ISO/IEC JTC1/SC7/WG6 N???

Date: 31-Aug-2011

ISO/IEC WD 25023

TITLE:

Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality

DATE: 2011-08-31

SOURCE: JTC1/SC7/WG6

WORK ITEM: Project

STATUS: Version 0.1.0 Working Draft for Sydney, Interim Meeting, November 2011.

DOCUMENT TYPE: WD

ACTION: WD for editor's review

PROJECT EDITOR: Prof. Motoei AZUMA, Japan, azumam@waseda.jp

DOCUMENT EDITOR: KeumSuk Lee, Korea

CO-EDITOR: Witold Suryn, Canada
Yangyang Zhang, China
Yukio Tanitsu, Japan
Atushi Yamada, Japan
Juyeun Han, Korea
Nigel Bevan, UK
Jean-Marc Desharnais, Canada

ISO/IEC JTC 1/SC 7 N_____

Date: **2011-08-31**

ISO/IEC 25023

ISO/IEC JTC 1/SC 7/WG 6

Secretariat: SCC

Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electro technical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 25023 makes a part of SQuaRE series of standards and was prepared by Joint Technical Committee ISO/IEC JTC 1, information technology, Subcommittee SC 7, Software and System Engineering.

This first edition of ISO/IEC 25023 aggregates and replaces ISO/IEC 9126-2 and ISO/IEC 9126-3, which have been technically revised.

SQuaRE series of standards consists of the following divisions under the general title Systems and software product Quality Requirements and Evaluation:

ISO/IEC 2500n - Quality Management Division,

ISO/IEC 2501n - Quality Model Division,

ISO/IEC 2502n - Quality Measurement Division,

ISO/IEC 2503n - Quality Requirements Division, and

ISO/IEC 2504n - Quality Evaluation Division.

Introduction

The general goal of creating the SQuaRE series of standards is to move to a logically organised, enriched and unified series covering three complementary processes: requirements specification, measurement and evaluation. The purpose of the SQuaRE series of standards is to assist those developing and acquiring system/software products with the specification and evaluation of quality requirements. It establishes criteria for the specification of system/software product quality requirements and their evaluation. It includes a two-part quality model for aligning customer definitions of quality with characteristics of the system/software product. In addition, the series provides recommended measures of system/software product quality characteristics that can be used by developers, acquirers, and evaluators.

It has to be stressed that the SQuaRE series of standards is dedicated to system/software product quality only. The Quality Management Division of the SQuaRE series deals with system/software products, and is separate and distinct from the "Quality Management" of processes which is defined in the ISO 9000 family of standards.

The major benefits of the SQuaRE series over its predecessor standards include:

the coordination of guidance on system/software product quality measurement and evaluation,

guidance for the specification of system/software product quality requirements, and

harmonization with ISO/IEC 15939 in the form of Quality Measurement Reference model presented in ISO/IEC 25020 - Software engineering: System/software product Quality Requirements and Evaluation (SQuaRE) Measurement reference model and guide.

The major differences between ISO/IEC 9126, ISO/IEC 14598 and SQuaRE series of standards are:

the introduction of the new general reference model,

the introduction of dedicated and detailed guides for each division,

the introduction of Quality Measure Elements within the Quality Measurement Division,

the introduction of the Quality Requirements Division,

incorporating and revising the evaluation processes,

the introduction of guidance for practical use in the form of examples,

co-ordination and harmonization of the content with ISO/IEC 15939.

SQuaRE consists of the following five divisions:

Quality Management Division (2500n),

Quality Model Division (2501n),

Quality Measurement Division (2502n),

Quality Requirements Division (2503n), and

Quality Evaluation Division (2504n).

SQuaRE provides:

Terms and definitions,

Reference models,

General guide,

Individual division guides, and

Standards for requirements specification, planning and management, measurement and evaluation purposes.

SQuaRE includes international standards on quality model and measures, as well as on quality requirements and evaluation. SQuaRE replaces the current ISO/IEC 9126 series and the 14598 series.

ISO/IEC 25010 defines terms for the software quality characteristics and how these characteristics are decomposed into subcharacteristics. ISO/IEC 25010, however, does not describe how any of these subcharacteristics could be measured. ISO/IEC 25022 defines quality–in use measures and ISO/IEC 25023 defines product quality measures which aggregates previous ISO/IEC 9126-2(external measures) and ISO/IEC 9126-3(internal measures) for measurement of the characteristics or the subcharacteristics of system and software product. Internal measures measure the software itself, external measures measure the behaviour of the computer-based system that includes the software, and quality in use measures measure the effects of using the software in a specific context of use.

This International Standard provides the required quality measures for measuring attributes of eight quality characteristics of system/software product quality model defined in ISO/IEC 25010. The set of required measures in this International Standard was selected by means of a survey conducted involving several large commercial and academic institutions. This set represents a default kernel of measures, which are proven to be beneficial and in common use, together with a wide range of other measures that are required for more specific purposes.

All the quality measures described in the SQuaRE series are derived from one or more quality measure elements described in the ISO/IEC 25021. When evaluating the quality measure, the user should first evaluate the relevant quality measure elements presented, described and defined in that International Standard.

The quality measures listed in this International standard are not intended to be an exhaustive set. Developers, evaluators, quality managers and acquirers may select measures from this standard for defining requirements, evaluating system/software products, measuring quality aspects and other purposes. They may also modify the measures or use measures which are not included here. This standard is applicable to any kind of system/software product, although each of the measures is not always applicable to every kind of system/software product.

This international standard is intended to be used together with ISO/IEC 25010. It is strongly recommended that the user refer to ISO/IEC 2500n, ISO/IEC 2501n, ISO/IEC 2503n, and ISO/IEC 2504n division of standards prior to using this International standard and the associated measurement standards, particularly if the user is not familiar with the use of software measures for product specification and evaluation. These standards discuss the planning and use of the software quality measures defined in the ISO/IEC 2502n series on system/software product quality measurement.

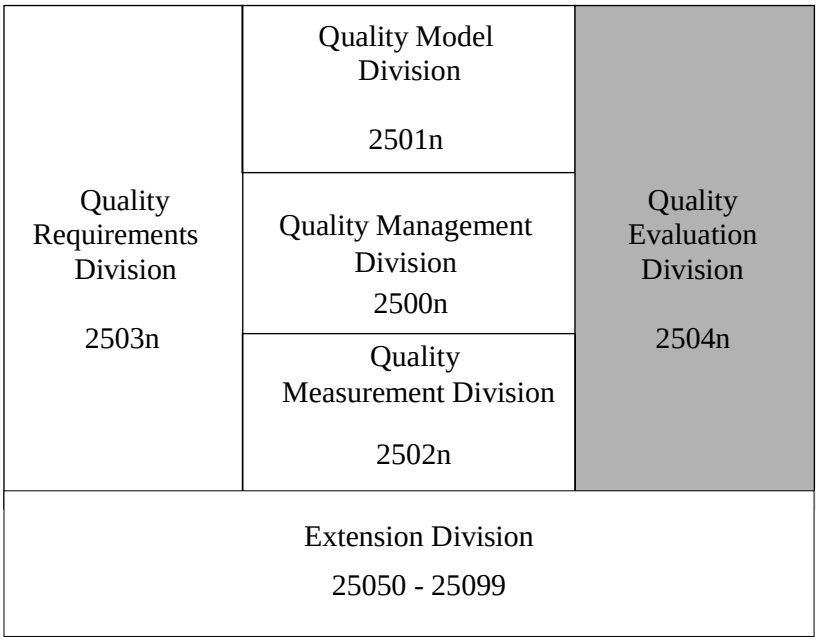

Figure 1- Organisation of the SQuaRE series of international standards

Figure 1 illustrates the organisation of the SQuaRE series representing families of standards, further called Divisions.

The Divisions within SQuaRE model are:

ISO/IEC 2500n - Quality Management Division. The standards that form this division define all common models, terms and definitions referred further by all other standards from SQuaRE series. Referring paths (guidance through SQuaRE documents) and high level practical suggestions in applying proper standards to specific application cases offer help to all types of users. The division provides also requirements and guidance for a supporting function which is responsible for the management of system/software product requirements specification and evaluation.

ISO/IEC 2501n - Quality Model Division. The standard that forms this division present detailed quality models for software, quality in use and data. Practical guidance on the use of the quality model is also provided.

ISO/IEC 2502n - Quality Measurement Division. The standards that form this division include a system/software product quality measurement reference model, mathematical definitions of quality measures, and practical guidance for their application. This division presents internal measures of software quality, external measures of software quality and quality in use measures. Quality measure elements forming foundations for the latter measures are presented, described and defined. ,

ISO/IEC 2503n - Quality Requirements Division. The standard that forms this division helps specifying quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a system/software product to be developed or as inputs for an evaluation process. The requirements definition process is mapped to technical processes defined in ISO/IEC 15288 – Information Technology - Life Cycle Management - System Life Cycle Processes,

ISO/IEC 2504n - Quality Evaluation Division. The standards that form this division provide requirements, recommendations and guidelines for system/software product evaluation, whether performed by independent evaluators, acquirers or developers. The support for documenting a measure as an Evaluation Module is also presented.

