ISO/IEC JTC1/SC7/WG6 N-625R4(Louveciennes meeting)

Date: June 3rd, 2011

TITLE:	ISO/IEC DIS 25021 (for the next DIS ballot) Systems and software engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE) – Quality measure elements
DATE:	**June 3rd**, 2011
SOURCE:	JTC1/SC7/WG6
WORK ITEM:	Project
STATUS:	Version 4.2
DOCUMENT TYPE:	DIS (Draft International Standard)
ACTION:	DIS ballot
ISO/IEC 25000 SQuaRE Series PRIME EDITOR:	Prof. Motoei AZUMA Department of Industrial and Management Systems Eng. Waseda University 3-4-1, Okubo, Shinjuku-ku, Tokyo 169-8555, Japan az-mo@mtd.biglobe.ne.jp
PROJECT EDITOR:	Dr. Jean-Marc Desharnais (Canada)
CO-EDITORS:	Dr. Keum-Suk Lee (Korea) Mr. Atsushi Yamada (Japan)

ISO/IEC JTC1/SC7

Software Engineering

Secretariat: CANADA (SCC)

Reference number of working document: **ISO/JTC 1/SC 7 N**

Date: 2011-06-03

Reference number of document: **ISO/IEC FCD 25021**

Committee identification: ISO/JTC 1/SC 7/WG 6

Secretariat: SC7

Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality measure elements

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Contents

Page

Foreword

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

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

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

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

ISO/IEC TR 25021 was published by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, Software and System Engineering. This document is a revision of the TR 25021 and a propose approach to prepare an IS

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

- ISO/IEC 2500n - Quality Management Division,
- ISO/IEC 2501n - Quality Model Division,
- ISO/IEC 2502n - Quality Measurement Division,
- ISO/IEC 2503n - Quality Requirements Division, and
- ISO/IEC 2504n - Quality Evaluation Division.

Introduction

The purpose of this document is to define and/or design an initial set of Quality Measure Elements (QME) to be used throughout the product life cycle for the purpose of Systems and Software Product Quality Requirements and Evaluation (SQuaRE). The document also gives a set of rules to design a QME or verify the design of an existing QME. The content of this document constitutes the link between the ISO/IEC 9126 series of standards and the subsequent SQuaRE series of standards.

A number of QMEs for quality measures that quantify some of the characteristic and sub-characteristic represent an initial list, which is to be used during the construction of the quality measures as referenced in ISO/IEC TR 9126-2, ISO/IEC TR 9126-3 and ISO/IEC TR 9126-4. Quality measures presented in the SQuaRE series (Figure 1, 2) were extracted from ISO/IEC TR 9126 series but it is not the only source. When evaluating selected quality measures, the user should first understand the definition of each property related to a QME used within the selected quality measures. .

The main purpose of defining and using the quality measures elements in this document are:

- To provide guidance for organizations developing and implementing their own QMEs,
- To promote the consistent use of specific QME for measuring the product.,
- To help identifying the set of QMEs that are uniquely required to derive all the quality measures for a given characteristics or a sub-characteristics of a product.

A QME is a common component of a number of quality measures. The intended usage of this document is that users will be able to select measures from valid QMEs to define internal quality measures, external quality measures, data quality measures or quality-in-use measures. Then, these can be used for quality requirements definition, products evaluation and quality assessment but not necessary limited to those. It is therefore recommended to use this document prior or together with the ISO/IEC 2502n series of standards.

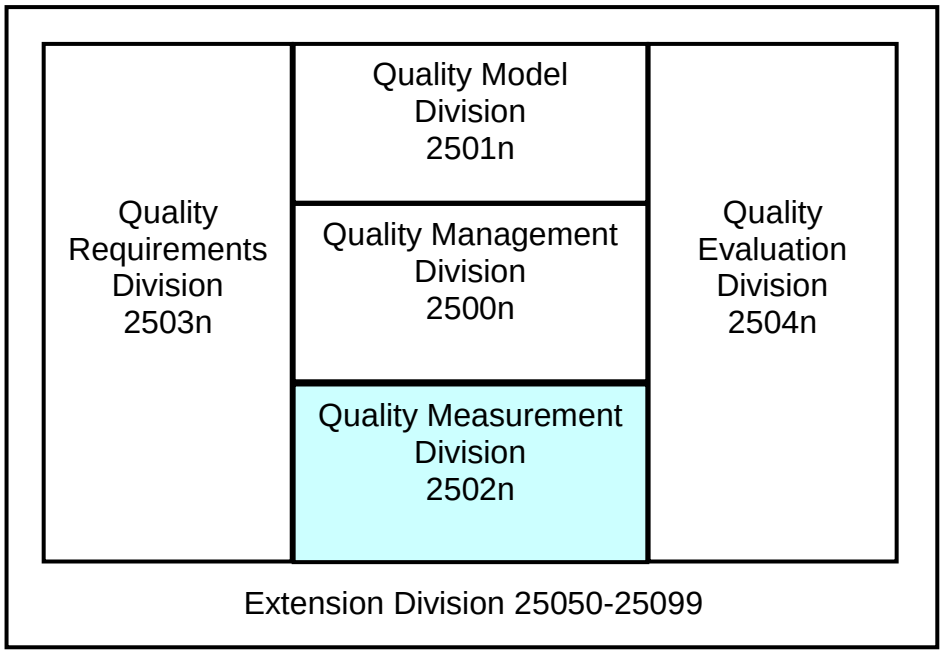


Figure 1 - Organisation of the SQuaRE series of international standards

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

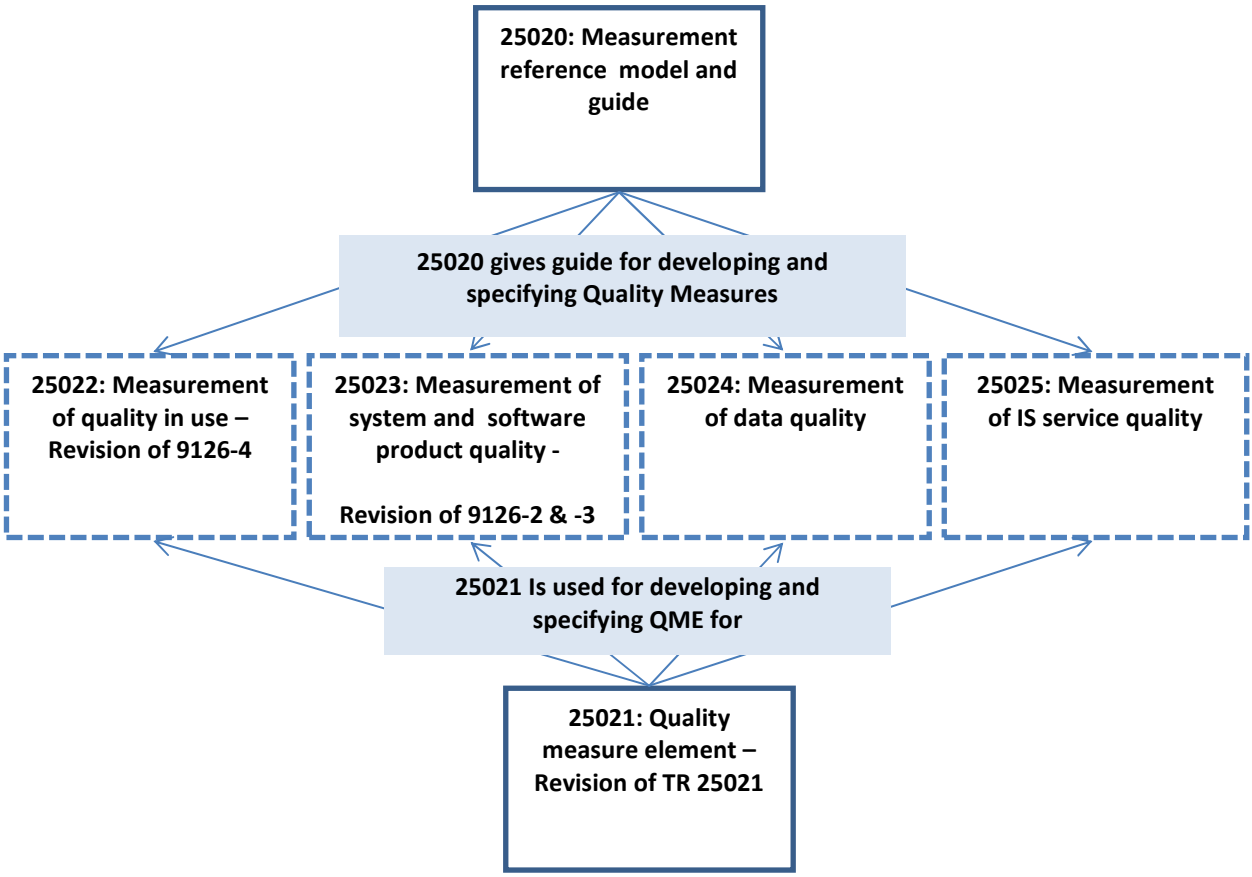


Figure 2 - Structure of the Quality Measurement Division

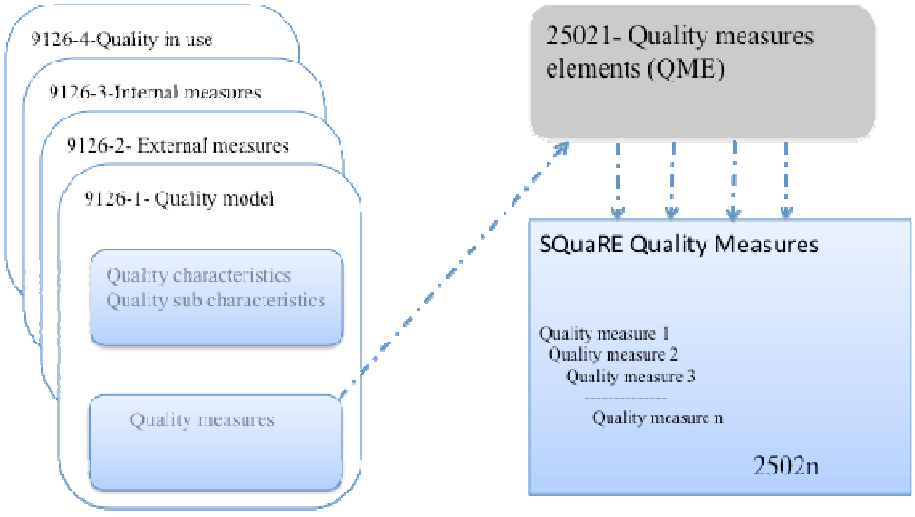


Figure 3 - The relationship of ISO/IEC 25021 as a link between the 9126 series and the SQuaRE series of standard

ISO/IEC 9126 series of international standard is composed of four documents that list and describe a set of quality characteristics, sub-characteristics and quality metrics (measures) that refer as the quality model. ISO/IEC 25010 SQuaRE quality models categorize product quality into characteristics which are

further subdivided into sub-characteristics and quality properties. For each quality measure within ISO/IEC 9126 series there are at least two QMEs. The properties (of a product) are linked to the QME (ISO/IEC 25020), using a measurement method. The 2502n series design and describe QMEs and quality measures for the quality model.

Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality measure elements

1 Scope

This International Standard contains the following information:

- Provide requirements for defining QMEs based on the product quality requirements with examples (tables 1 and 2 in clause 6.2);

NOTE: Product quality includes system quality, software product quality, data quality and eventually system service quality

- Provide an initial set of QMEs (table A.1 in Annex A)

- Provide a guideline for applying the measurement method to define and quantify the property of the product (target entity) for QMEs (Annex B)

This document is intended for, but not limited to, developers, acquirers and independent evaluators of product, particularly those responsible for defining product quality requirements and for product evaluation.

2 Conformance

When users define QMs for a product, they shall specify each QME according to the measurement method specified in table 1 (clause 6.1). The same should be applied for modifying a QME. The user of this standard should be adopted QMEs based on the product quality requirements and evaluation criteria.

3 Normative references

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

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

ISO/IEC 25010 *Systems and software engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models*

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

ISO/IEC 15939: 2007, Systems and software engineering — Measurement process

ISO/IEC *TR 9126-2, Software engineering — Product quality — Part 2: External metrics [Technical Report]*

ISO/IEC *TR 9126-3, Software engineering — Product quality — Part 3: Internal metrics [Technical Report]*

ISO/IEC *TR 9126-4, Software engineering — Product quality — Part 4: Quality in use metrics [Technical Report]*

Basic and General Terms in Metrology (VIM), International Organisation for Standardization, Geneva, Switzerland, 1993.

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 25000, ISO/IEC 25010, ISO/IEC 25020, ISO/IEC 15939 and International Vocabulary of Basic and General Terms in Metrology are applied. The following definitions are replicated here for the convenience of the user of this standard. Unattributed references are from ISO/IEC 25000.

4.1
data quality

degree to which the characteristics of data satisfy stated and implied needs when used under specified conditions

[ISO/IEC 25012:2008]

4.2
external measure of software quality

measure of the degree to which a software product enables the behavior of a system to satisfy stated and implied needs for the system including the software to be used under specified conditions

NOTE1 The behavior can be verified and/or validated by executing the software product during testing and operation.

NOTE2 Based on the ISO/IEC 25000:2005 definition of external software quality.

NOTE3 This definition was adapted from ISO/IEC 25010:2011.

4.3
internal measure of software quality

measure of the degree to which a set of static properties of a software product satisfies stated and implied needs for the software product to be used under specified conditions

NOTE1 Static properties include those that relate to the software architecture, structure and its components.

NOTE2 Static properties can be verified by review, inspection, simulation and/or automated tools.

NOTE3 This definition was adapted from ISO/IEC 25010:2011.

EXAMPLE Depending of the context specifications faults, design faults and code faults could be used as internal quality measures.

NOTE4 Based on the ISO/IEC 25000:2005 definition of internal software quality.

4.4
measure (noun)

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

NOTE The term “measures” is used to refer collectively to base measures, measures, and indicators.

[ISO/IEC 15939:2007]

4.5
measure (verb)

make a measurement

[ISO/IEC 25000].

4.6
measurement

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

[ISO/IEC 15939:2007, based on the definition in International Vocabulary of Basic and General Terms in Metrology, 1993].

NOTE Measurement can be nominal, ordinal, interval, ratio and absolute scale type.

4.7
measurement function

algorithm or calculation performed to combine two or more quality measure elements

4.8
measurement method

logical organization of operations, described generically, used in measurement

NOTE this definition is modified from ISO/IEC15939:2007 definition of measurement method.

4.9
measurement procedure

logical organization of operations, applied specifically, used in the performance of particular measurements according to a given measurement method

NOTE1 this definition is modified from ISO/IEC15939:2007 definition of measurement procedure.

NOTE2 A measurement procedure is usually recorded in a document that is sometimes itself called a "measurement procedure" and is usually in sufficient detail to enable an operator to carry out a measurement without additional information.

4.10
model

specification of the concepts, relationships and rules that are used to define a methodology

[ISO/IEC 24744:2007 Software Engineering — Model for Development Methodologies]

4.11
precision

the degree of exactness or discrimination with which a quantity is stated

[ISO/IEC 24765, Systems and Software Engineering Vocabulary]

4.12
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

NOTE1 a software artifact is an example of a target entity.

NOTE2 a sub-property is related to a property.

4.13
quality in use measure

measure of the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk, satisfaction and context coverage in specific contexts of use

NOTE2 Based on the ISO/IEC 25010:2011 definition of quality in use.

4.14
quality measure

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

4.15
quality measure element

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

4.16
repeatability (of results of measurement)

closeness of the agreement between the results of successive measurements of the same measurand carried out under the same conditions of measurement

[ISO/IEC TR 14143-3:2003]

4.17
reproducibility (of results of measurement)

closeness of the agreement between the results of measurements of the same measurand carried out under changed conditions of measurement

[ISO/IEC TR 14143-3:2003]

NOTE Repeatability and reproducibility may be expressed quantitatively in terms of the dispersion characteristics of the results.

4.18
target entity

fundamental thing of relevance to the user, about which information is kept, and need to be measured

4.19
unit (of measure)

a particular quantity defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity

NOTE1 Only quantities expressed in the same units of measurement are directly comparable. Examples of units include the number of faults and the number of failures. Hour and meter are also unit of measure.

NOTE2 Units of measurement have conventionally assigned names and symbols.

NOTE3 Based on the ISO/IEC 25000:2005 definition of unit of measurement.

5 Abbreviated terms

For the purposes of this Report, the symbols and abbreviations given in ISO/IEC 25000, ISO/IEC 25020 and the following apply:

a) QME - Quality measure element;

b) QM - Quality measure;

c) SPQM-RM - Systems and Software Product Quality Measurement Reference Model.

6 Quality measure elements concept

6.1 Presentation of the measurement method model

In order to understand and indicate quality (sub) characteristics, QM is defined. Then QMEs are defined based on the QM.

A Measurement function is applied to QME to generate QM. A Measurement method shall be applied to a property to define and identify a way to quantify QME.

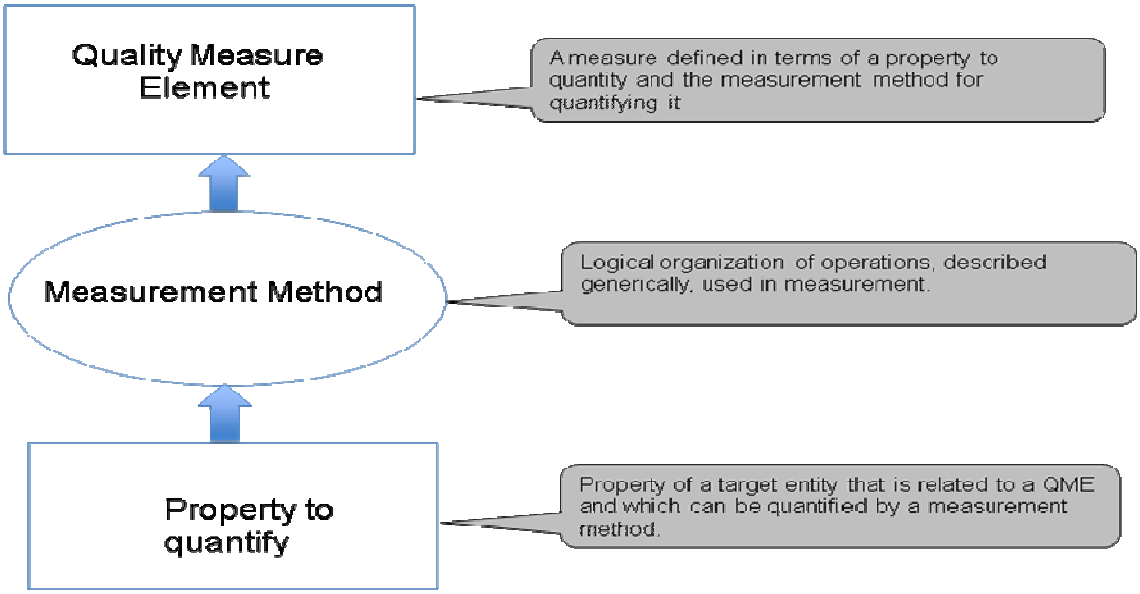


Figure 4 – Relationship between property to quantify, measurement method and QME

The user of the measurement method shall identify and collect data related to the quantification of property (Figure 4). Depending of the context usage and objective(s) of the QME, a number of properties and sub-properties can be identified. These are the input of the measurement method. Those properties are extracted and defined from the artifacts of the software (ex: documentation, code). The user of the measurement method produces different outputs that are the identification of the properties to finally determine the numerical assignment rules.

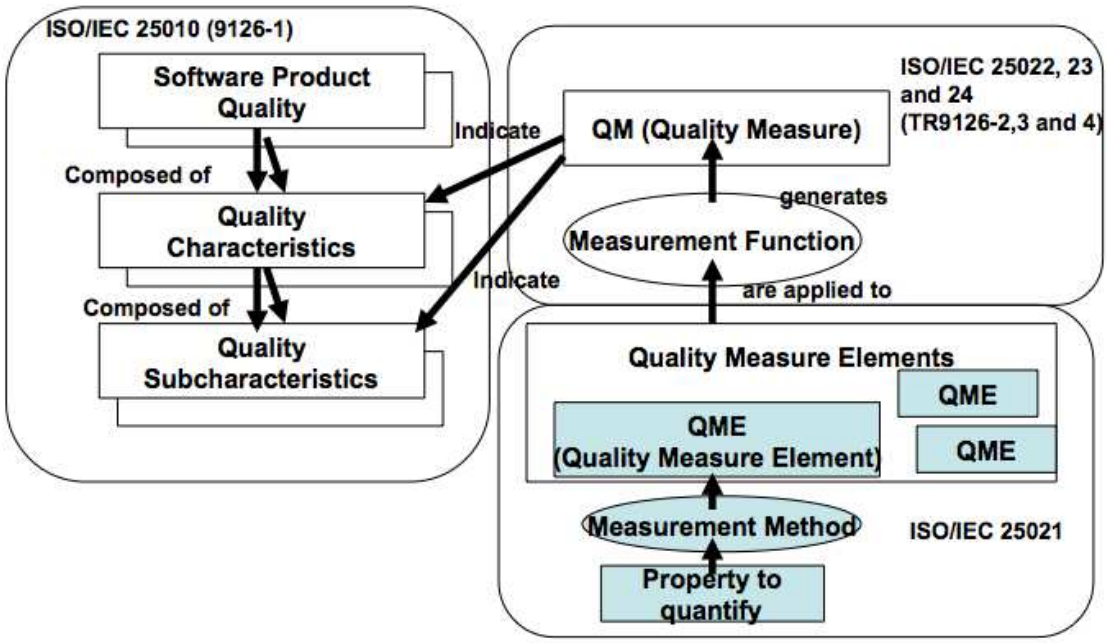


Figure 5 – Relationship between property to quantify, measurement method, QME and QM

Figure 5 shows that

a) product quality is composed of quality characteristics which in turn may be composed of sub-characteristics;

b) product quality measures are used to indicate the quality characteristics and sub-characteristics of interest;

c) the relationship between property to quantify, measurement method and QME.

NOTE Relations between QMs and QMEs can be represented in the form of a two-dimensional cross-reference table (annex C). This Figure 5 is based on SPQM-RM defined in ISO/IEC 25020.

