KISDI
KOREA INFORMATION SOCIETY DEVELOPMENT INSTITUTE
Asian Journal of Information and Communications

The principle of one software quality view and decomposition of product quality model of ISO/IEC 25000 series SQuaRE

Seokha Koh*
Chungbuk National University

Abstract

This paper critically reviews the product quality model of ISO/IEC 25000 Series SQuaRE. This paper classifies the characteristics of the product quality model into intrinsic quality characteristics, contingency quality characteristics, and activity quality characteristics, which are defined as the evaluation criteria of the software product, contingencies which involve the product, and software activity performed on the product in the contingencies, respectively. This paper classifies activity quality characteristics into a hierarchy which can serve as a prototype of software activity quality model. This paper also classifies measures of the product quality model into those intrinsic, of contingency, and of activity. Discussions show that the measures do not properly specify the process to assign values to the corresponding characteristics. The results show well how useful it is to build a system of software quality models which consist with homogeneous items.

Keywords: software quality, views on software quality, ISO/IEC 25000 series SQuaRE, software product quality model

1. Introduction

ISO/IEC (The International Standard Organization/The International Electrotechnical Commission) issued ISO/IEC 9126 in 1991 and has been issuing the more extensive 25000 series SQuaRE (System and software product Quality Requirements and Evaluation) since 2005 to replace ISO/IEC 9126. SQuaRE is designed to assist software engineers to develop high quality software products. It contains a huge and elaborated body of knowledge about software quality, which deserves appreciation. It is generally recognized as one of the most important software quality models (Miguel et al. 2014). It, however, suffers from ambiguity, inconsistency, and contradictions making it un-suitable for ordinary software engineers to measure the design quality of software products (Al-Kilidar 2005; Haboush et al. 2014; Kitchenham and Pfleeger 1996; Koh 2016, 2017; Koh and Whang 2016).

SQuaRE defines three distinctive views of software quality and quality requirements (ISO/IEC 25030:2007) as follows:

. **In use view**: *The software quality in use view is related to application of the software in its operational environment, for carrying out specific tasks by specific users.*[1)]

* shkoh@cbnu.ac.kr

. **External view**: *External software quality provides a 'black box' view of the software and addresses properties related to the execution of the software on computer hardware and applying an operating system.*

. **Internal view**: *Internal software quality provides a 'white box' view of software and addresses properties of the software product that typically is available during the development. Internal software quality is mainly related to static properties of the software.*

SQuaRE defines two quality models regarding system and software product: product quality model and quality in use model. According to SQuaRE, the quality in use model rests on the in-use view, and the product quality model rests on both the internal view and the external view simultaneously. Other basic software quality models such as, for example, Boehm (Boehm et al., 1978), Dormey (1995), FURPS (Grady, 1992), and McCall (McCall et al., 1977), generally do not adopt the notion of quality view explicitly. A valid model of software quality views, however, may make it possible to organize the extensive body of knowledge regarding software quality systematically.

It is noticeable that the external view is regarding execution while the internal view is regarding source code of a software product. The product quality model defines the same quality characteristics for two very contrasting types of software entity. Koh (2017) proposes the following principle of one quality view:

. **Principle of one view**: *A software quality model should correspond to one and only one software quality view.*

According to the principle, SQuaRE product quality model should be decomposed into, at least, two distinct models.

Koh (2016, 2017) also suggests the following model for software quality view:

. **End view**: *It represents the effort to find out what the software product should be good for. The quality characteristic of this view corresponds to the effect of good quality of the software product.*
  - **Activity view**: *It represents the effort to identify the types of activity, for which a software product should be good. It focuses on the activity itself or the immediate effect of the activity.*

. **Means view**: *It represents the effort to make the software product good for various ends. The characteristic of this view corresponds to the cause of desirable effects.*
  - **Intrinsic view**: *It represents the effort to find out intrinsic characteristics which facilitate software activities.*
  - **Contingency view**: *It represents the effort to find out non-intrinsic characteristics such that the degree to which they facilitate software activities varies depending on contingencies or context.*

Koh (2017) also argues that all the activity view, intrinsic view, and contingency view are involved in the product quality model. As the result, he argues that the product quality model should be decomposed into, at least, three distinct quality models, which is the purpose of this paper. The results show well that his approach can eliminate the inconsistency and conceptual vagueness associated with SQuaRE.