ISO/IEC 25050 to ISO/IEC 25099 is reserved to be used for SQuaRE extension International Standards and/or Standards.

This International Standard is part of the 2502n – Quality Measurement Division that currently consists of the following International Standards:

ISO/IEC 25020 – Measurement reference model and guide: provides a reference model and guide for measuring the quality characteristics defined in ISO/IEC 2501n Quality Model Division. The associated standards within the Quality Measurement Division provide suggested measures of quality throughout the product life-cycle.

ISO/IEC 25021 – Quality measure elements: offers quality measure elements that can be used to construct software quality measures.

ISO/IEC 25022 – Measurement of quality in use : provides measures for the characteristics in the quality in use model.

ISO/IEC 25023 – Measurement of system and software product quality : provides measures for the characteristics in the product quality model.

Figure 2 depicts the relationship between this standard and the measures tables contained in the other ISO/IEC 2502n division of standards. Developers, evaluators, quality managers, acquirers, suppliers, maintainers and other users of software may select measures from these standards for the measurement of quality characteristics of interest. In practice this may be with respect to defining requirements, evaluating system/software products, quality management and other purposes. They may also modify the measures or use measures which are not included in those standards.

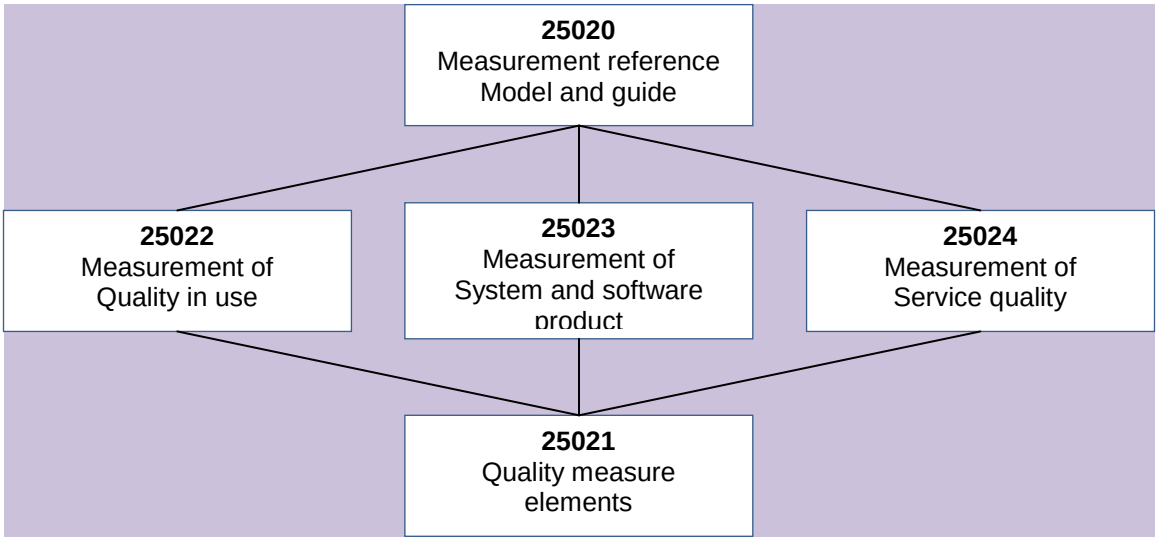

Figure 2 - Structure of the Quality Measurement division

Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality

1. Scope

This International Standard defines quality measures for quantitatively measuring system and software product quality in terms of characteristics and subcharacteristics defined in ISO/IEC 25010, and is intended to be used together with ISO/IEC 25010 and ISO/IEC 25021.

This International Standard contains:

an explanation of how to apply system and software product quality measures

a required set of quality measures for each subcharacteristic

an example of how to apply quality measures during the software product life cycle

This International Standard does not assign ranges of values of these quality measures to rated levels or to grades of compliance, because these values are defined for each system and software product or a part of the software product, by its nature, depending on such factors as category of the software, integrity level and users' needs. Some attributes may have a desirable range of values, which does not depend on specific user needs but depends on generic factors; for example, human cognitive factors.

This International Standard can be applied to any kind of system and software product for any application. Users of this International Standard can select or modify and apply quality measures from this International Standard or may define application-specific quality measures for their individual application domain. For example, the specific measurement of quality characteristics such as safety or security may be found in International Standard or International Standard provided by IEC 65 and ISO/IEC JTC1/SC27.

Intended users of this International Standard include:

Acquirer (an individual or organization that acquires or procures a system, software product or software service from a supplier);

Evaluator (an individual or organization that performs an evaluation. An evaluator may, for example, be a testing laboratory , the quality department of a software development organization, a government organization or an user);

Developer (an individual or organization that performs development activities, including requirements analysis, design, and testing through acceptance during the software life cycle process);

Maintainer (an individual or organization that performs maintenance activities);

Supplier (an individual or organization that enters into a contract with the acquirer for the supply of a system, software product or software service under the terms of the contract) when validating system and software product quality at qualification test;

User (an individual or organization that uses the system/software product to perform a specific function) when evaluating quality of system/software product at acceptance test;

Quality manager (an individual or organization that performs a systematic examination of the system/software product or software services) when evaluating system and software product quality as part of quality assurance and quality control.

This document provides a list of required quality measures for use in conjunction with ISO/IEC 2503n – Software product quality requirements division and ISO/IEC 2504n – Software product quality evaluation division standards or to more generally meet users information needs. The selected measures correspond to the quality characteristics of the product quality model in ISO/IEC 25010. The measures can be constructed from quality measure elements that are documented in ISO/IEC 25021.

2. Conformance

Any quality requirement, quality specification, or quality evaluation that conforms to this International Standard shall either:

a) use the quality measures defined in 7.1; or

b) modify the quality measures giving the rationale for any changes.

3. Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

1. ISO/IEC 25000 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE

2. ISO/IEC 25010 Systems and software engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE) – Quality model

3. ISO/IEC 25020 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Measurement reference model and guide

4. ISO/IEC 25021 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality measure elements

5. ISO/IEC 15939: 2002, Software Engineering – Software measurement process

4. Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 25000 and ISO 15939 apply.

5. Symbols (and abbreviated terms)

The following symbols and abbreviations are used in this International Standard..

1. SPQM-RM – Software Product Quality Measurement Reference Model
2. SQA - Software Quality Assurance (Group)
3. SLCP – Software Life Cycle Processes
4. QME – Quality Measure Element

6. Use of System and software product quality Measures

This International Standard provides a basic set of system and software quality measures (external and internal measures) to be used with the ISO/IEC 25010 Quality model. The user of this standard may modify the measures defined, and/or may also use measures not listed. When using a modified or a new measure not identified in this International Standard, the user should specify how the measures relate to the ISO/IEC 25010 quality model or any other substitute quality model that is being used.

The user of this International Standard should select the quality characteristics and subcharacteristics to be evaluated, from ISO/IEC 25010; identify the appropriate direct and indirect quality measures , identify the relevant quality measures and then interpret the measurement result in a objective manner. The user of this International Standard also may select product quality evaluation processes during the software life cycle from the ISO/IEC 2504n series of standards. These give methods for measurement, assessment and evaluation of system/software product quality. They are intended for use by developers, acquirers and independent evaluators, particularly those responsible for system/software product evaluation (see Figure 1).

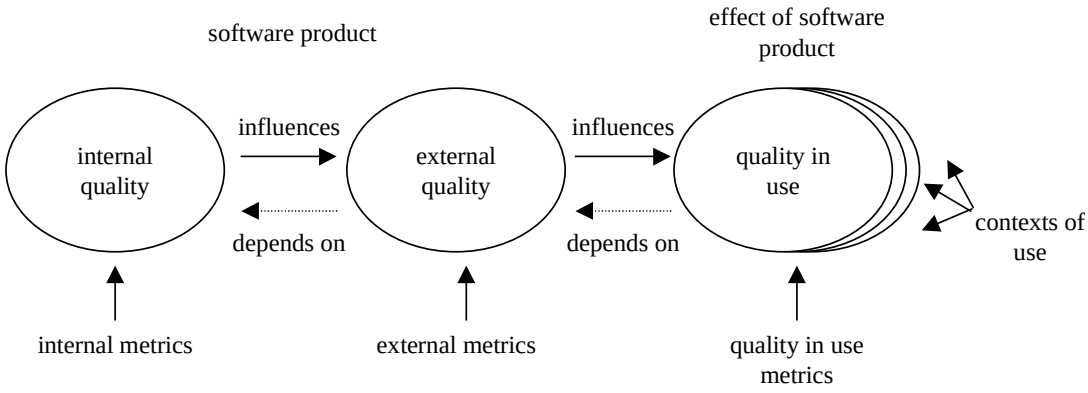

Figure 3 – Relationship between types of quality measures

The internal quality measures may be applied to a non-executable system/software product during its development stages (such as request for proposal, requirements definition, design specification or source code). Internal quality measures provide the users with the ability to measure the quality of the intermediate deliverables and thereby predict the quality of the final product. This allows the user to identify quality issues and initiate corrective action as early as possible in the development life cycle.

