INTERNATIONAL STANDARD

ISO/IEC 25021

First edition
2012-11-01

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

Ingénierie des systèmes et du logiciel — Exigences de qualité et évaluation des systèmes et du logiciel (SQuaRE) — Éléments de mesure de la qualité

Reference number
ISO/IEC 25021:2012(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

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

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 25021 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and System Engineering*.

This first edition of International Standard ISO/IEC 25021 cancels and replaces the first edition of Technical Report ISO/IEC TR 25021:2007.

The SQuaRE series of standards consists of the following divisions under the general title *Systems and Software Quality Requirements and Evaluation (SQuaRE)*:

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

Introduction

The purpose of this International Standard 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 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 subcharacteristic 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 (Figures 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 purposes of defining and using the Quality Measures Elements (QMEs) in this document are:

- To provide guidance for organisations developing and implementing their own QMEs;
- To promote the consistent use of specific QME for measuring and using the product properties that are relevant to different product quality characteristics and subcharacteristics;
- To help identify a set of QMEs that are uniquely required to derive all the quality measures for a given set of characteristics or a set of subcharacteristics of a product.

The QMEs are the common components of a number of quality measures. The intended usage of this International standard is that users will be able to select and define relevant valid QMEs to define internal, external, data or quality-in-use quality 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.

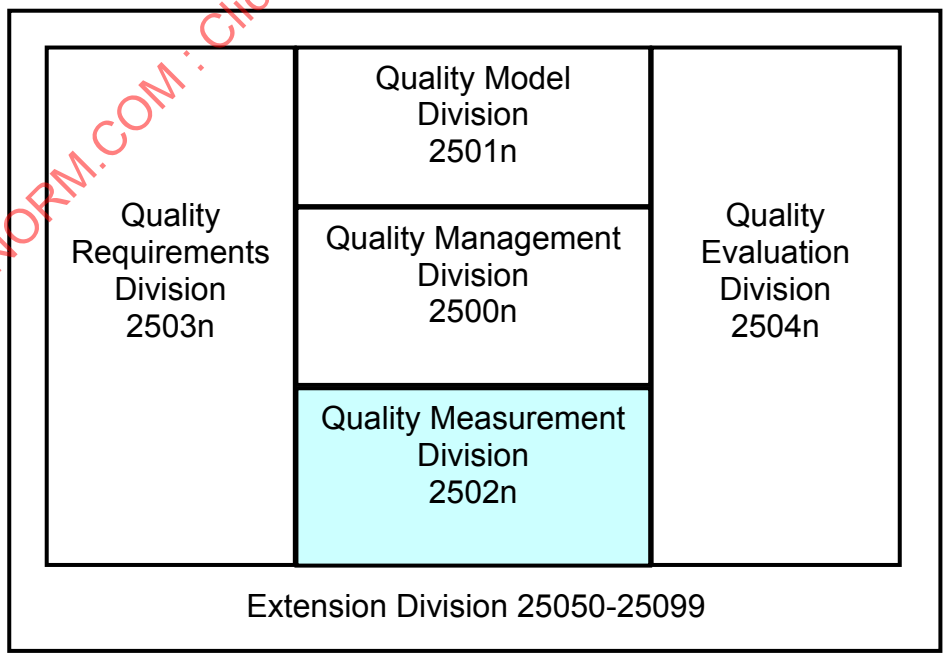

Figure 1 — Organisation of the SQuaRE series of international standards

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

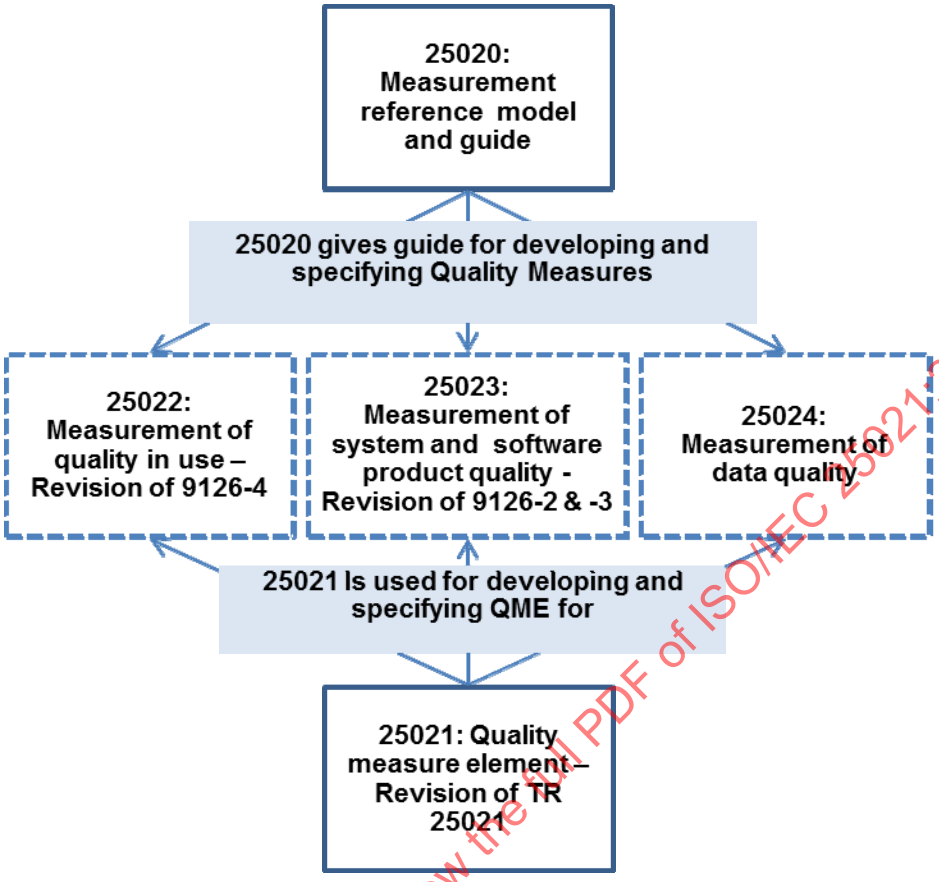

Figure 2 — Structure of the Quality Measurement Division

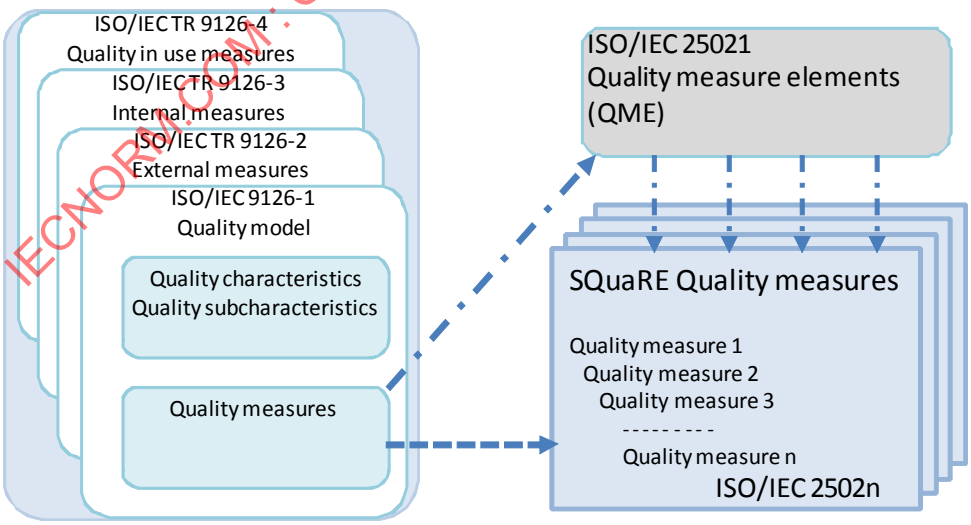

Figure 3 — The relationship of ISO/IEC 25021 as a link between the 9126 series and the SQuaRE series of standard

