiency matters in SQuaRE. For playing, however, efficiency may not matter at all. For example, the time taken may not matter as long as it is fun or interesting. This point of argument implies that usability should be specialized for specific types of software.

It is especially noticeable that efficiency is excluded from Gonzalez Sanchez et al.'s [2009b] playability. This seems very reasonable since minimizing time taken may be irrelevant in playing the video game. Effectiveness, however, is defined virtually the same as the typical efficiency. So, what is really excluded from the model is effectiveness. This seems very reasonable since the goal other than fun or pleasure can be hardly perceived as the chief goal of playing the video game. Again, however, the time taken to play is really so important? It seems not so. Efficiency may be irrelevant for some type of software such as, for example, the game software, implying that the general software quality model may be an illusion.

3.2 Satisfaction: User's View

According to Oxford Living Dictionaries, satisfaction means "*fulfillment of one's wishes, expectations, or needs, or the pleasure derived from this.*" Definitions as software quality characteristics can be classified into two groups: the fulfillment of user's needs and the pleasure the user feels. The satisfaction in quality in use corresponds to the one while those of Gonzalez Sanchez et al. [2009a, 2009b] correspond to the latter (refer <Table 5>). The satisfaction as the fulfillment of user's needs, however, is virtually redundant with the typical definition of effectiveness. SQuaRE uses it as the pleasure the user feels, in fact: it uses satisfaction as the synonym

〈Table 5〉 Definitions of satisfaction: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a, 2009b]

Quality Model	Definition	Remarks
Quality in Use	Degree to which user needs are satisfied when a product or system is used in a specified context of use.	The performer's view. Can be evaluated both subjectively or objectively
GS et al. [2009a]	Degree to which users (players) are satisfied in a context of use, the video game.	
GS et al. [2009b]	Gratification or pleasure derived from playing a complete video game or from some aspect of it.	Defined to be free from context.

〈Table 6〉 Sub-characteristics of satisfaction: ISO 25000 SQuaRE

Quality Model	Name	Definition	Remarks
Quality in Use	Comport	Degree to which the user is satisfied with physical comfort.	Do not include all the reasons why users are satisfied: People may be satisfied, for example, because it is exciting or hard to play.
	Pleasure	Degree to which a user obtains pleasure from fulfilling their personal needs.	
	Trust	Degree to which a user or other stakeholder has confidence that a product or system will behave as intended.	
	Usefulness	Degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use.	

of pleasure in the definition of the sub-characteristics of satisfaction (refer <Table 6>). From now on in this paper, satisfaction will be used to mean the pleasure the user feels.

Satisfaction represents the user's view. It cannot represent the third party's view while effectiveness and efficiency can be defined to represent the user's view as the followings, for example:

- **Effectiveness**: The user's evaluation on how well his/her motivation or goals are achieved.
- **Efficiency**: The user's evaluation on how much the cost is expended.
- **Satisfaction**: The user's evaluation on how much he/she is satisfied.

It is noticeable that context is not referred in the definitions as in Gonzalez Sanchez et al. [2009b]. Context is the concept associated with sampling. There is no need to include it in the definition of software quality. It is also noticeable that the goals are set by the user and that it is not necessary for the goals to be specified. The definition of efficiency is almost the same as that of SQuaRE. It, however, can be evaluated subjectively by the user. The subjective evaluation of time taken may not be congruent with the objective physical time taken. For example, the user can feel the time taken to play a game software product to be short regardless of physical duration if it was interesting.

People may be satisfied because his/her using experience was comfortable, pleasant, trustworthy, or useful. People may be satisfied for various other reasons too. One may be satisfied because it is exciting, interesting, fascinating, stimulating. One may be satisfied because it is hard to play a game product. One may be satisfied because

〈Table 7〉 Metrics of Satisfaction: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a]