The external quality measures may be used to measure the quality of the system/software product by measuring the behaviour of the system of which it is a part. The external quality measures can only

be used during the testing stages of the life cycle process and during any operational stages. The measurement is performed when executing the system/software product in the system environment in which it is intended to operate.

The quality in use quality measures measure whether a product meets the needs of specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in a specified context of use. This can be only achieved in a realistic system environment.

User quality needs can be specified as quality requirements by quality in use quality measures, by external quality measures, and sometimes by internal quality measures. These requirements specified by quality measures should be used as criteria when a product is evaluated.

It is recommended to use internal quality measures having a relationship as strong as possible with the target external quality measures so that they can be used to predict the values of external quality measures. However, it is often difficult to design a rigorous theoretical model that provides a strong relationship between internal quality measures and external quality measures. Therefore, a hypothetical model that may contain ambiguity may be designed and the extent of the relationship may be modelled statistically during the use of quality measures.

Recommendations and requirements related to validity and reliability are given in ISO/IEC 25010, clause A.4. Additional detailed considerations when using quality measures are given in Annex A of this International Standard.

7. Format for Documenting Quality Measures

The quality measures listed in clause 8 are categorised by the characteristics and subcharacteristics in ISO/IEC 25010. The following information is given for each measure in the table:

- **Quality measure name:** A quality measure shall have a unique name and should be identified with a serial number, if necessary.
- **Description:** This is expressed as the question to be answered by the application of the measure.
- **QMEs**: Relevant quality measure elements which is used as input to this quality measures
- **Measure type:** Types used are; Size type (e.g. Function size, Source size) , Time type (e.g. Elapsed time, User time) , Count type (e.g. Number of changes, Number of failures).
- **Measurement focus** – Considerations connected with the object of measurement (e.g. software product itself, software product in a system, software product in a system used by a specified user in a specified scenario),

Note The quality measures table does not provide any definition for a concrete evaluation method. This is because it's not possible to determine uniformly as there are various measurement and evaluation methods for each evaluation purposes. It is preferable to refer to the definition of measurement methods of relevant QMEs defined in the ISO/IEC 25021 or other references when actually using the quality measure.

8. Required Quality Measures

The measures listed in this clause are the minimum required set which shall be applied to every system and software product for the quality evaluation. They are listed by software quality characteristics and subcharacteristics, in the order introduced in ISO/IEC 25010.

Quaoity measures, which may be applicable, are not limited to these listed here. Additional specific quality measures for particular purposes are provided in Annex A of this International Standards and other related documents, such as functional size measurement or precise time efficiency measurement.

NOTE: It is recommended to refer a specific measure or measurement form specific standards, Standards or guidelines. Functional size measurement is defined in ISO/IEC 14143. An example of precise time efficiency measurement can be referred from ISO/IEC 14756.

Quality measures should be validated before application in a specific environment (see Annex A).

NOTE: This list of measures is not finalised, and may be revised in future versions of this International Standard. Readers of this International Standard are invited to provide feedback.

8.1. Functional suitability measures

Functional suitability measures should be able to measure the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.

NOTE Functional suitability is only concerned with whether the functions meet stated and implied needs, not the functional specification

8.1.1. Functional completeness measures

Functional completeness measures should be able to measure the degree to which the set of functions covers all the specified tasks and user objectives.

Internal completeness measures indicate a set of attributes for assessing explicitly functions to prescribed tasks, and for determining their adequacy for performing the tasks.

An external completeness measure should be able to measure an attribute such as the occurrence of an unsatisfying function or the occurrence of an unsatisfying operation during testing and user operation of the system.

An unsatisfying function or operation may be:

a) Functions and operations that do not perform as specified in user manuals or requirement specification.

b) Functions and operations that do not provide a reasonable and acceptable outcome to achieve the intended specific objective of the user task.

Measure Name	Description	QMEs	Measure type	Measurement focus

Functional implementation coverage	How complete is the implementation according to requirement specifications?	A = The number of functions stated in requirement specification. B = the number of missing or incorrect functions	Count/ Count	External/ Internal

8.1.2. Functional correctness measures

Functional correctness measures should be able to measure the degree to which a product or system provides the correct results with the needed degree of precision.

Internal correctness measures indicate a set of attributes for assessing the capability of the system/software product to achieve correct or agreeable results.

An external correctness measure should be able to measure an attribute such as the frequency of users encountering the occurrence of inaccurate matters which includes:

a) Incorrect or imprecise result caused by inadequate data; for example, data with too few significant digits for accurate calculation;

b) Inconsistency between actual operation procedures and described ones in the operation manual;

c) Differences between the actual and reasonable expected results of tasks performed during operation

Measure Name	Description	QMEs	Measure type	Measurement focus
Correctness	How much the required specific accuracy standard is complied with?	A = The # of data item implemented with the specific standard of accuracy B = The total # of data item implemented	Count/ Time	External/ Internal
Computational Accuracy	How often do the inaccurate results occurs?.	A= Number of inaccurate computations encountered T= Operation time.	Count/ Time	External/ Internal

8.1.3. Functional appropriateness measures

Functional appropriateness measures should be able to measure the degree to which the functions facilitate the accomplishment of specified tasks and objectives

An example of functional appropriateness is: the user is only presented with the necessary steps to complete a task, excluding any unnecessary steps.

Measure Name	Description	QMEs	Measure type	Measurement focus
Functional appropriateness	How many functions with no problem are implemented for the appropriate functions for pursuing a specific task.	A = the number of functions for pursuing specific tasks B = the number of functions from which a problem is detected.	Count/ Count	External/ Internal

8.2. Performance efficiency measures

Performance efficiency measures should be able to measure the performance relative to the amount of resources used under stated conditions. Resources can include other software products, the software and hardware configuration of the system, and materials (e.g. print paper, storage media).

Internal efficiency measures are used for predicting the efficiency of behavior of the system/software product during testing or operating. To measure efficiency, the stated conditions should be defined, i.e., the hardware configuration and the software configuration of a reference environment (which has to be defined in the software specifications) should be defined. When citing measured time behavior values the reference environment should be referred.

An external efficiency measure should be able to measure such attributes as the time consumption and resource utilisation behaviour of computer system including software during testing or operations.

It is recommended that the maximal and distribution time are investigated for many cases of testing or operations, because the measure is affected strongly and fluctuates depending on the conditions of use, such as load of processing data, frequency of use, number of connecting sites and so on. Therefore, efficiency measures may include the ratio of measured actual value with error fluctuation to the designed value with allowed error fluctuation range, required by specification.

It is recommended to list and to investigate the role played by factors such as “CPU” and memory used by other software, network traffic, and scheduled background processes. Possible fluctuations and valid ranges for measured values should be established and compared to requirement specifications.

It is recommended that a task be identified and defined to be suitable for software application: for example, a transaction as a task for business application: a switching or data packet sending as a task for communication application; an event control as a task for control application; and an output of data produced by user callable function for common user application.

8.2.1. Time behaviour measures

Time behaviour measures should be able to measure the degree to which the response and processing times and throughput rates of a product or system when performing its functions meet requirements.

Internal time behaviour measures indicate a set of attributes for predicting the time behaviour of the computer system including the software product during testing or operating.

An external time behaviour measure should be able to measure such attributes as the time behaviour of computer system including software during testing or operations.

NOTE:
1. Response time: Time needed to get the result from pressing a transmission key. This means that response time includes processing time and transmission time. Response time is applicable only for an interactive system. There is no significant difference when it is a standalone system. However, in the case of Internet system or other real time system, sometimes transmission time is much longer.

2. Processing time: The elapsed time in a computer between receiving a message and sending the result. Sometimes it includes operating overhead time, other times it only means time used for an application program.

3. Turnaround time: Time needed to get the result from a request. In many cases one turnaround time includes many responses. For example, in a case of banking cash dispenser, turnaround time is a time from pressing initial key until you get money, meanwhile you must select type of transaction and wait for a message, input password and wait for the next message etc.

Measure Name	Descript.ion	QMEs	Measure type	Measurement focus
(Mean) Response time	Duration from giving a command to start a batch of tasks till receiving the first response	A = time of entering a command B = time of receiving the first response	Time - Time	External/ Internal
(Mean) Turnaround time	Duration from giving an instruction to start a job till completion of the job.	A = time of starting a job B = time of completing the job	Time - Time	External/ Internal
(Mean) Throughput	How many tasks can be processed per unit of time.	A = number of completed tasks T = observation time period	Count/ Time	External/ Internal

8.2.2. Resource utilisation measures

Resource utilisation measures should be able to measure the degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements.

Internal resource utilisation measures indicate a set of attributes for predicting the utilization of hardware resources by the computer system including the software product during testing or operating.

An external resource utilisation measure should be able to measure such attributes as the utilised resources behaviour of computer system including software during testing or operating.

