ISO/IEC JTC 1 N 9638

2009-07-16

ISO/IEC JTC 1
Information Technology

Document Type:	**Proposed NP**
Document Title:	**NWIP, Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Quality measure elements**
Document Source:	**SC 7**
Reference:	**Resolution 1169**
Document Status:	**This document is circulated to JTC 1 National Bodies for concurrent review. If the JTC 1 Secretariat receives no objections to this proposal by the due date indicated, we will so inform the SG 7 Secretariat**
Action ID:	**ACT**
Due Date:	**2009-10-16**
No. of Pages:	**57**

ISO/IEC JTC1/SC7
Software and Systems Engineering
Secretariat: CANADA (SCC)

ISO/IEC JTC1/SC7 N4379

2009-07-16

Document Type	**NWIP**
Title	NWIP, Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality measure elements
Source	WG6
Project	**25021**
Status	**NWIP**
References	Resolution 1169
Action ID	ACT
Due Date	2009-10-16
Start Date	2009-07-16
Distribution	SC7 AG
Medium	PDF
No. of Pages	57
Note	Please vote using the ISO Electronic Balloting Facilities (Resolution 937)

Address reply to: ISO/IEC JTC1/SC7 Secretariat
École de technologie supérieure – Departement of Software and IT Engineering
1100 Notre Dame Ouest, Montréal, Québec Canada H3C 1K3
secretariat@jtc1-sc7.org

www.jtc1-sc7.org

New Work Item Proposal

July 2009

PROPOSAL FOR A NEW WORK ITEM

Date of presentation of proposal: 2009-07-16	Proposer: JTC1/SC7/WG6
Secretariat: Witold Suryn, JTC1/SC7 National Body	**ISO/IEC JTC 1/SC7** ISO/IEC JTC 1/SC7/WG6

A proposal for a new work item shall be submitted to the secretariat of the ISO/IEC joint technical committee concerned with a copy to the ISO Central Secretariat.

Presentation of the proposal - to be completed by the proposer.

Title (subject to be covered and type of standard, e.g. terminology, method of test, performance requirements, etc.) **Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality Measure Elements (QME)**

Scope **(and field of application) Introduction**

This is a major revision of ISO/IEC TR25021, which is intended to define an initial set of quality measure elements to be used throughout the software product life cycle.
This is a part of ISO/IEC SQuaRE series. The project is permitted to produce multiple standards with a range of numbers forming a family of related guide.
Quality Measure Elements (QMEs) shall be used throughout the software product life cycle for the purpose of Software Product Quality Requirement and Evaluation.

Purpose and justification –

Purpose

- ✓ To provide a set of QMEs
- ✓ To provide a guides to use a QME including a procedure for the measurement function (informative for 25022-23-24).
- ✓ To provide a procedure for develop QMEs for users' own purpose

Justification

Some important benefits from defining and using the Quality Measure Elements in this document are:

- ✓ To provide guidance for different organizations to develop their own QMEs following a defined procedure.
- ✓ To improve the consistency of measurement between different software within or across different organizations.
- ✓ To help identifying the set of *Quality Measure Elements* that are uniquely required to derive all the *quality measures* for a given characteristics or a sub-characteristics of a product.

Programme of work

If the proposed new work item is approved, which of the following document(s) is (are) expected to be developed?
__X__ a single International Standard (One of ISO/IEC 25000 SQuaRE series of standards)
___ more than one International Standard (expected number: ....)
____ a multi-part International Standard consisting of .... parts
____ an amendment or amendments to the following International Standard(s) ....
____ a technical report , type ....

And which standard development track is recommended for the approved new work item?