Koh (2017) defines software activity as the activity which is performed on the software product by a person or a group of persons. Among 8 product quality characteristics, usability, portability, and maintainability are directly related with software activities such as using, transferring, and modifying, respectively. Koh and Whang (2016) call such characteristics activity quality characteristics. They also suggest to use the suffix

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

'-ability' only for the software activity characteristic to make it clear whether a characteristic is an activity quality characteristic or not. This paper follows their naming scheme.

SQuaRE does not distinguish the characteristics of system and those of software product explicitly (Koh 2017). The main theme of this paper is the quality of software. Without any further mention, the terms quality and product quality model is used to denote the quality of software and the product quality model of SQuaRE, respectively. System is regarded as a contingency factor which affects the execution of a software product. The characteristics and measures of the product quality model are classified into those intrinsic, of contingency, and of activity in sections 2 and 3. The implications of the results are discussed in section 4.

2. Decomposition of the characteristics of SQuaRE's product quality model

The product quality model of has 8 characteristics (compatibility, functional suitability, maintainability, performance efficiency, portability, reliability, security, and usability), 31 sub-characteristics, and 86 measures (refer to Table 1). 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'* (ISO/IEC 25010, 2011: p.12). SQuaRE defines quality in use as '*the degree to which a product can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use'* (ISO/IEC 25010, 2011: p.8). It explains 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.'* It also add *'context of use is relevant to both quality in use and some product quality characteristics*'(ISO/IEC 25010, 2011: pp.9, 12). So, it is logical to interpret usability defined by SQuaRE as a triple of effectiveness, efficiency, and satisfaction associated with using a software product.

Table 1. Characteristics and sub-characteristics of SQuaRE product quality model

Characteristic Sub-Cha.	**Key Words**	**Activity Involved**
Compatibility	Sharing same environment	–
CCo[2)]	Performing required functions efficiently	–
CIn	Exchanging and using information	–
Functional Suit.	Functions meeting stated and implied needs	–
FAp	Facilitating the accomplishment of specified tasks and objectives	–
FCp	Covering specified tasks and user objectives	–
FCr	Providing correct results	–
Maintainability	Effectiveness, efficiency	Modifying
MAn	Effectiveness, efficiency	Analyzing
MMd	Effectiveness, efficiency	Modifying
MMo	Being composed of discrete components	–
MRe	Reusing an asset	*Reusing*
MTe	Effectiveness, efficiency	Testing

Characteristic Sub-Cha.	Key Words	Activity Involved
Performance Eff.	The amount of resources used	–
PCa	Maximum limits of parameter	–
PRu	Amount and types of resources used	–
PTb	Times and throughputs rates	–
Portability	Effectiveness, efficiency	Transferring
PAd	Effectiveness, efficiency	Adapting
PIn	Effectiveness, efficiency	Installing
PRe	Replacing another product (in the sense of similarity)	–
Reliability	Performing specified functions	–
RAv	Being operational and accessible when required	–
RFt	Operating as intended under presence of faults	–
RMa	Meeting the needs for reliability under normal operation	–
RRe	Recovering data, re-establishing the desired state	*Recovering*
Security	Protecting information and data	–
SAc	Tracing the entity of an action uniquely	*Tracing*
SAu	Proving the identity of a subject or resource	*Proving*
SCo	Data are accessible to only the people authorized	–
SIn	Preventing unauthorized access or modification	–
SNo	Preventing repudiation of events or actions	*Proving*
Usability	Effectiveness, efficiency, satisfaction	Using
UAc	Characteristics and capabilities of people	
UEp	Protecting users against making errors	
UIn	Pleasing and satisfying interaction	
UAp	Recognizing appropriateness	–
ULe	Effectiveness, efficiency, satisfaction, freedom from risk	Studying
UOp	Easiness	*Operating*

Abbreviation of sub-characteristics: CCo=Co-existence, CIn=Interoperability, FAp=Functional appropriateness, FCp=Functional completeness, FCr=Functional correctness, MAn=Analyzability, MMo=Modularity, MRe=Reusability, MMd=Modifiability, MTe=Testability, PAd=Adaptability, PCa=Capacity, PIn=Installability, PRe=Replaceability, PRu=Resource utilization, PTb=Time behavior, RAv=Availability, RFt=Fault tolerance, RMa=Maturity, RRe=Recoverability, UAc=Accessibility, UAp=Appropriateness recognizability, SAc=Accountability, SAu=Authenticity, SCo=Confidentiality, SIn=Integrity, SNo=Non-repudiation, UEp=User error protection, UIn=User interface aesthetics, ULe=Learnability, UOp=Operability
Source: Koh (2017)

2) The first letter of abbreviation of sub-characteristic denotes its super-characteristics.

