INTERNATIONAL STANDARD

ISO/IEC 25023

First edition
2016-06-15

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

Ingénierie des systèmes et du logiciel — Exigences de qualité et évaluation des systèmes et du logiciel (SQuaRE) — Mesurage de la qualité du produit logiciel et du système

Reference number
ISO/IEC 25023:2016(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical 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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 7, *Software and systems engineering*.

This first edition of ISO/IEC 25023, which is a part of the SQuaRE series of standards, cancels and replaces ISO/IEC TR 9126-2:2003 and ISO/IEC TR 9126-3:2003, with the following changes:

— the quality measures contained in ISO/IEC/TR 9126-2 and ISO/IEC/TR 9126-3 are reviewed and adopted or rejected according to the practical usefulness;

— in addition, the other quality measures are given for the revised system/software product quality model in ISO/IEC 25010;

— the internal and external measures are aggregated and represented with a simplified format in one table.

The SQuaRE series of International Standards consist of the following divisions, under the general title *Systems and software Quality Requirements and Evaluation (SQuaRE)*:

— *ISO/IEC 2500n — Quality Management Division*

— *ISO/IEC 2501n — Quality Model Division*

— *ISO/IEC 2502*n *— Quality Measurement Division*

— *ISO/IEC 2503n — Quality Requirements Division*

— *ISO/IEC 2504n — Quality Evaluation Division*

— *ISO/IEC 25050 to ISO/IEC 25099 — SQuaRE Extension Division*

Annexes A, B and C are for information only.

Introduction

This International Standard is a part of the SQuaRE series of International Standards. It provides a set of quality measures for the characteristics of system/software products that can be used for specifying requirements, measuring and evaluating the system/software product quality, in conjunction with other SQuaRE series of International Standards, especially ISO/IEC 25010, ISO/IEC 25030, ISO/IEC 25040 and ISO/IEC 25041.

The set of quality measures in this International Standard were selected based on their practical value and are categorized into two levels of reliability. They are not intended to be exhaustive and users of this International Standard are encouraged to refine them if necessary.

Quality measurement division

This International Standard is a part of the ISO/IEC 2502n series 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.

— ISO/IEC 25021 — **Quality measure elements**: provides a format for specifying quality measure elements and some examples of quality measure elements (QMEs) that can be used to construct software quality measures.

— ISO/IEC 25022 — **Measurement of quality in use**: provides measures including associated measurement functions for the quality characteristics in the quality in use model.

— ISO/IEC 25023 — **Measurement of system and software product quality**: provides measures including associated measurement functions for the quality characteristics in the product quality model.

— ISO/IEC 25024 — **Measurement of data quality**: provides measures including associated measurement functions for the quality characteristics in the data quality model.

Figure 1 depicts the relationship between this International Standard and the other International Standards in the ISO/IEC 2502n division. Developers, evaluators, quality managers, acquirers, suppliers, maintainers and users of target system/software product can select measures from these International Standards for the measurement of quality characteristics of interest. This could be for defining requirements, evaluating system/software products, performing quality management activities or for other purposes.

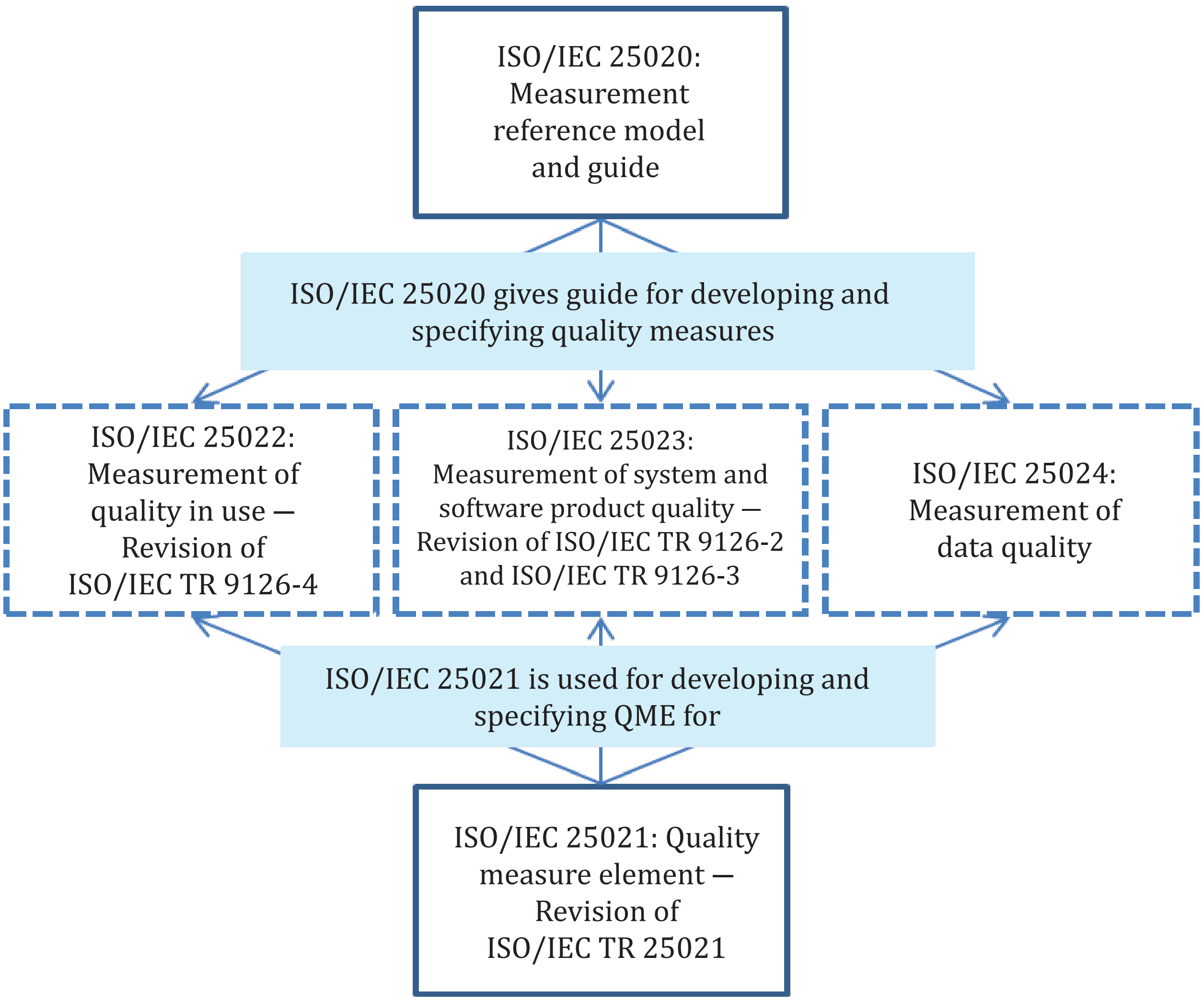

ISO/IEC 25020: Measurement reference model and guide
ISO/IEC 25020 gives guide for developing and specifying quality measures
ISO/IEC 25022: Measurement of quality in use — Revision of ISO/IEC TR 9126-4
ISO/IEC 25023: Measurement of system and software product quality — Revision of ISO/IEC TR 9126-2 and ISO/IEC TR 9126-3
ISO/IEC 25024: Measurement of data quality
ISO/IEC 25021 is used for developing and specifying QME for
ISO/IEC 25021: Quality measure element — Revision of ISO/IEC TR 25021

Figure 1 — Structure of the Quality Measurement Division

Outline and organization of SQuaRE series

The SQuaRE series consists of five main divisions and extension division. Outline of each divisions within SQuaRE series are as follows.

— 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. The division also provides requirements and guidance for the planning and management of a project.

— ISO/IEC 2501n — **Quality Model Division**. The standards that form this division provide quality models for system/software products, quality in use, and data. A service quality model is under development. 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, definitions of quality measures, and practical guidance for their application. This division presents internal measures of software quality, external measures of software quality, quality in use measures, and data quality measures. Quality measure elements forming foundations for the quality measures are defined and presented.

— ISO/IEC 2503n — **Quality Requirements Division**. The standards that form this division help specify quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a system/software product to be developed, designing a process for achieving necessary quality, or as inputs for an evaluation process.

— 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 are reserved for SQuaRE extension International Standards, which currently include ISO/IEC 25051 and ISO/IEC 25060 to ISO/IEC 25069.

Systems and software engineering — 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 evaluating 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. It can be used in conjunction with the ISO/IEC 2503n and the ISO/IEC 2504n standards or to more generally meet user needs with regard to software product or system quality.

This International Standard contains the following:

— a basic set of quality measures for each characteristic and subcharacteristics;

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

It includes, as informative annexes, considerations for the use of quality measures (Annex A), QMEs used to define product or system quality measures (Annex B), and detailed explanation of measurement types (Annex C).

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

The proposed quality measures are primarily intended to be used for quality assurance and improvement of system and software products during or post the development life cycle process.

The main users of this International Standard are people carrying out quality requirement specification and evaluation activities as part of the following:

— development: including requirements analysis, design specification, coding and testing through acceptance during the life cycle process;

— quality management: systematic examination of the software product or computer system, for example, when evaluating system or software product quality as part of quality assurance, quality control and quality certification;

— supply: a contract with the acquirer for the supply of a system, software product or software service under the terms of a contract, for example, when validating quality at qualification test;

— acquisition: including product selection and acceptance testing, when acquiring or procuring a system, software product or software service from a supplier;

— maintenance: improvement of the software product or system based on quality measurement.

2 Conformance

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

a) select the quality characteristics and/or subcharacteristics to be specified or evaluated as defined in ISO/IEC 25010;

b) for each selected characteristic or subcharacteristic, all the Generic (G) quality measures defined in Clause 8 should be used. If any are excluded, then provide a rationale;

c) optionally select any Specific (S) quality measures in Clause 8 that are relevant;

d) if any quality measure is modified, provide the rationale for the changes;

e) define any additional quality measures and QMEs as per ISO/IEC 25021 that are not included in this International Standard.

3 Normative references

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

ISO/IEC 25000, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*

ISO/IEC 25010:2011, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*

ISO/IEC 25021:2012, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality measure elements*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 25000 and ISO/IEC 25010 and the following apply.