__X __a. Default Timeframe

_____b. Accelerated Timeframe

__ __c. Extended Timeframe

Relevant documents to be considered

These standards should be consistent and leverage from other standards and relevant documents. These standards will describe relationships to other standards and relevant documents, including:

ISO/IEC 9126 series Software engineering — Product quality

ISO/IEC 15939 Systems and software engineering — Measurement process

ISO/IEC 12207 Software Lifecycle Processes

ISO/IEC 25000 SQuaRE series including

- ISO/IEC 25020 IS Software engineering — Measurement reference model and guide
- ISO/IEC 25021 TR Software engineering — Quality Measure Elements

Co-operation and liaison Group WG-26 (Testing)
Preparatory work offered with target date(s): NP 06/30, WD-CDR-CD combined 11/09, FCD 06/2010
Signature: Motoei AZUMA, Convener, JTC1/SC7/WG6
Will the service of a maintenance agency or registration authority be required ....No............ - If yes, have you identified a potential candidate? .... - If yes, indicate name .... Are there any known requirements for coding? ....No............ -If yes, please specify on a separate page Does the proposed standard concern known patented items? . No............ - If yes, please provide full information in an annex

Comments and recommendations of the JTC 1 or SC 07 Secretariat - attach a separate page as an annex, if necessary

Comments with respect to the proposal in general, and recommendations thereon: It is proposed to assign this new item to JTC1 /SC7/WG6 (JTC1/SC7 o7N4349 Resolution Hyderabad 1149)

Voting on the proposal - Each P-member of the ISO/IEC joint technical committee has an obligation to vote within the time limits laid down (normally three months after the date of circulation).

Date of circulation: 2009-07-16	**Closing date for voting:** 2009-10-16	**Signature of Secretary:** W. Suryn

NEW WORK ITEM PROPOSAL - PROJECT ACCEPTANCE CRITERIA		
Criterion	**Validity**	**Explanation**
A. Business Requirement		
A.1 Market Requirement	Essential __X_ Desirable ___ Supportive ___	
A.2 Regulatory Context	Essential ___ Desirable ___ Supportive ___ Not Relevant X	
B. Related Work		
B.1 Completion/Maintenance of current standards	Yes _X__ No	Major revision of ISO/IEC TR 25021

B.2 Commitment to other organisation	Yes ___ No X	
B.3 Other Source of standards	Yes X No___	ISO/IEC 25000 SQuaRE series ISO/IEC 9126-2,-3,-4 ISO/IEC 15939
C. Technical Status		
C.1 Mature Technology	Yes ___ No X	
C.2 Prospective Technology	Yes X No___	
C.3 Models/Tools	Yes X No___	
D. Conformity Assessment and Interoperability		
D.1 Conformity Assessment	Yes ___ No X	
D.2 Interoperability	Yes ___ No X	
E. Cultural and Linguistic Adaptability	**Yes**____ No X	
F. Other Justification		

Notes to Proforma

A. Business Relevance. That which identifies market place relevance in terms of what problem is being solved and or need being addressed.

A.1 Market Requirement. When submitting a NP, the proposer shall identify the nature of the Market Requirement, assessing the extent to which it is essential, desirable or merely supportive of some other project.

A.2 Technical Regulation. If a Regulatory requirement is deemed to exist - e.g. for an area of public concern e.g. Information Security, Data protection, potentially leading to regulatory/public interest action based on the use of this voluntary international standard - the proposer shall identify this here.

B. Related Work. Aspects of the relationship of this NP to other areas of standardisation work shall be identified in this section.

B.1 Competition/Maintenance. If this NP is concerned with completing or maintaining existing standards, those concerned shall be identified here.

B.2 External Commitment. Groups, bodies, or fora external to JTC 1 to which a commitment has been made by JTC for Co-operation and or collaboration on this NP shall be identified here.

B.3 External Std/Specification. If other activities creating standards or specifications in this topic area are known to exist or be planned, and which might be available to JTC 1 as PAS, they shall be identified here.

C. Technical Status. The proposer shall indicate here an assessment of the extent to which the proposed standard is supported by current technology.