Measure Name	Description	QMEs	Measure type	Measurement focus
(Mean) CPU utilization	How much CPU time is used to perform a given task	A =operation time B = the amount of CPU time actually used to perform a task	Time/ Time	External/ Internal
(Mean) Memory utilizatio	How much memory space is used to perform a given task	A =total amount of memory spaces B = the amount of memory spaces actually used to perform a task	Size/ Size	External/ Internal
(Mean) I/O devices utilization	How much I/O time is used to perform a given task	A =operation time B = the amount of I/O device(s) busy time to perform a task	Time/ Time	External/ Internal

8.2.3. Capacity measures

Capacity measures should be able to measure the degree to which the maximum limits of a product or system parameter meet requirements.

Measure Name	Description	QMEs	Measure type	Measurement focus
(Max.)No. of online requests	How many online requests can be processed per unit of time	A =operation time B = the maximum no. of online requests processed	Count/ Time	External/ Internal
(Max.) No. of simultaneous accesses	How many users can access the system simultaneously at a certain time.	A =operation time B = The maximum no. of simultaneous accesses	Count/ Time	External/ Internal

(Max.) Bandwith of transmission system	How much is the absolute limit value of transmission required to comply with the functions.	A = operation time B = the maximum amount of data transmission	Size/Time	External

8.3. Compatability measures

Compatability measures should be able to measure the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment

8.3.1 Co-existence measures

Co-existence measures should be able to measure the degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product

Internal co-existence measures indicate a set of attributes for predicting the impact the software product may have on other software products sharing the same operational hardware resources.

An external co-existence measure should be able to measure such attributes as the behaviour of the system or the user who is trying to use the software with other independent software in a common environment sharing common resources.

Measure Name	Description	QMEs	Measure type	Measurement focus
Available co-existence	How flexible is the product in sharing its environment with other products without adverse impacts on other products.	A= Number of entities with which product can co-exist as specified B= Number of entities in operation environment that require co-existence	Count/ Count	External/ Internal

8.3.2 interoperability measures

Interoperability measures should be able to measure the degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Internal Interoperability measures indicate a set of attributes for assessing the capability of the system/software product's interaction with designated systems.

An external interoperability measure should be able to measure an attribute such as the number of functions or occurrences of less communicativeness involving data and commands, which are transferred easily between the system/software product and other systems, other system/software products, or equipment which are connected.

Measure Name	Description	QMEs	Measure type	Measurement focus
Connectivity with external system	How correctly have the external interface protocols been implemented?	A = number of correctly implemented interfaces with other software or systems B = total number of external interfaces	Count/ Count	External/ Internal

Data exchangeability (Data format based)	How accurately is implementation of data exchange format determined between linking systems.	A = number of data formats regarded as being smoothly exchanged with other software or systems B = total number of data formats to be exchanged.	Count/ Count	External/ Internal

8.4. Usability measures

Usability measures should be able to measure 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.

Internal usability measures are used for predicting the extent to which the software in question can be understood, learned, operated and attractive. .

It should be possible for the measures taken to be used to establish acceptance criteria or to make comparisons between products. This means that the measures should be counting items of known value. Results should report the mean value and the standard error of the mean

Many external usability measures are tested by users attempting to use a function. The results will be influenced by the capabilities of the users and the host system characteristics. This does not invalidate the measurements, since the evaluated software is run under explicitly specified conditions by a sample of users who are representative of an identified user group. (For general-purpose products, representatives of a range of user groups may be used.) For reliable results a sample of at least eight users is necessary, although useful information can be obtained from smaller groups. Users should carry out the test without any hints or external assistance.

Measures for appropriateness recognisability, learnability and operability have two types of method of application: user test or test of the product in use.

NOTES:

1. **User test**

Users attempting to use a function test many external measures. These measures can vary widely among different individuals. A sample of users who are representative of an identified user group should carry out the test without any hints or external assistance. (For general-purpose products, representatives of a range of user groups may be used.) For reliable results a sample of at least eight users is necessary, although useful information can be obtained from smaller groups.

It should be possible for the measures to be used to establish acceptance criteria or to make comparisons between products. This means that the measures should be counting items of known value. Results should report the mean value and the standard error of the mean.

Many of these measures can be tested with early prototypes of the software. Which measures are to be applied will depend on the relative importance of different aspects of usability, and the extent of subsequent quality in use testing.

2. **Test of the product in use**

Rather than test specific functions, some external measures observe the use of a function during more general use of the product to achieve a typical task as part of a test of the quality in use (ISO/IEC 9126-4). This has the advantage that fewer tests are required. The disadvantage is that some functions may only rarely be used during normal use.

It should be possible for the measures to be used to establish acceptance criteria or to make comparisons between products. This means that the measures should be counting items of known value. Results should report the mean value and the standard error of the mean.

8.4.1 Appropriateness recognisability measures

Users should be able to select a system/software product which is suitable for their intended use. The measures for appropriateness recognisability should be able to measure the degree to which users can recognise whether a product or system is appropriate for their needs.

Appropriateness recognisability measures assess whether new users can understand:

whether the software is suitable

how it can be used for particular tasks.

Measure Name	Description	QMEs	Measure type	Measurement focus
Description completeness	What proportion of functions (or types of function) are described as understandable in the product description?	A= Number of functions (or types of functions) described as understandable in the product description B= Total number of functions (or types of functions)	Count/ Count	External/ Internal
Demonstration capability	What proportion of functions requiring demonstration have such capability?.	A= Number of functions implemented with demonstration capability B= Total number of functions requiring demonstration capability	Count/ Count	External/ Internal

8.4.2 Learnability measures

Learnability measures should be able to measure the degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use

An internal or external learnability measure should be able to assess how long users take to learn how to use particular functions, and the effectiveness of help systems and documentation.

Learnability is strongly related to appropriateness recognisability, and appropriateness recognisability measurements can be indicators of the learnability potential of the software.

Measure Name	Description	QMEs	Measure type	Measurement focus
Completeness of user documentation and/or help facilit	What proportion of functions are correctly described in the user documentation and/or help facility?.	A= Number of functions described correctly B= Total of number of functions implemented	Count/ Count	External/ Internal

8.4.3 Operability measures

Operability measures should be able to measure the degree to which a product or system has attributes that make it easy to operate and control

An internal or external operability measure should be able to assess whether users can operate and control the software. Operability measures can be categorised by the dialogue principles in ISO 9241-10:

- suitability of the software for the task
- self-descriptiveness of the software
- controllability of the software
- conformity of the software with user expectations
- error tolerance of the software
- suitability of the software for individualisation

The choice of functions to test will be influenced by the expected frequency of use of functions, the criticality of the functions, and any anticipated usability problems.

Measure Name	Description	QMEs	Measure type	Measurement focus
Operational consistency	How consistently can similar operations be carried out ?	A = number of operations that behave inconsistently B= total number of operations that behave similarly.	Count/ Count	External/ Internal
Message clarity	How easily can messages from a system be understood ?	A = number of messages that are understood easily B = total number of implemented messages	Count/ Count	External/ Internal
Customizing possibility	How many functions and operational procedures can a user customize for his convenience?	A=Number of implemented functions which can be customised during operation B=Number of functions requiring the customization capability	Count/ Count	External/ Internal

8.4.4 User error protection measures

User error protection measures should be able to measure the degree to which the system protects users against making errors.

Measure Name	Description	QMEs	Measure type	Measurement focus
Input validity checking	What proportion of input items provide checking for valid data.	A = Number of input items checked for valid data B = Number of input items which need checking for valid data	Count/ Count	External/ Internal
Avoidance of incorrect operation	How many functions have incorrect operation avoidance capability.	A = number of functions implemented to avoid critical or serious malfunctions being caused by incorrect operation B = total number of incorrect operation patterns	Count/ Count	External/ Internal

8.4.5 User interface aesthetics measures

User interface aesthetics measures should be able to measure the degree to which the user interface enables pleasing and satisfying interaction for the user.

An internal or external user interface aesthetics measure should be able to assess the appearance of the software, and will be influenced by factors such as screen design and colour. This is particularly important for consumer products.

Measure Name	Description	QMEs	Measure type	Measurement focus
Appearance customizability of user interface	What proportion of user interface elements can be customised in appearance..	A=Number of types of interface elements that can be customised. B=Total number of types of interface elements	Count/ Count	External/ Internal

8.4.6. Accessibility measures

Accessibility measures should be able to measure the degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

The range of capabilities includes disabilities associated with age.

Accessibility for people with disabilities can be specified or measured either as the extent to which a product or system can be used by users with specified disabilities to achieve specified goals with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use, or by the presence of product properties that support accessibility.

Measure Name	Description	QMEs	Measure type	Measurement focus
Physical accessibility	What proportion of functions can a user with a physical handicap access.	A = number of functions accessible by the disabled person. B = total number of functions implemented	Count/ Count	External/ Internal

8.5 Reliability measures

Reliability measures should be able to measure the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

Internal reliability measures are used for predicting if the system/software product in question will satisfy prescribed reliability needs, during the development of the system/software product.