NOTE The essential definitions from ISO/IEC 25000 SQuaRE series and the other ISO standards are reproduced here.

4.1
external measure (of system or software quality)
measure of the degree to which a system or software product enables the behaviour to satisfy stated and implied needs for the system including the software to be used under specified conditions

Note 1 to entry: Attributes of the behaviour can be verified and/or validated by executing the system or software product during testing and operation.

EXAMPLE The failure density against test cases found during testing is an external measure of software quality related to the reliability of the computer system. The two measures are not necessarily identical since testing may not find all faults, and a fault may give rise to apparently different failures in different circumstances.

[SOURCE: ISO/IEC 25000:2014, 4.11, modified]

4.2
internal measure (of software quality)
measure of the degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions

Note 1 to entry: Static attributes include those that relate to the software architecture, structure and its components.

Note 2 to entry: Static attributes can be verified by review, inspection, simulation and/or automated tools.

EXAMPLE The number of lines of code, complexity measures and the number of faults found in a walk through are all internal measures of software quality made on the product itself.

[SOURCE: ISO/IEC 25000:2014, 4.16, modified]

4.3
job
user-defined unit of work that is to be accomplished by a computer

[SOURCE: ISO/IEC/IEEE 24765:2010, 3.1542, modified]

4.4
measure
variable to which a value is assigned as the result of measurement

Note 1 to entry: The term "measures" is used to refer collectively to base measures, derived measures and indicators.

Note 2 to entry: In this International Standard, whenever the word "measure" is used qualified by a quality characteristic or sub-characteristic, it refers to a quality measure as defined in 4.8 below.

[SOURCE: ISO/IEC 15939:2007, 2.15, modified]

4.5
measurement
set of operations having the object of determining a value of a measure

Note 1 to entry: Measurement can include assigning a qualitative category such as the language of a source program (ADA, C, COBOL, etc.).

[SOURCE: ISO/IEC 15939:2007, 2.17, modified]

4.6
measurement function
algorithm or calculation performed to combine two or more quality measure elements

[SOURCE: ISO/IEC 25021:2012, 4.7, modified]

4.7
property to quantify
property of a target entity that is related to a quality measure element and which can be quantified by a measurement method

Note 1 to entry: A software artifact is an example of a target entity.

[SOURCE: ISO/IEC 25021:2012, 4.11, modified]

4.8
quality measure
derived measure that is defined as a measurement function of two or more values of quality measure elements

[SOURCE: ISO/IEC 25021:2012, 4.13]

4.9
quality measure element
QME
measure defined in terms of a property and the measurement method for quantifying it, including optionally the transformation by a mathematical function

[SOURCE: ISO/IEC 25021:2012, 4.14, modified]

4.10
quality model
defined set of characteristics, and of relationships between them, which provides a framework for specifying quality requirements and evaluating quality

[SOURCE: ISO/IEC 25000:2014, 4.27]

4.11
quality characteristic (of software product or system)
category of quality attributes that bears on software product or system quality

[SOURCE: ISO/IEC 25000:2014, 4.34, modified]

4.12
task
set or sequence of activities required to achieve a given goal

Note 1 to entry: These activities can be physical or cognitive.

Note 2 to entry: Role and responsibilities can determine goals and tasks.

[SOURCE: ISO/IEC 9241-11:1998, 3.9, modified]

5 Abbreviated terms

The following abbreviated term is used in this International Standard.

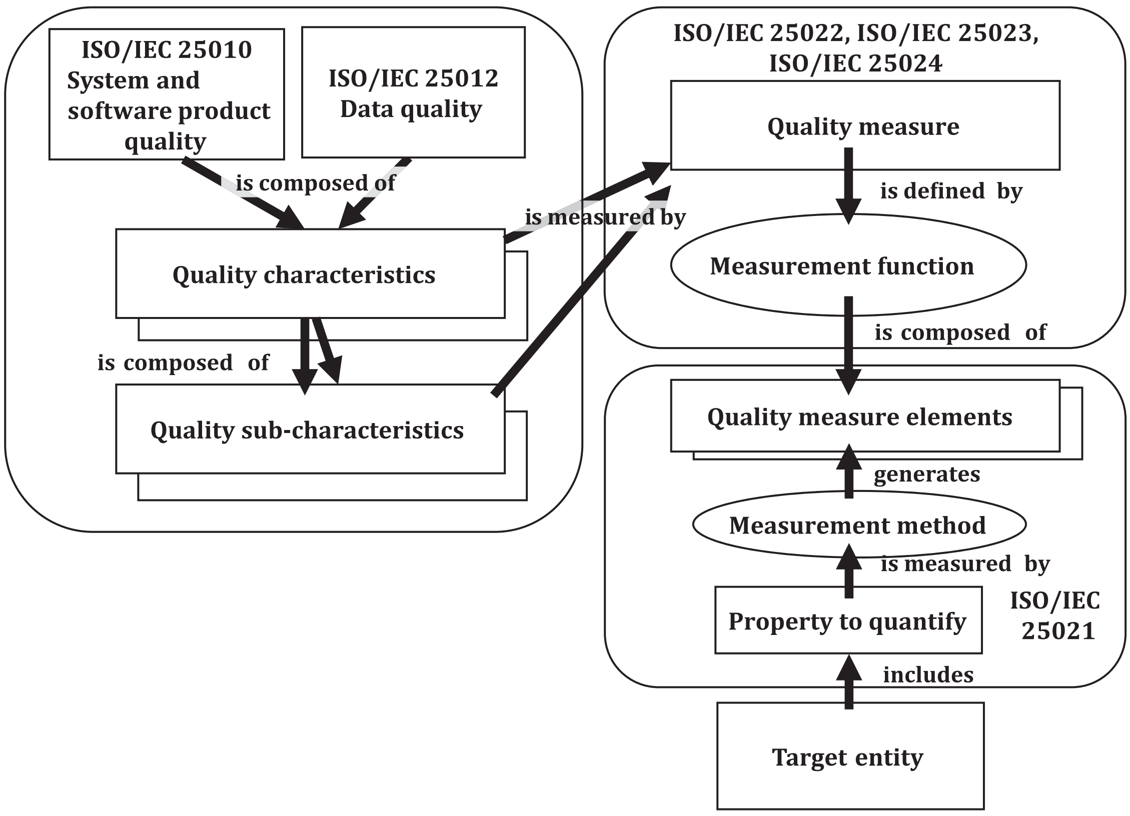
QME	quality measure element

6 Use of system and software product quality measures

6.1 System/software product quality measurement concepts

The quality of a system/software product is the degree to which it satisfies the stated and implied needs of its various stakeholders, and thus provides value. These stated and implied needs are represented in the SQuaRE series of standards by quality models that categorize system/software product quality into characteristics, which in some cases are further subdivided into subcharacteristics. The measurable quality-related properties of a system/software product are called properties to quantify and can be associated with quality measures. These properties are measured by applying a measurement method. A measurement method is a logical sequence of operations used to quantify properties with respect to a specified scale. The result of applying a measurement method is called a quality measure element.

The quality characteristics and subcharacteristics can be quantified by applying measurement functions. A measurement function is an algorithm used to combine quality measure elements. The result of applying a measurement function is called a quality measure. In this way, quality measures become quantifications of the quality characteristics and subcharacteristics. More than one quality measure can be used for the measurement of a quality characteristic or subcharacteristic (see Figure 2).



NOTE Target entity can be a system, a software product, data or a user (see ISO/IEC 25010:2011, Figure 5).

Figure 2 — Relationship among quality model, QM, QME, property to quantify, target entity

6.2 Approach to quality measurement

User needs for quality include requirements for system quality in use in specific contexts of use. These identified needs can be considered when specifying external and internal measures of quality using software product quality characteristics and subcharacteristics.

Software product quality can be evaluated by measuring internal properties (typically static measures of intermediate products), or by measuring external properties (typically by measuring the behaviour of the code when executed), or by measuring quality in use properties (when the product is in real or simulated use). Appropriate internal properties of the software are a prerequisite for achieving the required external behaviour and appropriate external behaviour is a prerequisite for achieving quality in use (see Figure 3).

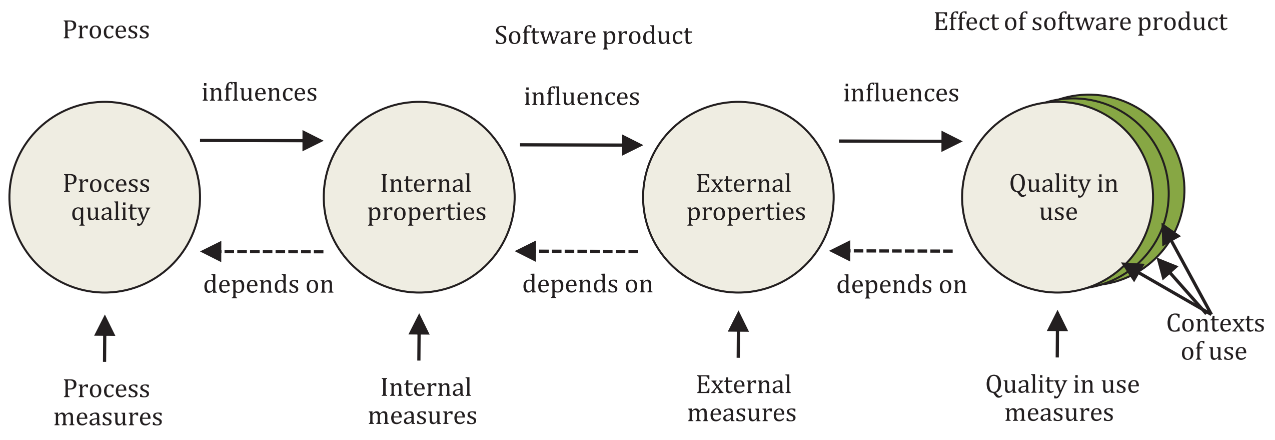


Figure 3 — Relationship between types of quality measures

The internal measures can 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) which can be verified by review, inspection, simulation and/or automated tools. Internal 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. For example, complexity measures and the number, severity and failure frequency of faults found in a walk through are internal measures of software quality made on the product itself.