C.1 Mature Technology. Indicate here the extent to which the technology is reasonably stable and ripe for standardisation.

C.2 Prospective Technology. If the NP is anticipatory in nature based on expected or forecasted need, this shall be indicated here.

C.3 Models/Tools. If the NP relates to the creation of supportive reference models or tools, this shall be indicated here.

D. Conformity Assessment and Interoperability

D.1 Indicate here if Conformity Assessment is relevant to your project. If so, indicate how it is addressed in your project plan.

D.2 Indicate here if Interoperability is relevant to your project. If so, indicate how it is addressed in your project plan

E. Cultural and Linguistic Adaptability   Indicate here if cultural and linguistic adaptability is applicable to your project. If so, indicate how it is addressed in your project plan.

F. Other Justification   Any other aspects of background information justifying this NP shall be indicated here

DRAFT only with Review Comments from Co-editor

ISO/IEC JTC1/SC7/WG6

Date: November-2008

ISO/IEC 25021

TITLE: **Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality measure elements**

DATE:	June-2009
SOURCE:	JTC1/SC7/WG6
WORK ITEM:	Project
STATUS:	Version 1.2
DOCUMENT TYPE:	Pre-WD Revision of TR-25021
PROJECT EDITOR:	Dr. Jean-Marc Desharnais (Canada)
CO-EDITORS:	Dr. Keum-Suk Lee (Korea)
	Dr. Jiri Vanicek (Czech Republic)
	Mr. Atsushi Yamada (Japan)

=

ISO/IEC TR 25021:2006(E)

ISO/IEC JTC 1/SC 7/WG 6

Secretariat:

Software Engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality Measure Elements

Contents

Page

Foreword

ISO (the International Organisation 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 organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, 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.

In exceptional circumstances, the joint technical committee may propose the publication of a TR of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard (“state of the art”, for example).

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, which is a Technical Report of type 2, was prepared 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. For now it is call ‘document’.

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

Quality Model Division,
Quality Measurement Division,
Quality Requirements Division, and
Quality Evaluation Division

Introduction

The purpose of this *document* is to list, define and/or design an initial set[1] of Quality Measure Elements (QME) to be used throughout the software product life cycle for the purpose of Software Product Quality Requirement and Evaluation (SQuaRE). The content of this document constitutes the link between the ISO/IEC 9126 series of standards and the subsequent SQuaRE series of standards.

The necessary QMEs for quality measures that quantify some of the characteristics and sub-characteristics 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. When evaluating selected quality measures, the user should first understand the definition of each attribute of the quality measure element(s) listed in this document.

Some important benefits from defining and using the quality measures elements in this document are:

- To provide guidance for different organisations to develop their own QMEs following a procedure to implement the measurement method as defined in this document.
- To improve the consistency of measurement between different software within or across different organisations.
- To help identifying the set of *quality measure elements* that are uniquely required to derive all the *quality measures* for a given characteristics or a sub-characteristics of a product.

The *quality measure elements* are the common components of a number of quality measures. The intended usage of this document is that users will select measures from *quality measure elements* to define internal, external or quality in use quality measures. Then, these can be used for quality requirements definition, software 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.

[1] Note The justification of the selection of the initial set of QMEs is explained in annex A