An external reliability measure should be able to measure attributes related to the behaviours of the system of which the software is a part during execution testing to indicate the extent of reliability of the software in that system during operation. Systems and software are not distinguished from each other in most cases.

8.5.1 Maturity measures

Maturity measures should be able to measure the degree to which a system meets needs for reliability under normal operation.

Internal maturity measures indicate a set of attributes for assessing the maturity of the software.

An external maturity measure should be able to measure such attributes as the software freedom of failures caused by faults existing in the software itself.

Measure Name	Description	QMEs	Measure type	Measurement focus
Fault removal	What proportion of detected faults have been corrected?	A=Number of corrected faults in design /coding/testing phase B= Number of faults detected in review or testing	Count/ Count	External/ Internal
Test coverage	How much of required test cases have been executed during testing?	A= Number of actually performed test cases representing operation scenario during testing B= Number of test cases to be performed to cover requirements	Count/ Count	External/ Internal
Mean time between failures (MTBF)	How frequently does the system/software fail in operation?	A= operation time B= total number of actually detected failures	Time/ Count	External

8.5.2 Availability measures

Availability measures should be able to measure the degree to a system, product or component is operational and accessible when required for use.

Externally, availability can be assessed by the proportion of total time during which the system, product or component is in an up state. Availability is therefore a combination of maturity (which governs the frequency of failure), fault tolerance and recoverability (which governs the length of down time following each failure).

Measure Name	Description	QMEs	Measure type	Measurement focus
Service time ratio	What proportion of system service time is actually provided?	A = system service time actually provided B = system service time regulated in the operational schedule	Time/ Time	External
Mean down time	What is the average time the system stays unavailable when a failure occurs before gradual start up?	A= Total down time B= Number of observed breakdowns	Time/ Count	External

8.5.3 Fault tolerance measures

Fault tolerance measures should be able to measure the degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

An internal or external fault tolerance measure should be related to the system/software products capability of maintaining a specified performance level in cases of operation faults or infringement of its specified interface.

Measure Name	Description	QMEs	Measure type	Measurement focus

Failure avoidance	How many fault patterns were brought under control to avoid critical and serious failures?	A= Number of avoided critical and serious failure occurrences against test cases of fault pattern B= Number of executed test cases of fault pattern (almost causing failure) during testing	Count/ Count	External
Redundancy (components)	How many types of system components are installed redundantly to avoid system failure?	A = number of system component types redundantly installed. B = total number of system component types	Count/ Count	External/ Internal

8.5.4 Recoverability measures

Fault tolerance measures should be able to measure the degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

An internal or external recoverability measure should be able to measure such attributes as the software with system being able to re-establish its adequate level of performance and recover the data directly affected in the case of a failure.

Measure Name	Description	QMEs	Measure type	Measurement focus
Mean recovery time	What is the average time the system takes to complete recovery from failure?	T= Time to recovery downed software system at each opportunity N= Number of cases which observed software system entered into recovery	Time/ Count	External

8.6 Security measures

Security measures should be able to measure the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorisation

Internal security measures indicate a set of attributes for assessing the capability of the system/software product to avoid illegal access to the system and/or data.

An external security measure should be able to measure an attribute such as the number of functions with, or occurrences of security problems, which are:

a) Failing to prevent leak of secure output information or data;

b) Failing to prevent lost of important data;

c) Failing to defend against illegal access or illegal operation.

NOTE: 1. It is recommended that penetration tests be performed to simulate attack, because such a security attack does not normally occur in the usual testing. Real security measures may only be taken in "real life system environment", that is "quality in use".

2. Security protection requirements vary widely from the case of a stand-alone-system to the case of a system connected to the Internet. The determination of the required functionality and the assurance of their effectiveness have been addressed extensively in related standards. The user of this standard should determine security functions using appropriate methods and standards in those cases where the impact of any damage caused is

important or critical. In the other case the user may limit his scope to generally accepted “Information Technology (IT)” protection measures such as virus protection backup methods and access control.

8.6.1 Confidentiality measures

Confidentiality measures should be able to measure the degree to which a product or system ensures that data are accessible only to those authorized to have access.

Measure Name	Description	QMEs	Measure type	Measurement focus
Access controllability	How controllable is the accesses to the system?.	A= Number of detected different types of illegal operations B= Number of types of illegal operations in the specification	Count/ Count	External/ Internal
Data encryption	How correctly is the encryption/decryption of data items implemented as stated in the requirement spec.	A = number of data items correctly encrypted/decrypted B = number of data items to be required encryption/decryption.	Count/ Count	External /Internal

8.6.2 Integrity measures

Integrity measures should be able to measure the degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

Measure Name	Description	QMEs	Measure type	Measurement focus
Data corruption prevention	To what extent can the data corruption be prevented?	A = number of data corruption instances actually occurring B = number of accesses where data damage or breakage is expected to occur.	Count/ Count	External/ Internal

8.6.3 Non-repudiation measures

Non-repudiation measures should be able to measure the degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.

Measure Name	Description	QMEs	Measure type	Measurement focus
utilization of digital signature	What proportion of events requiring non-repudiation are processed using digital signature?	A = number of events processed using digital signature B = number of events requiring non-repudiation property.	Count/ Count	External/ Internal

8.6.4 Accountability measures

Accountability measures should be able to measure the degree to which the actions of an entity can be traced uniquely to the entity.

Measure Name	Description	QMEs	Measure type	Measurement focus
Access auditability	How complete is the audit trail concerning the user access to the system and data?	A = number of accesses to system and data recorded in the system log B = number of accesses actually occurred.	Count/ Count	External/ Internal

8.6.5 Authenticity measures

Authenticity measures should be able to measure the degree to which the identity of a subject or resource can be proved to be the one claimed.

Measure Name	Description	QMEs	Measure type	Measurement focus
Authentication methods	How well does the system authenticate the identity of a subject or resource?	A = number of provided authentication methods (e.g., ID/password or IC card)	Count	External/ Internal

8.7 Maintainability measures

Maintainability measures should be able to measure the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers

Internal maintainability measures are used for predicting the level of effort required for modifying the system/software product.

An external maintainability measure should be able to measure such attributes as the behaviour of the maintainer, user, or system including the software, when the software is maintained or modified during testing or maintenance.

8.7.1 Modularity measures

Modularity measures should be able to measure the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Measure Name	Description	QMEs	Measure type	Measurement focus
Condensability	How strong is the relation between the components in a system or computer program?	A = number of components which are not impacted from changes of other components. B = total number of discrete components.	Count/ Count	Internal

8.7.2 Reusability measures

Reusability measures should be able to measure the degree to which an asset can be used in more than one system, or in building other assets.

Measure Name	Description	QMEs	Measure type	Measurement focus

Execution of reusability	How many assets can be reused ?	A = number of assets reused B = total number of assets in the reusable library	Count/ Count	Internal

8.7.3 Analysability measures

Analysability measures should be able to measure the degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified

Internal analyzability measures indicate a set of attributes for predicting the maintainer's or user's spent effort or spent resources in trying to diagnose for deficiencies or causes of failure, or for identification of parts to be modified in the system/software product.

An external analysability measure should be able to measure such attributes as the maintainer's or user's effort or spent of resources when trying to diagnose deficiencies or causes of failures, or for identifying parts to be modified.

Measure Name	Description	QMEs	Measure type	Measurement focus
Audit trail capability	Can users easily identify specific operation which caused failure?	A= Number of data actually recorded during operation B= Number of data planned to be recorded enough to monitor status of system/software during operation	Count/ Count	External/ Internal
Diagnosis function sufficiency	To what extent are diagnostic functions prepared, or to what extent do they work for causal analysis?	A = number of implemented diagnostic functions B = number of diagnostic functions required in the spec.	Count/ Count	External/ Internal

8.7.4 Modifiability measures

Modifiability measures should be able to measure the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality

Internal Modifiability measures indicate a set of attributes for predicting the maintainer's or user's spent effort when trying to implement a specified modification in the system/software product.

An external changeability measure should be able to measure such attributes as the maintainer's or user's effort by measuring the behaviour of the maintainer, user or system including the software when trying to implement a specified modification.

Measure Name	Description	QMEs	Measure type	Measurement focus
Localization degree of correction impact	To what extent can the trouble caused as an aftereffect of maintenance be prevented.	A= Number of failures emerged after failure is resolved by change during specified period B= Number of resolved failures	Count/ Count	External/ Internal
Modification complexity	Can the maintainer easily modify the software to resolve problem?.	A= Work time spent to modify B= Number of modifications	Time/ Count	External

Modification success rate	To what extent can software system be operated without failures after maintenance.	A = number of troubles within a certain period before maintenance B = number of troubles in the same period after maintenance.	Count/ Count	External

8.7.5 Testability measures

Testability measures should be able to measure the degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Internal testability measures indicate a set of attributes for predicting the amount of designed and implemented autonomous test aid functions present in the system/software product

An external testability measure should be able to measure such attributes as the maintainer's or user's effort by measuring the behaviour of the maintainer, user or system including software when trying to test the modified or non-modified software.