The ISO/IEC 9126 series is composed of four documents that list and describe the characteristics, subcharacteristics and quality measures that are referred to as the quality model. The SQuaRE quality models categorize product quality into characteristics which are further subdivided into subcharacteristics and quality properties (ISO/IEC 25010). Each quality measure within ISO/IEC 9126 series is composed of at least two QMEs. The properties (of a product) are linked to the QME (ISO/IEC 25020), using a measurement method. The 2502n series designs and describes quality measures and associated QMEs for all the quality (sub)characteristics in 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:

- Requirements for defining QMEs as part of the specification of the product quality requirements with examples (see 6.2 Tables 1 and 2);

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

- An initial set of QMEs, as examples (see Annex A Table A.1);
- A guideline for defining and quantifying the property of the product (target entity) for QMEs (see Annex B)

This document is intended for, but not limited to, developers, acquirers and independent evaluators of products, particularly those responsible for defining product quality requirements and for product evaluation. This International Standard is applicable when defining the QMEs to be used to implement quality measures such as those specified in ISO/IEC 25022, ISO/IEC 25023 and ISO/IEC 25024.

2 Conformance

When users define quality measures for a product, each of the referred QME shall be described according to the information items of format specified in Table 1 (see 6.2). The same should be applied for modifying an existing QME.

3 Normative references

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

ISO/IEC 25000:2005, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*

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

ISO/IEC 25020:2007, *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 Guide 99:2007, *International vocabulary of metrology — Basic and general concepts and associated terms (VIM)*

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, ISO/IEC Guide 99, and the following apply. The following definitions are replicated here for the convenience of the user of this document.

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

NOTE 1 The behavior can be verified and/or validated by executing the software product during testing and operation.

NOTE 2 Based on the ISO/IEC 25000:2005 definition of external software quality.

NOTE 3 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

NOTE 1 Static properties include those that relate to the software architecture, structure and its components.

NOTE 2 Static properties can be verified by review, inspection, simulation and/or automated tools.

NOTE 3 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.

NOTE 4 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 25000:2005]

4.6
measurement
set of operations having the object of determining a value of a measure

[ISO/IEC 15939:2007]

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

4.7
measurement function
algorithm or calculation performed to combine two or more quality measure elements

NOTE This definition is modified from ISO/IEC 15939:2007 definition of measurement method.

4.8
measurement method
logical organisation of operations, described generically, used in measurement

NOTE This definition is modified from ISO/IEC 15939:2007 definition of measurement method.

4.9
measurement procedure
logical organisation of operations, applied specifically, used in the performance of particular measurements according to a given measurement method

NOTE 1 This definition is modified from ISO/IEC 15939:2007 definition of measurement procedure.

NOTE 2 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
property to quantify
property of a target entity that is related to a quality measure element and which can be quantified by a measurement method

NOTE 1 A software artifact is an example of a target entity.

NOTE 2 A sub-property is related to a property.

4.12
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

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

4.13
quality measure
derived measure that is defined as a measurement function of two or more values of quality measure elements

4.14
quality measure element (QME)
measure defined in terms of a property and the measurement method for quantifying it, including optionally the transformation by a mathematical function

4.15
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]

4.16
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.

4.17
target entity
fundamental thing of relevance to the user, about which information is kept, and need to be measured

4.18
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

NOTE 1 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.

NOTE 2 Units of measurement have conventionally assigned names and symbols.

NOTE 3 Based on the ISO/IEC 25000:2005 definition of unit of measurement.

5 Abbreviated terms

In this International Standard, the following abbreviations are used:

a) QME - Quality measure element;

b) QM - Quality measure;

6 Quality measure elements concept

6.1 Presentation of the measurement method model

To understand and indicate quality (sub)characteristics, QM is defined and then QMEs are defined.

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 a QME.

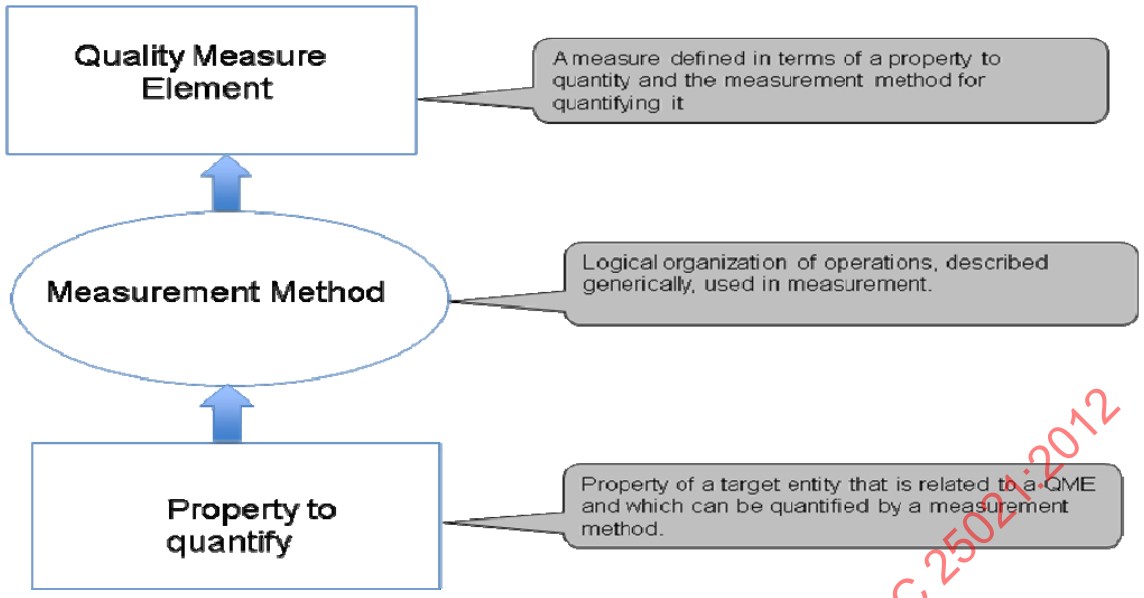

Figure 4 — Relationship between property to quantify, measurement method and QME

The user of the measurement method shall identify and collect data related to quantifying the property (Figure 4). Depending on the context of 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 artefacts, components, content or behaviour of the target entity (e.g. documentation, code).