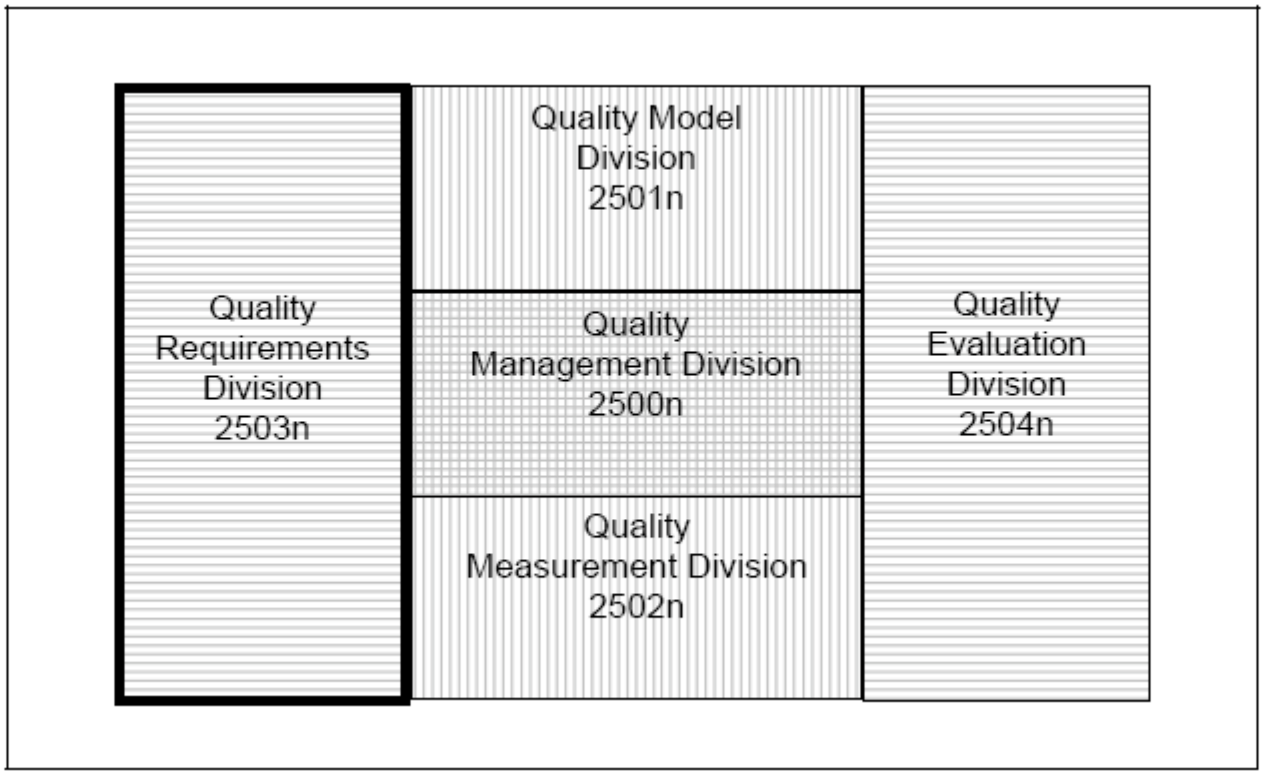

Quality Model Division 2501n
Quality Requirements Division 2503n
Quality Management Division 2500n
Quality Evaluation Division 2504n
Quality Measurement Division 2502n