Measure Name	Description	QMEs	Measure type	Measurement focus
Functional completeness of embedded test functions	How completely are test functions and facilities implemented.	A = number of test functions implemented as specification B = number of required test functions.	Count/ Count	Internal
Autonomous testability	How independently can software be tested.	A = number of tests that can be simulated by stub out of the tests depending on other systems B = total number of test dependencies to other systems.	Count/ Count	Internal
Test restartability	How easily can the operation test be carried out from the restart point after maintenance.	A = Number of cases in which maintainer can pause and restart executing test run at desired points to check step by step B= Number of cases of pause of executing test run	Count/ Count	External

8.8 Portability measures

Portability measures should be able to measure the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

Internal Portability measures are used for predicting the effect the system/software product may have on the behaviour of the implementer or system during the porting activity

An external portability measure should be able to measure such attributes as the behaviour of the operator or system during the porting activity.

8.8.1 Adaptability measures

Adaptability measures should be able to measure the degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Internal adaptability measures indicate a set of attributes for predicting the impact the system/software product may have on the effort of the user who is trying to adapt the system/software product to different specified environments

An external adaptability measure should be able to measure such attributes as the behaviour of the system or the user who is trying to adapt software to different specified environments. When a user has to apply an adaptation procedure other than previously provided by software for a specific adaptation need, user's effort required for adapting should be measured.

Measure Name	Description	QMEs	Measure type	Measurement focus
Hardware environmental adaptability	Is software system capable enough to adapt itself to hardware environment?.	A= Number of operational functions of which tasks were not completed or not enough resulted to meet adequate levels during combined operating testing with environmental hardware B= Total number of functions which were tested.	Count/ Count	External/ Internal
System software environmental adaptability	Is software system capable enough to adapt itself to system software environment?.	A= Number of operational functions of which tasks were not completed or were not enough resulted to meet adequate level during combined operating testing with system software or concurrent application software B= Total number of functions which were tested.	Count/ Count	External/ Internal
Organisational environment adaptability	Is software system capable enough to adapt itself to the operational environment?	A= Number of operated functions in which the tasks were not completed or not enough resulted to meet adequate levels during operational testing with user's business environment B= Total number of functions which were tested	Count/ Count	External/ Internal

8.8.2 Installability measures

Installability measures should be able to measure the degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

Internal Installability measures indicate a set of attributes for predicting the impact the system/software product may have on the effort of the user who is trying to install the software in a user specified environment.

An external installability measure should be able to measure such attributes as the behaviour of the system or the user who is trying to install the software in a user specific environment.

Measure Name	Description	QMEs	Measure type	Measurement focus
Installation time efficiency	How much time and trouble is required to make an install?	A = total elapsed time to install a system successfully. B = number of retries to install a system	Count/ Count	External

Ease of installation	Can user or maintainer easily install software to operation environment?	A = Number of cases which a user succeeded to in changing the install operation for his/her convenience B = Total number of cases which a user attempted to change the install operation for his/her convenience	Count/ Count	External

8.8.3 Replaceability measures

Replaceability measures should be able to measure the degree to which a product can be replaced by another specified software product for the same purpose in the same environment

Internal replaceability measures indicate a set of attributes for predicting the impact the software product may have on the effort of the user who is trying to use the software in place of other specified software in a specified environment and context of use.

An external replaceability measure should be able to measure such attributes as the behaviour of the system or the user who is trying to use the software in place of other specified software in the environment of that software.

Measure Name	Description	QMEs	Measure type	Measurement focus
User support function consistency	How consistent is the new component with the existing user interface.	A = number of new functions that are considered not to be consistent with user's expectations B = number of new functions	Count/ Count	External/ Internal
Functional inclusiveness	Can the similar functions easily be used after replacing this to previous one?	A = number of functions which produce similar results as previously and where changes have not been required B = number of tested functions which are similar to functions provided by another software to be replaced	Count/ Count	External
Continuous usage of data	Can the same data easily be used after replacing this software to previous one?	A = number of data which are used in other software to be replaced and are confirmed that they are able to be continuously used B = number of data which are used in other software to be replaced and planned to be continuously reusable.	Count/ Count	External

Annex A（Informative）Recommended Quality Measures

***** TBD**

A.1. Functional suitability

degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions

A.1.1 Functional completeness

Degree to which the set of functions covers all the specified tasks and user objectives

Measure Name	Description	QMEs	Measure type	Measurement focus

A.1.2 Functional correctness

Degree to which a product or system provide the correct results with the necessary degree of precision

Measure Name	Description	QMEs	Measure type	Measurement focus

A.1.3 Functional appropriateness

degree to which the functions facilitate the accomplishment of specified tasks and objectives

Measure Name	Description	QMEs	Measure type	Measurement focus

A.2. Performance efficiency

performance relative to the amount of resources used under stated conditions

A.2.1. time behaviour

degree to which the response and processing times and throughput rates of a product or system when performing its functions meet requirements

Measure Name	Description	QMEs	Measure type	Measurement focus

A.2.2. resource utilisation

degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements

Measure Name	Description	QMEs	Measure type	Measurement focus

A.2.3. capacity

degree to which the maximum limits of a product or system parameter meet requirements

Measure Name	Description	QMEs	Measure type	Measurement focus

A.3. compatibility

degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment

A.3.1. co-existence

degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product

Measure Name	Description	QMEs	Measure type	Measurement focus

A.3.2. interoperability

degree to which two or more systems, products or components can exchange information and use the information that has been exchanged

Measure Name	Description	QMEs	Measure type	Measurement focus

A.4. usability

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.

A.4.1. appropriateness recognisability

degree to which users can recognise whether a product or system is appropriate for their needs

Measure Name	Description	QMEs	Measure type	Measurement focus

A.4.2. learnability

degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use

Measure Name	Description	QMEs	Measure type	Measurement focus

A.4.3. operability

degree to which a product or system has attributes that make it easy to operate and control

Measure Name	Description	QMEs	Measure type	Measurement focus

A.4.4. user error protection

degree to which the system protects users against making errors

Measure Name	Description	QMEs	Measure type	Measurement focus

A.4.5. User interface aesthetics

degree to which the user interface enables pleasing and satisfying interaction for the user

Measure Name	Description	QMEs	Measure type	Measurement focus

A.4.6. Accessibility

degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use

Measure Name	Description	QMEs	Measure type	Measurement focus

A.5. Reliability

degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

A.5.1. Maturity

degree to which a system meets needs for reliability under normal operation

Measure Name	Description	QMEs	Measure type	Measurement focus

A.5.2. Availability

degree to which a system, product or component is operational and accessible when required for use

Measure Name	Description	QMEs	Measure type	Measurement focus

A.5.3. Fault tolerance

degree to which a system, product or component operates as intended despite the presence of hardware or software faults

Measure Name	Description	QMEs	Measure type	Measurement focus

A.5.4. Recoverability

degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system

Measure Name	Description	QMEs	Measure type	Measurement focus

A.6. Security

degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorisation

A.6.1. Confidentiality

degree to which a product or system ensures that data are accessible only to those authorized to have access

Measure Name	Description	QMEs	Measure type	Measurement focus

A.6.2. Integrity

degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data

Measure Name	Description	QMEs	Measure type	Measurement focus

A.6.3. Non-repudiation

degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later

Measure Name	Description	QMEs	Measure type	Measurement focus

A.6.4. Accountability

degree to which the actions of an entity can be traced uniquely to the entity

Measure Name	Description	QMEs	Measure type	Measurement focus

A.6.5. Authenticity

degree to which the identity of a subject or resource can be proved to be the one claimed

Measure Name	Description	QMEs	Measure type	Measurement focus

A.7. Maintainability

degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers

A.7.1. Modularity

degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components

Measure Name	Description	QMEs	Measure type	Measurement focus

A.7.2. Reusability

degree to which an asset can be used in more than one system, or in building other assets

Measure Name	Description	QMEs	Measure type	Measurement focus

A.7.3. Analysability

degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified

Measure Name	Description	QMEs	Measure type	Measurement focus

A.7.4. Modifiability

degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality

Measure Name	Description	QMEs	Measure type	Measurement focus

A.7.5. Testability

degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met

Measure Name	Description	QMEs	Measure type	Measurement focus

A.8. Portability

degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another

A.8.1. Adaptability

degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments

Measure Name	Description	QMEs	Measure type	Measurement focus

A.8.2. Installability

degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment

Measure Name	Description	QMEs	Measure type	Measurement focus

A.8.3. Replaceability

degree to which a product can be replaced by another specified software product for the same purpose in the same environment

Measure Name	Description	QMEs	Measure type	Measurement focus

Annex B（Informative） Considerations When Using Measures

B.1 Interpretation of measures

B.1.1 Potential differences between test and operational contexts of use

When planning the use of measures or interpreting measures it is important to have a clear understanding of the intended context of use of the software, and any potential differences between the test and operational contexts of use. For example, the “time required to learn operation" measure is often different between skilled operators and unskilled operators in similar software systems. Examples of potential differences are given below.

a) Differences between testing environment and the operational environment

Are there any significant differences between the testing environment and the operational execution in user environment?