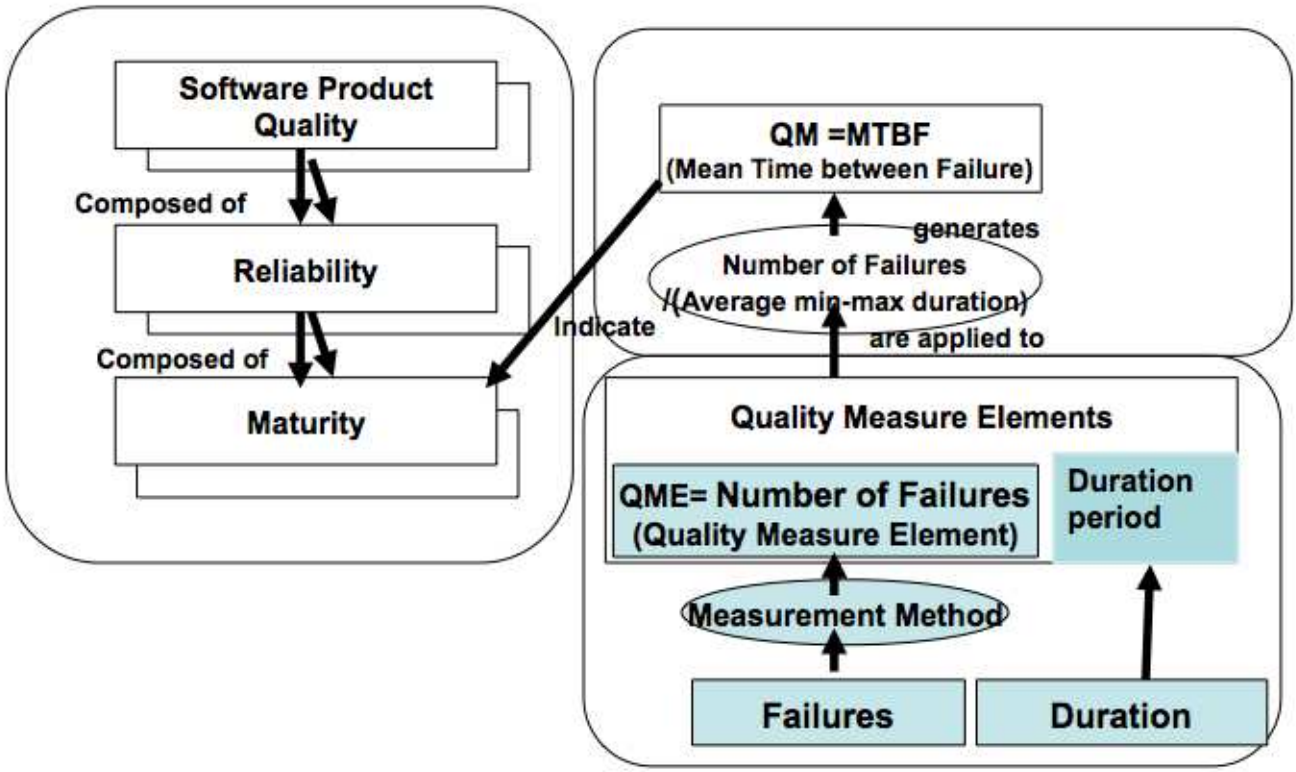


Figure 6 – An Example of Relationship among property to quantify, measurement method, QME and QM

Figure 6 shows an example that a QME is derived by applying a measurement method to a property to quantify.

Table 1 shows the measurement information items for QME which shall be used to describe QME.

NOTE1 a QME can be identified when quality characteristic or sub characteristic are selected and/or QM is defined to indicate it. The same QME may be used by some different QMs.

NOTE 2 Guide for designing a QME is provided in Annex B.

6.2 Tables presentation

The table format of the measurement method (table 1) shall be used to define and quantify a QME.

Table 1 — Table format of measurement method for QME

NOTE Each Item of the below table is organized into four groups such that “a)” for QME identification, “b) – d)” for what QME is, “e) – k)” for how to measure QME, and “l) – n)” for management of QME application.

a) QME Name	A QME shall have a unique name and should be identified with a serial number, if necessary. Most of the time it begins as "number of… (ratio scale)”.
b) Target entity	A QME shall have a target object that is to be characterized by measuring its property Target entity should be work products to be developed or behaviors of system, software, or stakeholders such as users, operators, developers, testers, or maintainers.
c) Objectives and property to quantify	Identification of the property to quantify is usually related to the name of the QME. (A list of QMEs is provided in Table A.1). Property of a target entity for QME (Example: QME = number of faults, target property = faults). Selected property to quantify should be the one which is most relevant to the measurement of information needed. A given property may be incorporated in multiple measurement constructs. A measurement method results in only one measurement construct. Ex: “number of software faults” is the QME and “fault” is the property of the software to quantify. Objective of QME should be specified to describe such as the followings: - what is intended to know by making definition of this QME; - what is information needed that is expected to be represented by this QME. Examples: For the identified property to quantify, what need to be measured shall be determined (ex: Line of Code, number of defects, duration). It is helpful to describe what kinds of component or event in the designated target entity are needed to be identified and quantify. For examples: a) Lines, functions, paths or tokens having specified feature in program source code may be identified and quantify; b) Events every when the software under testing fails to specified test cases may be identified and quantify; c) Events every when the user of the system fails to user’s intended tasks may be identified and quantify.
d) Relevant Quality measure(s)	Reference to specific quality measure(s) which use this QME shall be specified. NOTE For example, quality measures in the ISO/IEC 9126 series, 25000 SQuaRE series and other documents.

e) Measurement method	Measurement method gives how to collect and how to transform to the value quantifying property by a mathematical function. The following information: context of QME, Software Life Cycle process, measurement constraints and numerical rules are parts of the measurement method. The measurer can give optionally a name to the measurement method to facilitate the distinction between QME name, property to quantify name and Measurement method. For example, the functional measurement methods had the following name: IFPUG FPA, COSMIC, Mark II, etc.
f) List of sub properties related to the property to quantify (optional)	An identified property to quantify can be related to some different sub-properties, if necessary. This relation between properties should be expressed as a schema or a formula. This constitutes the measurement method model. For example, within the COSMIC method, a functional process is one property that can be expressed in a model with some sub-properties like entry, read, write and exit. This can help to identify the property to quantify of "data movement" which is relevant to functional size based measure.
g) Definition of each sub property (optional)	If there is a list of sub-properties, each sub property should be defined.
h) Input for the QME	The input shall be described in enough detail to identify what quantitative information is used to measure the QME. Any of sources providing the input also should be identified such as the documented work products, behaviors of system and software, or human behaviors of users, operators, developers, testers, or maintainers. Then, the input may be sub-properties or quantitative information relating to them. For example, the measurer could identify in a data model the information to retrace the entity of a read type (data movement) in COSMIC function point.
i) Unit of measurement for the QME	The unit of measurement and, if appropriate, the formula used. Examples of units include number of X, percentage and rank.
j) Numerical rules	A numerical assignment rule shall be described from a practitioner view (generally a text form) or from a theoretical point of view (generally a mathematical expression). The internal consistency is often a problem when assigning a numerical rule. It is important to have consistency between property and sub-properties that need to be measured. For this reason it is important to demonstrate that when adding two entities they are related by a common property. For example, measuring faults will give the number of faults. But if there is a distinction between major and minor faults, a more precise measure will be obtained by adding separately the major and minor faults. The interpretation must consider the limit of the result apply to each property and sub properties.
k) Scale type	Scale type shall be identified. Scale type could be nominal, ordinal, interval or ratio (annex E)

l) Context of QME	It gives information about the intended use of the measurement results. This is helpful to describe typical examples of quality characteristics, quality sub characteristics or quality measures (QM) which are mainly intended to use of the QME measurement results. NOTE The QME is able to be employed by any of quality measure (QM) to measure any of quality (sub) characteristics. Then, they are not limited to appearing ones in this table. Assumptions and prerequisite conditions of target entities, its environments and circumstances to which the measurement method of QME is enable to be applied shall be described.
m) Software Life Cycle process(es)	The typical appropriate life cycle process(es) should be identified here for each QME, which creates or realizes specific target entity enabling to obtain actual measured value of QME. NOTE 1 In some cases, the estimation may be available in some of life cycle processes based on historical data before actual measurement of QME. However, life cycle process(es) is expected to be identified to obtain actual measured results of QME here. Related life cycle process(es) after obtaining actual data, additional actual measurement or use of the measured results are available. For example, the number of faults in code is actually measurable by applying code review, code analyzing tools or unit testing during construction process (coding and unit testing). Also the number of faults in code is also able to be additionally measured when code is corrected to resolve failures in integration or qualification testing processes. Besides, the number of faults in code may be estimated through estimated coding size from page volume of requirements specifications based on historical data. NOTE 2 the basic software life cycle process(es) in ISO/IEC 12207:2008 include such requirements analysis, architectural design, detailed design, software construction, Integration, qualification testing, maintenance and so on (ref. ISO/IEC 12207:2008, Systems and Software Engineering - Software Life Cycle Processes). NOTE 3 if the methodology in use does not include any life cycle process(es) described in ISO/IEC 12207 then the measurer shall also mention the methodology and the particular process(es) that will be used. NOTE 4 The followings are typical example life cycle process(es) in ISO/IEC 12207 and 15288. a) STAKEHOLDER REQUIREMENTS DEFINITION b) SYSTEM/ SOFTWARE REQUIREMENTS ANALYSIS c) SYSTEM/ SOFTWARE ARCHITECTURAL DESIGN d) SOFTWARE DETAILED DESIGN e) SOFTWARE CONSTRUCTION (coding and unit testing) f) SYSTEM/ SOFTWARE INTEGRATION g) SYSTEM/ SOFTWARE QUALIFICATION TESTING h) SYSTEM/ SOFTWARE OPERATION i) SYSTEM/ SOFTWARE MAINTENANCE

n) Measurement Constraints **(optional)**	Any constraints related to the measurement method should be described, if necessary. The QME may have measurement constraints such measurement errors or fluctuations due to dependency of the followings: scope of investigation, way of investigation, volatile of specification or tactics of test cases. NOTE 1 For example, the number of faults in code may be different frequency between newly developed code in scope and scoping reused code in scope. Each of various ways of code investigation gives different number of faults in code such review meeting, walk-through meeting, inspection, individual inspection by expertise, pair programming, code analysis tools, unit testing, causal analysis of failure in integration testing etc. NOTE 2 For example, in case of counting the number of specification defects, the specification document should be available and not much volatile.

The following table is an example on how to complete the Table 1.

Table 2 — EXAMPLE: measurement method for fault (code)

QME Name	Number of faults (code)
Target entity	Program source code
Objectives and property to quantify	The objective is to find fault at coding phase using the design specifications. What need to be measured are erroneous lines of code. Fault is the property to quantify. Definition of fault:(1) a manifestation of an error in software (ISO/IEC 24765:2009 Systems and software engineering vocabulary) (2) an incorrect step, process, or data definition in a computer program (ISO/IEC 24765:2009 Systems and software engineering vocabulary) Note: A fault, if encountered, may cause a failure. NOTE1 By definition it is an accidental condition that causes a functional unit to fail to perform its required function. IEEE 982.1-1988 IEEE Standard Dictionary of Measures to Produce Reliable Software. It is also a manifestation of an error in software.
Relevant Quality measure(s)	To measure the reliability of the software by utilizing fault density • Finding fault detection rate during coding phase • Finding fault removal rate with corrected faults in code phase Maturity (sub-characteristic) and Reliability (characteristic) level of the software
Measurement method	Software Fault Measurement Method in the code Review or analyze differences of the revised program source code and identify corrected lines of code which consist of modified lines, added lines and deleted lines of code. NOTE Program source code is usually revised as the results of verification and validation activities such code review, unit testing, causal analysis to resolve failures in integration testing.