The external measures can 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 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 tested and/or intended to operate. For example, the number of failures found during testing is an external measure of software quality related to the number of faults present in the computer system.

It is recommended, where possible, to use internal measures that have a strong relationship with the target external measures so that they can be used to predict the values of external measures.

This International Standard provides a suggested 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 International Standard can modify the quality measures defined and can also define and use quality measures not identified or defined in this International Standard.

NOTE 1 For example, the specific measurement of quality characteristics, such as safety or security, can be found in the International Standards provided by IEC 65 and ISO/IEC JTC 1/SC 27.

When using a modified or a new quality measure not identified in this International Standard, the user should specify how the measure relates to the ISO/IEC 25010 quality model or any other substitute quality model that is being used.

Most quality measures use a measurement function, which normalizes the result value within a range of 0,0 to 1,0. Closer to 1,0 is better. When this is not true, the interpretation is described in a NOTE.

Some quality measures produce a result that is relative to a target value that needs to be established as part of requirements.

NOTE 2 Some measurements are normalized against the target value specified in a requirement specification, a design specification, or a user documentation. Such target value is able to be determined and required as the threshold by developers or maintainers to improve architecture, design, implementation, assembles, operational procedures, user interface or performance of the software product or system. The target value is also able to be specified as one of agreed requirements by acquirers and suppliers to specify quality requirements or to examine conformance for acquisition. A requirements specification is usually changed and revised during development and affects the quality measures based on it. Some of requirements to be specified might be missing or inconsistent, or some of the target values might be insufficient and need to be changed because it is very difficult to specify completely both of stated and implied needs derived from stakeholder or system requirements at the beginning of development. Accordingly, users of quality measures are expected to take account of evolving and revising a requirements specification and to apply quality measures not at once but iteratively during development and/or evaluation.

NOTE 3 Some quality measures (such as mean response time) can be difficult to interpret in isolation. The following are ways that quality measures can be applied so that they are easier to understand and interpret:

a) conformance: comparing measures with a specific business or usage requirements (e.g. the maximum acceptable response time is 0,5 seconds);

b) benchmarks: comparing measures with a benchmark for the same or a similar product or system used for the same purpose (e.g. the mean response time of the new system in no more than the mean response time of the old system);

c) time series: comparing trends over time (e.g. how does the mean response time change during the day).

7 Format used for documenting the quality measures

The following information is given for each quality measure in the tables in Clause 8:

a) ID: identification code of quality measure; each ID consists of the following three parts:

— abbreviated alphabetic code representing the quality characteristics as capital X and subcharacteristics as one capital X followed by lowercase x (for example, “PTb” denotes “Time behaviour” measures for “Performance efficiency”);

— serial number of sequential order within quality subcharacteristic;

— G (Generic) or S (Specific) expressing potential categories of quality measure; where, Generic measures can be used whenever appropriate and Specific measures could be used when relevant in a particular situation;

b) Name: quality measure name;

c) Description: the information provided by the quality measure;

d) Measurement function: mathematical formula showing how the quality measure elements are combined to produce the quality measure.

NOTE Useful QMEs which can be used frequently to construct quality measures are specified briefly in Annex B to help comprehend and apply measurement function for the quality measures.

8 System and software product quality measures

8.1 General

The quality measures in Clause 8 are listed by quality characteristics and subcharacteristics in the order used in ISO/IEC 25010.

Quality measures can be used with different evaluation techniques that could be chosen according to quality characteristics and evaluation rating levels depending on whether it is used as internal or external measures. Accordingly, some quality measures listed in Clause 8 can be used at different stages of evaluation such as static review of design specification or dynamic analysis of executable products.

Quality measures, which may be applicable, are not limited to these listed here. It is recommended to refer a specific measure or measurement from specific International Standards or guidelines. For example, functional size measurement is defined in ISO/IEC 14143 and an example of precise time efficiency measurement can be referred from ISO/IEC 14756.

NOTE 1 This list of quality measures is not finalized and might be revised in future versions of this International Standard. Readers of this International Standard are invited to provide feedback.

NOTE 2 In this clause, the word measure means quality measure unless otherwise mentioned. For example, “Functional suitability measures” means “Functional suitability quality measures”.

8.2 Functional suitability measures

Functional suitability measures are used to assess the degree to which a product or system provided functions that meet stated and implied needs when used under specified conditions.

NOTE 1 Functional suitability is concerned with whether the functions meet stated and implied needs.

NOTE 2 A function referred to here could be an elementary process as defined in functional user requirements in ISO/IEC 14143.

NOTE 3 Similar measures with other QMEs like functional size can be defined as a way to weight the result with better accuracy, as unit ratios do not indicate the quantum of functionality that is missing.

8.2.1 Functional completeness measures

Functional completeness measures are used to assess the degree to which the set of functions covers all the specified tasks and user objectives.

Table 1 — Functional completeness measures

ID	Name	Description	Measurement function
FCp-1-G	**Functional coverage**	What proportion of the specified functions has been implemented?	X = 1 – A/B A = Number of functions missing B = Number of functions specified
NOTE 1 Functions can be specified in a requirement specification, a design specification, a user manual or all of these. NOTE 2 A missing function is detected when the system or software product does not have the ability to perform a function that is specified.			

8.2.2 Functional correctness measures

Functional correctness measures are used to assess the degree to which a product or system provides the correct results with the needed degree of precision.

Table 2 — Functional correctness measures

ID	Name	Description	Measurement function
FCr-1-G	**Functional correctness**	What proportion of functions provides the correct results?	X = 1 – A/B A = Number of functions that are incorrect B = Number of functions considered
NOTE 1 An incorrect function is one that does not provide a reasonable and acceptable outcome to achieve the specific intended objective. NOTE 2 The functions considered for evaluation may be all the functions of a product or a specific set of functions required for a particular usage. NOTE 3 Developer or maintainer possibly examines an individual function by reviewing or testing and determines whether the function successfully provides suitable outcomes to specific objectives as defined in the requirements specification or not. In such a case, the degree of correctness is determined per an individual function.			

8.2.3 Functional appropriateness measures

Functional appropriateness measures are used to assess the degree to which the functions facilitate the accomplishment of specified tasks and objectives.

Table 3 — Functional appropriateness measures

ID	Name	Description	Measurement function
FAp-1-G	**Functional appropriateness of usage objective**	What proportion of the functions required by the user provides appropriate outcome to achieve a specific usage objective?	X = 1 – A/B A = Number of functions missing or incorrect among those that are required for achieving a specific usage objective B = Number of functions required for achieving a specific usage objective
NOTE 1 This function will typically be considered for the most important or most frequently identified usage objectives. Thus, this quality measure is first calculated for each of the defined usage objectives that can be pursued in the system, and then the next quality measure, i.e. FAp-2-G “Functional appropriateness of the system”, can be calculated collectively across all usage objectives to provide a system measure. NOTE 2 Users of this International Standard could also consider measuring the proportion of user objectives that are achievable in order to get a better understanding of the actual impact on user’s intended usage.			
FAp-2-G	**Functional appropriateness of system**	What proportion of the functions required by the users to achieve their objectives provides appropriate outcome?	$X = \sum_{i=1 \text{ to } n} A_i / n$ A_i = Appropriateness score for usage objective i, that is, the measured value of FAp-1-G for i-th specific usage objective n = Number of usage objectives

8.3 Performance efficiency measures

Performance efficiency measures are used to assess 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).

NOTE 1 The performance efficiency 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, performance efficiency measures might include the ratio of estimated or measured 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 estimated or measured values can be established and compared to requirement specifications.

NOTE 2 It is also recommended that a task be identified and defined to be suitable for performance efficiency or capacity measures; for example, a transaction as a task for a business application, a switching or data packet sent as a task for a communication application, an event control as a task for a control application and an output of data produced by a user callable function as a task for a common user application.

8.3.1 Time behaviour measures

Time behaviour measures are used to assess the degree to which the response and processing times and throughput rates of a product or system when performing its functions meet the requirements.

Table 4 — Time behaviour measures

ID	Name	Description	Measurement function
PTb-1-G	**Mean response time**	How long is the mean time taken by the system to respond to a user task or system task?	$X = \sum_{i=1 \text{ to } n} (A_i) / n$ A_i = Time taken by the system to respond to a specific user task or system task at i-th measurement n = Number of responses measured
PTb-2-G	**Response time adequacy**	How well does the system response time meet the specified target?	X = A/B A = Mean response time measured by PTb-1-G B = Target response time specified
NOTE 1 Result of a smaller value is better and less than or equal to 1 is good. NOTE 2 Response time is the time from the submission of a request until the first response is produced, i.e. the time it takes to start responding, not the time it take to output the response. NOTE 3 An alternative to this measure is nth percentile response time under expected load conditions. It is also useful to apply it on individual functions or classes of functions.			
PTb-3-G	**Mean turnaround time**	What is the mean time taken for completion of a job or an asynchronous process?	$X = \sum_{i=1 \text{ to } n} (B_i - A_i) / n$ A_i = Time of starting a job i B_i = Time of completing the job i n = Number of measurements
PTb-4-G	**Turnaround time adequacy**	How well does the turnaround time meet the specified targets?	X = A/B A = Mean turnaround time measured by PTb-3-G B = Target turnaround time specified
NOTE 1 Result of a smaller value is better and less than or equal to 1 is good. NOTE 2 In the case of a pipeline (e.g. a systems chain), the elapsed time in each stage of the pipeline has to be considered and bottlenecks in one stage can affect overall turnaround time. NOTE 3 It is recommended to use this measure in conjunction with specified payload and/or workload.			

Table 4 *(continued)*

ID	Name	Description	Measurement function
PTb-5-G	**Mean throughput**	What is the mean number of jobs completed per unit time?	$X = \sum_{i=1 \text{ to } n} (A_i / B_i) / n$ A_i = Number of jobs completed during the i-th observation time B_i = i-th observation time period n = Number of observations
NOTE 1 Jobs could be fine-grained operations like microprocessor operations or coarse grained transaction processing units like those defined by Transaction Processing Performance Council (TPC) or higher level abstractions like functions. So, the results of this measure when used in different contexts should be interpreted appropriately. NOTE 2 Mean throughput is able to be compared to a target threshold of throughput to calculate the throughput adequacy. When such a target threshold under specific condition is specified as one of requirements, the result value is required to be larger than 1.			