Beside usability, SQuaRE defines learnability (ULe), maintainability, analyzability (MAn), modifiability (MMd), testability (MTe), portability, adaptability (PAd), and installability (PIn) as the multiple-valued characteristics composed with effectiveness, efficiency, satisfaction, and/or freedom from risk. It is noticeable that such characteristics are those associated with software activity such as learning, maintaining (modifying), analyzing, testing, transferring, adapting, or installing (and/or uninstalling) respectively. They are of activity view.

Reusability (MRe), recoverability (RRe), accountability (SAc), authenticity (SAu), non-reputation (SNo), and operability (UOp) (those written in italic font in Table 1) can be also regarded as activity quality characteristics although they are not defined in terms of effectiveness, efficiency, satisfaction, or freedom from risk. They involve reusing an asset, recovering the data and re-establishing the desired state of the system, tracing the actions of an entity, proving the identity of a subject or resource, proving actions or events to have taken place, and operate and control a product, respectively. Besides, security also involves protecting information and data. Availability also involves accessing the system. These tasks may be performed by the system or software entirely. They, however, may also involve human activity. In this regard, they can be classified as software activity.

The software activity quality characteristics implied in the product quality model can be classified as follows. Some of the characteristics are renamed according to Koh and Whang's (2016) name scheme (those in parenthesis are the original names in the product quality model).

. Access-ability (availability)
. Changeability (maintainability)
  − Adaptability
  − Analyzability
  − Modifiability
  − Reusability
  − Testability
. Customizability
  − Functional customizability
  − User interface customizability
. Portability
  − Install-ability
  − Un-install-ability (installability)
. Protect-ability (security)
  − Entity trace-ability (accountability)
  − Identity prove-ability (authenticity)
  − Repudiation preventability (non-reputation)
. Study-ability (learnability)
  − On-the-product study-ability
  − Off-the-product study-ability
. System operability (reliability)
  − Recoverability
. Usability
  − Control-ability (operability)

It is noticeable that study-ability is used instead of learnability and excluded from the sub-characteristics of usability. 'Learning' and 'recognizing appropriateness' are not software activities since they are mental phenomenon that occurs inside a person and that cannot be observed externally (Koh, 2017). Moreover, they may happen in the course of using or studying, or simply as the result of contact with advertisements or word-of-marketing (Koh, 2017). On the other hand, the term studying typically means *'the activity of learning or gaining knowledge'* (Oxford Learner's Dictionary, 2016) or *'spending time learning about a particular subject or subjects'* (Collins Cobuild Advanced Learner's English Dictionary, 2016). It specifies and emphasizes 'spending time to perform an activity.' Studying is a type of software activity, which can happen independently of using. Study-ability is classified further into on-the-product study-ability and off-the-product study-ability according to whether it is performed using the user interface of the product or not.

Change-ability is also used instead of maintainability. Although the term 'maintenance' is customarily used for both software and hardware, it means the opposite for software and hardware: Maintaining software is changing it whereas maintaining hardware is preventing it from changing (Koh and Han, 2015). For hardware, maintenance generally means preserving the original state of an item. For example, NF EN 13306 (2001) defines maintenance, irrespective of the type of items except software, as *"combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function"*. For hardware, preventive maintenance is one of the most important types of maintenance, which is intended to *reduce the probability of failure and the degradation of the functioning of an item* (NF EN 13306, 2001). Software, however, is never degraded or deteriorated by using. So, maintenance is needless for software. In this regard, Koh and Han (2015) propose to use the term 'post lifecycle change' instead of maintenance. In this paper, however, 'changing' is used to emphasize that changing is important even during the development stage of a software product.

Customizing is defined as a type of software activity performed on the execution through user interface to make the product better to use. Customizing page on a web site is a good example. It is involved in the measures of operability in the product quality model: functional customizability (*the proportion of functions and operational procedures which a user customizes for his convenience*) and user interface customizability (*the proportion of user interface elements which can be customized in appearance*) (ISO/IEC 25023, 2016: p.17). Customizability is separated from usability too. Customizing is not regarded as a sub-type or a sub-activity of using.