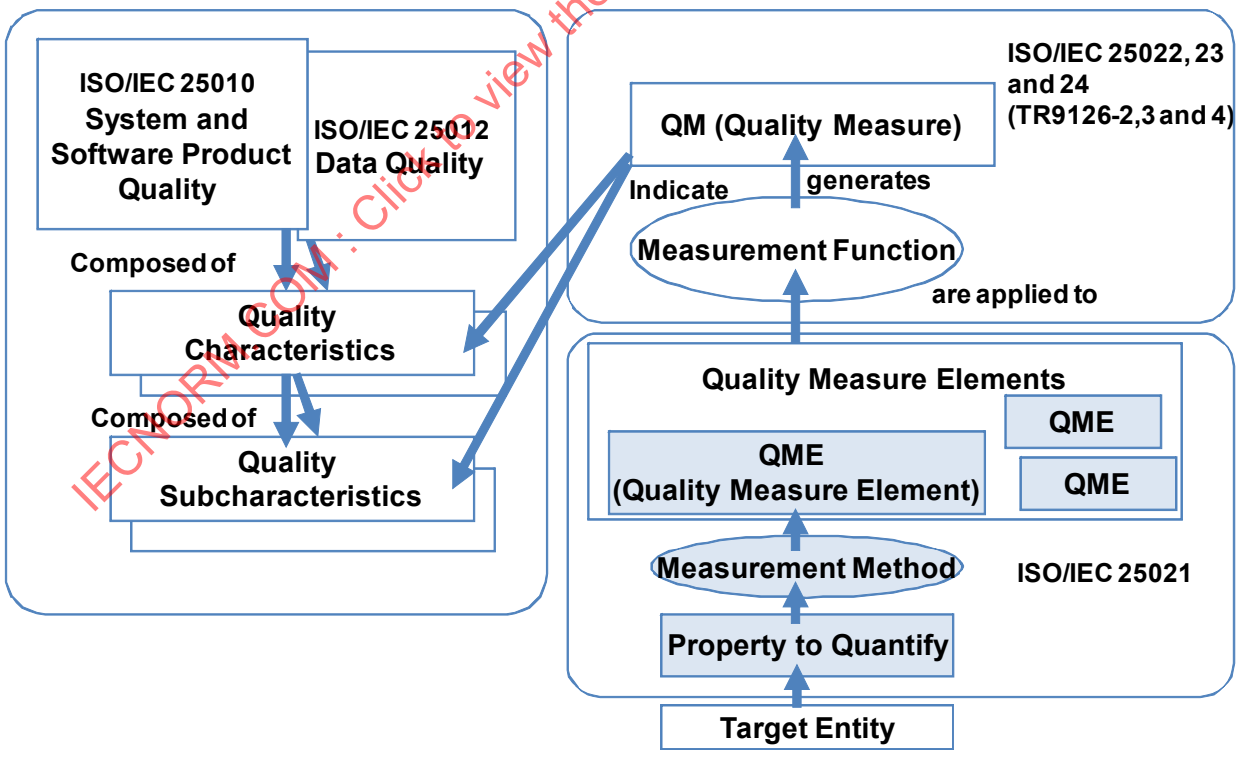

Figure 5 — Relationship among property to quantify, measurement method, QME and QM

Figure 5 shows that

a) product quality is expressed as a set of quality characteristics which in turn are composed of subcharacteristics;

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

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

NOTE This Figure 5 is based on the Systems and Software Product Quality Measurement Reference Model defined in ISO/IEC 25020.

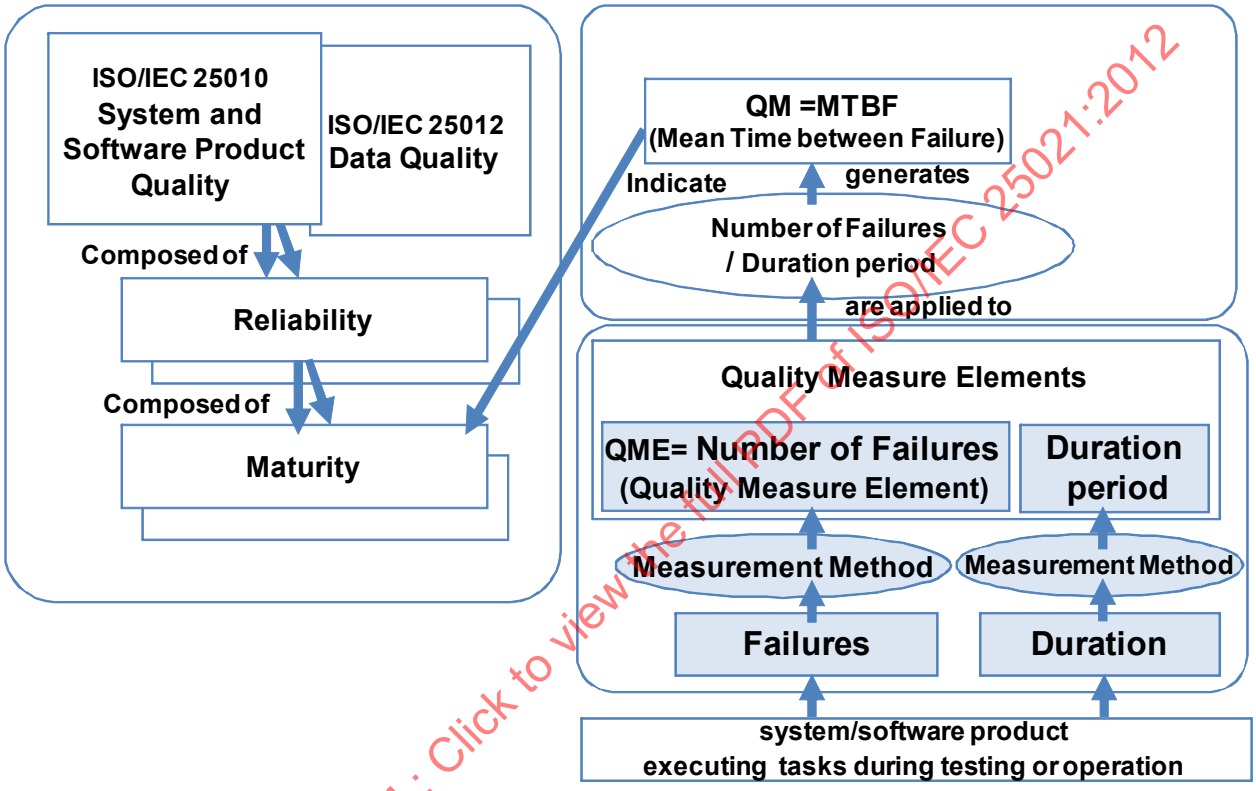

Figure 6 — An Example of Relationship among property to quantify, measurement method, QME and QM

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.

NOTE 1 A QME can be identified when quality characteristic or subcharacteristic are selected and/or a QM is defined to indicate it. The same QME may be used by different QMs.

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

6.2 Table format of QMEs

The information items listed in table format of the QMEs (see Table 1) shall be used to define[1] and/or design a QME to provide necessary or helpful information.

Table 1 — Table format for QMEs

NOTE Items in the below table are organized into four groups which are a) for QME identification, b) – d) for "what the QME is", e) – k) for "how to measure the QME" and l) – n) for management of QME application.

a) QME Name	A QME should have a unique name and should be identified with a serial number, if necessary. Most of the time it begins with "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 a work product or behavior of a 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. 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. For example, "number of software faults" is the QME and "fault" is the property of the software to quantify. Objective of QME should be specified with the definition of the property to quantify to describe such as the followings: - What is intended to know by making definition of the property to quantify of this QME? - What is the information needed that is expected to be represented by this QME? From the identification and definition of the property to quantify, what needs to be measured is determined (e.g. line of code, defects, duration). It is helpful to describe what kinds of component or event in the designated target entity need to be identified, defined and quantified. The followings are examples: 1) Lines, functions, paths or tokens having specified feature in program source code may be identified, defined and quantified; 2) Events every when the software under testing fails to specified test cases may be identified, defined and quantified; 3) Events every when the user of the system fails to user's intended tasks may be identified, defined and quantified.
d) Relevant Quality measure(s)	Reference to specific quality measure(s) which use this QME shall be specified. Examples of quality measures can be found in the ISO/IEC 9126 series, 25000 SQuaRE series and other documents. An exhaustive list of quality measures is not required.