8.3.2 Resource utilization measures

Resource utilization measures are used to assess the degree to which the amounts and types of resources used by a product or system when performing its functions meet the requirements.

Table 5 — Resource utilization measures

ID	Name	Description	Measurement function
PRu-1-G	**Mean processor utilization**	How much processor time is used to execute a given set of tasks compared to the operation time?	$X = \sum_{i=1 \text{ to } n} (A_i / B_i) / n$ A_i = Processor time actually used to execute a given set of tasks in observation i B_i = Operation time to perform the tasks in observation i n = Number of observations
NOTE Result value varies from greater than 0 to 1. Usually, the smaller is better.			
PRu-2-G	**Mean memory utilization**	How much of memory is used to execute a given set of tasks compared to the available memory?	$X = \sum_{i=1 \text{ to } n} (A_i / B_i) / n$ A_i = Size of memory actually used to perform a given set of tasks for i-th sample processing B_i = Size of memory available to perform the tasks during i-th sample processing n = Number of samples processed
NOTE Result value varies from greater than 0 to 1. Usually, the smaller is better.			
PRu-3-G	**Mean I/O devices utilization**	How much of I/O device busy time is used to perform a given set of tasks compared to the I/O operation time?	$X = \sum_{i=1 \text{ to } n} (A_i / B_i) / n$ A_i = Duration of I/O device(s) busy time to perform a given set of tasks for i-th observation B_i = Duration of I/O operations to perform the tasks for i-th observation n = Number of observations

Table 5 *(continued)*

ID	Name	Description	Measurement function
NOTE 1 Result value varies from greater than 0 to1. Usually, the smaller is better. NOTE 2 Busy time means the period of time during which a system or a device is actually working.			
PRu-4-S	**Bandwidth utilization**	What proportion of the available bandwidth is utilized to perform a given set of tasks?	X = A/B A = Bandwidth of actual transmission measured over time to perform a given set of tasks B = Bandwidth capacity available to perform a given set of tasks
NOTE 1 In case there is a concern whether the relevant type of resource is well utilized during specific time period or not, for example, to complete specified tasks with maximum resource utilization by avoiding interrupting processing, the result value of closer to optimal is better. In this case, the optimal value depends on the circumstance. NOTE 2 The measurer has to consider the possible communication traffic limitations (e.g. dropping or throttling) which can affect the resulting statistical values including average.			

8.3.3 Capacity measures

Capacity measures are used to assess the degree to which the maximum limits of a product or system parameter meet the requirements.

NOTE 1 Capacity measures are expected to be measured through dynamic analysis, such as volume testing of the system, or can be measured by system integration testing or simulation. Maximum value and distribution of the duration can be investigated for many cases of static analysis, dynamic testing or operations.

NOTE 2 The maximum limit is expected to be specified as a target value which can theoretically be beyond a possible realistic value.

Table 6 — Capacity measures

ID	Name	Description	Measurement function

PCa-1-G	**Transaction processing capacity**	How many transactions can be processed per unit time?	X = A/B
			A = Number of transactions completed during observation time
			B = Duration of observation

NOTE 1 Result value varies from 0 to maximum limit. Usually, the larger is better. NOTE 2 This measure can be useful only if there is sufficient workload to test. NOTE 3 Task can be alternatively used, as well as transaction.			
PCa-2-G	**User access capacity**	How many users can access the system simultaneously at a certain time?	$X = \sum_{i=1 \text{ to } n} A_i / n$ A_i = Maximum number of users who can simultaneously access the system at i-th observation n = Number of observations

Table 6 *(continued)*

ID	Name	Description	Measurement function
NOTE Result value varies from 0 to maximum limit. Usually, the result of larger value is better.			

PCa-3-S	**User access increase adequacy**	How many users can be added successfully per unit time?	X = A/B
			
			A = Number of users successfully added during observation time
			
			B = Duration of observation

NOTE 1 Result value varies from 0 to maximum limit. Usually, the larger is better. NOTE 2 This measure indicates the degree to which the capability of software or system to have enough capacity to accept accesses from a lot of users, even during rapid increase of users in a given moment, e.g. an extremely large number of users could simultaneously access the system or software in an instance through the internet.			

8.4 Compatibility measures

Compatibility measures are used to assess 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.

8.4.1 Co-existence measures

Co-existence measures are used to assess 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.

Table 7 — Co-existence measures

ID	Name	Description	Measurement function
CCo-1-G	**Co-existence with other products**	What proportion of specified software products can share the environment with this software product without adverse impact on their quality characteristics or functionality?	X = A/B A = Number of other specified software products with which this product can co-exist B = Number of other software products specified to co-exist with this product in the operation environment

8.4.2 Interoperability measures

Interoperability measures are used to assess the degree to which two or more systems, products or components can exchange information and successfully use the information that has been exchanged.

Table 8 — Interoperability measures

ID	Name	Description	Measurement function
CIn-1-G	**Data formats exchangeability**	What proportion of the specified data formats is exchangeable with other software or systems?	X = A/B A = Number of data formats exchangeable with other software or systems B = Number of data formats specified to be exchangeable
Cln-2-G	**Data exchange protocol sufficiency**	What proportion of the specified data exchange protocols is supported?	X = A/B A = Number of data exchange protocols supported B = Number of data exchange protocols specified to be supported
NOTE For the details of data quality, refer to Con-I-1 in ISO/IEC 25024.			
CIn-3-S	**External interface adequacy**	What proportion of the specified external interfaces (interfaces with other software and systems) is functional?	X = A/B A = Number of external interfaces that are functional B = Number of external interfaces specified

8.5 Usability measures

Usability measures are used to assess 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.

NOTE 1 Internal usability measures are used for predicting the extent to which the software in question can be understood, learned and operated and will enable pleasing and satisfying interaction for the user.

NOTE 2 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 could be used.) For reliable results, a sample of a large group of committed users is necessary, although useful information can be obtained from smaller groups. Users carry out the test without any hints or external assistance.

NOTE 3 Internal and external measures for usability make comparisons between stated design conventions, specific guidelines or specification for usability and actually developed documented design, prototype or executable system/software. Therefore, it is very important to elicit end user's requirements and create well specific specification for usability by considering characteristics and measures of quality in use as well as user centred design concept and human ergonomics view. For example, usability specific guideline, templates, or check list are necessary to explain in detail what kinds of messages are easy to understand for end users.

NOTE 4 In this International Standard, the target entities of usability measures are limited to any system or software product only. Usability measures concerning the effectiveness, efficiency and satisfaction in a specified context of use can be found in ISO/IEC 25022.

NOTE 5 The usability measures would inevitably generate somewhat subjective results. In case of difficulties in measuring with a ratio scale, an ordinal scale can be used as an alternative depending on the situation, e.g., 1,0 for excellent, 0,8 for good, 0,6 for average, 0,4 for poor, and 0,2 for bad.

8.5.1 Appropriateness recognizability measures

Users have to be able to select a system/software product which is suitable for their intended use. The quality measures for appropriateness recognizability are used to assess the degree to which users can recognize whether a product or system is appropriate for their needs.

NOTE Appropriateness recognizability measures can be used to assess whether new users can understand:

— whether the software product or system is suitable for their purposes or not;

— how it can be used for particular tasks

Table 9 — Appropriateness recognizability measures

ID	Name	Description	Measurement function
UAp-1-G	**Description completeness**	What proportion of usage scenarios is described in the product description or user documents?	X = A/B A = Number of usage scenarios described in the product description or user documents B = Number of usage scenarios of the product
UAp-2-S	**Demonstration coverage**	What proportion of tasks has demonstration features for users to recognize the appropriateness?	X = A/B A = Number of tasks with demonstration features B = Number of tasks that could benefit from demonstration features
UAp-3-S	**Entry point self-descriptiveness**	What proportion of the commonly used landing pages on a website explains the purpose of the website?	X = A/B A = Number of landing pages that explain the purpose of website B = Number of landing pages in a website

8.5.2 Learnability measures

Learnability measures are used to assess 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.

Table 10 — Learnability measures

ID	Name	Description	Measurement function
ULe-1-G	**User guidance completeness**	What proportion of functions is explained in sufficient detail in user documentation and/or help facility to enable the user to apply the functions?	X = A/B A = Number of functions described in user documentation and/or help facility as required B = Number of functions implemented that are required to be documented
NOTE 1 Learnability is strongly related to appropriateness recognizability and appropriateness recognizability measurements can be indicators of the learnability potential of the software. NOTE 2 Help facility includes, for example, on-line help, operational guide video, operational instruction system, etc. NOTE 3 More sophisticated measure is possibly formulated when views from ergonomics or user experiences are well applied, for example, the degree of matching on mapping of system conceptual model to user mental model.			
ULe-2-S	**Entry fields defaults**	What proportion of entry fields that could have default values are automatically filled with default values?	X = A/B A = Number of entry fields whose default values have been automatically filled in during operation B = Number of entry fields that could have default values
NOTE The default values for entry fields are helpful for beginners to learn how to operate the product comprehensively and quickly.			

Table 10 *(continued)*

ID	Name	Description	Measurement function
ULe-3-S	**Error messages understandability**	What proportion of the error messages state the reason why the error occurred and how to resolve it?	X = A/B A = Number of error messages which state the reason of occurrence and suggest the ways of resolution where this is possible B = Number of error messages implemented
ULe-4-S	**Self-explanatory user interface**	What proportion of information elements and steps presented to the user enable common tasks to be completed by a first-time user without prior study or training or seeking external assistance?	X = A/B A = Number of information elements and steps that are presented in a way that the user could understand B = Number of information elements and steps needed to complete common tasks for a first time user
NOTE This measure is particularly relevant for public system and website.			

8.5.3 Operability measures

Operability measures are used to assess the degree to which a product or system has attributes that make it easy to operate and control.

NOTE 1 Operability measures are expected to be measured through operational testing by representatives of operators or end users, or can be measured through static analysis such as review of requirement, design specification or user manuals.

NOTE 2 An internal or external operability quality measure is used to assess whether users can operate and control the software. Operability measures can be categorized by the following dialogue principles in ISO 9241-110:

— 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 individualization.