Koh (2016) regards modularity as the only pure intrinsic characteristic of the product quality model. He classifies the other characteristics as contingency characteristics. Koh (2016, 2017) argues that there is cause-and-effect relationship among the activity quality characteristics and the contingency characteristics (refer to Table 2). It is noticeable that functional suitability (degree to which a product or system provides functions that meet stated and implied needs when user under specified conditions), for example, is not classified as an activity quality characteristic since it is the product itself which 'provides' functions.

Table 2. Influence and dependency of contingency quality characteristics: a subjective opinion.

Characteristic	Sub-Characteristic	Depends on	Influences
Compatibility	Co-existence	System (software portfolio)	Installing
	Interoperability		
Functional suitability	Functional appropriateness	Social and organizational environment (work and task)	Using
	Functional completeness		
	Functional correctness		
Performance efficiency	Capacity	System (data volume and structure)	
	Resource utilization		
	Time behavior		
Reliability	Availability	System	
	Fault tolerance		
	Maturity		
	Recoverability		
Usability	User error protection	System, environment (work and task, physical)	
	User interface aesthetics	System (interface device), Physical environment (illumination, noise, etc.)	

Source: Koh (2016)

3. Decomposition of the measures of the product quality model

Every measure of the product quality model is a scalar defined to be the ratio of 'A/B' or '1-A/B'. As a matter of course, *usability measures are used to assess usability* (ISO/IEC 25023, 2016: p.14). SQuaRE, however, does not provide any hint nor guideline regarding how to convert 22 scalar values of usability measures to the triple of effectiveness, efficiency, and satisfaction. It is obvious that simply summating singletons makes a singleton. How should the three different values of effectiveness, efficiency, and satisfaction be derived from 22 scalar values? This problem applies to every activity characteristic of product quality model. That is, the characteristics cannot be obtained from their measures.

Koh (2017) converts the measures of the product quality model into check items for developers or maintainers. He classifies the items into those regarding data processing (valid, correct, and efficient), data interface, user interface (information providing, data inputting, process controlling, and presentation/appearance/composite), composite, software engineering (architecture, coding, and diagnosis/testing), system (capacity, runtime behavior, and operation management), and software activity. The fact that most of the measures can be converted to check items for developers or maintainers implies that they are of means view.

According to Koh's (2017) classification, all the measures of usability are regarding user interface (refer to Table 3). User interface affects usability. However, user interface is not effectiveness, efficiency, nor satisfaction by itself. Moreover, the measures regarding valid, exact, and efficient dada processing are classified chiefly into functional suitability and performance efficiency. This may be interpreted to imply that the effectiveness and efficiency associated using a software product are affected chiefly by user interface, but not by data processing. The relationship implied in the model is very incomplete and misleading.

Table 3. Discordance between activity characteristic and measures in SQuaRE product quality model

Characteristic Sub-Cha.	**Activity Involved**	**Variables to be Measured**	
		In Definition	**Koh's (2017) Classification**
Maintainability	Changing	Et, Ec	--
MAn	Analyzing	Et, Ec	SWE (DT), System (runtime behavior)
MMd	Modifying	Et, Ec	Activity view (developing & maintaining)
MMo	--	Modularity	SWE (architecture)
MRe	*Reusing*	Reusability	SWE (architecture, coding)
MTe	Testing	Et, Ec	SWE (DT)
Portability	Transferring	Et, Ec	--
PAd	Adapting	Et, Ec	Composite
PIn	Installing	Et, Ec	Activity view (installing), System (OM)
PRe	--	Replaceability	Data interface, Composite
Usability	Using	Et, Ec, St	--
UAc	Accessing	Accessibility	UI (IP, Etc)
UAp	--	Recognizability	UI (IP)
UEp	--	Protection	UI (data inputting)
UIn	--	Aesthetics	UI (Etc)
ULe	Studying	Et, Ec, St, FR	UI (IP, Data inputting)
UOp	*Operating*	Easiness	UI (IP, data inputting, process controlling, Etc)
Reliability,	--	Reliability	--
RAv	*Accessing*	Availability	System (runtime behavior)
RRe	*Recovering*	Recoverability	System (runtime behavior, OM)
Security	--	Security	--
SAc	*Tracing*	Accountability	System (OM)
SAu	*Proving*	Authenticity	System (OM)
SNo	*Preventing*	Non-repudiation	System (OM)

Abbreviation: DT=diagnosis and Test, Ec= efficiency, Et=effectiveness, FR= freedom from risk, IP=information provision, OM=operation management, PAC=presentation, appearance, composite, St=satisfaction, SWE=software engineering, UI=user interface.