Figure 1 — Organisation of SQuaRE series of standards

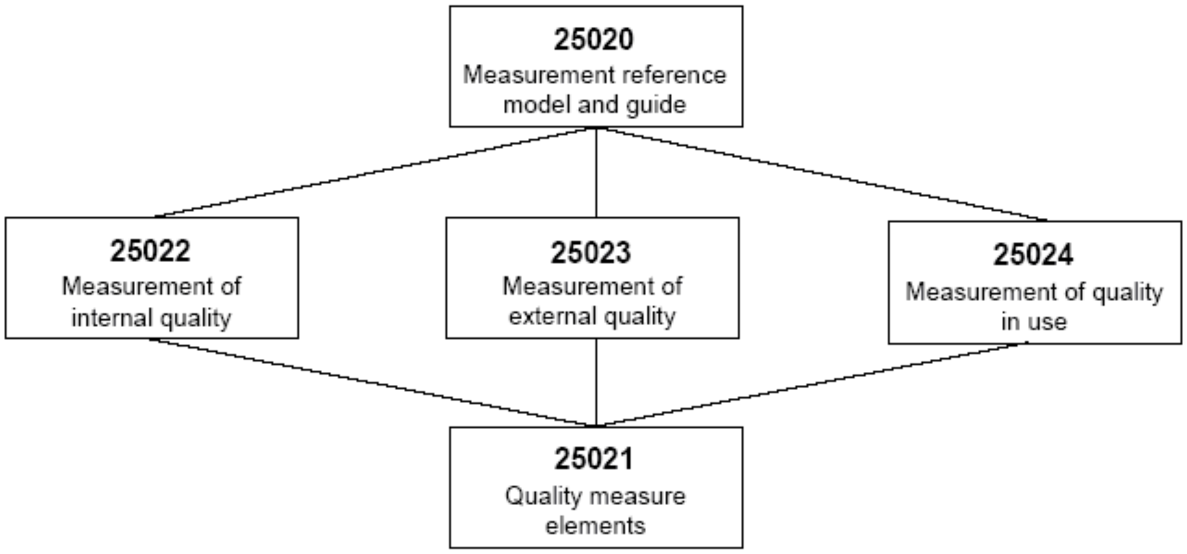

25020
Measurement reference model and guide
25022
Measurement of internal quality
25023
Measurement of external quality
25024
Measurement of quality in use
25021
Quality measure elements