Table 11 — Operability measures

ID	Name	Description	Measurement function
UOp-1-G	**Operational consistency**	To what extent do interactive tasks have a behaviour and appearance that is consistent both within the task and across similar tasks?	X = 1 – A/B A = Number of specific interactive tasks that are performed inconsistently B = Number of specific interactive tasks that need to be consistent
UOp-2-G	**Message clarity**	What proportion of messages from a system conveys the right outcome or instructions to the user?	X = A/B A = Number of messages that convey the right outcome or instructions to the user B = Number of messages implemented
NOTE Messages that provide all available information that could help the user, and when possible explain how to resolve the error.			

Table 11 *(continued)*

ID	Name	Description	Measurement function
UOp-3-S	**Functional customizability**	What proportion of functions and operational procedures can a user customize for his convenience?	X = A/B A = Number of functions and operational procedures which can be customized for user's convenience B = Number of functions and operational procedures for which users could benefit from customization
UOp-4-S	**User interface customizability**	What proportion of user interface elements can be customized in appearance?	X = A/B A = Number of user interface elements that can be customized B = Number of user interface elements that could benefit from customization
UOp-5-S	**Monitoring capability**	What proportion of function states can be monitored during operation?	X = A/B A = Number of functions having state monitoring capability B = Number of functions that could benefit from monitoring capability
NOTE 1 Monitoring and management of operational state of some functions are very important in case of, for example, distributed system, embedded system and so on. NOTE 2 For a better measurement, it is helpful to find which function has a benefit from monitoring capability from a view of usability during operational scenario review or operational testing by user. Such a function is also able to be specified as requirements.			
UOp-6-S	**Undo capability**	What proportion of tasks that has a significant consequence provides an option for re-confirmation or undo capability?	X = A/B A = Number of tasks that provide undo capability or prompt for re-confirmation B = Number of tasks for which users could benefit from having re-confirmation or undo capability
UOp-7-S	**Understand-able categorization of information**	To what extent does the software organize information in categories that are familiar to the intended users and convenient for their tasks?	X = A/B A = Number of information structures that are familiar and convenient for the intend-ed users B = Number of information structures used
EXAMPLE The online shop of a department store organizes the goods in a similar way to the physical layout of the goods in the store.			
UOp-8-S	**Appearance consistency**	What proportion of user interfaces with similar items has a similar appearance?	X = 1 – A/B A = Number of user interfaces with similar items but with different appearances B = Number of user interfaces with similar items
NOTE For example, the "OK" and "Cancel" in screens are always located at the same position.			
UOp-9-S	**Input device support**	To what extent can tasks be initiated by all appropriate input modalities (such as keyboard, mouse or voice)?	X = A/B A = Number of tasks that can be initiated by all appropriate input modalities B = Number of tasks supported by the system
EXAMPLE Within a search form, the search button can be activated by using the mouse or by pressing the "Enter" key on the keyboard.			

8.5.4 User error protection measures

User error protection measures are used to assess the degree to which the system protects users against making errors.

NOTE User error protection measures are expected to be measured through operational testing by representatives of operators or end users, or can be measured through static analysis such as review of requirement, design specification or user manuals.

Table 12 — User error protection measures

ID	Name	Description	Measurement function
UEp-1-G	**Avoidance of user operation error**	What portion of user actions and inputs are protected against causing any system malfunction?	X = A/B A = Number of user actions and inputs that are protected from causing any system malfunction B = Number of user actions and inputs that could be protected from causing any system malfunction
NOTE 1 This includes the system requesting confirmation before carrying out an action that cannot be undone and that would have significant consequences. EXAMPLE When erasing files within an application, the user is required to confirm each deletion. NOTE 2 For a better measurement, it is helpful to find user actions and inputs where a user often makes an error during operational testing. Such protections of erroneous user actions and inputs are also able to be specified as requirements.			
UEp-2-S	**User entry error correction**	To what extent does the system provide a suggested correct value for detected user entry errors with an identifiable cause?	X = A/B A = Number of entry errors for which the system provides a suggested correct value B = Number of entry errors detected
NOTE For the details of related data quality, refer to Cre-I-1 in ISO/IEC 25024.			
UEp-3-S	**User error recoverability**	What proportion of user errors can be corrected or recovered by the system?	X = A/B A = Number of user errors that are designed and tested to be recovered by the system B = Number of user errors which can occur during operation

8.5.5 User interface aesthetics measures

User interface aesthetics measures are used to assess the degree to which the user interface enables pleasing and satisfying interaction for the user.

Table 13 — User interface aesthetics measures

ID	Name	Description	Measurement function
UIn-1-S	**Appearance aesthetics of user interfaces**	To what extent are user interfaces and the overall design aesthetically pleasing in appearance?	X = A/B A = Number of display interfaces aesthetically pleasing to the users in appearance B = Number of display interfaces
NOTE 1 An internal or external user interface aesthetics quality measure is used to assess the appearance of the user interfaces and will be influenced by factors such as screen design and colour. This is particularly important for consumer products. NOTE 2 Good colour combinations can help users to quickly read the text or identify the image. Then, it can be helpful for better aesthetics measurement to address bad colour combinations, such as light blue on grey, red on orange, green on blue and so on. NOTE 3 This quality measure often depends on an individual of users. Then, either expertise usability designers or testers on behalf of users, or representatives from target user groups are expected to be involved to measure this.			

8.5.6 Accessibility measures

Accessibility measures are used to assess 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.

NOTE For the additional criteria for accessibility, refer to ISO 9241-171.

Table 14 — Accessibility measures

ID	Name	Description	Measurement function
UAc-1-G	**Accessibility for users with disabilities**	To what extent can potential users with specific disabilities successfully use the system (with assistive technology if appropriate)?	X = A/B A = Number of functions successfully usable by the users with a specific disability B = Number of functions implemented
NOTE 1 Specific disabilities include cognitive disability, physical disability, hearing/voice disability, and visual disability. NOTE 2 The range of capabilities includes disabilities associated with age. NOTE 3 Any person becomes possibly a user with limited cognitive, physical, hearing or visual ability under specific situations or environments, for example, in darkness, in low atmospheric pressure at high altitude, in water and so on.			
UAc-2-S	**Supported languages adequacy**	What proportion of needed languages is supported?	X = A/B A = Number of languages actually supported B = Number of languages needed to be supported
NOTE When users are trying to use a system or software with different language from their own native one, they frequently suffer from operational errors and sometimes give up to achieve their intended goals. Such case is one of decreasing accessibility and caused by misunderstanding of description and messages. Then, it has to be considered, specified and implemented, which languages are to be supported for possible variation of users.			

8.6 Reliability measures

Reliability measures are used to assess 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 completed system/software product in question will satisfy prescribed reliability needs during the development of the system/software product.

External reliability quality measures are used to assess attributes related to the behaviour 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.

8.6.1 Maturity measures

Maturity measures are used to assess the degree to which a system, product or component meets the needs for reliability under normal operation.

NOTE The concept of maturity can also be applied to other quality characteristics to indicate the degree to which they meet the required needs under normal operation (see ISO/IEC 25010).

Table 15 — Maturity measures

ID	Name	Description	Measurement function
RMa-1-G	**Fault correction**	What proportion of detected reliability-related faults has been corrected?	X = A/B A = Number of reliability-related faults corrected in design /coding/testing phase B = Number of reliability-related faults detected in design/coding/testing phase
NOTE For example, inadequate error handling is a kind of reliability-related faults.			

RMa-2-G	**Mean time between failure (MTBF)**	What is the MTBF during the system/software operation?	X = A/B
			A = Operation time
			B = Number of system/software failures actually occurred

NOTE 1 Result value varies from 0 to infinite. Usually, the larger is better. NOTE 2 MTBF itself can be used to compare the reliabilities of different systems or software products.			

RMa-3-G	**Failure rate**	What is the average number of failures during a defined period?	X = A/B
			A = Number of failures detected during observation time
			B = Duration of observation

NOTE 1 The period used in this measure could be different for testing and operations purposes, which refers to actual usage or testing time. NOTE 2 A reliability estimation model can use this measure as an input. NOTE 3 The usefulness of this quality measure depends on the adequacy of test cases or the extent of system usage during testing, e.g. normal, exceptional and abnormal cases.			
RMa-4-S	**Test coverage**	What percentage of the system or software capabilities, operational scenarios or functions that are included in their associated test suites are actually performed?	X = A/B A = Number of system or software capabilities, operational scenarios or functions that are actually performed B = Number of system or software capabilities, operational scenarios or functions which are included in their associated test suites

8.6.2 Availability measures

Availability measures are used to assess the degree to which a system, product or component is operational and accessible when required for use.

Table 16 — Availability measures

ID	Name	Description	Measurement function
RAv-1-G	**System availability**	For what proportion of the scheduled system operational time is the system actually available?	X = A/B A = System operation time actually provided B = System operation time specified in the operation schedule
NOTE This measure can be extended to special days, such as holidays and weekend, in addition to regular operational days.			

RAv-2-G	**Mean down time**	How long does the system stay unavailable when a failure occurs?	X = A/B
			A = Total down time
			B = Number of breakdowns observed

NOTE 1 Result value varies from 0 to infinite. Usually, the smaller is better. NOTE 2 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).			

8.6.3 Fault tolerance measures

Fault tolerance measures are used to assess the degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

NOTE An internal or external fault tolerance measure can 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.

Table 17 — Fault tolerance measures

ID	Name	Description	Measurement function
RFt-1-G	**Failure avoidance**	What proportion of fault patterns has been brought under control to avoid critical and serious failures?	X = A/B A = Number of avoided critical and serious failure occurrences (based on test cases) B = Number of executed test cases of fault pattern (almost causing failure) during testing
RFt-2-S	**Redundancy of components**	What proportion of system components is installed redundantly to avoid system failure?	X = A/B A = Number of system components redundantly installed B = Number of system components
NOTE For example, in many safety-critical systems, some parts of the control system could be duplicated with the intention of increasing reliability of the system.			
RFt-3-S	**Mean fault notification time**	How quickly does the system report the occurrence of faults?	$X = \sum_{i=1 \text{ to } n} (A_i - B_i) / n$ A_i = Time at which the fault i is reported by the system B_i = Time at which fault i is detected n = Number of faults detected
NOTE Result value varies from 0 to infinite. Usually, the closer to 0 is the better.			