SQuaRE's measures of activity quality characteristics generally do not measure their characteristics which they are supposed to measure. Instead, they generally measure the attributes regarded to facilitate the associated software activities. However, the cause-and effect relationship established in the model is very incomplete and misleading.

A comprehensive and valid system of software quality models should be established first. Then, theories regarding the cause-and-effect relationships among software quality characteristics and measures can be developed. The inconsistencies in the product quality model shows well the necessity for this point of view to be stressed explicitly. First of all, software quality characteristics and measures should be classified according their corresponding quality views.

4. Discussions

4.1 Software quality characteristics and types of software entity

SQuaRE defines the terms software quality characteristic, attribute, measure, and measurement, as *category of software attributes that bear on software quality,' 'inherent property or characteristic of an entity that can be distinguished qualitatively or qualitatively by human or automated means,' 'variable to which a value is assigned as the result of measurement', and 'set of operations having the object of determining a value of a measure*, respectively (ISO/IEC 25010, 2011: p.19). SQuaRE's definitions are not rigorous and involve tautology. Especially, some characteristics and measures are not inherent to the software product. It is also noticeable that some measures of SQuaRE can be regarded as software quality characteristics by their definitions.

Especially, by letting the external quality and the internal quality share the same model of characteristics, SQuaRE virtually lets the distinction of the internal view and the external view be the issue restricted to measurement. According to SQuaRE, 21 measures can be measured only externally. If SquaRE is right, this implies that these measures are strongly affected by the system, which means that they as the measures of the software product quality are with low validity. The logical conclusion is that they are appropriate for the measures of the quality of associated contingencies rather than for the measures of software quality.

According to SQuaRE, 3 measures can be measured only internally. SQuaRE presumes that the external product quality is affected by the target computer system and that the quality in use is affected by both the human-computer system including the target computer system and the particular context in which the product is being used (ISO/IEC 25010, 2011: pp.4-5; ISO/IEC 25022, 2016: p.7). However, the property of system cannot be loaded on internal measures. So, these 3 measures are solely for software quality and are of intrinsic view.

According to SQuaRE, the other 62 measures can be measured both internally and externally. SQuaRE adds following explanation (ISO/IEC 25023, 2016): *Internal measures for performance efficiency are applicable to static design documents or source code. These measured values are able to be contained by estimation of theoretical calculation amount of designed algorithms, number of function calls or steps of executable code. However, applications of external measures for performance efficiency on intermediate executable prototype during design are helpful to understand actual gaps between internal and external measures and calibrate estimation for internal measures. Internal measures for usability are applicable to static mock-up of screen display, specification for usability design, a set of message text files, user manuals, source code for user interfaces and so on. However, applications of external measures for usability on intermediate executable* *prototype during development are helpful to understand actual gaps between internal and external measures. If available, application of quality in use measures to executable prototype during development is also very helpful*. The explanation illustrates that the internal measurement and external measurements of these measures involve different forms of the software product.

A software program can take three distinctive types of entities: the source code, the executable (*that can be executed on the target machine architecture*), and the execution (*the solution provided through running of the program-executable code on some hardware and is a model of some virtually constructed environment created to satisfy the intended requirements*) (Younessi, 2002: p.23). For example, the failure (*it is said to have occurred when the software product does not satisfy expectations*), fault (*an incorrect state entered by a program executable or an incorrect transformation undergone; it is characteristics of the executable code and does not necessarily cause software failure, which is also impacted by how the software is used*), defect (*an imperfection in the software engineering work product that require rectification), and bug (a defect that cause the generation of a fault; some defects such as the provision of a comment on the wrong line or in the wrong section of the source code do not generate faults in the executable*) should be distinguished, although they are very closely related (Younessi, 2002: pp.23-25). They are associated with distinctive software entity types. Failure is a transient phenomenon associated with the execution. Fault and bug are persistent and static property associated with the executable or source code, respectively. Failure is a symptom while fault and bug are its causes. Hence, they should be defined distinctively in their concepts, and should be detected by distinctive ways. So the software quality measures should be measured.

In this paper, intrinsic quality characteristics, contingency quality characteristics, and activity quality characteristics are defined as follows.