Quality Model	Characteristic	Metric	Description	Remarks
Quality in Use	Satisfaction	Overall satisfaction	Overall satisfaction of the user.	Can be evaluated for each using instance.
	Comport	Comfort	Extent to which the user is comfortable compared to the average for this type of system.	
	Pleasure	Pleasure	Extent to which the user obtains pleasure compared to the average for this type of system.	
	Trust	Trust	Extent to which the user trusts the system.	
	Usefulness	Satisfaction with features	Satisfaction of the user with specific system features	
		Discretionary usage	Proportion of potential users choosing to use a system or function.	Aggregated information which cannot be obtained from a using instance. Objective metrics. May be unreliable.
		Feature utilization	Proportion of an identified set of users of the system who use a particular feature.	
		The same as description	Proportion of users making complaints	
		The same as description	Proportion of user complaints about a particular feature.	
GS et al. [2009a]	Satisfaction	Satisfaction scale	How satisfied is the player?	Can be evaluated for each using instance.
		Satisfaction questionnaire	How satisfied is the user with specific software feature?	
		Discretionary usage	What proportion of potential users choose to use the system	Aggregated information. Objective metrics. May be unreliable.
		Socialization	What proportion of potential users choose to use the system	

it was good for just spending time.

Effectiveness and efficiency can be the reason of satisfaction too. That is, effectiveness and efficiency can be classified as the sub-characteristics of satisfaction in the user's view. SQuaRE regards usefulness as "*user's perceived achievement of pragmatic goals, including the results of use and the consequences of use*" [ISO/IEC 25010:2011, p. 9]. In this regard, satisfaction can be used as the synonym of the user's view on usability. The usability in user's view can be estimated statistically to identify the response of the users as customers of the product in the market.

Satisfaction can be measured using the five point rating scale by asking 'how much are you satisfied?" It can be also measured by asking "how much do you feel following feelings or emotions?" for various relevant feelings and emotions such as, for example, comfort, pleasure, trust, and/or usefulness and summating the results to increase the reliability of measurement. Gonzalez Sanchez et al.'s [2009a] metrics of satisfaction is inferior to those of SQuaRE in respect of reliability (refer <Table 7>).

Satisfaction of the user can be measured objectively. 'Discretionary usage,' 'feature utilization,' 'proportion of users making complaints,' 'proportions of user complaints about a particular feature,' and socialization are designed to measure satisfaction objectively. However, they cannot measure the user's satisfaction on in-

dividual using instances since they need multiple observations to be measured. Other objective measures can be devised. It is well known, however, that questioning generally produces more valid and reliable data about people's opinion or emotion than observation if its procedure is executed adequately [Cooper and Schindler, 2008]. For example, users may complain about a product for various reasons even if they think it is useful. Users also may use a product or function simply because there is no other alternative available despite they are dissatisfied. Moreover, "*potential users may choose to use a system of function*" [ISO/IEC 25022:2016, p. 14] as the result of being exposed to advertising, and then become dissatisfied. Logically, potential users cannot choose to use a system or function because they are satisfied with it. In this respect, 'proportion of users complaining,' and 'proportion of user complaints about a particular feature' can be regarded as indicators reflecting the effectiveness of marketing activities. It is generally not a good idea to attempt to measure the satisfaction of the user objectively.

4. What Should not be Included in Usability: Information that Cannot be Obtained from Each Individual Using Instance

4.1 Context Coverage: Information that Should be Obtained by Aggregating the Information Obtained from Each Individual Using Instance

SQuaRE defines context of use as "*users, equipment, (hardware, software and materials), and the physical and social environments in which a product is used*" [ISO/IEC 25010:2011, p. 18]. SQuaRE presumes that both the usability and quality in use are affected by the context of use and recommends estimating them contextually [ISO/IEC 25010:2011, pp. 4-5; ISO/IEC 25022:2016, pp. 7-8]. SQuaRE also recommends to let data obtained from a sufficient number of users performing tasks to obtain the desired level of statistical confidence of effectiveness, efficiency, or satisfaction that the target values have been achieved [ISO/IEC 25022:2016, p. 9]. Context and contextual estimation can be regarded to correspond to stratum and stratified sampling respectively. Well performed, stratified sampling provides more accurate estimate than the simple random sampling [Cooper and Schindler, 2008].