[1] For the definitions see c) (objectives and properties to quantify) and g) (definition of each sub-properties).

e) Measurement method	Measurement method explains how to collect data and how to transform it to a value quantifying the property through a numerical rule. 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 could have names: 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 different sub properties, if necessary. This relationship 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 details to identify what quantitative information is used to measure the QME. Any sources providing the input should also be identified such as the documented work products, behavior of system and software, or human behavior 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 consider the limit of the result applied to each property and sub properties.
k) Scale type	Scale type shall be identified. Scale type could be nominal, ordinal, interval or ratio (see Annex D).

l) Context of QME	This gives information about the intended use of the measurement results. It is helpful to understand possibility of using QME to express quality (sub) characteristics by describing typical examples of quality characteristics, quality subcharacteristics or quality measures (QM) which are mainly intended to use of the QME measurement results. NOTE The QME is able to be employed by a number of quality measure (QM) to measure any of quality (sub)characteristics. Assumptions and prerequisite conditions of target entities, its environments and circumstances to which the measurement method of QME is to be applied is described here.
m) Software Life Cycle process(es)	The typical appropriate life cycle process(es) that are suited for actual measurement of this QME with respect to a target entity should be identified here (e.g. process(es) in which a specific target entity is created or realized enough to enable the obtaining an 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) listed here are those where we can obtain actual measured results of the QME. Related life cycle process(es) after obtaining actual data, additional actual measurement or use of the measured results are also specified here. 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), such as stakeholder requirements definition, software requirements analysis, software architectural design, software detailed design, software construction, software integration, software qualification testing, software installation, software acceptance support, software operation, software maintenance, software disposal and so on, are Defined in ISO/IEC 12207:2008, Systems and Software Engineering - Software Life Cycle Processes. The basic system process(es), such as stakeholder requirements definition, requirements analysis, architectural design, implementation, integration, verification, transition, validation, operation, maintenance, disposal and so on, are defined in ISO/IEC 15288:2008, Systems and Software Engineering - System Life Cycle Processes. NOTE 3 If the methodology in use does not include any life cycle process(es) described in neither ISO/IEC 12207 nor ISO/IEC 15288 then the measurer may also mention the methodology and the particular process(es) that will be used.
n) Measurement Constraints **(optional)**	If necessary, any constraints related to the measurement method should be described, if necessary. The QME may have measurement constraints such as measurement errors or fluctuations due to dependency of the followings: scope of investigation, way of investigation, volatility of specification or tactics of test cases. NOTE 1 For example, the number of faults in code may vary between newly developed code and reused code in scope. Each of the various ways of code investigation such as review, walk-through, inspection, individual inspection by expertise, pair programming, code analysis tools, unit testing, causal analysis of failure in integration testing etc gives different number of faults in code. NOTE 2 For example, in case of counting the number of specification defects, the specification document should be available and not very volatile.

The following table is an example on how to use the format of Table 1.

Table 2 — EXAMPLE: application of Table 1 for fault (of code)

a) QME Name	Number of faults (of code)
b) Target entity	Program source code
c) Objectives and property to quantify	The objective is to measure the number of faults in code with reference to the design specifications and / or coding standards. What needs to be measured is the number of erroneous lines of code. Fault is the property to quantify. Definition of fault:(1) a manifestation of an error in software (ISO/IEC 24765:2010 Systems and software engineering vocabulary) (2) an incorrect step, process, or data definition in a computer program (ISO/IEC 24765:2010 Systems and software engineering vocabulary). NOTE A fault, if encountered, may cause a failure.
d) 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 (subcharacteristic) and Reliability (characteristic) level of the software
e) 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 a result of verification and validation activities such as code review, unit testing, causal analysis to resolve failures in integration testing.
f) List of sub properties related to the property to quantify (optional)	Related sub properties: executable statements, erroneous lines of code, corrected lines of code.

g) Definition of each sub property (optional)	Executable Statements: The statements which can be categorized as labeled statements, expressions, selection statements, iteration statements and jump statements. Non-Executable Statements: The statements that can be categorized as declarations and declaration specifiers. Erroneous Lines of Code: Lines of code which contains fault(s). Specifications should tell if the source code is erroneous or not. Correct Lines of Code: Lines of code with no fault. NOTE It is also possible that the actual lines of code are correct and the specifications should be changed. This should not be counted as erroneous lines of code.
h) Input for the QME	Source code, design specifications and coding standards
i) Unit of measurement for the QME	Lines of code
j) Numerical rules	Adding total erroneous lines of code A numerical assignment rule from a practitioner view using 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
K) Scale type	Ratio
l) Context of QME	This QME is mainly chosen to measure the Maturity (subcharacteristic) and Reliability (characteristic) level of the software
m) Software Life Cycle process(es)	Software Construction (coding and unit testing), Implementation process.
n) Measurement Constraints (optional)	Source codes shall be available to be able to compare the actual lines of code with the design specifications. Stable design specifications shall be available to be able to compare verify the actual lines of code with the design specifications and identify faults. A tool or checklist shall be available to check for coding standards.

Annex A
(informative)

Examples of QMEs

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

Table A.1 — List of Initial set of QMEs

NO	QME	Description
1	**a) QME name**	**Number of accessible functions**
	b) Target entity	User callable functions
	c) Objectives and property to quantify	To know how many user callable functions are accessible by disabled users. Definition of **callable function**: function provided to users by a system for users to access, call and use to perform specified tasks.
	d) Relevant quality measure(s)	QMs such as number of (not) accessible functions under specific context in use, or by specific type of user during testing or operation for quantifying context coverage in quality in use and accessibility in usability
	e) Measurement method	Review or test specified cases that disabled users callable and operable functions of system/software and count the number of functions that they could not be able to successfully use.
	f) List of sub properties related to the property to quantify (optional)	-to be completed, if necessary
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	Review or test results on specified cases of operation by disabled users, User manuals
	i) Unit of measurement for the QME	Number of user callable functions
	j) Numerical rules	x-y, x: The number of user callable functions which are reviewed or tested on specified cases of operation by disabled users y: The number of user callable functions that disabled user could not be able to successfully use.
	k) Scale type	Ratio
	l) Context of QME	This QME is usable to context coverage in quality in use and accessibility in usability
	m) Software life cycle process(es):	Implementation and Operation
	n) Measurement constraints (optional)	-to be completed, if necessary