. **Intrinsic quality characteristic**: Evaluation criterion of the software product, which has one and only one unique value for a product.
. **Contingency quality characteristic**: Evaluation criterion of the contingency including the product, which has one and only one unique value for a contingency.
. **Activity quality characteristics**: Evaluation criterion of the activity performed on the product in the contingency, which has one and only one unique value for a software activity instance.

Any evaluation criterion can be a characteristic. Theoretical interest, however, is only in the generally important attributes. An inherent attribute of the software product can be an intrinsic quality characteristic. The value of an intrinsic quality characteristic of a software product is not changed unless the product itself is changed. A contingency is the composite which is composed with the target product and its environment. It has its own persistent and emerging attributes, which are the most important candidates of contingency quality characteristics. A contingency quality characteristic of a software product can take multiple values since multiple contexts can exist for a product. It is needless to define an attribute invariant over contingencies as a contingency quality characteristic, since it is more effective to define it as an intrinsic quality characteristic. An activity quality characteristic can have more values per a software product than a contingency quality characteristic since multiple instances of software activity can be taken place in a contingency. An inherent attribute cannot be an activity quality characteristic by the very definitions. The set of relevant activity quality characteristics may vary according to the type of activity too, since the set of relevant attributes of activity may vary according to the type of activity.

In this paper, the quality measure is defined as the following:

. **Quality measure**: an operational definition of a quality characteristic, which specifies the process to assign a value to the corresponding characteristic.

If there is no measurement error, it is enough to measure an intrinsic quality characteristic of a product only once. Multiple measurements, however, are generally needed for a contingency quality characteristic and an activity quality characteristic per a product. If a contingency quality characteristic or an activity quality characteristic is defined to be measured internally (by examining the source code of the product), then it will be neither reliable nor valid.

4.2 Definition of software quality measure congruent with the associated software quality view

The product quality measures also can be classified into those intrinsic, of contingency, and of activity (refer to Table 4). It is noticeable that 18 measures both internal and external are classified as intrinsic. For example, 'functional correctness (the proportion of functions which provide the correct values)' does not change unless the software is changed. So, it is intrinsic. However, it can be confirmed by both white-box testing and black-box testing. It is obvious, however, that white-box testing and black-box testing cannot be performed by a process. So, at least, two distinctive measures of 'functional correctness' should be defined if it is to be measured both internally and externally. Generally speaking, although the conceptual definition of a variable is unique, multiple measures of the variable are required whenever it is measured by distinctive processes. In this respect, intrinsic measures of the product quality model can be regarded as quality characteristic, whose operational definition is not defined yet.

Table 4. Reclassification of SQuaRE product quality measures according to Koh's (2017) framework

Product Quality Model of SQuaRE		**Koh (2017)**	
View	**Measures**	**Category**	**View**
Internal	(SIn) Buffer overflow prevention	DInput-UI	Intrinsic
	(MRe) Coding rules conformity	Coding	
	(MMo) Cyclomatic complexity adequacy	Architecture	
Both	(MMo) Coupling of components; (MRe) Reusability of assets		
	(FCr) Functional correctness	Correct DP	
	(UAp) Description completeness, Demonstration coverage, Entry point self-descriptiveness; (ULe) User guidance completeness, Error messages understandability, Self-explanatory user interface; (UOp) Message clarity, Monitoring capability	Information Provision -UI	
	(ULe) Entry fields defaults; (UEp) User entry error correction	DInput-UI	
	(UOp) Functional customizability, Undo capability	PC-UI	
	(UOp) Operational consistency, Understandable categorization of information, Appearance consistency	PAC-UI	