8.6.4 Recoverability measures

Recoverability measures are used to assess 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.

Table 18 — Recoverability measures

ID	Name	Description	Measurement function
RRe-1-G	**Mean recovery time**	How long does it take for the software/system to recover from failure?	$X = \sum_{i=1\,\text{to}\,n} A_i / n$ A_i = Total time to recover the downed software /system and re-initiate operation for each failure i n = Number of failures
NOTE 1 Result value varies from 0 to infinite. Usually, the smaller is better. NOTE 2 When this quality measure is compared to a target threshold for mean recovery time, that is specified in agreed requirements by acquirer and supplier, the measure is able to be used to examine conformance.			
RRe-2-S	**Backup data completeness**	What proportion of data items is backed up regularly?	X = A/B A = Number of data items actually backed up regularly B = Number of data items requiring backup for error recovery

8.7 Security measures

Security measures are used to assess 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 authorization.

NOTE 1 Penetration tests can be performed to simulate an attack because such a security attack does not normally occur in the usual testing.

NOTE 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 security functions and the assurance of their effectiveness have been addressed extensively in related International Standards. The user of this International Standard has to determine what kind of security functions need to be used in each case depending on the level of risk.

8.7.1 Confidentiality measures

Confidentiality measures are used to assess the degree to which a product or system ensures that data are accessible only to those authorized to have access.

Table 19 — Confidentiality measures

ID	Name	Description	Measurement function
SCo-1-G	**Access controllability**	What proportion of confidential data items are protected from unauthorized accesses?	X = 1 − A/B A = Number of confidential data items that can be accessed without authorization B = Number of data items that require access control
SCo-2-G	**Data encryption correctness**	How correctly is the encryption/decryption of data items implemented as stated in the requirement specification?	X = A/B A = Number of data items encrypted/decrypted correctly B = Number of data items that require encryption/decryption
NOTE For the details of related data quality, refer to Cnf-I-1 in ISO/IEC 25024.			
SCo-3-S	**Strength of cryptographic algorithms**	What proportion of cryptographic algorithms has been well-vetted?	X = 1 − A/B A = Number of cryptographic algorithms broken or unacceptably risky in use B = Number of cryptographic algorithms used
NOTE 1 It is important to select a well-vetted algorithm that is currently considered to be strong by experts in the field and to select well-tested implementations. As with some cryptographic mechanisms, the source code has to be available for analysis. For example, US government systems require FIPS 140-2 certification. NOTE 2 There are other ways of measuring the strength of cryptographic algorithms, for example, using ethical hacking.			

8.7.2 Integrity measures

Integrity measures are used to assess the degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

Table 20 — Integrity measures

ID	Name	Description	Measurement function
SIn-1-G	**Data integrity**	To what extent is the data corruption or modification by unauthorized access prevented?	X = 1 − A/B A = Number of data items which are actually corrupted by unauthorized access B = Number of data items for which data corruption or modification have to be prevented
SIn-2-G	**Internal data corruption prevention**	To what extent are the available prevention methods for data corruption implemented?	X = A/B A = Number of data corruption prevention methods actually implemented B = Number of data corruption prevention methods available and recommended
NOTE Examples of internal methods for data corruption prevention are back up data frequently, compare data to reference data periodically, store data in multiple mirror sites.			
SIn-3-S	**Buffer overflow prevention**	What portion of memory accesses with user input in software modules has been done bounds checking for preventing buffer overflow?	X = A/B A = Number of memory accesses with user input that are bounds checked B = Number of memory accesses with user input in software modules
NOTE A buffer overflow occurs when data written to a buffer corrupts data values in memory addresses adjacent to the destination buffer due to insufficient bounds checking. This can occur when copying data from one buffer to another without first checking that the data fits within the destination buffer.			

8.7.3 Non-repudiation measures

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

Table 21 — Non-repudiation measures

ID	Name	Description	Measurement function
SNo-1-G	**Digital signature usage**	What proportion of events requiring non-repudiation is processed using digital signature?	X = A/B A = Number of events that ensure non-repudiation using digital signature B = Number of events requiring non-repudiation using digital signature
NOTE Certificates and security algorithms are also helpful to improve non-repudiation.			

8.7.4 Accountability measures

Accountability measures are used to assess the degree to which the actions of an entity can be traced uniquely to the entity.

Table 22 — Accountability measures

ID	Name	Description	Measurement function
SAc-1-G	**User audit trail completeness**	How complete is the audit trail concerning the user access to the system or data?	X = A/B A = Number of accesses recorded in all logs B = Number of accesses to system or data actually tested
SAc-2-S	**System log retention**	For what percent of the required retention period is the system log retained in stable storage?	X = A/B A = Duration for which the system log is actually retained in stable storage B = Retention period specified for keeping the system log in stable storage
NOTE 1 A stable storage is a classification of computer data storage technology that guarantees atomicity for any given write operation and allows software to be written that is robust against some hardware and power failures. Most often, stable storage functionality is achieved by mirroring data on separate disks via RAID technology. NOTE 2 Result value varies from 0 to infinite. Usually, larger than 1 is better.			

8.7.5 Authenticity measures

Authenticity measures are used to assess the degree to which the identity of a subject or resource can be proved to be the one claimed.

Table 23 — Authenticity measures

ID	Name	Description	Measurement function
SAu-1-G	**Authentication mechanism sufficiency**	How well does the system authenticate the identity of a subject?	X = A/B A = Number of authentication mechanisms provided (e.g., User ID/password or IC card) B = Number of authentication mechanisms specified
NOTE What is relevant for security is the strength of the authentication model and the ability to have multi-level multi-factor authentication and threat detection. Number of factors and degree of authenticity of provided protocol can also be used as authenticity measure.			
SAu-2-S	**Authentication rules conformity**	What proportion of the required authentication rules is established?	X = A/B A = Number of authentication rules implemented B = Number of authentication rules specified

8.8 Maintainability measures

Maintainability measures are used to assess the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.

8.8.1 Modularity measures

Modularity measures are used to assess 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.

Table 24 — Modularity measures

ID	Name	Description	Measurement function
MMo-1-G	**Coupling of components**	How strongly are the components independent and how many components are free from impacts from changes to other components in a system or computer program?	X = A/B A = Number of components which are implemented with no impact on others B = Number of specified components which are required to be independent
NOTE Such a threshold is helpful to determine whether the degree of impact from changes of other components is minimal or not, for example, the frequency of changes of the component caused by changes of other components or the number of externally shared data bases that the component directly accesses.			
MMo-2-S	**Cyclomatic complexity adequacy**	How many software modules have acceptable cyclomatic complexity?	X = 1– A/B A = Number of software modules which have a cyclomatic complexity score that exceeds the specified threshold B = Number of software modules implemented
NOTE Such a threshold is used to determine whether a value of cyclomatic complexity is acceptable or not for each module. This is defined by each project or organization and is possibly a different value for a programming language, a type of module or function.			

8.8.2 Reusability measures

Reusability measures are used to assess the degree to which an asset can be used in more than one system or in building other assets.

Table 25 — Reusability measures

ID	Name	Description	Measurement function
MRe-1-G	**Reusability of assets**	How many assets in a system can be reusable?	X = A/B A = Number of assets which are designed and implemented to be reusable B = Number of assets in a system
NOTE In this measure, assets could be work products such as requirements documents, source code modules, testing modules, and specific hardware, etc.			
MRe-2-S	**Coding rules conformity**	How many modules conform to required coding rules?	X = A/B A = Number of software modules conforming to coding rules for a specific system B = Number of software modules implemented
NOTE 1 Coding rules for a specific system might include rules which contribute to, for example, modularity, traceability, and conciseness. NOTE 2 This quality measure can also be applied to different characteristics and subcharacteristics such as analysability.			

8.8.3 Analysability measures

Analysability measures are used to assess 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 of its parts, or to diagnose a product for deficiencies or causes of failure, or to identify parts to be modified.

Table 26 — Analysability measures

ID	Name	Description	Measurement function
MAn-1-G	**System log completeness**	To what extent does the system record its operations in logs so that they are to be traceable?	X = A/B A = Number of logs that are actually recorded in the system B = Number of logs for which audit trails are required during operation
MAn-2-S	**Diagnosis function effectiveness**	What proportion of the diagnosis functions meets the requirements of causal analysis?	X = A/B A = Number of diagnostic functions useful for causal analysis B = Number of diagnostic functions implemented
MAn-3-S	**Diagnosis function sufficiency**	What proportion of the required diagnosis functions has been implemented?	X = A/B A = Number of diagnostic functions implemented B = Number of diagnostic functions required
NOTE Analysability measures are used to assess such attributes as the maintainer's or user's effort or resources used when trying to diagnose deficiencies or causes of failures or for identifying parts to be modified.			

8.8.4 Modifiability measures

Modifiability measures are used to assess the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

NOTE Modifiability measures are used to assess 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 make a specified modification.

Table 27 — Modifiability measures

ID	Name	Description	Measurement function
MMd-1-G	**Modification efficiency**	How efficiently are the modifications made compared to the expected time?	$X = \sum_{i=1 \text{ to } n} (A_i / B_i) / n$ A_i = Total work time spent for making a specific type of modification i B_i = Expected time for making the specific type of modification i n = Number of modifications measured
NOTE 1 X greater than 1 represents inefficient modifications and X less than 1 represents very efficient modifications. NOTE 2 Expected time for making a specific type of modification can be based on historical data or industry averages.			
MMd-2-G	**Modification correctness**	What proportion of modifications has been implemented correctly?	X = 1 – (A/B) A = Number of modifications that caused an incident or failure within a defined period after being implemented B = Number of modifications implemented
MMd-3-S	**Modification capability**	To what extent are the required modifications made within a specified duration?	X = A/B A = Number of items actually modified within a specified duration B = Number of items required to be modified within a specified duration

8.8.5 Testability measures

Testability measures are used to assess 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.

NOTE 1 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.

NOTE 2 External testability measures are used to assess 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.