Contexts can be grouped to generate bigger sub-populations. SQuaRE classifies contexts into specified contexts and unspecified contexts according to whether they are specified or not. The proficiency of users can be the classification criterion too. Task type can be the criterion too. Role and responsibilities can determine goals and tasks [ISO/IEC 25023:2016, p. 4]. Task types may be categorized according to the role and responsibilities they rest on.

The means of resulting sub-populations can be defined as follows, for example:

- **Context usability**: The sub-population mean corresponding to a specific context.
- **Specified context usability**: The sub-population mean corresponding to specified contexts.
- **Unspecified context usability**: The sub-population mean corresponding to un-specified contexts.

- **Proficient user usability**: The sub-population mean corresponding to proficient users.
- **Low-proficient user usability**: The sub-population mean corresponding to un-proficient users.
- **Task type/function usability**: The sub-population mean corresponding to a task type or a function.
- **Role/responsibility usability**: The sub-population means corresponding to a role or a responsibility.

The difference between the definition of quality in use and that of context coverage rest on whether the clause "*in/beyond contexts initially explicitly identified*" is added or not (refer Table 8 and 9). The definition of SQuaRE's context coverage is virtually the same as that of quality in use from which context coverage is excluded. That is, context coverage generates circular reference in which the whole includes itself as its part. On the other hand, context usability including its varieties is conceptually clear. Context coverage should be excluded from the sub-characteristics of quality in use.

〈Table 8〉 Definitions of Context Coverage, Freedom from Risk, Flexibility, and Safety: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a]

Quality Model	Title	Definition	Remarks: It is regarding
Quality in Use	Context coverage	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.	Virtually the same as quality in use from which context of coverage is excluded.
	Freedom from risk:	Degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment.	Regarding long-term and aggregated impact on various entities.
GS et al. [2009a]	Flexibility	Degree to which the video game can be used in different contexts or by different player or game profiles.	Shares similar problems of context coverage and freedom from risk
	Safety	Acceptable level of risk to the player health or data in a context of use, the video game.	

〈Table 9〉 Sub-Characteristics of Context Coverage and Freedom from Risk: ISO 25000 SQuaRE

Characteristic	Title	Definition	Remarks
Context coverage	Context completeness	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use.	Division of contexts: Correspond to specified context usability and unspecified context usability, respectively
	Flexibility	Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements.	
Freedom from risk	Economic risk mitigation	Degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use.	Only one value is possible per product. Require special domain knowledge which ordinary SW engineer cannot be expected to have.
	Environmental risk mitigation	Degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use.	
	Health and safety risk mitigation:	Degree to which a product or system mitigates the potential risk to people in the intended contexts of use.	

Among four measures of 'context coverage,' 'product flexibility' is regarding changing the product, but not regarding using the product (refer <Table 10>). In fact, it is redundant with modifiability of product quality model [ISO/IEC 25023:2016, pp. 21-22]. It is the intrinsic and invariant property of the software product [Koh, 2017a, 2017b].

〈Table 10〉 Metrics of Context Coverage, Freedom from Risk, Flexibility, and Safety: ISO 25000 SQuaRE and Gonzalez Sanchez et al. [2009a]

Quality Model	Characteristic	Metric	Definition	Remarks
Quality in Use	Context completeness	Context completeness	Proportion of the intended contexts of use in which a product or system can be used with acceptable usability and risk	Require the aggregated information regarding contexts or sub-populations.
	Flexibility	Flexible context of use	Extent to which the product can be used in additional contexts of use with no modifications or only simple modifications	
		Proficiency independence	Extent to which the product can be used by people without specific knowledge, skills or experience	
		Product flexibility	Ease with which a product can be modified to meet additional user requirements	Regarding changing.
	Economic risk mitigation	ROI	Return on investment	Require long-term and aggregated information with which business people usually deal much better than SW engineers.
		Time to achieve ROI	Time taken to achieve the expected ROI	
		Business performance	Profitability or sales compared to a target	
		Benefits of IT Investment	Measure of the benefits of IT investment compared to a target	
		Service to customers	Extent to which the intended level of service to customers is achieved	
		Website visitors converted to customers	Proportion of visitors to a particular web page(s) who become customers	
		Revenue from each customer	Revenue from each customer	
		Errors with economic consequences	Proportion of usage situations where there are human or system errors with economic consequences	
	Environmental risk mitigation	Environmental impact	Environmental impact of the manufacture and use of the product or system compared to a target	Chiefly determined by system
	Health and safety risk mitigation	User health reporting frequency	Proportion of users of the product who report health problems arising from usage	Require long-term and aggregated information with which ordinary SW engineers cannot deal properly.
		User health and safety impact	Health and safety impact on users of the product	
		Safety of people affected by use of the system	Incidence of hazard to people affected by use of the system	
GS et al [2009a]	Flexibility	Accessibility	What proportion of the goals can be achieved by using alternative ways of interaction?	Chiefly determined by system
		Personalization	What proportion of the personalization options are used by the players?	Regarding customizing
	Safety	User health and safety	What is the incidence of health problems among users of the product?	Ordinary users cannot provide necessary information.
		Software damage	What is the incidence of software damage?	