Figure 2 — Structure of the Quality Measurement Division

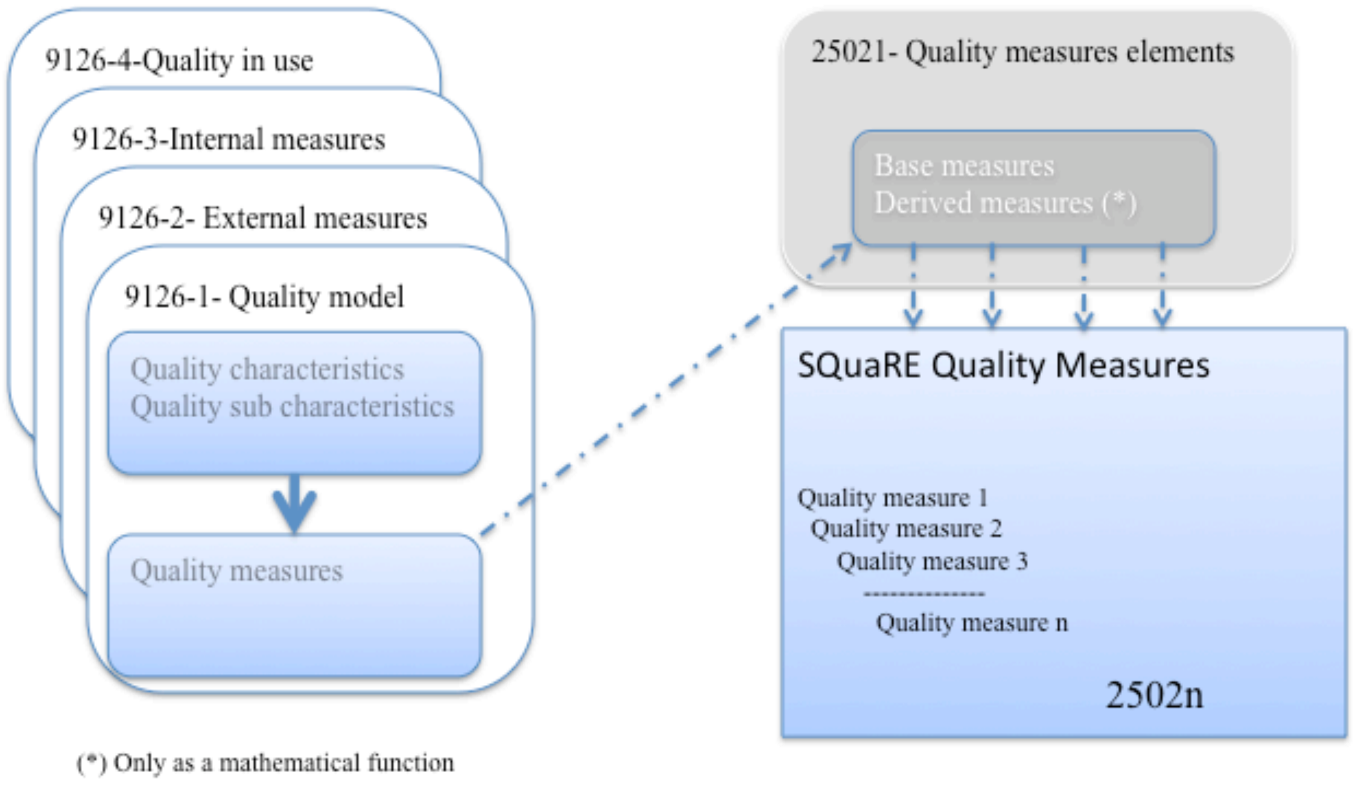


Figure 3 —The relationship of 25021 as a link between the 9126 series and the SQuaRE series of standards

ISO/IEC 9126 series is compose of 4 documents list and describe characteristics, sub characteristics and quality measures that refer as the quality model. The SQuaRE quality models categorise product quality into characteristics which are further subdivided into subcharacteristics and quality attributes (ISO/IEC 25010). For each quality measure within ISO/IEC 9126 series there are at least 2 quality measures elements (QMEs). The attributes can be measured by quality measure elements (ISO/IEC 25020). Another important document is ISO/IEC 15939 which link attribute and base measure (similar to QME) via a measurement method. The 2502n series list, design and describe QMEs and quality measures for the quality model.

1 Scope

This document provides a procedure to apply the measurement method. The application of a measurement method is recommend in ISO/IEC 15939 to generate, from an attribute, a *Quality Measure Element* (QME). It also provides a set of QMEs already designed and a list of QMEs that should be defined in the future. Finally, an informative annex (B), suggest a procedure for applying the measurement function to obtain quality measures. 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). This informative annex will be helpful when selecting and using different quality measures to evaluate the quality of the software product within the software product life cycle.

Every quality *measure* for internal quality measure, external quality measure and quality in use measure described in the SQuaRE series of standards is from measures that are selected from a set of quality measure elements in the ISO/IEC TR-9126 series.

This document contains:

- Conformance and normatives references
- Terms and definitions necessary for this document;
- Symbols (and abbreviation terms);
- Quality measure elements concepts in SPQM-RM
- Procedure to design a Quality Measure Elements (QME) and the measurement method table of information;
- An initial set of QMEs;

In Annex:

Annex A – Justification of the selection of the initial QMEs

Annex B – Procedure for the measurement function

Annex C Cross-reference table representing the Life cycle phases and the use of QMEs

Annex D Cross reference tables for quality measures

Annex E Measurement scale type

Annex F Bibliography

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

2 Conformance

Applicable QMEs (quality measurement elements) shall be associated with a defined set of QMEs in this document when such QMEs are used to construct QM (quality measure). When audiences of this document modify or develop and use their own QMEs, they should be defined and presented by following the procedure to design the measurement method for QME in this document.

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

ISO/IEC 25021 TR Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality measure elements

ISO/IEC FDIS15939: 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 25020, ISO/IEC 15939 and International Vocabulary of Basic and General Terms in Metrology apply. The following definitions are replicated here for the convenience of the user of this standard. Unattributed references are from ISO/IEC 25000.

4.1 Attribute

inherent property or characteristic of an entity that can be distinguished quantitatively or qualitatively by human or automated means

NOTE 1 based on ISO/IEC 15939:2007.

NOTE 2 ISO 9000 distinguishes two types of attributes: a permanent characteristic existing inherently in something and an assigned characteristic of a product, process or system (e.g. the price of a product, the owner of a product). The assigned characteristic is not an inherent quality characteristic of that product, process or system.