Table 28 — Testability measures

ID	Name	Description	Measurement function
MTe-1-G	**Test function completeness**	How completely are test functions and facilities implemented?	X = A/B A = Number of test functions implemented as specified B = Number of test functions required
MTe-2-S	**Autonomous testability**	How independently can the software be tested?	X = A/B A = Number of tests that can be simulated by stub among the tests which depend on other systems B = Number of tests which depend on other systems
NOTE A stub is a skeletal or special-purpose implementation of a software module used to develop or test a module that calls or is otherwise dependent on it.			
MTe-3-S	**Test restartability**	How easily can the operation test be carried out from the restart point after maintenance?	X = A/B 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 in which executing test run can be paused

8.9 Portability measures

Portability measures are used to assess 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.

8.9.1 Adaptability measures

Adaptability measures are used to assess 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.

Table 29 — Adaptability measures

ID	Name	Description	Measurement function
PAd-1-G	**Hardware environmental adaptability**	Is software or system capable enough to adapt itself to different hardware environment?	$X = 1 - A/B$ A = Number of functions which were not completed or results which were insufficient to meet requirements during testing B = Number of functions which were tested in different hardware environment
PAd-2-G	**System software environmental adaptability**	Is software or system capable enough to adapt itself to different system software environment?	$X = 1 - A/B$ A = Number of functions which were not completed or results which were insufficient to meet requirements during testing B = Number of functions which were tested in different system software environment
NOTE 1 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 have to be measured. NOTE 2 System software may include operating systems, middleware, database management system, compiler, network management system, etc.			
PAd-3-S	**Operational environment adaptability**	Is software or system capable enough to adapt itself to different operational environment?	$X = 1 - A/B$ A = Number of functions which were not completed or results which were insufficient to meet requirements during operational testing with user's environment B = Number of functions which were tested in different operational environment

8.9.2 Installability measures

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

Table 30 — Installability measures

ID	Name	Description	Measurement function
PIn-1-G	**Installation time efficiency**	How efficient is the actual installation time compared to expected time?	$X = \sum_{i=1 \text{ to } n} (A_i / B_i) / n$ A_i = Total work time spent for making an installation i B_i = Expected time for making an installation i n = Number of installations measured
NOTE 1 X greater than 1 represents inefficient installation, and X less than 1 represents very efficient installation. NOTE 2 Expected time for making an installation can be based on historical data or industry averages.			
PIn-2-G	**Ease of installation**	Can users or maintainers customize the installation procedure for their convenience?	$X = A/B$ A = Number of cases in which a user succeeds to customize the installation procedure B = Number of cases in which a user attempted to customize the installation procedure for user's convenience
NOTE Such changes of installation procedure are able to be recognized as customization of installation by user.			

8.9.3 Replaceability measures

Replaceability measures are used to assess the degree to which a product can replace another specified software product for the same purpose in the same environment.

Table 31 — Replaceability measures

ID	Name	Description	Measurement function
PRe-1-G	**Usage similarity**	What proportion of user functions of the replaced product can be performed without any additional learning or workaround?	X = A/B A = Number of user functions which can be performed without any additional learning or workaround B = Number of user functions in the replaced software product
NOTE User functions are those which user can call and use to perform its intended tasks including user interfaces.			
PRe-2-S	**Product quality equivalence**	What proportion of the quality measures is satisfied after replacing previous software product by this one?	X = A/B A = Number of quality measures of the new product which are better or equal to the replaced product B = Number of quality measures of the replaced software product that are relevant
NOTE Some of the critical product qualities relevant to replaceability are interoperability, security and performance efficiency.			
PRe-3-S	**Functional inclusiveness**	Can the similar functions easily be used after replacing previous software product by this one?	X = A/B A = Number of functions which produce similar results as before B = Number of functions which have to be used in the replaced software product
PRe-4-S	**Data reusability/ import capability**	Can the same data be used after replacing previous software product by this one?	X = A/B A = Number of data which can be used continuously as before B = Number of data which are to be used continuously in the replaced software product

Annex A
(informative)

Considerations for the use of quality measures

This Annex deals with a number of considerations in the selection and application of quality measures. Each quality measure defined in Clause 8 can be used in case of measuring internal properties (typically static measures of intermediate products), external properties (typically by measuring the behaviour of the code when executed), or both of them.

NOTE 1 When an iterative or incremental model is applied to development or maintenance, both of internal and external measures can be used for each cycle of iteration or increment. Iteratively increased or improved system/software specification, architectural design, detailed design, component and code are able to be measured by review with internal measures, while iteratively integrated system/software are able to be measured by executing them for build testing tasks with external measures during each of iteration or increment. In case executable testing is able to be conducted very frequently during iterations or increments, external measures (or quality in use measure) are possibly employed more than internal measures. Then, these quality measures can be repeatedly used to monitor evolving quality trends through multiple iterations or increments. For example, measured value of functional coverage, which is one of quality measures for functional suitability, is possibly lower in the early iterations and is expected to be increasing to higher in the later iterations.

NOTE 2 Internal measures for performance efficiency are applicable to static design documents or source code. These measured values are able to be obtained by estimation of theoretical calculation amount of designed algorithms, number of function calls or steps of executable code. However, applications of external measures for performance efficiency on intermediate executable prototype during design are helpful to understand actual gaps between internal and external measures and calibrate estimation for internal measures.

NOTE 3 Internal measures for usability are applicable to static mock-up of screen display, specification for usability design, a set of message text files, user manuals, source code for user interfaces and so on. However, applications of external measures for usability on intermediate executable prototype during development are helpful to understand actual gaps between internal and external measures. If available, application of quality in use measures to executable prototype during development is also very helpful.

In addition, the quality measures can be classified according to the recommendation level such as

— HR: highly recommended, which means “use this quality measure always”,

— R: recommended, which means “use this quality measure when appropriate”, and

— UD: used at user’s discretion, which means “use this quality measure as a reference when developing a new quality measure” because the measure has unknown reliability.

Table A.1 represents this kind of considerations related to the usage of each quality measure.

Table A.1 — Summary table for the usage of quality measures

Quality characteristic	Quality subcharacteristic	ID	Quality measure name	Internal/ External/Both	Recommenda-tion level
Functional suitability	Functional completeness	FCp-1-G	Functional coverage	Both	HR
	Functional correctness	FCr-1-G	Functional correctness	Both	HR
	Functional appropriateness	FAp-1-G	Functional appropriateness of usage objective	Both	HR
		FAp-2-G	Functional appropriateness of system	Both	HR
Performance efficiency	Time behaviour	PTb-1-G	Mean response time	Both	HR
		PTb-2-G	Response time adequacy	Both	R
		PTb-3-G	Mean turnaround time	Both	R
		PTb-4-G	Turnaround time adequacy	Both	R
		PTb-5-G	Mean throughput	Both	R
	Resource utilization	PRu-1-G	Mean processor utilization	External	HR
		PRu-2-G	Mean memory utilization	External	R
		PRu-3-G	Mean I/O devices utilization	External	R
		PRu-4-S	Bandwidth utilization	External	UD
	Capacity	PCa-1-G	Transaction processing capacity	Both	R
		PCa-2-G	User access capacity	Both	R
		PCa-3-S	User access increase adequacy	External	UD
Compatibility	Co-existence	CCo-1-G	Co-existence with other products	External	HR
	Interoperability	CIn-1-G	Data formats exchangeability	Both	HR
		CIn-2-G	Data exchange protocol sufficiency	Both	R
		CIn-3-S	External interface adequacy	Both	HR

Table A.1 *(continued)*

Quality characteristic	Quality subcharacteristic	ID	Quality measure name	Internal/External/Both	Recommendation level
Usability	Appropriateness recognizability	UAp-1-G	Description completeness	Both	HR
		UAp-2-S	Demonstration coverage	Both	UD
		UAp-3-S	Entry point self-descriptiveness	Both	UD
	Learnability	ULe-1-G	User guidance completeness	Both	HR
		ULe-2-S	Entry fields defaults	Both	R
		ULe-3-S	Error message understandability	Both	R
		ULe-4-S	Self-explanatory user interface	Both	UD
	Operability	UOp-1-G	Operational consistency	Both	HR
		UOp-2-G	Message clarity	Both	R
		UOp-3-S	Functional customizability	Both	UD
		UOp-4-S	User interface customizability	Both	UD
		UOp-5-S	Monitoring capability	Both	UD
		UOp-6-S	Undo capability	Both	R
		UOp-7-S	Understandable categorization of information	Both	R
		UOp-8-S	Appearance consistency	Both	UD
		UOp-9-S	Input device support	Both	UD
	User error protection	UEp-1-G	Avoidance of user operation error	Both	HR
		UEp-2-S	User entry error correction	Both	HR
		UEp-3-S	User error recoverability	Both	R
	User interface aesthetics	UIn-1-S	Appearance aesthetics of user interfaces	Both	UD
	Accessibility	UAc-1-G	Accessibility for users with disabilities	Both	R
		UAc-2-S	Supported languages adequacy	Both	UD

Table A.1 *(continued)*

Quality characteristic	Quality subcharacteristic	ID	Quality measure name	Internal/ External/Both	Recommenda-tion level
Reliability	Maturity	RMa-1-G	Fault correction	Both	HR
		RMa-2-G	Mean time between failures (MTBF)	External	HR
		RMa-3-S	Failure rate	External	R
		RMa-4-S	Test coverage	External	R
	Availability	RAv-1-G	System availability	External	HR
		RAv-2-G	Mean down time	External	R
	Fault tolerance	RFt-1-G	Failure avoidance	External	HR
		RFt-2-S	Redundancy of components	Both	R
		RFt-3-S	Mean fault notification time	External	UD
	Recoverability	RRe-1-G	Mean recovery time	External	HR
		RRe-2-S	Backup data completeness	Both	R
Security	Confidentiality	SCo-1-G	Access controllability	Both	HR
		SCo-2-G	Data encryption correctness	Both	R
		SCo-3-S	Strength of cryptographic algorithms	Both	UD
	Integrity	SIn-1-G	Data integrity	Both	HR
		SIn-2-G	Internal data corruption prevention	Both	R
		SIn-3-S	Buffer overflow prevention	Internal	UD
	Non-repudiation	SNo-1-G	Digital signature usage	Both	R
	Accountability	SAc-1-G	User audit trail completeness	Both	HR
		SAc-2-S	System log retention	Both	R
	Authenticity	SAu-1-G	Authentication mechanisms sufficiency	Both	HR
		SAu-2-S	Authentication rules conformity	Both	R