The other measures represent some useful concepts. However, they can be more clearly expressed in terms of context usability and its varieties. For example, 'proficiency independence' can be replaced with the ratio or difference of proficient user usability and low-proficient user usability. The pair of proficient user usability and low-proficient user usability provides more information more clearly than 'proficiency independence' is seemingly expected to provide. The pair of specified context usability and unspecified context usability provides more information more clearly than 'flexible context of use' is seemingly expected to provide too. The ratio or difference of specified context usability and unspecified context usability can be used as an index. 'Context completeness' can be rephrased as 'the proportion of the intended contexts whose context usability is above the acceptable criteria.'

'Context coverage' and its sub-elements represent some useful concepts. They, however, cannot be sub-elements of quality in use nor usability. Context usability and its varieties are conceptually clear and can provide more information than they can.

Gonzalez Sanchez et al.'s [2009a] flexibility corresponds to and shares the same problem with 'context coverage.' Its metrics, however, are quite different from those of 'context coverage' and do not measures what is seemingly expected to represented by flexibility.

4.2 Freedom from Risk: Long-term and Aggregated Effects

SQuaRE defines risk as "*a function of the probability of occurrence of a given threat and the potential adverse consequences of that threat's occurrence*" [ISO/IEC 25010:2011, p. 9]. It is obvious that probability or proportion cannot be inferred from a using instance. It is aggregated information which cannot be obtained from an individual using instance unlike effectiveness, efficiency, and satisfaction.

The measures of 'freedom from risk,' however, do not measure probability or proportion in fact. They measures the long-term and aggregated impact which various activities performed on a software product such as using, developing, changing, and so on exert on various entities. For example, 'return on investment' requires long-term and aggregated information to measure. They cannot be obtained from a single using instance. The metrics that can be obtained from individual using instances respectively and the metrics that can be obtained by aggregating such data should be separated into distinctive models.

Moreover, they require the domain knowledge which ordinary software engineers cannot be expected to have. For example, business people generally can measure 'return on investment' of a software product more validly than software engineers. It is the typical issue which business people traditionally have dealt with. It is better to leave such issues to the specialists of corresponding domains.

Among 12 measures of 'freedom from risk,' 'service to customers' is regarding service level. It is not an item associated with the product itself. It is not a software quality metric. Other measures and sub-characteristics may have to

be redefined to be more useful according to the appropriate software quality view: The long-term view. It is beyond of this paper, however, to review and redefine individual items in depth and details. Gonzalez Sanchez et al.'s [2009a] safety is virtually the subset of and shares the same problems with 'freedom from risk.'

5. Discussions

The critical review in sections 3 and 4 shows the discordance among characteristics, between characteristic and its sub-characteristics, between characteristics and their metrics, and between the titles and definitions prevails in quality in use model of SQuaRE and Gonzalez Sanchez et al.'s [2009a, 2009b] playability. There exist virtually the same phenomena in product quality model of SQuaRE [Koh 2017a, 2017b]. This implies that the phenomena can prevail in many other software quality models too.