Product Quality Model of SQuaRE		**Koh (2017)**	
View	**Measures**	**Category**	**View**
Both	(FAp) Functional appropriateness of usage objective, Functional appropriateness of system; (FCp) Functional coverage	Valid DP	Contingency
	(CIn) Data formats exchangeability, Data exchange protocol sufficiency, External interface adequacy	Data Interface	
	(UEp) Avoidance of user operation error, User error recoverability; (UOp) Input device support	Data Input-UI	
	(UAc) Accessibility for users with disabilities, Supported languages adequacy; (UIn) Appearance aesthetics of UI; (UOp) UI customizability	PAC-UI	
	(PRe) Usage similarity, Product quality equivalence	Composite	
	(MAn) Diagnosis function effectiveness, Diag. function sufficiency; (MTe) Test function completeness, Autonomous testability, Test restartability	Diagonsis & Test	
	(RRe) Backup data completeness; (SAc) User audit trail completeness, System log retention; (SAu) Authentication mechanism sufficiency, Authentication rules conformity; (SCo) Access controllability, Data encryption correctness, Strength of cryptographic algorithms; (SIn) Data integrity, Internal data corruption prevention; (SNo) Digital signature usage	Operation Management -System	
	(MAn) System log completeness; (PTb) Turnaround time adequacy	RB-System	
	(PCa) Transaction processing capacity, User access capacity; (RFt) Redundancy of components; (PTb) Mean response time, Response time adequacy, Mean turnaround time, Mean throughput	Capacity -System	
External	(PCa) User access increase adequacy		
	(RAv) System availability, Mean downtime; (RFt) Mean fault notification time; (RMa) Mean time between failure, Failure rate; (RRe) Mean recovery time	Runtime Behavior -System	
	(PIn) Ease of installation	OM-System	
	(PRe) Data reusability/import capability	Valid DP	
	(PRu) Mean processor utilization, Mean memory utilization, Mean I/O devices utilization, Bandwidth utilization	Efficient Data Process	
	(CCo) Co-existence with other products; (PAd) Hardware environmental adaptability, System software environmental adaptability, Operational environment adaptability; (PRe) Functional inclusiveness	Composite	
	(PIn) Installation time efficiency	Installing	Activity
	(RFt) Failure avoidance; (RMa) Test coverage	Developing & Changing	
Both	(MMd) Modification efficiency, Modification correctness, Modification capability; (RMa) Fault correction		

Abbreviation: DInput (data input), DP (data process), OM (operation management), PAC (presentation, appearance & composite), PC (process control), RB(Runtime Behavior), UI (user interface)

It is especially noticeable that 40 measures both internal and external are classified as of contingency. However, it is not sure how 'accessibility for users with disabilities (*the ratio of functions successfully usable by the users with a specific disability*),' for example, can be determined by examining the software product internally. This example illustrates the necessity to critically review and revise the measures of the product quality model according to the framework suggested in this paper, although it is beyond the scope of this paper to review and revise each items one by one. Especially, the quality of software and the quality of system should be strictly distinguished since a software product can be executed on multiple systems. The designer of such a product should address the quality of the product, but should not address the quality of a specific system.

On the other hand, 18 measures classified to be contingent are defined as external by SQuaRE. For example, 'user access increase adequacy (*how many users can be added successfully per unit time*)' is considered to be able to be measured only externally. This implies that it is regarded to be strongly affected by contingency factors (for example, the system and communication networks). So, it amount to declaring implicitly that it is not of internal view. It will be better to explicitly classify it into a contingency quality model.

Four measures both internal and external are classified as of activity. However, for example, how can 'modification efficiency (*how efficiently are the modifications made compared to the expected time*) be measured internally? It can be measured only by observing the instance of modification. It is not an attribute of a software product.

The discussions show well how serious problems SQuaRE has and how to resolve them. It is necessary to critically review and revise every measure of SQuaRE product quality model too, which is beyond the scope of this paper.

5. Conclusions

This paper critically reviews SQuaRE's product quality model according to Koh's (2016, 2017) frame of software quality. First, this paper defines intrinsic quality characteristics, contingency quality characteristics, and activity quality characteristics as the evaluation criteria of the software product, contingencies, and activity performed on the product in a contingency, respectively. The contingency is defined as the composite of a software product and the environment in which the product is executed. A contingency generally has its own persistent and emerging attributes. An intrinsic quality characteristic is regarding the source code while a contingency quality characteristic is regarding contingencies. In this regard, an inherent attribute of the product, for example, modularity is intrinsic. A persistent and emergent attribute of the contingency is generally an important candidate of the contingency quality characteristic. For example, user interface aesthetics is of contingency.

Second, this paper proposes a hierarchy of activity quality characteristics which are referred to explicitly or implied implicitly in the definitions quality characteristics or measures of the product quality model. The hierarchy can serve as a prototype of software activity quality model. This paper also classifies the measures of product quality model into those intrinsic, of contingency, and of activity.