The following are examples:

- testing with higher / comparable / lower performance of CPU of operational computer;
- testing with higher / comparable / lower performance of operational network and communication;
- testing with higher / comparable / lower performance of operational operating system;
- testing with higher / comparable / lower performance of operational user interface.

b) Differences between testing execution and actual operational execution

Are there any significant differences between the testing execution and operational execution

in user environment?

The following are examples:

- coverage of functionality in test environment;
- test case sampling ratio;
- automated testing of real time transactions;
- stress loads;
- 24 hour 7 days a week (non stop) operation
- appropriateness of data for testing of exceptions and errors;
- periodical processing;
- resource utilisation.

- levels of interruption
- production preassures
- distractions

c) User profile under observation

Are there any significant differences between test user profiles and operational user profiles?

The following are examples:

- mix of type of users;
- user skill levels;
- specialist users or average users;
- limited user group or public users.
-

B.1.2. Issues affecting validity of results

The following issues may affect the validity of the data that is collected.

procedures for collecting evaluation results:

- automatically with tools or facilities/ manually collected / questionnaires or interviews;

source of evaluation results

- developers' self reports / reviewers' report / evaluator's report;

results data validation

- developers' self check / inspection by independent evaluators.
-

B.1.3. Balance of measurement resources

Is the balance of measures used at each stage appropriate for the evaluation purpose?

It is important to balance the effort used to apply an appropriate range of measures for internal, external and quality in use measures.

B.1.4. Correctness of specification

Are there significant differences between the software specification and the real operational needs?

Measurements taken during software product evaluation at different stages are compared against product specifications. Therefore, it is very important to ensure by verification and validation that the product specifications used for evaluation reflect the actual and real needs in operation.

B.2. Validation of Measures

B.2.1. Desirable Properties for Measures

To obtain valid results from a quality evaluation, the measures should have the properties stated below. If a measure does not have these properties, the measure description should explain the associated constraint on its validity and, as far as possible, how that situation can be handled.

Reliability (of measure): Reliability is associated with random error. A measure is free of random error if random variations do not affect the results of the measure.

Repeatability (of measure): repeated use of the measure for the same product using the same evaluation specification (including the same environment), type of users, and environment by the same evaluators, should produce the same results within appropriate tolerances. The appropriate tolerances should include such things as fatigue, and learning effect

Reproducibility (of measure): use of the measure for the same product using the same evaluation specification (including the same environment), type of users, and environment by different evaluators, should produce the same results within appropriate tolerances.

NOTE: It is recommended to use statistical analysis to measure the variability of the results

Availability (of measure): The measure should clearly indicate the conditions (e.g. presence of specific attributes) which constrain its usage.

Indicativeness (of measure): Capability of the measure to identify parts or items of the software which should be improved, given the measured results compared to the expected ones.

NOTE: The selected or proposed measure should provide documented evidence of the availability of the measure for use, unlike those requiring project inspection only.

Correctness (of measure): The measure should have the following properties:

1) Objectivity (of measure): the measure results and its data input should be factual: i.e., not influenced by the feelings or the opinions of the evaluator, test users, etc. (except for satisfaction or attractiveness measures where user feelings and opinions are being measured).

2) Impartiality (of measure): the measurement should not be biased towards any particular result.

3) Sufficient precision (of measure): Precision is determined by the design of the measure, and particularly by the choice of the material definition used as the basis for the measure. The measure user will describe the precision and the sensitivity of the measure.

Meaningfulness (of measure): the measurement should produce meaningful results about the software behaviour or quality characteristics.

The measure should also be cost effective: that is, more costly measures should provide higher value results.

B.2.2. Demonstrating the Validity of Measures

The users of measures should identify the methods for demonstrating the validity of measures, as shown below:

(a) Correlation

The variation in the quality characteristics values (the measures of principal measures in operational use) explained by the variation in the measure values, is given by the square of the linear coefficient.

An evaluator can predict quality characteristics without measuring them directly by using correlated measures.

(b) Tracking

If a measure M is directly related to a quality characteristics value Q (the measures of principal measures in operational use), for a given product or process, then a change value Q(T1) to Q(T2), would be accompanied by a change measure value from M(T1) to M(T2), in the same direction (for example, if Q increases, M increases).

An evaluator can detect movement of quality characteristics along a time period without measuring directly by using those measures which have tracking ability.

(c) Consistency

If quality characteristics values (the measures of principal measures in operational use) Q1, Q2,..., Qn, corresponding to products or processes 1, 2,..., n, have the relationship Q1 > Q2 > ...> Qn, then the correspond measure values would have the relationship M1 > M2 > ...> Mn.

An evaluator can notice exceptional and error prone components of software by using those measures which have consistency ability.

(d) Predictability

If a measure is used at time T1 to predict a quality characteristic value Q (the measures of principal measures in operational use) at T2, prediction error, which is {(predicted Q(T2) - actual Q(T2)) / actual Q(T2)}, would be within allowed prediction error range.

An evaluator can predict the movement of quality characteristics in the future by using these measures, which measure predictability.

(e) Discriminative

A measure would be able to discriminate between high and low quality software.

An evaluator can categorise software components and rate quality characteristics values by using those measures which have discriminative ability.

B.3. Use of Measures for Estimation (Judgement) and Prediction (Forecast)

Estimation and prediction of the quality characteristics of the software product at the earlier stages are two of the most rewarding uses of measures.

B.3.1. Quality characteristics prediction by current data

(a) Prediction by regression analysis

When predicting the future value (measure) of the same characteristic (attribute) by using the current value (data) of it (the attribute), a regression analysis is useful based on a set of data that is observed in a sufficient period of time.

For example, the value of MTBF (Mean Time Between Failures) that is obtained during the testing stage (activities) can be used to estimate the MTBF in operation stage.

(b) Prediction by correlation analysis

When predicting the future value (measure) of a characteristic (attribute) by using the current measured values of a different attribute, a correlation analysis is useful using a validated function which shows the correlation.

For example, the complexity of modules during coding stage may be used to predict time or effort required for program modification and test during maintenance process.

B.3.2. Current quality characteristics estimation on current facts

(a) Estimation by correlation analysis

When estimating the current values of an attribute which are directly unmeasurable, or if there is any other measure that has strong correlation with the target measure, a correlation analysis is useful.

For example, because the number of remaining faults in a software product is not measurable, it may be estimated by using the number and trend of detected faults.

Those measures which are used for predicting the attributes that are not directly measurable should be estimated as explained below:

Using models for predicting the attribute;

Using formula for predicting the attribute;

Using basis of experience for predicting the attribute;

Using justification for predicting the attribute.

Those measures which are used for predicting the attributes that are not directly measurable may be validated as explained below:

Identify measures of attributes which are to be predicted;

Identify the measures which will be used for prediction;

Perform a statistical analysis based validation;

Document the results;

Repeat the above periodically;

B.4. Detecting deviations and anomalies in quality problem prone components

The following quality control tools may be used to analyse deviations and anomalies in software product components:

(a) process charts (functional modules of software)

(b) Pareto analysis and diagrams

(c) histograms and scatter diagrams

(d) run diagrams, correlation diagrams and stratification

(e) Ishikawa (Fishbone) diagrams

(f) statistical process control (functional modules of software)

(g) check sheets

The above tools can be used to identify quality issues from data obtained by applying the measures.

B.5. Displaying Measurement Results

(a) Displaying quality characteristics evaluation results

The following graphical presentations are useful to display quality evaluation results for each of the quality characteristic and subcharacteristic.

Radar chart; Bar chart numbered histogram, multi-variates chart, Importance Performance Matrix, etc.

(b) Displaying measures

There are useful graphical presentations such as Pareto chart, trend charts, histograms, correlation charts, etc.

Annex C
(Informative)
Detailed explanation of measurement types

C.1 Measurement Types

C.1.0 General

In order to design a procedure for collecting data, interpreting fair meanings, and normalizing measures for comparison, a user of measures should identify and take account of the measure type of measurement employed by a metric.

C.1.1 Size Measure Type

C.1.1.0 General

A measure of this type represents a particular size of software according to what it claims to measure within its definition.

NOTE: software may have many representations of size (like any entity can be measured in more than one dimension - mass, volume, surface area etc.).

Normalizing other measures with a size measure can give comparable values in terms of units of size. The size measures described below can be used for software quality measurement.

C.1.1.1 Functional Size Type

Functional size is an example of one type of size (one dimension) that software may have. Any one instance of software may have more than one functional size depending on, for example:

- the purpose for measuring the software size (It influences the scope of the software included in the measurement);
- the particular functional sizing method used (It will change the units and scale).

The definition of the concepts and process for applying a functional size measurement method (FSM Method) is provided by the standard ISO/IEC 14143--1.