To overcome the inconsistency in product quality model of SQuaRE, Koh [2017a, 2017b] proposes a model of the views regarding software quality, which classifies the views of software quality broadly into end view and means view. The discussions in the sections 3 and 4 show that end view can be divided into long-term view and short-term view which can be divided further into performer's view on software activity and third party's view on software activity. The resulting model of software quality view becomes as the following:

- **End view**: It represents the effort to find out for what the software product should be good. The quality characteristics in this view correspond to the effects of good quality in means view.
  - **Short-term view**: It focuses on short-term effects of various software activity types.
    - **Performer's view on software activity**: It represents the performer's subjective evaluation of the software activity that he/she has performed.
    - **Third party's view on software activity**: It represents the interests of stakeholders other than the performer, which are associated with individual software activities.
  - **Long-term view**: It focuses on the long-term and aggregated effects on various stakeholders.
- **Means view**: It represents the effort to make the software product good for various ends. The elements of this view correspond to the causes of desirable effects. Software engineers should be able to manipulate the elements to improve the quality in end view.
  - **Intrinsic view**: It identifies static and invariant properties of the software product, which affect the achievement of ends. It does not change unless the product is changed.
  - **Contingency view**: It identifies static and invariant emerging properties of contingencies, which affect the achievement of ends. It can change even if the product is not changed.

Koh [2017a, 2017b] proposes the principle of one view stating that a software quality model should correspond to one and only one software quality view. According to principle of one view

and the new model of software quality view, at least, five software quality models are required: Long-term effect model, activity quality model in performer's view, activity quality model in third party's view, intrinsic quality model, and contingency quality model.

Quality in use model includes the elements of performer's view on software activity, third party's view on software activity, and long-term view. Product quality model includes the elements of intrinsic view and contingency view as well as end view [Koh 2017a, 2017b]. As the result, SQuaRE includes all the five views in its two quality models of software and system violating the principle of one view.

Product quality model and quality in use model are too big and complex even for expert software engineers to comprehend properly [Koh 2016, 2017a, 2017b]. They include even the elements that can be obtained by aggregating the data regarding other elements. It is the main reason why there are so many discords in the models. It is almost impossible for ordinary software engineers to deal with the models properly. The elements in SQuaRE should be reclassified into a set of smaller models which ordinary software engineers can deal with properly. The cause-and-effect relationships among the elements of the models should be elucidated too.

<Table 11> shows another example of discordance. Gonzalez Sanchez et al.'s [2009b] playability consists with satisfaction, effectiveness, and other 5 characteristics (refer <Table 11>). Among them, it is uncertain how emotion is differentiated form satisfaction. Emotion can be regarded as an aspect of satisfaction. It is classified as a sub-characteristic of satisfaction in Gonzalez Sanchez et al.'s [2009a].

However, the definitions of immersion, motivation, and socialization do not describe what their titles typically represent. They represent the factors which affect the titles without specifying what the factors are. So, the audiences should figure out by themselves both what the titles mean and the factors that affect the titles. Confusion arises as the result. The meaning of immersion, motivation, and socialization should be defined specifically and clearly in a model and the factors which contribute to increase emotion, immersion, motivation, and socialization should

〈Table 11〉 The Rest Sub-Characteristics of Playability: Gonzalez Sanchez et al. [2009b]

Title	Definition	Remarks
Immersion	Capacity of the video game contents to be believable, such that the player becomes directly involved in the virtual game world.	Titles are of player's view while definitions are of means view without specifying the factors that cause the effects specified by titles.
Motivation	Set of game characteristics that prompt a player to realize specific actions and continue undertaking them until they are completed.	
Socialization	Set of game attributes elements and resources that promote the social dimension of the game experience in a group scenario.	
Emotion	Player's involuntary impulse in response to the stimulus of the video game that induces feelings or a chain reaction of automatic behaviors.	It is uncertain how it is differentiate from satisfaction.
Learnability	Player's capacity to understand and master the game's system and mechanics (objectives, rules, how to interact with the video game, and so on).	Contingency factor: the trait of player

be specified in the other models. The theory which explains the cause-and-effect relationship among the elements in the models should be developed too.

Learnability also represents the factor which can affect playability. It, however, is not regarding the software. It is contingency factor.