Third, this paper defines the measure of a quality characteristic as an operational definition of the quality characteristic, which specifies the process to assign a value to the corresponding characteristic. If there is no measurement error, it is enough to measure an intrinsic quality characteristic of a product only once. Multiple measurements, however, are generally needed for a contingency quality characteristic since there are generally multiple contingencies associated a software product. Even more measurements may be needed for an activity quality characteristic since multiple instance of an activity type can be performed in a contingency.

The definitions of software characteristics and measures in this paper resolve the conceptual vagueness and tautologies associated with those of SQuaRE. It is especially noticeable that the most of the product quality measures are not the measures as defined in this paper. It is obvious that the same measurement procedure cannot be executed both internally and externally. The sheer fact that 62 measures defined to be both internal and external by SQuaRE shows well that they cannot be valid quality measures.

SQuaRE's product quality model contains too much items for ordinary software engineers to deal with properly. This paper shows how to resolve the problem: decompose the model into three distinctive models corresponding to intrinsic view, contingency view, and activity view respectively. The decomposition will make the items included in each model more homogeneous. The quality characteristics and measures of the product quality model should be revised and rigorously redefined to be included in the models. More items may be needed for the models to be completed. Additional views may be necessary to construct a comprehensive system of software quality models. Further researches are required to address this issue rigorously.

A body of theories to explain the cause-and-effect relationships among software quality models should be developed too. The system of software quality models and the body of theories to explain the relationships among the models will together constitute a body of knowledge on which software engineers can rely to make the software product with which various stakeholders are satisfied.

References

Al-Kilidar, H., Cox, K. and Kitchenham, B. (2005). The use and usefulness of the ISO/IEC 9126 quality standard. In *Proceedings of International Symposium on Empirical Software Engineering* 2005, IEEE, 126-132.

Boehm, B.W., Brown, H. and Lipow, M. (1978). Quantitative evaluation of software quality. TRW Systems and Energy Group.

Collins Cobuild Advanced Learner's English Dictionary. Reference date: 05/16/2016.

Dormey, R.G. (1995). A model for software product quality. *IEEE Transactions on Software Engineering*, *21*(2), 146-162.

Grady, R. B. (1992). *Practical Software Metrics for Project Management and Process Improvement.* Prentice Hall: Englewood Cliffs, NJ, USA.

Haboush, A., Alnabhan, M., Al-Badareen, A., Al-Nawayseh, M., and El-Zaghmouri, B. (2014). Investigating software maintainability development: a case for ISO 9126. *International Journal of Computer Science Issues (IJCSI)*, *11*(2), 18-23.

ISO/ IEC 25030:2007 (2005). *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – Quality Requirements,* ISO.

_______ 25010:2011 (2011). *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models*, ISO.

_______ 25022:2016 (2016). *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – Measurement of Quality in Use*, ISO.

_______ 25023:2016 (2016). *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and Software Product Quality*, ISO.

Kitchenham, B. and Pfleeger, S. L. (1996). Software quality: the elusive target [special issues section]. *IEEE software*, *13*(1), 12-21.

Koh, S. (2016). Cause-and-effect perspective on software quality: application to ISO/IEC 25000 series SQuaRE's product quality model. *Journal of Information Technology Applications & Management*, *23*(3), 71-86.

______ (2017). The checklist for system and software product quality implied in the product quality model of ISO/IEC 25000 series SQuaRE. In *Proceedings of the 17th International Conference on IT Applications and Management (ITAM17): Smart and Intelligent Civilization*, 126-136.

Koh, S. and Han, M. P. (2015). Purposes, results, and types of software post life cycle changes. *Journal of Information Technology Applications & Management, 22*(3), 143-167.

_______ and Whang, J. (2016). A critical review on ISO/IEC 25000 SQuaRE Model. In *Proceedings of the 15th International Conference on IT Applications and Management (ITAM15): Mobility, Culture and Tourism in the Digitalized World*, 42-52.

McCall, J. A., Richards, P. K. and Walters, G. F. (1977). *Factors in Software Quality, Volumes I, II, and III US Rome Air Development Center Reports,* US Department of Commerce, USA.

Miguel, J. P., Mauricio, D. and Rodríguez, G. (2014). A review of software quality models for the evaluation of software products. *International Journal of Software Engineering & Applications (IJSEA), 5*(6), 31-54.

NF EN 13306 (2001). *Terminologie de la Maintenance.*

Oxford Learner's Dictionary. Reference date: 05/16/2016.

Younessi, H. (2002). *Object-Oriented Defect Management of Software.* Prentice Hall PTR: Upper Saddle River, NJ.