In order to use functional size for normalization it is necessary to ensure that the same functional sizing method is used and that the different software being compared have been measured for the same purpose and consequently have a comparable scope.

Although the following often claim that they represent functional sizes, it is not guaranteed they are equivalent to the functional size obtained from applying a FSM Method compliant with ISO/IEC 14143--1. However, they are widely used in software development:

- **number of spread sheets;**
- **number of screens;**
- **number of files or data sets which are processed;**
- **number of itemized functional requirements described in user requirements specifications.**

C.1.1.2 Program size type

In this clause, the term ‘programming’ represents the expressions that when executed result in actions, and the term ‘language’ represents the type of expression used.

1. Source program size

The programming language should be explained and it should be provided how the non executable statements, such as comment lines, are treated. The following measures are commonly used:

Non-comment source statements (NCSS)

Non-comment source statements (NCSS) include executable statements and data declaration statements with logical source statements.

NOTE

1. New program size

A developer may use newly developed program size to represent development and maintenance work product size.

2. Changed program size

A developer may use changed program size to represent size of software containing modified components.

3. Computed program size

Example of computed program size formula is new lines of code + 0.2 x lines of code in modified components (NASA Goddard).

It may be necessary to distinguish a type of statements of source code into more detail as follows:

Statement Type

- Logical Source Statement (LSS). The LSS measures the number of software instructions. The statements are irrespective of their relationship to lines and independent of the physical format in which they appear.
- Physical Source Statement (PSS). The PSS measures the number of software source lines of code.

Statement attribute

- Executable statements;
- Data declaration statements;
- Compiler directive statements;
- Comment source statements.

Origin

- Modified source statements;
- Added source statements;

- Removed source statements;
- Newly Developed source statements: (= added source statements + modified source statements);
- Reused source statements: (= original - modified - removed source statements);

2. Program word count size

The measurement may be computed in the following manner using the Halstead's measure:

Program vocabulary = n1+n2; Observed program length = N1+N2, where:

- n1: Is the number of distinct operator words which are prepared and reserved by the program language in a program source code;
- n2: Is the number of distinct operand words which are defined by the programmer in a program source code;
- N1: Is the number of occurrences of distinct operators in a program source code;
- N2: Is the number of occurrences of distinct operands in a program source code.

3. Number of modules

The measurement is counting the number of independently executable objects such as modules of a program.

C.1.1.3 Utilized resource measure type

This type identifies resources utilized by the operation of the software being evaluated. Examples are:

(a) Amount of memory, for example, amount of disk or memory occupied temporally or permanently during the software execution;

(b) I/O load, for example, amount of traffic of communication data (meaningful for backup tools on a network);

(c) CPU load, for example, percentage of occupied CPU instruction sets per second (This measure type is meaningful for measuring CPU utilization and efficiency of process distribution in multi-thread software running on concurrent/parallel systems);

(d) Files and data records, for example, length in bytes of files or records;

(e) Documents, for example, number of document pages.

It may be important to take note of peak (maximal), minimum and average values, as well as periods of time and number of observations done.

C.1.1.4 Specified operating procedure step type

This type identifies static steps of procedures which are specified in a human-interface design specification or a user manual.

The measured value may differ depending on what kinds of description are used for measurement, such as a diagram or a text representing user operating procedures.

C.1.2 Time measure type

C.1.2.0 General

The user of measures of time measure type should record time periods, how many sites were examined and how many users took part in the measurements.

There are many ways in which time can be measured as a unit, as the following examples show.

(a) Real time unit

This is a physical time: i.e. second, minute, or hour. This unit is usually used for describing task processing time of real time software.

(b) Computer machinery time unit

This is computer processor's clock time: i.e. second, minute, or hour of CPU time.

(c) Official scheduled time unit

This includes working hours, calendar days, months or years.

(d) Component time unit

When there are multiple sites, component time identifies individual site and it is an accumulation of individual time of each site. This unit is usually used for describing component reliability, for example, component failure rate.

(e) System time unit

When there are multiple sites, system time does not identify individual sites but identifies all the sites running, as a whole in one system. This unit is usually used for describing system reliability, for example, system failure rate.

C.1.2.1 System operation time type

System operation time type provides a basis for measuring software availability. This is mainly used for reliability evaluation. It should be identified whether the software is under discontinuous operation or continuous operation. If the software operates discontinuously, it should be assured that the time measurement is done on the periods the software is active (this is obviously extended to continuous operation).

(a) Elapsed time

When the use of software is constant, for example in systems operating for the same length of time each week.

(b) Machine powered-on time

For real time, embedded or operating system software that is in full use the whole time the system is operational.

(c) Normalized machine time

As in "machine powered-on time", but pooling data from several machines of different “powered-on-time” and applying a correction factor.

C.1.2.2 Execution time type

Execution time type is the time which is needed to execute software to complete a specified task. The distribution of several attempts should be analyzed and mean, deviation or maximal values should be computed. The execution under the specific conditions, particularly overloaded condition, should be examined. Execution time type is mainly used for efficiency evaluation.

C.1.2.3 User time type

User time type is measured upon time periods spent by individual users on completing tasks by using operations of the software. Some examples are:

(a) Session time

Measured between start and end of a session. Useful, as example, for drawing behaviour of users of a home banking system. For an interactive program where idling time is of no interest or where interactive usability problems only are to be studied.

(b) Task time

Time spent by an individual user to accomplish a task by using operations of the software on each attempt. The start and end points of the measurement should be well defined.

(c) User time

Time spent by an individual user using the software from time started at a point in time. (Approximately, it is how many hours or days user uses the software from beginning).

C.1.2.4 Effort type

Effort type is the productive time associated with a specific project task.

(a) Individual effort

This is the productive time which is needed for the individual person who is a developer, maintainer, or operator to work to complete a specified task. Individual effort assumes only a certain number of productive hours per day.

(b) Task effort

Task effort is an accumulated value of all the individual project personnel: developer, maintainer, operator, user or others who worked to complete a specified task.

C.1.2.5 Time interval of events type

This measure type is the time interval between one event and the next one during an observation period. The frequency of an observation time period may be used in place of this measure. This is typically used for describing the time between failures occurring successively.

C.1.3 Count measure type

If attributes of documents of the software product are counted, they are static count types. If events or human actions are counted, they are kinetic count types.

C.1.3.1 Number of detected fault type

The measurement counts the detected faults during reviewing, testing, correcting, operating or maintaining. Severity levels may be used to categorize them to take into account the impact of the fault.

C.1.3.2 Program structural complexity number type

The measurement counts the program structural complexity. Examples are the number of distinct paths or the McCabe's cyclomatic number.

C.1.3.3 Number of detected inconsistency type

This measure counts the detected inconsistent items which are prepared for the investigation.

(a) Number of failed conforming items

Examples:

Conformance to specified items of requirements specifications;

Conformance to rule, regulation, or standard;

Conformance to protocols, data formats, media formats, character codes

(b) Number of failed instances of user expectation

The measurement is to count satisfied/unsatisfied list items, which describe gaps between user's reasonable expectation and software product performance.

The measurement uses questionnaires to be answered by testers, customers, operators, or end users on what deficiencies were discovered.

The following are examples:

- Function available or not;
- Function effectively operable or not;
- Function operable to user's specific intended use or not;
- Function is expected, needed or not needed.

C.1.3.4 Number of changes type

This type identifies software configuration items which are detected to have been changed. An example is the number of changed lines of source code.

C.1.3.5 Number of detected failures type

The measurement counts the detected number of failures during product development, testing, operating or maintenance. Severity levels may be used to categorize them to take into account the impact of the failure.

C.1.3.6 Number of attempts (trial) type

This measure counts the number of attempts at correcting the defect or fault. For example, during reviews, testing, and maintenance.

C.1.3.7 Stroke of human operating procedure type

This measure counts the number of strokes of user human action as kinetic steps of a procedure when a user is interactively operating the software. This measure quantifies the ergonomic usability as well as the effort to use. Therefore, this is used in usability measurement. Examples are number of strokes to perform a task, number of eye movements, etc.

C.1.3.8 Score type

This type identifies the score or the result of an arithmetic calculation. Score may include counting or calculation of weights checked on/off on checklists. Examples: Score of checklist; score of questionnaire; Delphi method; etc.

Bibliography

ISO/IEC 25000 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE.

ISO/IEC 25010 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality model.

ISO/IEC 25020 Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Measurement reference model and guide.

ISO/IEC 25021 Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Quality measure elements. (to be published)

ISO/IEC 25030 Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Quality requirements

ISO/IEC 25042 Systems and software engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE) – Evaluation modules (to be published)

ISO/IEC TR 9126-2:2003, Software engineering - Product quality - Part 2: External measures

ISO/IEC TR 9126-3:2003, Software engineering - Product quality - Part 3: Internal measures

ISO/IEC TR 9126-4:2004, Software engineering - Product quality - Part 4: Quality in Use

ISO/IEC 15939:2007, System and software engineering – Measurement process

METI, Investigative Report on Measure for System/Software Product Quality Requirement Definition and Evaluation, March, 2011, Japan,