List of sub properties related to the property to quantify (optional)	List: executable statements, erroneous line of code, corrected line of code.
Definition of each sub property (optional)	Executable Statements: The statements which can be categorized to labeled statements, expressions, selection statements, iteration statements and jump statements. Non-Executable Statements: The statements that can be categorized to declarations and declaration specifiers. Erroneous Line of Code : Line of code which contains fault(s). Specifications should tell if the source code is erroneous or not. Correct Line of Code: Line of code with no fault. NOTE it is also possible that the actual lines of code are correct and the specifications should be change. This should not be counted as erroneous line of code.
Input for the QME	Source code and design specifications of source code
Unit of measurement for the QME	Erroneous line of code
Numerical rules	Adding total erroneous lines of code A numerical assignment rule from a practitioner view use the following measurement actions: a) Review or analyze differences of the revised program source code and identify corrected lines of code which consist of modified lines, added lines and deleted lines of code NOTE Program source code is usually revised as the results of verification and validation activities such code review, unit testing, causal analysis to resolve failures in integration testing.
Scale type	Ratio
Context of QME	This QME is mainly chosen to measure the Maturity (sub-characteristic) and Reliability (characteristic) level of the software
Software Life Cycle process(es)	SOFTWARE CONSTRUCTION (coding and unit testing), Coding phase
Measurement Constraints (optional)	Design specifications must be available to be able to compare the actual lines of code with the design specifications.

Annex A
(informative)
Initial set of QMEs

Various QMEs can be used together and combined to define QMs. Some QME are from the ISO/IEC 9126s and others are from industry market needs and existing standard such as functional size measurement. QMEs listed in this initial set are related to quality (sub) characteristics of product quality model defined in ISO/IEC 25010. The initial set of QMEs is suggested for users of this document to consider its applicability, when one is preparing QM for evaluation of product quality.

.

Table A.1 — List of Initial set of QMEs and concept to quantify and high level definition

NO	Initial set of QME	Definition and concepts related directly to QME
1	Number of access to help function	Accessibility: usability of a product, service, environment or facility by people with the widest range of capabilities. ISO/IEC 25062:2006, Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for usability test reports.4.1; ISO/IEC 26514, Systems and software engineering--requirements for designers and developers of user documentation.4.1 NOTE Although "accessibility" typically addresses users who have disabilities, the concept is not limited to disability issues. In this case it is the accessibility of the help function.
2	Number of user problems	Each time a user address a complaint about a product, it is registered by the organization (normally at the helpdesk level). Knowing the complaint could help to measure the user satisfaction over time period. For example, technical problems or functional problems are extracted from user complaints and sorted out by help desk. Problem: unknown underlying cause of one or more incidents (ISO/IEC 20000-1:2005 Information technology -- Service management -- Part 1: Specification, 2. a negative situation to overcome. ISO/IEC 24765:2009 Systems and software engineering vocabulary Note: A risk factor becomes a problem when a risk metric (an objective measure) crosses a predetermined threshold (the problem trigger). User: 1. person who performs one or more tasks with software; a member of a specific audience. ISO/IEC 26514:2008 Systems and software engineering--requirements for designers and developers of user documentation 2. person who interacts with the product. ISO/IEC 25062:2006 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Common Industry Format (CIF) for usability test reports 3. individual or organization who uses a software-intensive system in daily work activities or recreational pursuits (IEEE 1362-1998 (R2007) IEEE Guide for Information Technology-System Definition -Concept of Operation Document, 4. the person (or persons) who operates or interacts directly with a software-intensive system (5) individual or group that benefits from a system during its utilization. ISO/IEC 15288:2008 Systems and software engineering--System life cycle processes, ISO/IEC 15939:2007 Systems and software engineering--Measurement process
3	Number of records	Record: a set of related data items treated as a unit. ISO/IEC 24765 :2009

NO	Initial set of QME	Definition and concepts related directly to QME
		Systems and software engineering vocabulary
4	Duration	Duration:The total number of work periods (not including holidays or other nonworking periods) required to complete a schedule activity or work breakdown structure component. Usually expressed as workdays or workweeks. Sometimes incorrectly equated with elapsed time. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fourth Edition
5	Effort (in unit of time)	Effort: The number of labor units required to complete a schedule activity or work breakdown structure component. Usually expressed as staff hours, staff days, or staff weeks. *A* Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fourth Edition
6	Number of system failures	System failure: A complete system includes all of the associated equipment, facilities, material; computer programs, firmware, technical documentation, services, and personnel required for operations and support to the degree necessary for self sufficient use in its intended environment previously specified limits. Software failure: Termination of the ability of a product to perform a required function or its inability to perform within previously specified limits.
7	Number of failures	Failure: 1. termination of the ability of a product to perform a required function or its inability to perform within previously specified limits. ISO/IEC 25000:2005, Software Engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE.4.20. 2. an event in which a system or system component does not perform a required function within specified limits NOTE A failure may be produced when a fault is encountered.
8-1	Number of faults (code)	Fault: an incorrect step, process, or data definition in software code. NOTE: A fault, if encountered, may cause a failure. Fault: 1 a manifestation of an error in software. ISO/IEC 24765:2009 Systems and software engineering vocabulary 2. an incorrect step, process, or data definition in a computer program. ISO/IEC 24765:2009 Systems and software engineering vocabulary 3. a defect in a hardware device or component. ISO/IEC 24765:2009 Systems and software engineering vocabulary Note: A fault, if encountered, may cause a failure.
8-2	Number of faults (design)	Fault: an incorrect step, process, or data definition in software design specifications.
8-3	Number of faults (requirements)	Fault: an incorrect step, process, or data definition in software requirements.
9	Functional size of the product	Functional size: a size of the software derived by quantifying the functional user requirements. ISO/IEC 14143-1:2007 Information technology--Software measurement--Functional size measurement; Part 1
10	Number of interruptions	Interrupt: the suspension of a process to handle an event external to the process. ISO/IEC 24765:2009 Systems and software engineering vocabulary
11	Number of data items	Data item: The smallest identifiable unit of data within a certain context for which the definition, identification, permissible values and other information is specified by means of a set of properties. Data: 1. a representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means. ISO/IEC 24765:2009 Systems and software engineering

NO	Initial set of QME	Definition and concepts related directly to QME
		vocabulary 2. collection of values assigned to base measures, derived measures and/or indicators. ISO/IEC 25000:2005 Software Engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, ISO/IEC 15939:2007 Systems and software engineering--Measurement process 3. the representation forms of information dealt with by information systems and users thereof. ISO/IEC 10746-2:1996 Information technology -- Open Distributed Processing -- Reference Model: Foundations 4. a reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or communication, or processing. ISO/IEC 2382-1:1993 Information technology--Vocabulary--Part 1: Fundamental terms
12	Number of error messages	Error message: a message that the application gives when incorrect data is entered or when another processing error occurs. ISO/IEC 24570:2005 Software engineering -- NESMA functional size measurement method version 2.1 -- Definitions and counting guidelines for the application of Function Point Analysis
13	Number of errors	Error: 1. a human action that produces an incorrect result, such as software containing a fault. ISO/IEC 24765:2009 Systems and software engineering vocabulary 2. an incorrect step, process, or data definition. ISO/IEC 24765:2009 Systems and software engineering vocabulary 3. an incorrect result. ISO/IEC 24765:2009 Systems and software engineering vocabulary 4. the difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. ISO/IEC 24765:2009 Systems and software engineering vocabulary Example: omission or misinterpretation of user requirements in a software specification, incorrect translation, or omission of a requirement in the design specification See Also: failure, defect
14	Number of messages	Message: Information given to an end user of a software system for informing, directing, warning purposes. Message: a communication sent from one object to another. IEEE 1320.2-1998 (R2004) IEEE Standard for Conceptual Modeling Language Syntax and Semantics for IDEF1X97 (IDEFobject) Note: Message encompasses requests to meet responsibilities as well as simple informative communications.
15	Number of steps (of procedure)	Step: 1. one element (numbered list item) in a procedure that tells a user to perform an action (or actions). ISO/IEC 26514 Systems and software engineering--requirements for designers and developers of user documentation.4.47. 2. the simultaneous occurrence of a finite multiset of transition modes that are concurrently enabled in a marking. ISO/IEC 15909-1:2004 Software and system engineering -- High-level Petri nets -- Part 1: Concepts, definitions and graphical notation.2.1.26.4. 3. an abstraction of an action, used in a process, that may leave unspecified objects that participate in that action. ISO/IEC 15414:2006 Information technology -- Open distributed processing -- Reference model -- Enterprise language.6.3.6. NOTE: A step contains one or more actions. Responses by the software are not considered to be steps.
16	Number of tasks	Task: 1. required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process. ISO/IEC 12207:2008 Systems and software engineering--Software life cycle processes.4.34; ISO/IEC 15288:2008 Systems and software engineering--

NO	Initial set of QME	Definition and concepts related directly to QME
		System life cycle processes.4.34. 2. in software design, a software component that can operate in parallel with other software components. 3. the activities required to achieve a goal. ISO/IEC TR 9126-4:2004 Software engineering -- Product quality -- Part 4: Quality in use metrics.4.3. 4. a concurrent object with its own thread of control. 5. a sequence of instructions treated as a basic unit of work by the supervisory program of an operating system. 6. smallest unit of work subject to management accountability; a well-defined work assignment for one or more project members. IEEE 829-2008 IEEE Standard for Software and System Test Documentation.3.1.38
17	Number of test cases	Test case: a minimal independently executable part of the test suite of a software system which gives 2 possible results [fail, pass]. Test case: 1. a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. IEEE 1012-2004 IEEE Standard for Software Verification and Validation 2. documentation specifying inputs, predicted results, and a set of execution conditions for a test item. IEEE 1012-2004 IEEE Standard for Software Verification and Validation
18	Number of use cases	Use Case is the description of the interaction between an Actor (the initiator of the interaction) and the system itself. It is represented as a sequence of simple steps. The property is the functional size, but from the use case perspective. Use case: in UML, a complete task of a system that provides a measurable result of value for an actor. ISO/IEC 24765:2009 Systems and software engineering vocabulary Note: More formally, a use case defines a set of use case instances or scenarios. Use case specification: a document that describes a use case. ISO/IEC 24765:2009 Systems and software engineering vocabulary Note: A use case specification's fundamental parts are the use case name, brief description, precondition, basic flow, postcondition, and alternate flow.
19	Number of operations	Ongoing execution of activities that produce the same product or provide a repetitive service. Operation: 1. the process of running a computer system in its intended environment to perform its intended functions. ISO/IEC 24765:2009 Systems and software engineering vocabulary 2. an action needed to perform an activity. ISO/IEC 15940:2006 Information Technology -- Software Engineering Environment Services
20	Number of fatal errors	Fatal error: An error that results in the complete inability of a system or component to function. ISO/IEC 24765:2009 Systems and software engineering vocabulary
21	Size of a database	Number of occurrences in a database. Database: 1. a collection of interrelated data stored together in one or more computerized files. ISO/IEC 24765:2009 Systems and software engineering vocabulary 2. a collection of data organized according to a conceptual structure describing the characteristics of the data and the relationships among their corresponding entities, supporting one or more application areas. ISO/IEC 2382-1:1993 Information technology--Vocabulary--Part 1: Fundamental terms 3. collection of data describing a specific target area that is used and

NO	Initial set of QME	Definition and concepts related directly to QME
		updated by one or more applications. ISO/IEC 29881:2008 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method
22	Size of a memory	Amount of computer or device of a computer storage (express in multiple of bytes). Memory: the addressable storage space in a processing unit and all other internal storage that is used to execute instructions. ISO/IEC 2382-1:1993 Information technology--Vocabulary--Part 1: Fundamental terms Memory capacity: the maximum number of items that can be held in a given computer memory; usually measured in words or bytes. ISO/IEC 24765:2009 Systems and software engineering vocabulary

NOTE 2 Tables A.2 a and b do not show complete mapping between QMEs and their usage of quality characteristics and sub-characteristics especially. This initial set can be changed over time.