NO	QME	Description
2	**a) QME name**	**Number of user problems**
	b) Target entity	User problems during operation
	c) Objectives and property to quantify	To know how many occurrences of problems system/software users recognize during operation. User problem during system/software operation is the property to quantify Definition of **user problem**: Each time a user complains about a product, it is registered by the organisation (normally at the helpdesk level). Knowing the complaint could help to measure the degree of the user satisfaction over time period. For example, technical problems or functional problems are extracted from user complaints and sorted out by help desk. This QME can help users to determine problem during operation of the software but not necessarily limited to that properties.
	d) Relevant quality measure(s)	QMs such as number of (sever) user problems reported per week or distribution of spent days until resolution of user problems for quantifying usability, reliability and satisfaction
	e) Measurement method	List up problems from user claim reports, categorize them by severity and count them.
	f) List of sub properties related to the property to quantify (optional)	-to be completed, if necessary
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	User claim report sent to help desk
	i) Unit of measurement for the QME	Number of problems
	j) Numerical rules	Count the number of problems for individual severity level
	k) Scale type	Ratio
	l) Context of QME	This QME is usable to QMs applied during operation for usability, reliability and satisfaction.
	m) Software life cycle process(es):	Operation
	n) Measurement constraints (optional)	-to be completed, if necessary
3	**a) QME name**	**Number of records**
	b) Target entity	Data items treated as unit or records
	c) Objectives and property to quantify	It is used to quantify the complexity of a database. Definition of **record**: a set of related data items treated as a unit. (ISO/IEC 24765:2010 Systems and software engineering vocabulary)
	d) Relevant quality measure(s)	A large number of records can affect the maintainability.
	e) Measurement method	List up records and count up them
	f) List of sub properties related to the property to quantify (optional)	-to be completed, if necessary
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	Data items
	i) Unit of measurement for the QME	Each record
	j) Numerical rules	Adding each record
	k) Scale type	Ratio
	l) Context of QME	This QME is usable to QMs for data quality characteristics.
	m) Software life cycle process(es):	Maintenance

NO	QME	Description
	n) Measurement constraints (optional)	Need to obtain data items
4	**a) QME name**	**Duration**
	b) Target entity	A time period
	c) Objectives and property to quantify	Definition of **duration**: 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 days weeks or months. Sometimes incorrectly equated with elapsed time.(A Guide to the Project Management Body of Knowledge, (PMBOK® Guide) — Fourth Edition)
	d) Relevant quality measure(s)	This QME is useful to QMs for performance efficiency characteristic. This is also useful to QMs such as time to complete user's intended task, throughput, mean time between failures or mean time to repair which is using time period spent by operator, user, maintainer or system.
	e) Measurement method	Duration is based on the definition of the base quantity of time and related to the International System of Quantities (VIM)
	f) List of sub properties related to the property to quantify (optional)	-to be completed, if necessary
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	From time sheet in the organisation
	i) Unit of measurement for the QME	Days, weeks or months
	j) Numerical rules	Accumulating the work periods
	k) Scale type	Ratio
	l) Context of QME	This QME is usable for all QMs relating duration time, such as MTBF and average throughput per unit time for reliability and performance efficiency. Besides, combination with effort provides productivity measure.
	m) Software life cycle process(es):	All
	n) Measurement constraints (optional)	-to be completed, if necessary
5	**a) QME name**	**Effort (in unit of time)**
	b) Target entity	Effort in hours or days
	c) Objectives and property to quantify	For productivity measures in hours. Definition of **effort**: The number of labor units required to complete a scheduled activity or work breakdown structure component. Usually expressed as hours, days, or weeks. ('A Guide to the Project Management Body of Knowledge' (PMBOK® Guide) — Fourth Edition)
	d) Relevant quality measure(s)	This QME is useful to QMs for performance efficiency characteristic. This is also useful to QMs such as effort to complete user's intended task, effort for system recovery, or effort for maintenance which is using people effort spent by operator, user, developer, tester or maintainer.
	e) Measurement method	Effort is based on the definition of the base quantity of time and related to the International System of Quantities (VIM)

NO	QME	Description
	f) List of sub properties related to the property to quantify (optional)	-to be completed, if necessary
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	Time sheet from the organisation
	i) Unit of measurement for the QME	Generally in hours and days
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME for all QMs relating effort for especially efficiency, effectiveness, reliability, performance efficiency, usability and maintainability. Besides, combination with effort provides productivity measure or estimation.
	m) Software life cycle process(es):	All
	n) Measurement constraints (optional)	-to be completed, if necessary
6	**a) QME name**	**Number of system failures**
	b) Target entity	System failure
	c) Objectives and property to quantify	'System Failure Count' is intended to be used in the quality (derived) measures such as reliability, efficiency and quality of the software and applicable to system engineering and software engineering and management disciplines. Property if failure. Definition of **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.
	d) Relevant quality measure(s)	QMs such as system failure frequency or MTBF during testing or operation for reliability and efficiency of quality in use.
	e) Measurement method	Adding the number of System failures
	f) List of sub properties related to the property to quantify (optional)	to be completed, if necessary
	g) Definition of each sub property (optional)	**Structural System Failures**: Subsystem, aspect systems and phase systems not performing assigned functions **Software / Hardware System Failures**: These are malfunctions which are caused by problem of design and human errors. **Decision Making System Failures**: This type of failures refers to the mismatch between structure of organization and environment demands and also the mismatch between values and world view of decision makers and environment.
	h) Input for the QME	See definition of each sub properties (malfunctions, mismatch, non performance, etc,)
	i) Unit of measurement for the QME	Each failure
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	Related to reliability, efficiency of the system.

NO	QME	Description
	m) Software life cycle process(es):	Testing, operation and maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary
7	**a) QME name**	**Number of failures**
	b) Target entity	Failure
	c) Objectives and property to quantify	Intended to be used in measuring the maintainability and reliability of software. Definition of **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.
	d) Relevant quality measure(s)	QMs such as failure density or failure frequency during testing or operation for maturity of reliability
	e) Measurement method	"Quantification of the Software Failures"
	f) List of sub properties related to the property to quantify (optional)	There are a number of categories of failures. **Critical Failure**: It results in the termination of the software program, entire system stops performing. It should not be mixed with the down of the system since it is related with the software itself not with the hardware. **Serious Failure**: Important functions of the software become inoperable and with no alternative way of operating. Average Failure: Most functions are still available, but limited performance occurs with limited or alternate operation. **Small Failure:** A few functions experience limited performance with limited operation. It can be an absence of a variable that should be shown at the output, but the failure does not cause serious problems at all. Four categories of the software failures are divided into two sub concepts according to their dissolution state: **Resolved Failure**: The failure is detected and resolved afterwards. **Unresolved (Actually Detected) Failure**: The failure is detected but not resolved.
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	Log of failures within an organisation
	i) Unit of measurement for the QME	Failures by categories
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for reliability, maintainability, and portability.
	m) Software life cycle process(es):	Implementation and Maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary
8	**a) QME name**	**Number of faults**
	b) Target entity	Fault
	c) Objectives and property to quantify	The aim of this QME is to measure the software faults. The result can be used in reliability evaluation, estimating possible number of faults in the end of a project, ratio of faults between finished projects, gathering data for future projects. The result of this measurement can also be useful in quality assessment. Definition of **fault**: an incorrect step, process, or data definition in software code. NOTE A fault, if encountered, may cause a failure