SQuaRE's definitions of effectiveness and efficiency may be especially appropriate for business application software to be used in the workplace. They, however, may be inappropriate for some kind of software such as, for example, the video game. Gonzalez Sanchez et al.'s [2009a, 2009b] playability illustrates the point of argument very well. Sub-characteristics and metrics of usability should be defined variously according to the types of software.

6. Conclusions

This paper proposes the expended model of software quality view which classifies the views on software quality into two broad categories of end view and means view. End view includes long-term view and short-term view which is classified further into performer's view on software activity and third party's view on software activity. Means view includes intrinsic view and contingency view. According to the expended software quality view and Koh's [2017a, 2017b] principle of one view, at least, following five software quality models are required: Long-term effect model, activity quality model in performer's view, activity quality model in third party's view, intrinsic quality model, and contingency quality model.

Quality in use model of ISO 25000 Series SQuaRE includes the elements of performer's view on software activity, third party's view on software activity, and long-term view. It includes even the elements that can be obtained by aggregating the data regarding other elements. SQuaRE's product quality model includes the elements of intrinsic view and contingency view as well as end view [Koh 2017a, 2017b]. As the result, SQuaRE includes all the five views in its two quality models of software and system violating the principle of one view, which is the main reason why there are so many discords in the models. It is almost impossible for ordinary software engineers to deal with the models properly.

The elements in SQuaRE should be classified into a set of smaller models which ordinary software engineers can deal with properly. Effectiveness and efficiency should be classified into the third party's view on software activity; satisfaction into the performer's view on software activity; 'freedom form risk' into long-term view. 'Context coverage' should be eliminated.

This paper suggests restricting the term using to denote the type of software activity in which a person interacts with a software product through user interface. The software activity such as studying, testing, and customizing is excluded from using although they involve interacting with a software product through user interface. This approach sharply contrasts with that of ISO 25000 Series SQuaRE in which using encompasses various software activities performed by various stakeholders such as "*primary user (person who interact with the system to achieve the primary goals), secondary users*

(person who provide support, for example. content provider, system manager/administrator, security manager, maintainer, analyzer, porter, installer), and *indirect user (person who receives output, but does not interact with system)"* [ISO/IEC 25010:2011, pp. 5-6].

This paper also introduces the notion of using instance, usability function, and usability instance. Using instance is the sequence of using actions performed on a software product to accomplish specific goals. It includes the immediate results or effects of the actions too. Playing instance is a special type of using instance.

Usability function is the function by which the usability of using instance is determined. It is typically defined in terms of a set of evaluation criteria such as, for example, effectiveness, efficiency, and satisfaction, by which the goodness of the using instance is defined. Multiple usability function can be defined simultaneously. Usability instance is the value of usability function of the using instance. If multiple usability function is defined, then multiple distinctive usability instances can exist for a using instance.

The usability instance is assumed to be determined regardless the using instance is measured or not. The individual usability instance cannot be regarded as the property of the software product. The usability of a software product is defined as the population mean of usability instances from the product. The product usability can be regarded as the property of software product since the effects of contingency factors are factored out.

This paper analysis why SQuaRE is so difficult to comprehend. The results demonstrate well the need to decompose product quality model and quality in use model of SQuaRE into five, at least, small and easy-to-understand models which are consisted with homogeneous elements. This paper also provides the theoretical basis to customize usability according to various types of software.

It is beyond the scope of this paper to complete elaborated software quality models or to present precise definitions of various concepts or terminologies. The system of software quality models and theories to explain the relationships among the elements of the models should be developed. They will constitute a consistent and comprehensive body of knowledge regarding software quality.

References

[1] Al-Kilidar, H., Cox, K., and Kitchenham, B., "The Use and Usefulness of the ISO/IEC 9126 Quality Standard," *Proceedings of International Symposium on Empirical Software Engineering 2005*, IEEE, 2005, pp. 126-132.

[2] Cooper, D. R. and Schindler, P. S., *Business Research Methods*(10ed.), McGraw-Hill/Wirwin, International Edition 2008.

[3] Foraker Labs, "Introduction to User-Centered Design," http://www.usabilityfirst.com/about-usability/introduction-to-user-centered-design, 2002 (reference date: 17/04/2017).