The following tables define and quantify an initial set of QME using table 1. Annex B (informative) could be use in conjunction with table 1 to complete the definition and quantification of each QME or to help define new QME in your own organization.

Table A.2a — Mapping between some initial QMEs and their usage for Functional Suitability, Compatibility, Usability and Performance Efficiency of ISO/IEC 25010

Characteristics	**Functional Suitability**			**Compatibility**		**Usability**						**Performance efficiency**		
QME/subcharacteristics	Functional Completeness	Functional correctness	Functional appropriateness	Co-existence	Interoperability	Appropriateness recognisability	Learnability	Operability	User error protection	User interface aesthetics	Accessibility	Time behaviour	Resource utilisation	Capacity
1-Number of access to help function											X			
2-Number of user problems	X		X			X	X	X	X	X	X			
3-Number of records											X			X
4-Duration				X	X			X		X		X		
5-Effort							X	X					X	
6-Number of system failures							X	X	X					
7-Number of failures		X							X					
8-Number of faults									X					
9- Functional size of the product	X												X	
10-Number of interruptions	X													

Characteristics	Functional Suitability			Compatibility		Usability						Performance efficiency		
QME/subcharacteristics	Functional Completeness	Functional correctness	Functional appropriateness	Co-existence	Interoperability	Appropriateness recognisability	Learnability	Operability	User error protection	User interface aesthetics	Accessibility	Time behaviour	Resource utilisation	Capacity
11-Number of data items											X			
12-Number of error messages											X			X
13-Number of errors		X							X					
14-Number of messages											X			
15-Number of steps (of procedure)			X					X						
16-Number of tasks	X		X			X			X					
17-Number of test cases								X	X		X			
18-Number of use cases	X						X	X	X	X	X			
19-Number of operations	X						X	X	X	X	X			
20-Number of fatal errors		X								X				
21-Size of a database														X
22-Size of a memory														X

Table A.2b — Mapping between some initial QMEs and their usage for Reliability, Security, Maintainability and Portability of ISO/IEC 25010

Characteristics	Reliability				Security					Maintainability					Portability		
QME/subcharacteristics	Maturity	Availability	Fault tolerance	Recoverability	Confidentiality	Integrity	Non-repudiability	Accountability	Authenticity	Modularity	Reusability	Analysability	Modifiability	Testability	Adaptability	Installability	Replaceability
1-Number of access help functions																	
2-Number of user problems	X	X	X	X	X	X	X	X	X						X	X	X
3-Number of records				X	X	X	X	X	X						X		

Characteristics	Reliability				Security					Maintainability					Portability		
QME/subcharacteristics	Maturity	Availability	Fault tolerance	Recoverability	Confidentiality	Integrity	Non-repudiability	Accountability	Authenticity	Modularity	Reusability	Analysability	Modifiability	Testability	Adaptability	Installability	Replaceability
4-Duration												X	X	X			
5-Effort (in unit of time)												X	X	X			
6-Number of system failures												X	X	X			
7-Number of failures	X	X	X	X						X	X	X	X	X			
8-Number of faults	X	X	X	X						X	X	X	X	X			
9-Functional size of the product	X	X	X	X													
10-Number of interruptions														X			
11-Number of data items				X	X	X	X	X	X						X		
12-Number of error messages		X	X												X		
13-Number of errors	X	X	X	X													
14-Number of messages																	
15-Number of steps (of procedure)																	
16-Number of tasks															X		
17-Number of test cases														X			
18-Number of use cases			X	X	X	X	X	X	X								
19-Number of operations			X	X	X	X	X	X	X								
20-Number of fatal errors	X	X	X	X										X			
21-Size of a database																	
22-Size of a memory																	

Annex B
(informative)
Guide for Designing a Quality Measure Element (QME)

This document provides a procedure to apply the measurement method as recommend in ISO/IEC 15939. The measurement method generates a Quality Measure Element (QME) from an property to quantify. The intent is to assist users of the ISO/IEC 9126 series (ISO/IEC TR 9126-2, ISO/IEC TR 9126-3, ISO/IEC TR 9126-4) and users of SQuaRE series of quality measurement standards (ISO/IEC 25020, ISO/IEC 25022, ISO/IEC 25023 and ISO/IEC 25024) to apply the measurement method. This annex will be helpful when selecting and using different quality measures to evaluate the quality of the product within the product life cycle.

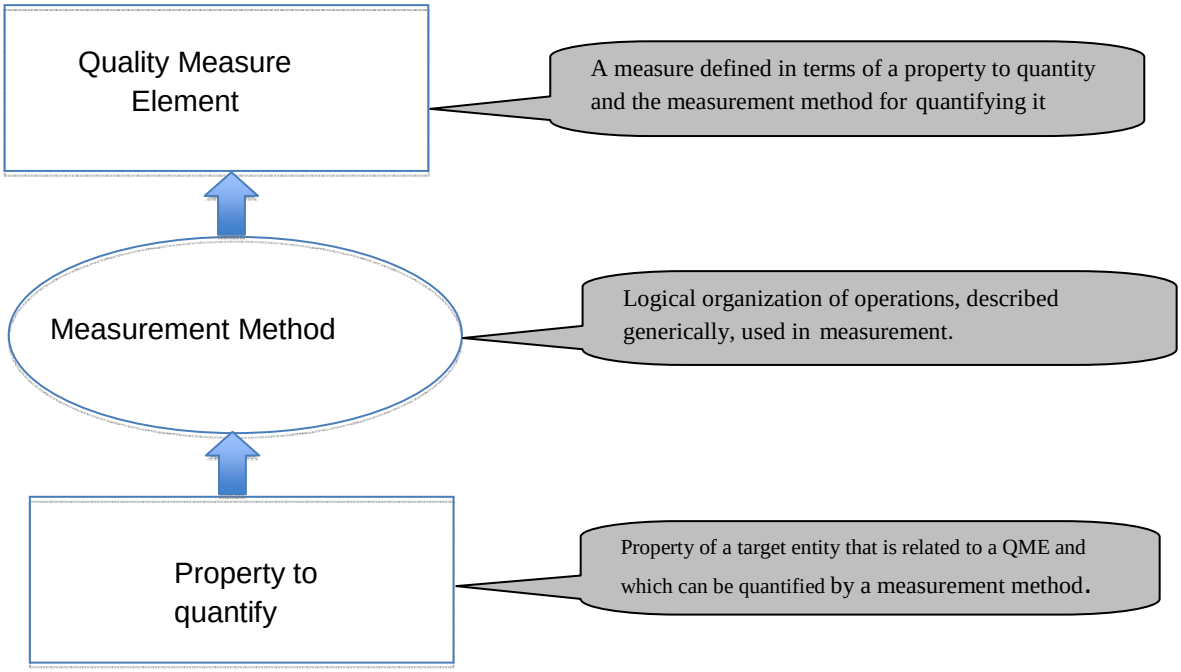


Figure B.1 – Relationship between property to quantify, measurement method and QME

As Figure B.1 suggest, to quantify the QME, the designer of the measurement method identify and collect data related to the quantification of property of a QME. Depending of the context usage and objective(s) of the QME, a number of sub-properties could be identified. These are the input of the measurement method. Those properties are extracted and defined from the artifacts of the software (ex: software life cycle). The designer of the measurement method produces different outputs that are the identification of the properties and sub-properties. He constructs the measurement principle and the description of its measurement method for the implementation of the numerical assignment rules.

This section describes the procedure (different steps) for designing the measurement method, from the identification of the QME through the numerical assignment (unit of measure). The following section will give examples of "resulting" QMEs using this measurement procedure.

Proposed steps to design QMEs:

1. identification of the QME and objectives (B.2)

2. identification of the property to quantify related to the QME (B.3)

3. definition of the property and sub-properties(B.4)

4. construction of the model of the properties to be quantified (B.5)

5. assignment of the unit of measurement (formula) and scale type (B..6)

Users of this document can consider each steps listed in this section to design, or verify the design, of a measurement method for a specific QME in order to avoid accidental misuse of the QME. In theory, a QME could be applied to any quality measure and at any stage during the whole product life cycle. In this document, the design process apply to implement a measurement method is independent of the QME and technology. However, considerations related to the objectives of the measurement can be defined when designing a specific QME. It is necessary because a specific QME is related to a quality measure that is related to a sub characteristic. This not excludes the usage of a QME for different characteristics and sub characteristics as long as the measurement objective does not change.

Note 1: All things being equal, the measurement results should be repeatable and reproducible across measurers, across groups measuring the same quality measures of the software and, as well, across organisations. The proposed steps to design QMEs should help to reach those objectives as much as the application of the measurement method.

B.1 Identification of the QMEs and objectives

A list of QME were extract from the quality measures (about 250) in the ISO/IEC 9126 series, part 2, 3 and 4. For each QME listed, an property (of a product) was identified. This work of identifying QME imply continuous update because it depends also of new identified characteristics and sub characteristics (ref: ISO/IEC 25010) that will bring also new "quality measures' and potential new QMEs with their respective properties. Finally, the identification of the QME in the context of his usage is important because it gives information about the objective of the measurement and the intended use of the measurement results. When a QME is identified it is possible to identify the permanent property use by the QME.

Consequently, the measurement design should contain the following descriptive information: name of the QME, name of the property to quantify, optional measurement method name, context of the measure (it is use by what quality measure), software life cycle, constraints, point of view, definition of the property to quantify and references, list of sub properties with references, relations between properties and sub properties, source of the information, input for the measure, unit of the QME, numerical rules for the QME and the scale type. About the objective, the designer should also clarify whether the measurement of the property will be performed from the user or developer point of view for example. Within ISO/IEC 9126 there are three points of view: internal (developer), external (user) and quality in use (when the software is use by the user). The document should clarify in which software development life cycle it is best to apply the measurement method. This will help to identify the properties of the property. The software life cycle phases may change based on the type of the software life cycle. In ISO/IEC 12207, basic life cycle phases are requirements analysis, design, coding, testing, and maintenance (ISO/IEC FCD 12207- 2006, Systems and Software Engineering - Software Life Cycle Processes).

Identification of the property to quantify related to the QME measurement method.

B.2 Identification of the property to quantify related to the QME

Software is an intangible product, but still, it can be made visible through multiple representations. A set of screens and reports for a user, a set of lines of code for a programmer, a set of software model representations for a software designer are good examples of elements of a software (IEEE 1233-1998). To quantify a property related to a QME the measurer can take in account the existence of those

elements. One main property to quantify is link to one QME. For example, the number of errors is the QME while the "error" is the main property to quantify.