NO	QME	Description
	d) Relevant quality measure(s)	QMs such as fault density during reviewing, modifying, testing for fault tolerance, maturity or testability
	e) Measurement method	Software fault quantification method
	f) List of sub properties related to the property to quantify (optional)	to be completed, if necessary
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	Log of faults within an organisation
	i) Unit of measurement for the QME	Could be a manifestation of an error in software or an incorrect step, process, or data definition in a computer program or a defect in a hardware device or component or a software fault, also known as a "crash" or "abend" is when the program directs the computer to go outside of its restricted memory boundary.
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for fault tolerance, maturity, modifiability and testability. QMs for such quality subcharacteristics are related to the quality of the code.
	m) Software life cycle process(es):	**C**oding, testing, maintenance
	n) Measurement constraints (optional)	**-** to be completed, if necessary
9	**a) QME name**	**Functional size of a product**
	b) Target entity	Requirements specifications. Details depend on the measurement method.
	c) Objectives and property to quantify	To measure the functional requirements of what user ask for. Definition of **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)
	d) Relevant quality measure(s)	This QME is useful to QMs for performance efficiency characteristic. This QME is also useful to normalize a value and calculate a density for the comparison of QM, such as fault density per functional size.
	e) Measurement method	Basically analyze functional user requirements in specification, categorize their record type and score them in accordance with weighted functions. (See ISO/IEC 14143-1:2007) NOTE There are typical four measurement methods.
	f) List of sub properties related to the property to quantify (optional)	to be completed, if necessary
	g) Definition of each sub property (optional)	to be completed, if necessary
	h) Input for the QME	Depend on the measurement method. Based generally on requirements.
	i) Unit of measurement for the QME	Depend on the measurement method
	j) Numerical rules	Depend on the measurement method
	k) Scale type	Ratio
	l) Context of QME	This QME is to size the software and is useful to calculate density. Then it is usable especially to maturity and other quality characteristics represented by QM using density.
	m) Software life cycle process(es):	Requirements analysis and the following phases
	n) Measurement constraints (optional)	To have the correct documentation.

NO	QME	Description
10	**a) QME name**	**Number of interruptions**
	b) Target entity	Interrupt
	c) Objectives and property to quantify	How much control a user has on the operation and how this affects the use of the software. Definition of **interrupt**: the suspension of a process to handle an event external to the process. (ISO/IEC 24765:2010 Systems and software engineering vocabulary)
	d) Relevant quality measure(s)	QMs such as appropriate frequency of interruptions by user represents user can control the system or software, but extreme frequent interruptions by users represents user's anxious and inconvenience for quantifying operability of usability, efficiency and effectiveness of quality in use.
	e) Measurement method	Monitor and count interruption events during operation
	f) List of sub properties related to the property to quantify (optional)	Event
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	Log of operations
	i) Unit of measurement for the QME	Interruption
	j) Numerical rules	Add
	k) Scale type	Ratio
	l) Context of QME	This QME quantifies control of operations and is usable especially for usability and operability.
	m) Software life cycle process(es):	Testing, Operation and Maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary
11	**a) QME name**	**Number of data items**
	b) Target entity	Data items
	c) Objectives and property to quantify	It measures the size of the structure of the database. See also number of records. Definition of **data item**: 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.
	d) Relevant quality measure(s)	QM such as degree of available data items which can be used even after modification or porting.
	e) Measurement method	Data items
	f) List of sub properties related to the property to quantify (optional)	- to be completed, if necessary
	g) Definition of each sub property (optional)	- to be completed, if necessary
	h) Input for the QME	Software requirement specifications, software design specifications, software manuals, the source code, the database schema (if applicable), can be used to identify data items in the software
	i) Unit of measurement for the QME	Data items
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for maintainability and portability.
	m) Software life cycle process(es):	From software requirements analysis through maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary

NO	QME	Description
12	**a) QME name**	**Number of error messages**
	b) Target entity	Error messages
	c) Objectives and property to quantify	To know if a system is reliable and secure enough. Definition of error message: a message that the application gives when incorrect data is entered or when another processing error occurs.
	d) Relevant quality measure(s)	QMs such as degree of detectable input data error, degree of detectable unauthorized system access, or degree of avoidable user operational error for qualifying reliability, security and usability.
	e) Measurement method	Error messages
	f) List of sub properties related to the property to quantify (optional)	- to be completed, if necessary
	g) Definition of each sub property (optional)	- to be completed, if necessary
	h) Input for the QME	-to be completed, if necessary
	i) Unit of measurement for the QME	Number of error messages
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for reliability, security and usability.
	m) Software life cycle process(es):	Implementation and Maintenance
	n) Measurement constraints (optional)	- to be completed, if necessary
13	**a) QME name**	**Number of errors**
	b) Target entity	Error
	c) Objectives and property to quantify	The method for error measurement can be used for testing (maintainability) Definition of **errors**: (1) A human action that produces an incorrect result, such as software containing a fault. (ISO/IEC 24765:2010 Systems and software engineering vocabulary) (2) An incorrect step, process, or data definition.(ISO/IEC 24765:2010 Systems and software engineering vocabulary) (3) An incorrect result. (ISO/IEC 24765:2010 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 24765:2010 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
	d) Relevant quality measure(s)	QMs such as number of errors per thousand of test cases during testing
	e) Measurement method	Error quantification
	f) List of sub properties related to the property to quantify (optional)	Compile time errors, link time errors, and run time errors.
	g) Definition of each sub property (optional)	**Compile time errors**: Error occurs while translating program from source code into machine code. These are generally, syntax errors, type check errors and template instantiating errors. **Link time errors**: Error occurs when linking compiled source codes. At that time, addresses of externally referenced variables and classes are fixed; type checks for externally referenced variables are done. Any error occurring at those steps identified as link time error.

NO	QME	Description
		Run time errors: Error occurs while program is executing. Run time errors usually indicate bugs in the program, or it could also be caused by running out of memory. Run time errors result in unexpected behaviour such as crashing the program or giving wrong output and etc.
	h) Input for the QME	Source code for compile errors
	i) Unit of measurement for the QME	# of compile time errors, # of link time errors, # of run time errors
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for reliability, security, usability and maintainability.
	m) Software life cycle process(es):	Testing and maintenance
	n) Measurement constraints (optional)	Coding shall be finished. In order to be able to detect compile time errors all codes has to be finished. For link time errors all compile time errors have to be identified and fixed. For run time errors all link time errors have to be fixed and program should be able to run.
14	**a) QME name**	**Number of messages**
	b) Target entity	Messages
	c) Objectives and property to quantify	Non encrypted messages can help to better use a software and learn it faster. For this reason the number of messages a human user can understand is important. However this could be put in perspective with the size of the software. Definition of **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 Modelling Language Syntax and Semantics for IDEF1X97 (IDEF object)). Message could be encrypted (mainly for a machine) or easily identified by a human user. NOTE Message encompasses requests to meet responsibilities as well as simple informative communications.
	d) Relevant quality measure(s)	QMs such as degree of guidable beginner user operation for usability (learnability). Sometime this can affect operability and aesthetic.
	e) Measurement method	Message Quantification
	f) List of sub properties related to the property to quantify (optional)	Functional Message Process (FMP) is one concept that can be expressed in a model with different sub properties like Information Message, Status Message, Warning Message and Error Message.
	g) Definition of each sub property (optional)	**Information Message**: A message from a computer system or application program that has no requirement for operator intervention. **Status Message**: A message from a computer system or application program that has no requirement for operator intervention, but is reporting current operational state of the reporting system. **Warning Message**: A message from a computer system or application program that indicates an event or problem during operation that does require operator intervention. This is almost always an indicator of some kind of no-good thing happening. In computer science it is a diagnostic message that is issued when a computer program detects an error or potential problem but continues processing. Generally a *warning message* is a modal dialog box, in-place message, notification, or balloon that alerts the user of a condition that might cause a problem in the future. **Error Message**: A message from a computer system or application program that indicates a significant event or problem during operation that does require operator intervention. This is almost always an indicator of some kind of very no-good thing happening. An error message is information displayed when an unexpected condition occurs, usually on a computer or other device. On modern operating systems with graphical user interfaces, error messages are often displayed using dialog boxes. Error messages are used when user intervention is required, to indicate that a desired operation has failed, or to relay important warnings (such as warning a computer user that they are almost out of hard disk space). Error messages are seen widely throughout computing, and are part of every operating system or computer hardware device