Table A.1 *(continued)*

Quality characteristic	Quality subcharacteristic	ID	Quality measure name	Internal/ External/Both	Recommenda-tion level
Maintainability	Modularity	MMo-1-G	Coupling of components	Both	R
		MMo-2-S	Cyclomatic complexity adequacy	Internal	UD
	Reusability	MRe-1-G	Reusability of assets	Both	HR
		MRe-2-S	Coding rules conformity	Internal	R
	Analysability	MAn-1-G	System log completeness	Both	HR
		MAn-2-S	Diagnosis function effectiveness	Both	R
		MAn-3-S	Diagnosis function sufficiency	Both	R
	Modifiability	MMd-1-G	Modification efficiency	Both	HR
		MMd-2-G	Modification correctness	Both	HR
		MMd-3-S	Modification capability	Both	UD
	Testability	MTe-1-G	Test function completeness	Both	R
		MTe-2-S	Autonomous testability	Both	UD
		MTe-3-S	Test restartability	Both	UD
Portability	Adaptability	PAd-1-G	Hardware environmental adaptability	External	HR
		PAd-2-G	System software environmental adaptability	External	HR
		PAd-3-S	Operational environment adaptability	External	UD
	Installability	PIn-1-G	Installation time efficiency	External	R
		PIn-2-G	Ease of installation	External	R
	Replaceability	PRe-1-G	Usage similarity	Both	HR
		PRe-2-S	Product quality equivalence	Both	R
		PRe-3-S	Functional inclusiveness	External	R
		PRe-4-S	Data reusability/import capability	External	UD

Annex B
(informative)

QMEs used to define product or system quality measures

Most of the QMEs used for the measurement function of various quality measures are already described in ISO/IEC 25021:2012, Annex A. If needed, a number of new QMEs can be defined or designed properly according to the procedure and table format provided in ISO/IEC 25021.

The following is a summary of general QMEs which are used frequently in the measurement functions of various quality measures.

NOTE For more information about QMEs, see the QME definitions in ISO/IEC 25021.

B.1 Number of functions

The count of all the functions that satisfy the condition given in the specific QME definitions.

NOTE The functions can be, for example, required, implemented, tested, essential, optional, or any combination of these and more.

B.2 Number of failures

The count of all failures which occur in a given time span and which also satisfy the condition given in the specific QME definitions.

Examples of QMEs: number of expected failures, number of detected failures, number of resolved failures, number of failures of a given severity level.

B.3 Number of faults

The count of software product faults detected (or estimated) in a given software product component and satisfy the condition given in the specific QME definitions, e.g., number of fault of a given category, number of faults of a given severity, number of faults successfully corrected, etc.

B.4 Product size

The count of software product components according to a desired criterion. This can be lines of code (LOC), function points, modules, classes, or visual structures such as diagrams or their parts.

NOTE Software product components can be counted only if some additional properties are satisfied, e.g. only executable lines of code, lines of code which also contain commenting, only comment lines, declaration or type casting, bracket/braces only, etc.

B.5 Duration

Refers to the interval between a starting time and an end time of any process described in the specific QME definitions (Duration = end time – start time). For example,

— execution time: refers to time measured internally by the computer clock, e.g. CPU time, I/O time, etc., or time measured via inserted code or software tools (e.g. test suites);

— observation time: refers to time measured externally by the observer, using an external clock, e.g. time to finish a transaction or user task;

— set-up time: a fixed time process or observation, but meaningful for a measure that is independent of action, e.g. required response time.

B.6 Number of test cases

Refers to the count of different test input data and scenarios that satisfy the condition given in the specific QME definitions, e.g. test cases designed, required, executed (successfully or failed), etc.

B.7 Number of restarts

Counts the number of attempts for a system to resume computation after a critical failure and satisfies the condition given in the specific QME definitions. It can be distinguished between system restart and recovery.

B.8 Number of I/O

The count of the number of I/O events that satisfy the condition given in the specific QME definitions. I/O events are distinguished from system messages to the observer.

— Interaction between observer and the system, e.g. a dialogue;

— transaction: the sequence of interactions between the observer and system which must be executed atomically to accomplish an operation, e.g. a wizard (with options).

B.9 Number of tasks

A task is the set or sequence of activities required to achieve a given goal. The number of tasks is the count of the tasks that satisfy the condition given in the specific QME definitions. It can be distinguished between:

— user tasks: activities performed by the user (using software product) towards a specified goal;

— system tasks: activities performed by the system to support user.

B.10 Number of user attempts (trials)

The count of attempts to perform the same operation to satisfy the condition given in the specific QME definitions. These attempts can be:

— evaluation: iterations with same input and same scenario (e.g. stress testing);

— cases: iterations with different input and/or different scenarios.

B.11 Number of data items

The count of different structures, classes, or formats of data that satisfy the condition given in the specific QME definitions.

B.12 Number of records

The count of records of the same structure, class or format that satisfy the conditions given in the specific QME definitions.

B.13 Number of requirements

The count of requirement clauses that satisfy the condition given in the specific QME definitions.

NOTE The requirements can be i.e. essential, optional, validated, or any combination of these and more.

B.14 Number of user operations

The count of number of operations performed by the user that satisfy the condition given in the Specific QME definition, where an operation is a sequence of steps required to perform a task.

B.15 Number of system operations

The count of complete operations performed by the system that satisfy the condition given in the Specific QME definitions.

NOTE This QME counts the number of complete operations, not the individual steps required in each operation.

B.16 Number of languages

The count of different languages supported by the system or software product to be used to perform the intended user functions.

B.17 Number of software modules

The count of software components which works independently from another. Conceptually, modules represent a separation of concerns and improve maintainability by enforcing logical boundaries between components.

B.18 Number of interfaces

The count of shared boundaries across which two separate components of computer systems exchange information. The exchange can be between software, hardware, peripheral devices, human and combinations of these.

Annex C
(informative)

Detailed explanation of measurement types

C.1 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 quality measure.

NOTE For the most external measures, testing can be carried out to collect the input data for measurement function. The measurement types explained in this Annex are closely related to the test design techniques and the types of testing defined in ISO/IEC/IEEE 29119-4. The quality-related types of testing and mapping quality characteristics/subcharacteristics to types of testing are described in detail in ISO/IEC/IEEE 29119-4:2015, Annex A.

C.2 Size measure type

C.2.1 General

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

NOTE Software could 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.2.2 Functional size type

Functional size is an example of one type of size (one dimension) that a software could have. Any one instance of software can 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 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.2.3 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.

C.2.3.1 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 measure is commonly used.

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

NOTE 1 New program size: A developer can use newly developed program size to represent development and maintenance work product size.

NOTE 2 Changed program size: A developer can use changed program size to represent size of software containing modified components.

It might 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)

C.2.3.2 Program word count size

The measurement can be performed in the following manner using the Halstead’s measure:

Program vocabulary = n1 + n2; Observed program length = $N_1 + N_2$,

where

n1 is the number of distinct operators (i.e. 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 operands (i.e. the number of distinct operand words which are defined by the programmer in a program source code);

N_1 is the total number of operators (i.e. the number of occurrences of distinct operators in a program source code);

N_2 is the total number of operands (i.e. the number of occurrences of distinct operands in a program source code).

C.2.3.3 Number of modules

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

C.2.4 Utilized resource measure type

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

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

C.2.5 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 can differ depending on what kinds of description are used for measurement, such as a diagram or a text representing user operating procedures.

C.3 Time measure type

C.3.1 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.3.2 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.3.3 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 analysed 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.3.4 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 as follows:

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.3.5 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.3.6 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 can be used in place of this measure. This is typically used for describing the time between failures occurring successively.

C.4 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.4.1 Number of detected fault type

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

C.4.2 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.4.3 Number of detected inconsistency type

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

a) Number of failed conforming items

EXAMPLE

— 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.4.4 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.4.5 Number of detected failures type

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

C.4.6 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.4.7 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.4.8 Score type

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

Bibliography

[1] ISO/IEC 9241-11:1998, *Ergonomic requirements for office work with visual display terminals (VDTs) — Part 11: Guidance on usability*

[2] ISO/IEC 9241-110, *Ergonomics of human-system interaction — Part 110: Dialogue principles*

[3] ISO/IEC 9241-171, *Ergonomics of human-system interaction — Part 171: Guidance on software accessibility*

[4] ISO/IEC 14143, *Information technology — Software measurement — Functional size measurement*

[5] ISO/IEC 14143-1, *Information technology — Software measurement — Functional size measurement — Part 1: Definition of concepts*

[6] ISO/IEC 14756, *Information technology — Measurement and rating of performance of computer-based software systems*

[7] ISO/IEC 15939:2007, *Systems and software engineering — Measurement process*

[8] ISO/IEC 25012, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model*

[9] ISO/IEC 25020, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Measurement reference model and guide*

[10] ISO/IEC 25022, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of quality in use*

[11] ISO/IEC 25024, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of data quality*

[12] ISO/IEC 25030, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements*

[13] ISO/IEC/IEEE 15288, *Systems and software engineering — System life cycle processes*

[14] ISO/IEC/IEEE 24765:2010, *Systems and software engineering — Vocabulary*

[15] ISO/IEC/IEEE 29119-4:2015, *Software and systems engineering — Software testing — Part 4: Test techniques*

[16] Ministry of Economy, Trade and Industry, Japan (METI). Investigative Report on Measure for System/Software Product Quality Requirement Definition and Evaluation, 2011, http://www.meti.go.jp/policy/mono_info_service/joho/cloud/2011/11_05.pdf

[17] SAC. *Measurement of System and Software Product Quality, GB/T 29831-GB/T29836*. Standards Press of China, 2013

[18] U.S. Department of Health and Human Services. *The Research-Based Web Design & Usability Guidelines, Enlarged/Expanded edition*. U.S. Government Printing Office, Washington, 2006

[11] OMG. *CISQ Specification for Automated Quality Characteristic Measures, CISQ-TR-2012-01*, 2012

ICS 35.080

Price based on 45 pages