For the organization, the choice of an property will be directly related to the choice of a QME. The choice of a QME should be related to the purpose of the measurement program in the organisation (What characteristics and sub characteristics the organization wants to quantify?). Each QME use by an organisation can be define by the organization if not already define within 25021

The table that provides the design of the measurement method for a specific QME should also provide the reference for definitions and designs when they exist. The reference could be from an existing standard or from publish authors. For each identified property, what need to be measured can be determined (ex: size, quality, etc.). An identified property is related to one or more sub-properties, which in turn could have two or more sub properties. The construction of a model is then necessary.

B.3 Definition of the property and sub-properties

The identified property to quantify of the QME can be decomposed into measurable sub-properties. For example the property to size a "USE CASE" can be decompose in three sub-properties "main scenario", "alternative paths" and "exceptions". The designer of the measurement method should make then a comprehensive literature review to find out how the properties related to the QME are defined and measured on previous researches. The designer should observe the similarities and the differences between the definition of the properties in the quality model and other bibliography references. This depends mainly of the objective and context usage of the QME (B.2). The results of the review should be compatible with the objective and usage of the QME.

For such properties, the characterization can be done by first stating implicitly how a property is decomposed into sub properties. This decomposition describes which role each sub property plays in the constitution of the property. Therefore, how properties are decomposed should be described in this step.

Here is an example of a decomposition of a property. The COSMIC method within ISO/IEC 19761 (COSMIC) is composed of the property of “data movement” and some others properties (layers, boundary and functional process) that help to understand and define “data movement". In this case we can also find the sub properties (of data movement) identify as entry type, exit type, read type and write type. Later, those sub properties will be useful to construct a model (B.5) and count the “data movement” to obtain the number (quantify) of CFP (B.6).

B.4 Construction of the model of the property to be quantified

The property to quantify of the QME is use to obtain the property(s) in the model. The relations between the defined property(s) or sub properties that represent a software or a part of a software constitute the model. The model describes how to recognize the property(s) and/or sub properties in the measurement method.

Sources of data used in the measurement method should be identified at this step. For example, the elements "requirement specification document", "test description document" and others, provide important information that help finding the measurable properties.

The input source of data, which is used to quantify the QME, should be identified. For example, the element could be a text from which a human should extract the necessary information to quantify the QME. The measurement method for the QME could involve a human judgment (example: counting manually the number of faults) or a tool (example: counting the number of failures following an automated test).

B.5 Assignment of the unit of measurement (formula) and scale type

The input source of data, which is used to quantify the QME, should be identified. For example, the element could be a text from which a human should extract the necessary information to quantify the QME. The measurement method for the QME could involve a human judgment (example: counting manually the number of faults) or a tool (example: counting the number of failures following an automated test).

Assignment of numerical rules is part of the design process. A numerical assignment rule can be described from a practitioner view (generally a text) or from a theoretical point of view (generally a mathematical expression). The internal consistency is often a problem when assigning a numerical rule. It is important to have consistency between two properties that need to be measured. For this reason, it is important to demonstrate that when adding the two properties (or sub properties) they are related by a common property. For example adding apples and oranges could be justify when considering the property of fruit, and the result is number of fruits. The interpretation must also consider the limit of the result. In the final result we know nothing about the number of apples and oranges, but only about the number of fruits.

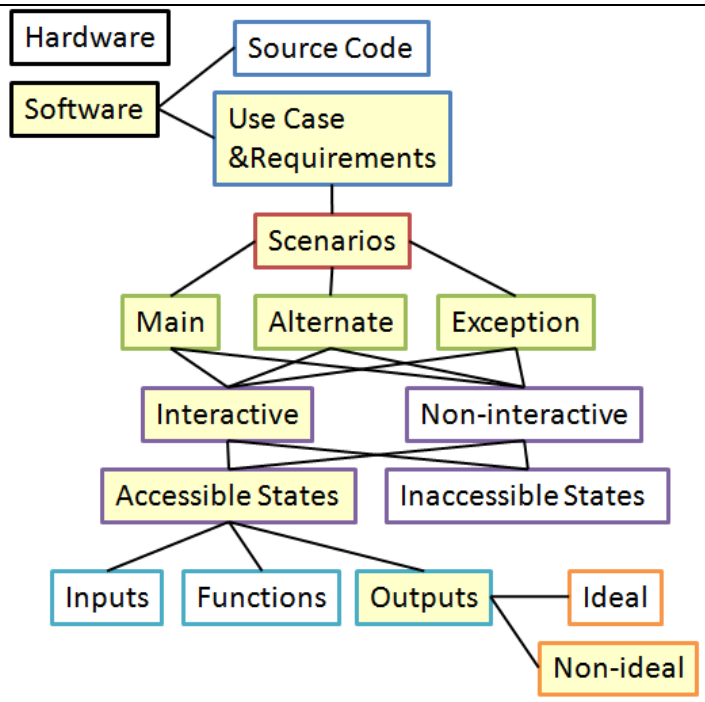
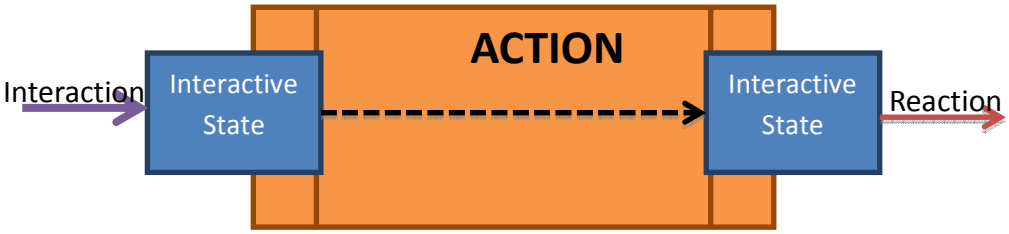
The interpretation is also related to the scale type (annex E) of the values and the mathematical relation between the values. When it is not done the interpretation could be erroneous. If the scale type is ordinal, the only interpretation is related to lower value or higher value relative to two results. User satisfaction of 3 is lower of user satisfaction of 5 assuming 1 is the lowest and 5 the highest.

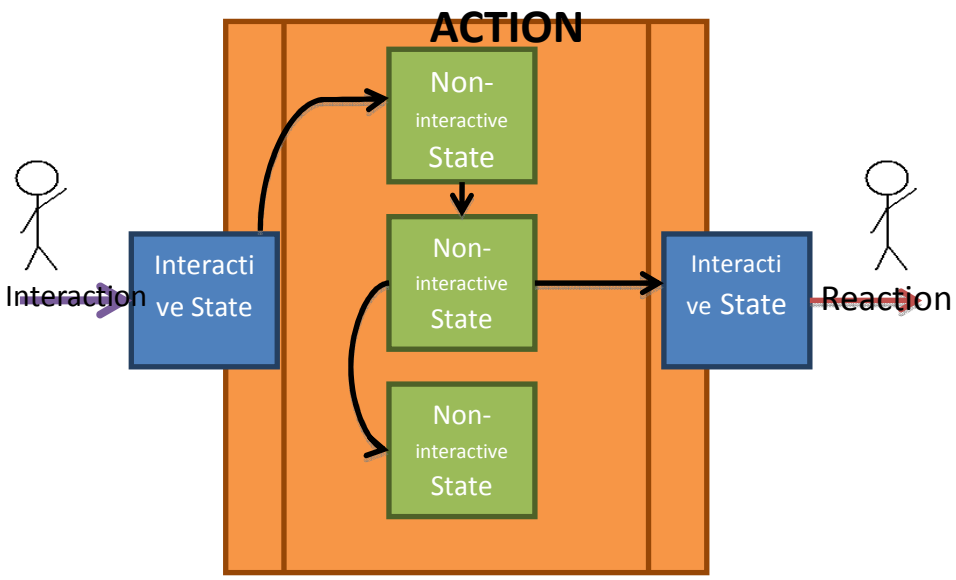
Annex C
(informative)
Examples of QME and proposed expansion

C.1 Examples of QME

Table C.1 — Example 1: failures
NOTE This table is an example from application of the approach in Annex B, then there may be some differences from 6.2 Table1.

QME Name	Number of failures
Target entity	Interactive Software
Objectives and property to quantify	The objective is to find interactive software failures during interactive use. Interactive Software failure is the property to quantify. Definition of the Property to quantify: An event in which a software or software component does not perform a required function within specified limits; concluding to a different result Point of View: - Software users: Desires failure and bug-free software - Support team: Ease of finding software fixes
Relevant Quality measure(s)	To measure the reliability of the software by utilizing mean time to failure during interactive use
Measurement method	Interactive Software Failures (SISF) Observe user interaction with software on system and catch events in which a software or software component does not perform a required function within specified limits; concluding to a different result.

List of sub properties related to the property to quantify (optional)	  Failure; - can be seen in software and hardware level. - is expected occur from a fault or unimplemented function. - can be found in interactive, non-interactive, accessible and inaccessible states.

Definition of each sub property (optional)	State: A unique configuration of information in a program or a machine Interactive State: A snapshot of action that requires user interaction Non-interactive State: A snapshot of action that does not require user interaction Accessible State: State that can be accessed during an action done by a user Inaccessible State: Opposite of accessible state Action : Set of states (operations) to create a possible event between interactive states Interaction: Input operation that triggers the action Reaction: Output operation done by the software; end-state of the action Interactive State: A snapshot of action that requires user interaction Output : Result of an action given by an informative screen or any perceptible way Ideal output: If and only if the output of an action is the way it is supposed to be, it is ideal Non-ideal output: Opposite of ideal output, failure Relations Between the Properties and Sub Properties  - Non-interactive states lack human interaction so a partial action or an action made up from these states is out of scope. - Inaccessible states are not meant to be measured and they are junk due to having no usage. - Any scenario not defined in use case or requirements document is not in the scope of the software so they are not counted in measure. This includes any scenario created from source codes. - Input or function failures are not in context of this measurement. Actions: - Has to have at least two interactive state (one for interaction, one for reaction) - Has to invoke at least one function - Has to have at least one output - Must be formed only by accessible states - Shall be defined either in requirements or use case document (or alike)Has to have at least one output

Input for the QME	Source of the information (Data element): Software documents (Use cases, Requirements document, etc…), Source codes Measurand (Input for the Measure): The actions found from the sources conclude to reactions -after an interaction by a user- that are either ideal results or opposite. These non-ideal results are the measurands.
Unit of measurement for the QME	Non-ideal reaction from an action
Numerical rules	ActDOC = Actions determined from documents ActSC = Actions obtained from source codes ActPOS = Possible actions to be measured determined from ActSC that are also found in fully or partially in ActDOC ActPOS = ActDOC ∩ ActSC ∑ (ActPOS) = Possible actions to be measured If an action of ActPOS is non-ideal for at least one value, that action is failure-possible. Failure Size = Count of the failure-possible unique actions FS = ∑ {Each ActPOS (For at least one Input) ≠ Ideal}
Scale type	Ratio
Context of QME	At the maintenance level
Software Life Cycle process(es)	Maintenance phase SYSTEM/ SOFTWARE OPERATION SYSTEM/ SOFTWARE MAINTENANCE
Measurement Constraints **(optional)**	• SISF is expected to be used on software. • Hardware related failures are not taken as measurement factors. • Component independency is a must during measurement process. • Software is supposed to run without any error before SISF is measured. • Software that quit with error all the time is not measurable since there is no activity found during any run period. • Target user of the application defines what a failure is what is not. Implemented functions are measured.