NO	QME	Description
	h) Input for the QME	The main concept is extracting the messages from the software, which can be done by extracting the Functional User Requirements from the design document, then by eliminating non-message related Functional User Requirements and getting the Functional User Message Requirements. Each extracted Functional User Message Requirement is classified by identification of message and Functional Message Process.
	i) Unit of measurement for the QME	Messages
	j) Numerical rules	Adding number of Information message, number of status message, number of warning message, number error message.
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for usability (learnability).
	m) Software life cycle process(es):	Design, coding, testing, and maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary
15	**a) QME name**	**Number of steps(of procedure)**
	b) Target entity	Steps (of procedure)
	c) Objectives and property to quantify	To have an idea of the complexity of a procedure. Hypothesis: more steps are more complex. Definition of **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 multi set 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.
	d) Relevant quality measure(s)	QMs such as number of steps for user operation procedure or maintenance procedure for quantifying usability, efficiency and maintainability
	e) Measurement method	Count the identified steps
	f) List of sub properties related to the property to quantify (optional)	- to be completed, if necessary
	g) Definition of each sub property (optional)	-to be completed, if necessary
	h) Input for the QME	Different procedures from requirements and design
	i) Unit of measurement for the QME	Step
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for usability, efficiency and maintainability.
	m) Software life cycle process(es):	From requirements analysis through maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary
16	**a) QME name**	**Task complexity**
	b) Target entity	Task

NO	QME	Description
	c) Objectives and property to quantify	To know the complexity of a task. The complexity of the software product can affect the software usability. Reliability and maintainability mainly. Determining the complexity of a specific task shows us the difficulty of performing this task. Definition of **task**: A task is a function that needs to be accomplished within a defined period of time. In this work, we look at software perspective and the function mentioned here is the software itself. Institute of Electrical and Electronics Engineers (simply IEEE) defines complexity as "the degree to which system or component has a design or implementation that is difficult to understand and verify". In this study interactor is user and difficulty of performing task is data or request inputting the software program and result extraction from the system. NOTE Task: (1) in software design, a software component that can operate in parallel with other software components. (2) concurrent object with its own thread of control. (3) sequence of instructions treated as a basic unit of work by the supervisory program of an operating system (ISO/IEC/IEEE 24765:2010 Systems and software engineering-Vocabulary). (4) 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) (ISO/IEC 15288:2008 Systems and software engineering — System life cycle processes)
	d) Relevant quality measure(s)	QMs such as degree of program task complexity to manipulate user interface functions or degree of program task complexity for quantifying usability, reliability and maintainability.
	e) Measurement method	Task complexity
	f) List of sub properties related to the property to quantify (optional)	Variables and arguments
	g) Definition of each sub property (optional)	The number of variables defined and used in a software system. A source code which uses more variables is harder to understand, debug and maintain. The number of arguments (parameters) involved in each function call in the source code. Function arguments (parameters) can be considered variables. A function with a large number of parameters is hard to understand, debug, and maintain.
	h) Input for the QME	In a statement, each variable used will contribute one point to complexity index. For function parameters: In a statement, each method/function invocation will contribute n points to complexity index (n = the number of arguments)
	i) Unit of measurement for the QME	Task complexity
	j) Numerical rules	As example: low, medium and high
	k) Scale type	Ordinal
	l) Context of QME	This QME is usable especially for usability, reliability and maintainability.
	m) Software life cycle process(es):	From design through maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary
17	**a) QME name**	**Number of test cases**
	b) Target entity	Test case
	c) Objectives and property to quantify	Aim to quantify the Pass/Fail Test. Definition of **test case**: a minimal independently executable part of the test suite of a software system which gives two possible results [fail, pass]. More precisely test case is: (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)

NO	QME	Description
	d) Relevant quality measure(s)	QMs such as number of test cases automatically executable, or number of detected failures per thousand of error handling test cases for quantifying maintainability (testability) and reliability (fault tolerance).
	e) Measurement method	test cases
	f) List of sub properties related to the property to quantify (optional)	-to be completed, if necessary
	g) Definition of each sub property (optional)	- to be completed, if necessary
	h) Input for the QME	Tests scenario and results
	i) Unit of measurement for the QME	Test cases
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for maintainability (testability) and reliability (fault tolerance).
	m) Software life cycle process(es):	Testing
	n) Measurement constraints (optional)	- to be completed, if necessary
18	**a) QME name**	**Number of use cases**
	b) Target entity	Use cases
	c) Objectives and property to quantify	It is a size measure of a use case. Mainly use in a productivity model it can help to quantify the relative resource utilisation. How much resource use a use case related to its size. Definition of **use case**: 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 24765:2010 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 24765:2010 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.
	d) Relevant quality measure(s)	QMs such as specified number of use cases for completing specific user intended task, for mitigating specific risks, for operating by specific sort of disabled user for quantifying the product capability of freedom from risk, effectiveness, context coverage and usability. It is also useful for QMs related to performance efficiency (resource utilisation).
	e) Measurement method	The rules to find the different type of actions for different scenarios are the following: • Use the sequential ordering of action descriptions (and hence their unique number identifiers) to indicate strict sequence between actions for the main scenario • Iterations and concurrent actions can be expressed in the same section of the Use Case, whereas alternative actions should be written in a different section. For alternative paths and extensions look all the possibilities NOTE Find number of Use Cases to measure the software and find relations. Sometimes it is better to use conditional logic or activity diagrams to describe Use Case with many rules and conditions.
	f) List of sub properties related to the property to quantify (optional)	Action
	g) Definition of each sub property (optional)	**Action:** (1) “a defined body of work to be performed, including its required input and output information”. (2) Set of cohesive tasks of a process (ISO/IEC 12207:2008 Systems and software engineering--Software life cycle processes)