[4] Gonzalez Sanchez, J. L., Montero Simarro, F., Padilla Zea, N., and Guitierrez Vela, F. L., "Playability as Extension of Quality in Use

in Video Games," *Proceedings of the Second International Workshop on the Interplay between Usability Evaluation and Software Development (I-USED'09)*, Uppsala, Sweden, August 24, 2009a.

[5] Gonzalez Sanchez, J. L., Zea, N. P., and Gutierrez Vela, F. L., "Playability: How to Identify the Player Experience in a Vedio Game," *Proceedings of IFIP Conference on Human-Computer Interaction: Human-Computer Interaction-INTERACT 2009*, 2009b, pp. 356-359.

[6] Haboush, A., Alnabhan, M., AL-Badareen, A., Al-Nawayseh, M., and EL-Zagmouri, B., "Investigating Software Maintainability Development: A Case for ISO 9126," *International J. of Computer Science Issues* (IJCSI), Vol. 11, No. 2, 2014, pp. 18-23.

[7] ISO/ IEC 25010:2011, *Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models*, ISO, 2011.

[8] ISO/IEC 9126-1:2001, *Software Engineering-Product Quality-Part I: Quality Model*, ISO, 2001.

[9] ISO/IEC 9241-11:1998, *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)-Part 11: Guidance on Usability*, ISO, 1998.

[10] ISO/IEC 25021:2012, *Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-Quality Measure Elements*, ISO, 2012.

[11] ISO/IEC 25022:2016, *Systems and Software EngineeringSystems and Software Quality Requirements and Evaluation (SQuaRE)-Measurement of Quality in Use*, ISO, 2016.

[12] ISO/IEC 25023:2016, *Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-Measurement of system and Software Product Quality*, ISO, 2016.

[13] Kitchenham, B. and Pfleeger, S. L., "Software Quality: The Elusive Target [special issue section]," *Software*, IEEE, Vol. 13, 1996, pp. 12-21.

[14] Koh, S. and Whang, J., "A Critical Review on ISO/IEC 25000 SQuaRE Model," *Proceedings of the 15th International Conference on IT Applications and Management: Mobility, Culture and Tourism in the* Digitalized *World*, (ITAM15), 2016, pp. 42-52.

[15] Koh, S., "Cause-and-Effect Perspective on Software Quality: Application to ISO/IEC 25000 Series SQuaRE's Product Quality Model," *Journal of Information Technology Applications & Management*, Vol. 23, No. 3, 2016, pp. 71-86.

[16] Koh, S., "The Checklist for System and Software Product Quality Implied in the Product Quality Model of ISO/IEC 25000 Series SQuaRE," Proceedings of 17th International Conference on IT Applications and Management: Babolsar, Iran, 2017a, pp. 126-136.

[17] Koh. S., "The Principle of One Quality View and Decomposition of Product Quality Model of ISO/IEC 25000 Series SQuaRE," *Asian Journal of Information and Communications*, Vol. 9, No. 1, 2017b, pp. 101-114.

[18] Microsoft Corporation, "Usability in Soft-

ware Design," https://msdn.microsoft.com/en-us/library/ms997577.aspx, 2000 (reference date: 17/04/2017).

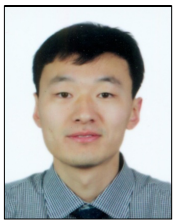
[19] Nielsen, J., "Usability 101: Introduction to Usability," Nielsen Norman Group, https://www.nngroup.com/articles/usability-101-introduction-to-usability, 2012 (reference date: 17/04/2017).

[20] Oxford Learner's Dictionary, reference date: 05/16/2016.

Author Profile

Seokha Koh

Seokha Koh is the professor of the Department of MIS, Chungbuk National University. His current primary research areas include Software Quality Management, Business Process Modeling, Software Architecture, Project Management, and Software Engineering.



Jialei Jiang

Jialei Jiang is a Ph.D. Candidate in College of Business Department of Chungbuk National University. He has received master's degree and bachelor's degree in The Dept. of Management information Systems from Chungbuk National University. His major research areas include Business Management, Information Technology, Business Statistics, Software Project Management, Management and IT.