Table C.2 — Example 2: Number of records
NOTE This table is an example for data quality measure according to 6.2 Table1.

QME Name	Number of records
Target entity	Records
Objectives and property to quantify	The objective is to determine data quality of target data. Record is “a set of related data items treated as a unit.” (ISO/IEC 24765)
Relevant Quality measure(s)	Measure of Accuracy • Data Quality Measure Name: syntactic accuracy of records • Measurement Function A/B • Quality Measure Elements A=number of records with the specified data item syntactically correct B=number of records
Measurement method	Review and analyze data records
List of sub properties related to the property to quantify (optional)	Data Item: the lowest component of a group of data File: a set of related records
Definition of each sub property (optional)	Data item: the lowest component of a group of data Physical file note: the measurer should consider the data dictionary to find physical files.
Input for the QME	Physical files of a data base
Unit of measurement for the QME	Number of records
Numerical rules	Adding total records
Scale type	Ratio

Context of QME	This QME is mainly chosen to measure the accuracy and completeness to a group of data, or to create measures that aggregate information about quality of data items
Life Cycle Processes	Software life cycle processes (ISO/IEC 12207): Software Qualification Testing Process (Clause 7.1.7) Software Quality Assurance Process (Clause 7.2.3) Software Verification Process (Clause 7.2.4) Software Validation Process (Clause 7.2.5) Software Audit Process (Clause 7.2.7) Software Problem Resolution Process (Clause 7.2.8) Reuse Asset Management Process (Clause 7.3.2) System life cycle processes (ISO/IEC 15288): Information Management Process (Clause 6.3.6) Measurement Process (Clause 6.3.7) System Qualification Testing Process (Clause 6.4.6) Software Maintenance Process (Clause 6.4.10)
Measurement Constraints **(optional)**	Before comparing results coming from different technology environments is suggested to verify the impact of the technology on the number of record generated for the same information item

Table C.3 — Example 3: use case

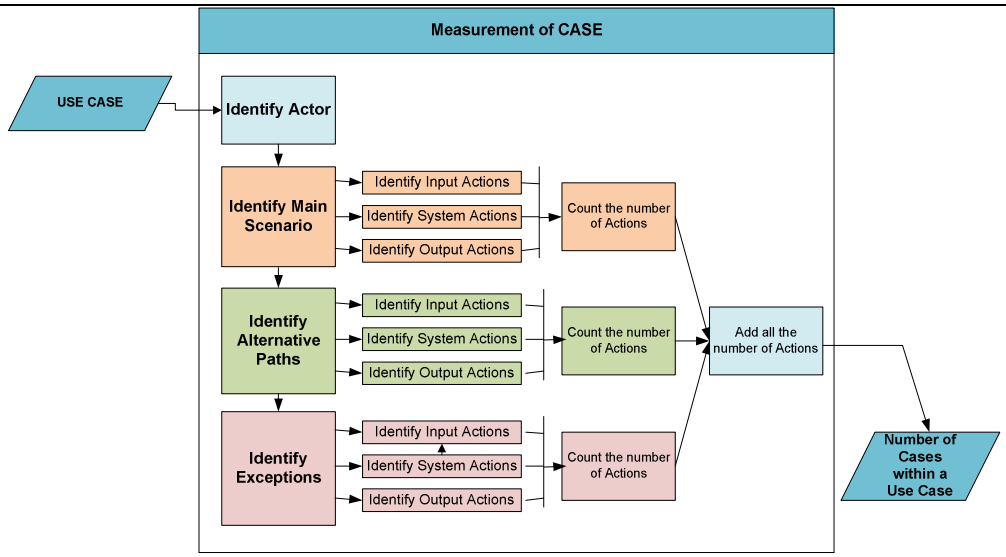
NOTE This table is an example from application of the approach in Annex B, then there may be some differences from 6.2 Table1.

<table>
<tr><td>QME Name</td><td>Number of use cases</td></tr>
<tr><td>Target entity</td><td>Specifications describing use cases or executable software or system</td></tr>
<tr><td>Objectives and property to quantify</td><td>The results of the measurement method will be used to assess external quality of software product when the use case is measurable.

Use case is the property to quantify.

Use Case is the description of the interaction between an Actor (the initiator of the interaction) and the system itself. It is represented as a sequence of simple steps. Each use case is a complete series of events, described from the point of view of the Actor.</td></tr>
<tr><td>Relevant Quality measure(s)</td><td>
<table>
<tr><th>Characteristic</th><th>Subcharacteristics</th><th>Sample Measurable Measurement</th></tr>
<tr><td>Reliability</td><td>Recoverability</td><td>Availability</td></tr>
<tr><td>Functionality</td><td>Interoperability</td><td>Data exchangeability</td></tr>
<tr><td>Usability</td><td>Understandability</td><td>Demonstration Accessibility in use</td></tr>
<tr><td></td><td>Learnability</td><td>Help frequency</td></tr>
<tr><td></td><td>Operability</td><td>Customizability</td></tr>
<tr><td>Maintainability</td><td>Analyzability</td><td>Status monitoring capability</td></tr>
<tr><td></td><td>Changeability</td><td>Parameterized modifiability</td></tr>
<tr><td></td><td>Stability</td><td>Change success ratio</td></tr>
<tr><td></td><td>Testability</td><td>Availability of built-in test function</td></tr>
<tr><td>Portability</td><td>Installability</td><td>Ease of installation</td></tr>
</table>
</td></tr>
</table>

Measurement method	MONOC (Measurement of Number of Case) Review and analyze specified use case description or observe user operation of software on system.
List of sub properties related to the property to quantify (optional)	Properties to quantify measure Action Line in the Use Case. An action can be a • Data group reference/retrieval • Data group insert/update • Derived data creation by transforming existing data • Mathematical formulas/calculations • Condition analysis to determine which are applicable • Data validation • Equivalent-value conversion • Data filtering/selection Actor, preconditions, main scenario, action, post-conditions, alternative paths, exceptions. This correspond to the actual vocabulary used in the literature for Use case [1] http://en.wikipedia.org/wiki/Use_case
Definition of each sub property (optional)	**An Actor** is “someone or something outside the system that either acts on the system – a primary actor – or is acted on by the system – a secondary actor. An actor may be a person, a device, another system or sub-system, or time. Actors represent the different roles that something outside has in its relationship with the system whose functional requirements are being specified.” **Preconditions** define all the conditions that must be true before the initiation of the use case. **Main scenario** is the description of the main success scenario in a sequential order. **Action** is the element of a step that a user performs during a procedure. **Post-conditions** “describe what the change in state of the system will be after the use case completes. Post-conditions are guaranteed to be true when the use case ends.” **Alternative paths;** “Use cases may contain secondary paths or alternative scenarios, which are variations on the main theme. Each tested rule may lead to an alternative path and when there are many rules the permutation of paths increases rapidly. Sometimes it is better to use conditional logic or activity diagrams to describe use case with many rules and conditions.” **Exceptions,** is the place “what happens when things go wrong at the system level are described, not using the alternative paths section but in a section of their own.” An example of an alternative path would be: "The system recognizes cookie on user's machine", and "Go to step 4 (Main scenario)". An example of an exception path would be: "The system does not recognize user's logon information", and "Go to step 1 (Main path)".
Input for the QME	Software requirements Specification documents can be used as input since they commonly include use cases. In addition, user manuals and screens can be used to extract main flows, alternative flows and exceptions after the software has been developed.

Unit of measurement for the QME	The unit of measurement and, if appropriate, the formula use. Examples of units include number of X, percentage and rank. Note: the percentage is possible when the QME is a mathematical function (ratio).
Numerical rules	 Mathematical expression: Number of Case = Σ number (Input Actions) + Σ number (System Actions) + Σ number (Output Actions)
Scale type	Ratio
Context of QME	The aim of this measurement method is to measure the size of the 'use case' and later use the results of the measurement to assess the reliability, maintainability, functionality, usability, and portability of the software and understand quantitatively the following characters and sub characters given in the measurement scope section. The results of this measurement will also be helpful in assessing the internal, external quality and quality in use of a software product. For example, when the number of cases or size of the cases is defined based on the case definition, we can make an assessment about the "easiness of installation". The measurement method of case can be applied to any type of software such data rich, control rich, real-time.
Software Life Cycle process(es)	The measurement of use case can be applied even in the early phases of a software life cycle. It is best to measure the case after requirements analysis phase. - STAKEHOLDER REQUIREMENTS DEFINITION - SYSTEM/ SOFTWARE REQUIREMENTS ANALYSIS - SYSTEM/ SOFTWARE QUALIFICATION TESTING - SYSTEM/ SOFTWARE OPERATION - SYSTEM/ SOFTWARE MAINTENANCE
Measurement Constraints **(optional)**	The level of detail of the use cases impacts the number of case.

C.2 Expansion set of Quality Measure Elements (QMEs)

Table C.4 — List of Expansion set of QMEs and concept to quantify and high level definition

NO	Expansion set of QME	Definition and concepts related directly to QME
1	Duration of Processing	Duration of Processing: time between start point to and finish point of specified task of system or software. Process: system of activities, which use resources to transform inputs into outputs [ISO 9000:2000],ISO/IEC 25000:2005 Edition:1 Activity duration: the time in calendar units between the start and finish of a schedule activity (A Guide to the Project Management Body of Knowledge (PMBOK® Guide) -- Fourth Edition) Actual duration: the time in calendar units between the actual start date of the schedule activity and either the data date of the project schedule if the schedule activity is in progress or the actual finish date if the schedule activity is complete. (A Guide to the Project Management Body of Knowledge (PMBOK® Guide) -- Fourth Edition) Duration (DU or DUR): the total number of work periods (not including holidays or other nonworking periods) required to complete a schedule activity or work breakdown structure component. Usually expressed as workdays or workweeks. Sometimes incorrectly equated with elapsed time. (A Guide to the Project Management Body of Knowledge (PMBOK® Guide) -- Fourth Edition) Data processing (DP): the systematic performance of operations upon data. (ISO/IEC 2382-1:1993 Information technology--Vocabulary--Part 1: Fundamental terms, 01.01.06) Example: arithmetic or logic operations upon data, merging or sorting of data, assembling or compiling of programs, or operations on text, such as editing, sorting, merging, storing, retrieving, displaying, or printing Note: The term data processing should not be used as a synonym for information processing. Syn: automatic data processing
2	Duration of Service Available (System Operational)	Duration of Service Available: period during which a system is working in a manner acceptable to its operator or user. Service: performance of activities, work, or duties associated with a product (ISO/IEC 12207:2008 Systems and software engineering--Software life cycle processes, 4.40) (ISO/IEC 15939:2007 Systems and software engineering--Measurement process, 3.36) See Also: product, result, deliverable
3	Duration of Response	Duration of Response (response time) Response time: A time period between the time when a sending of response request is triggered and the time when its response is received by the response confirmation role object., ISO/IEC 10164-22:2000 Edition:1 Response time: the elapsed time between the end of an inquiry or command to an interactive computer system and the beginning of the system's response (ISO/IEC 24765:2009 Systems and software engineering vocabulary) See Also: port-to-port time, think time, turnaround time
4	Duration of Operator Work	Duration of Operator Work: operation time period between the time when specified amount of tasks are started being manipulated by operator and the time when completed.
5	Duration of Operation	Duration of Operation: action needed to perform an Activity Note: One or more operations are necessary to execute an activity. An operation may consist of other operations. ISO/IEC 15940:2006 Edition:1