NO	QME	Description
	h) Input for the QME	Use cases
	i) Unit of measurement for the QME	Action
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for freedom from risk, effectiveness, context coverage and usability.
	m) Software life cycle process(es):	Requirements
	n) Measurement constraints (optional)	- to be completed, if necessary
19	**a) QME name**	**Number of operations**
	b) Target entity	Operation
	c) Objectives and property to quantify	To provide a measure of operational complexity for highly interactive high volume use software. In many environments operators are faced with the task of repetitively using the same feature sets of software applications. In such environments the minimization of operational complexity may result in significant reduction of fatigue and increase in usability and efficiency of quality in use. Typically, such operators have extensive experience with the software and choose the most effort efficient way to interact with the software in order to perform the operations that constitute their tasks. Definition of an **operation**: ongoing execution of activities that produce the same product or provide a repetitive service. More precisely an operation is: (1) The process of running a computer system in its intended environment to perform its intended functions. (ISO/IEC 24765:2010 Systems and software engineering vocabulary) (2) An action needed to perform an activity.(ISO/IEC 15940:2006 Information Technology — Software Engineering Environment Services)
	d) Relevant quality measure(s)	Degree of free from operational error which compare the number of occurred operational error to the total number of operations during the operation for quantifying usability (operation)
	e) Measurement method	Operation
	f) List of sub properties related to the property to quantify (optional)	-to be completed, if necessary
	g) Definition of each sub property (optional)	- to be completed, if necessary
	h) Input for the QME	List of controls which the operator will operate
	i) Unit of measurement for the QME	Operation
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for usability and efficiency of quality in use.
	m) Software life cycle process(es):	Software procurement - non functional requirement specification: As a usability requirement threshold. -Software procurement - COTS selection/comparison: As a selection factor. -Software development - UI design: As pre-determination of operational complexity. - Testing - non functional requirement compliance: As a comparison basis with respect to requirements. -Maintenance - Improvement opportunity determination: As a UI improvement identification facility
	n) Measurement constraints (optional)	- to be completed, if necessary

NO	QME	Description
20	**a) QME name**	**Number of fatal errors**
	b) Target entity	Fatal errors
	c) Objectives and property to quantify	To know the reliability of a software. Definition of **a fatal error**: an error that results in the complete inability of a system or component to function. (ISO/IEC 24765:2010 Systems and software engineering vocabulary)
	d) Relevant quality measure(s)	Number of fatal errors arising system down for representing fault tolerance, recoverability or number of data restorable fatal errors for representing recoverability in reliability
	e) Measurement method	Monitor the system or the software during testing or operation and record the fatal error occurrences such as system down or service suspended.
	f) List of sub properties related to the property to quantify (optional)	- to be completed, if necessary
	g) Definition of each sub property (optional)	- to be completed, if necessary
	h) Input for the QME	Testing and operation
	i) Unit of measurement for the QME	Fatal errors
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	This QME is usable especially for reliability (fault tolerance, recoverability) and freedom from risk.
	m) Software life cycle process(es):	Testing and Operation
	n) Measurement constraints (optional)	- to be completed, if necessary
21	**a) QME name**	**Size of a database**
	b) Target entity	Database
	c) Objectives and property to quantify	Definition of the **size of a database**: Number of occurrences in a database. This QME can be helpful to measure the installability of a new software when data need to be transferred. A database is: (1) A collection of interrelated data stored together in one or more computerized files. (ISO/IEC 24765:2010 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) A collection of data describing a specific target area that is used and updated by one or more applications. (ISO/IEC 29881:2008 Information technology — Software and systems engineering — FiSMA 1.1 functional size measurement method)
	d) Relevant quality measure(s)	Portability (installability)
	e) Measurement method	Database size
	f) List of sub properties related to the property to quantify (optional)	Occurrence
	g) Definition of each sub property (optional)	--to be completed, if necessary
	h) Input for the QME	Database
	i) Unit of measurement for the QME	Number of occurrences
	j) Numerical rules	Adding
	k) Scale type	Ratio

NO	QME	Description
	l) Context of QME	- This QME is usable especially for portability (installability)
	m) Software life cycle process(es):	Maintenance
	n) Measurement constraints (optional)	-to be completed, if necessary
22	**a) QME name**	**Size of a memory**
	b) Target entity	Memory size
	c) Objectives and property to quantify	To know the required amount of volatile and permanent memory that a software needs to run properly. Definition of the size of a memory: Amount of computer or device of a computer storage (express in multiple of bytes). Definition of 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)
	d) Relevant quality measure(s)	Performance efficiency (resource utilisation)
	e) Measurement method	Memory size
	f) List of sub properties related to the property to quantify (optional)	Permanent memory, Volatile Memory, RAM, Video Memory, Memory Profiler
	g) Definition of each sub property (optional)	**Permanent memory:** the place where data is held in an electromagnetic or optical form for access by a computer processor. Storage is frequently used to mean the devices and data connected to the computer through input/output operations - that is, hard disk and tape systems and other forms of storage that don't include computer memory and other in-computer storage. **Volatile Memory:** Computer memory that requires power to maintain the stored information, unlike storage memory which does not require a maintained power supply. **RAM:** Random-access memory (RAM) is a form of volatile memory. Today, it takes the form of integrated circuits that allow stored data to be accessed in any order (i.e., at random). "Random" refers to the idea that any piece of data can be returned in a constant time, regardless of its physical location and whether it is related to the previous piece of data. **Video Memory:** A form of volatile memory installed on a video adapter. Before an image can be sent to a display monitor, it is first represented as a bit map in an area of video memory called the frame buffer. The reason of handling video memory as a seperate sub concept is to determine graphics memory requirements of programs more accurately. **Memory Profiler:** The tool to investigate of a program's memory patterns using information gathered as the program executes.
	h) Input for the QME	Memory usage is pretty unique feature to high-level programming languages. It makes it possible to create the objects of any size during program executing at run time. The size of memory assignment matters the performance and size of the program hence the quality of the software. Programmer is responsible for keeping track of memory that is allocated at run time and free it whenever it is not required any longer. The maximum memory space, which is peak point, is necessary to highlight the minimum amount of physical memory the software requires. It is provide by a program mostly within the operation system.
	i) Unit of measurement for the QME	The unit of measure will be bytes. Kilobytes (KB), Megabytes (MB), and Gigabytes (GB) can also be used as unit of measure as they are 1024 multiples of byte unit.
	j) Numerical rules	Adding
	k) Scale type	Ratio
	l) Context of QME	resource utilisation
	m) Software life cycle process(es):	Maintenance mainly
	n) Measurement constraints (optional)	-to be completed, if necessary

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 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. The user of this standard could decide to complete the 'table format of QMEs' in 6.2 without considering this text if not interested of the logic behind the completion of the table that is explain in this informative Annex.

As Figure 4 (see 6.1) 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.1)
2. identification of the property to quantify related to the QME (B.2)
3. definition of the property and sub properties(B.3)
4. construction of the model of the properties to be quantified (B.4)
5. assignment of the unit of measurement (formula) and scale type (B.5)

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 applied to implement a measurement method that 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 subcharacteristic. This not excludes the usage of a QME for different characteristics and subcharacteristics 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 QMEs were extracted from the quality measures (about 250) in the ISO/IEC 9126 series, part 2, 3 and 4. For each QME listed, a property (of a product) was identified. This work of identifying QME implies continuous update because it depends also of new identified characteristics and subcharacteristics (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 the 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, target entity, objectives and property to quantify, relevant quality measure(s) (it is used by what quality measure), measurement method, list of sub properties related to the property to quantify, definition of each sub property, input for the QME, numerical rules for the QME, the scale type, context of QME, software life cycle and measurement constraints. 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. The software life cycle phases may change based on the type of the software life cycle. In ISO/IEC 12207:2008, basic life cycle phases are requirements analysis, design, coding, testing, and maintenance (ISO/IEC 12207:2008, Systems and Software Engineering — Software Life Cycle Processes).

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

For the organisation, the choice of a 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 subcharacteristics the organisation wants to quantify?). Each QME used by an organisation can be defined by the organisation if not already define by an International Standard.

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 decomposed 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) identified 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).