NO	Expansion set of QME	Definition and concepts related directly to QME
6	Duration of Restoration	Duration of Restoration: The time when the major portion of the interrupted load has been restored and the emergency is considered to be ended. Some of the loads interrupted may not have been restored due to local problems.
7	Duration of System Migration	Duration of System Migration: time period between the time when the specified migration is started and the time when completed.
8	Duration of Modification	Duration of Modification: time period between the time when the specified modification is started and the time when completed. Modification: The replacement of a sentence in the information base or conceptual schema by another one, thereby possibly changing the collection of sentences which are deducible., ISO/TR 9007:1987 Edition:1
9	Number of Users (User's Requests)	Number of Users (User's Requests): number of requests for processing of the system sourced by individual or organisation who uses the system to perform a specific function. User : individual or organisation that uses the system to perform a specific function Note: Users may include operators, recipients of the results of the software, or developers or maintainers of software. [ISO/IEC 15939:2002, ISO/IEC 25000:2005 Edition:1]
10	Number of Users (Number of authorized users)	Number of Users (Number of authorized users): number of individual or organisation who is authorized to use the system to perform a specific function.
11	Number of Interfaces	interface: A named set of operations that characterize the behavior of an element., ISO/IEC 19501:2005 Edition:1 This QME includes the following sub categories: Number of secured interfaces. NOTE This includes any kind of interface existing between functions, objects, software, systems and human.
12	Size of logs (Number of logs)	Log: a document used to record and describe or denote selected items identified during execution of a process or activity. Usually used with a modifier, such as issue, quality control, action, or defect.[PMBOK 4th Ed.]
13	Number of Document (including log records)	Document: 1. uniquely identified unit of information for human use, such as a report, specification, manual or book, in printed or electronic form , ISO/IEC TR 9294:2005 Edition:2 2. equivalent to an item of documentation. ISO/IEC 15910:1999 Edition:1
14	Number of System Handling Equipments and Site Locations	Number of System Handling Equipments and Site Locations: Equipments and their locations which are handled by the system. NOTE The more different remote locations and more types of equipments are to be handled, the harder the system is able to do. Location : information on the network point of attachment through which a user is currently accessing a network, including e.g. identification of the geographical, administrative and IP-topology domain, ISO/IEC TR 26927:2006 Edition:1 This QME includes the following sub categories: Number of System Handling Servers
15	Number of Connections	Connection: An association established between functional units for data transmission., ISO/IEC 2382-9:1995 Edition:2

NO	Expansion set of QME	Definition and concepts related directly to QME
16	Number of resources (including assets)	Asset: anything that has value to the organization Note: There are many types of assets, including: information (2.18); software, such as a computer program; physical, such as computer; services; people, and their qualifications, skills, and experience; and intangibles, such as reputation and image. [ISO/IEC 27000:2009]
17	Number of components	Component: (1) an entity with discrete structure, such as an assembly or software module, within a system considered at a particular level of analysis (ISO/IEC 15026:1998 Information technology -- System and software integrity levels, 3.1) (2) one of the parts that make up a system (IEEE 829-2008 IEEE Standard for Software and System Test Documentation, 3.1.6) (3) set of functional services in the software, which, when implemented, represents a well-defined set of functions and is distinguishable by a unique name (ISO/IEC 29881:2008 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method, A.4) Note: A component may be hardware or software and may be subdivided into other components. The terms "module," "component," and "unit" are often used interchangeably or defined to be subelements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized. A component may or may not be independently managed from the end-user or administrator's point of view. Software component (SC). (1) a general term used to refer to a software system or an element, such as module, unit, data, or document (IEEE 1061-1998 (R2004) IEEE Standard for Software Quality Metrics Methodology, 2.2) (2) a functionally or logically distinct part of a software configuration item, distinguished for the purpose of convenience in designing and specifying a complex SCI as an assembly of subordinate elements (ISO/IEC 24765:2009 Systems and software engineering vocabulary) System element. (1) member of a set of elements that constitutes a system (ISO/IEC 12207:2008 Systems and software engineering--Software life cycle processes, 4.49) (ISO/IEC 15288:2008 Systems and software engineering--System life cycle processes, 4.32) Example: hardware, software, data, humans, processes (e.g. processes for providing service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities (e.g., water, organisms, minerals), or any combination Note: A system element is a discrete part of a system that can be implemented to fulfill specified requirements. Configuration item (CI). (1) entity within a configuration that satisfies an end use function and that can be uniquely identified at a given reference point. (ISO/IEC 12207:2008 Systems and software engineering--Software life cycle processes, 4.7) (2) item or aggregation of hardware or software or both that is designed to be managed as a single entity (ISO/IEC 19770-1:2006 Information technology -- Software asset management -- Part 1:
18	Data volume	Data: reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing Note 1: Data can be processed by humans or by automatic means. [ISO/IEC 2382-1:1993] Note 2: The definition in ISO/IEC 25000 is different because it refers to data which relate to the result of the measurement. , ISO/IEC 25012:2008 Edition:1
19	Number of requirements	Requirement: (1) a condition or capability needed by a user to solve a problem or achieve an objective (ISO/IEC 24765:2009 Systems and software engineering vocabulary) (2) a condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents (ISO/IEC 24765:2009 Systems and software engineering vocabulary) (3) a documented

NO	Expansion set of QME	Definition and concepts related directly to QME
		representation of a condition or capability as in (1) or (2) (ISO/IEC 24765:2009 Systems and software engineering vocabulary) (4) a condition or capability that must be met or possessed by a system, product, service, result, or component to satisfy a contract, standard, specification, or other formally imposed document. Requirements include the quantified and documented needs, wants, and expectations of the sponsor, customer, and other stakeholders (A Guide to the Project Management Body of Knowledge (PMBOK® Guide) -- Fourth Edition) See Also: design requirement, functional requirement, implementation requirement, interface requirement, performance requirement, physical requirement Allocated requirement: (1) requirement that levies all or part of the performance and functionality of a higher level requirement on a lower level architectural element or design component (ISO/IEC 24765:2009 Systems and software engineering vocabulary)
20	Throughput (QME for Performance efficiency)	Throughput: (1) the amount of work that can be performed by a computer system or component in a given period of time (ISO/IEC 24765:2009 Systems and software engineering vocabulary) (2) the rate (i.e., the average number per time unit with respect to the rating interval) of all tasks of a task type submitted to the SUT (ISO/IEC 14756:1999 Information technology -- Measurement and rating of performance of computer-based software systems, 4.24) Note: Usually throughput is defined by the rate of terminated tasks during a period of time. Performance efficiency: performance relative to the amount of resources used under stated conditions NOTE Resources can include other software products, the software and hardware configuration of the system, and materials (e.g. print paper, storage media).
21	Number of Tolerable Faults (QME for Fault Tolerance)	Fault tolerance: degree to which a system, product or component operates as intended despite the presence of hardware or software faults. [ISO/IEC 25010:2011]
22	Number of Change Requests on User Interface (QME for User interface aesthetic of usability)	User interface aesthetics: degree to which the user interface enables pleasing and satisfying interaction for the user NOTE This refers to properties of the product or system that increase the pleasure and satisfaction of the user, such as the use of colour and the nature of the graphical design.[ISO/IEC 25010:2011]

Annex D
(informative)
Cross reference tables for quality measures

The layout of the cross-reference table is shown in Table 1. In this table the quality measures (a combination of measures) the properties of the QMEs (and /or base measures) plus the qualifier can be any of those stated in this document. This table works in both ways: by row, a cross associates a QME and suggested extension to all the quality measures where its usage applies; by column, all the crosses indicate the QMEs that are required for the quality measures plus their qualifier in that column.

Table D.1 − Layout of the cross-reference table representing the relationships between QME and quality measures

	Quality measure 1	Quality measure 2	Quality measure n
QME 1	X	X	
QME 2	X		X
QME n			X

This is a recommend table to construct when using a number of quality measures in an organisation. When constructing the table the measurer should add the qualifiers because in practice there is always a context or an objective related to a quality measure. Table D.1 should be constructed from the list of QMEs (Annex A) and quality measures. The list is able to be extracted from Quality Measures Elements related to quality in ISO/IEC TR 9126 - 2,3,4 or ISO/IEC 25022,3,4. Note that some QMEs like cost, effort, duration and size are already defined in different standards, then they will not be redefined, they will be listed only. The cross-reference table D.1 should use the list of properties of QME with the qualifiers of the quality measure (Annex B). Annex D presents the cross-reference table representing the relationship between quality measures, characteristics and sub characteristics.

Table D.2 − Layout of the cross-reference table representing the relationships between quality measures and characteristics and sub characteristics

	Quality characteristics and sub characteristics 1	Quality characteristics and sub characteristics 2	Quality characteristics and sub characteristics 3
Quality measures 1	X	X	
Quality measures 2	X		X
Quality measures n			X

This is a recommend table (table 2) when using a number of quality measures with sub characteristics and characteristics. Those tables will be a part of ISO/IEC 25022, 25023 and 25024 documents.

Annex E
(informative)
Measurement scale type

The type of scale depends on the nature of the relationship between values on the scale. Five types of scales are commonly used and connected considerations are:

NOTE Scale is defined in ISO/IEC 25000. The following are just examples of types of scale.

Nominal scale type - The purpose of nominal scale type measures in a set of QME is to classify measured attributes. No ordering is implied even if numbers are used. The numbers assigned to measures in nominal scale type measurement identify only the category (type) of measured attribute. Therefore the order, minimum, maximum, median, arithmetic mean, percentages etc. from these numbers have not any empirical meaning – these correspond to inadmissible mathematical operations.

EXAMPLE Identification of football players with numbers.

Ordinal scale type - The purpose of ordinal scale type measures in a set of QME is to assign the order to the measured attributes. The ordinal scale type is often useful to augment the nominal scale with information about an ordering of the classes or categories. The minimum, maximum and median from the measures in the ordinal scale type measurement have empirical meaning. The arithmetic mean, percentage etc. do not have an empirical meaning..

EXAMPLE Software product failure by severity (e.g. negligible, marginal, critical, catastrophic)

Interval scale type - The purpose of interval scale type measures in a set of QME is to measure a difference between measures. If a ratio is calculated from the QMEs of the same type, it does not have an empirical meaning.

EXAMPLE The temperature in Celsius can be add or subtract, but not multiply or divide (25 C and 10 C only mean 15 C difference).

Ratio Scale type - The ratio scale is similar to the interval scale but includes the value zero, representing the total lack of an attribute. Ratio scale type QMEs include ordered rating scales, where the difference between two measures, and the proportion of two measures, have the same empirical meaning. Ratio and average have meaning to the values.

EXAMPLE Effort (time) spent to change, Buffer size, Number of detected faults

Bibliography

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

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

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

ISO/IEC 25022 Systems and software engineering —Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of quality in use (revision of ISO/IEC 9126-4) (to be published)

ISO/IEC 25023 Systems and software engineering —Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality (revision of ISO/IEC 9126-2 and 9126-3) (to be published)

ISO/IEC 25024 Systems and software engineering —Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of data quality (to be published)

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

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

ISO/IEC 2382-1:1993 – Information technology – Vocabulary – Part 1: Fundamental terms

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

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

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

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

Desharnais, J-M. , Abran A., Suryn W., ‘Attributes and Related Base Measures within ISO 9126: A Pareto Analysis’, Software Quality Management 2009 and INSPIRE 2009 Conferences, British Computer Society, April 2009.