4.2 Base measure

measure defined in terms of an attribute and the method for quantifying it

NOTE A base measure is functionally independent of other measures. A base measure captures information about a single attribute.

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

4.3 Derived measure

measure that is defined as a function of two or more values of base measures. ISO/IEC 25000:2005 Software Engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE. 4.11. [ISO/IEC 15939:2002, based on the definition in International Vocabulary of Basic and General Terms in Metrology, 1993].

NOTE A transformation of a base measure using a mathematical function can also be considered as a derived measure.

4.4 Element

a component of a system; may include equipment, a computer program, or a human*. IEEE 1233-1998 (R2002) IEEE Guide for Developing System Requirements Specifications.*3.7
EXAMPLE documents, requirements specifications, test cases, source code, installation information, and read-me files

4.5 External software quality

capability of a software product to enable the behavior of a system to satisfy stated and implied needs when the system is used under specified conditions

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

EXAMPLE The number of failures found during testing is an external software quality measure related to the number of faults present in the program. The two measures are not necessarily identical since testing may not find all faults, and a fault may give rise to apparently different failures in different circumstances.

4.6 Indicator

measure that provides an estimate or evaluation of specified attributes from a model with respect to defined information needs

[ISO/IEC 15939:2002]

NOTE In ISO/IEC 14598 this definition was: "a measure that can be used to estimate or predict another measure".

4.7 Information need

insight necessary to manage objectives, goals, risks, and problems

[ISO/IEC 15939:2002]

4.8 Internal software quality

capability of a set of static attributes of a software product to satisfy stated and implied needs when the software product is used under specified conditions.

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

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

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

4.9 Measurer

an individual who apply a measurement method

4.10 Qualifier

any information from the measurer, that helps to determine the object to measure with the quality measures. For example, the qualifier tells if the quality measure address a part of a software or all the software or the part that is implemented or only the design part.

4.11 Derived measure

measure that is defined as a function of two or more values of base measures

[ISO/IEC 15939:2002, based on the definition in International Vocabulary of Basic and General Terms in Metrology, 1993].

NOTE A transformation of a base measure using a mathematical function can also be considered as a derived measure.

4.12 Measurable concept

abstract relationship between attributes of entities and information needs. [ISO/IEC 15939:2007, Systems and software engineering — Measurement process.3.14]

Note: within the context of 25021 the word attribute (of the QME) has a different meaning of attribute of entities. The latest is in the context of modelling.

4.13 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]

4.14 Measure (verb)

make a measurement

[ISO/IEC 14598-1:1999]

4.15 Measurement

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

[ISO/IEC 15939:2002, based on the definition in International Vocabulary of Basic and General Terms in Metrology, 1993]

NOTE Measurement can include assigning a qualitative category such as the language of a source program (ADA, C, COBOL, etc.).

4.16 Measurement function

algorithm or calculation performed to combine two or more base measures.

Note: In the context of square it is a combination of two or more QMEs to obtain a quality measure.

[ISO/IEC 15939:2002]

4.17 Measurement method

logical sequence of operations, described generically, used in quantifying an attribute with respect to a specified scale. [ISO/IEC 15939:2002, based on the definition in International Vocabulary of Basic and General Terms in Metrology, 1993].

4.18 Measurement procedure (look the difference and put a reference)

set of operations, described specifically, used in the performance of particular measurements according to a given method

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

4.19 Precision

the degree of exactness or discrimination with which a quantity is stated. ISO/IEC 24765, Systems and Software Engineering Vocabulary. See also: accuracy

4.20 QME design

a process that perform different steps from the definition of what we are measuring to an operational description of the procedure(s) to be used to obtain the unit (of measurement) of a QME.

4.21 Quality in use (measure)

the extent to which a product used by specific users meets their needs to achieve specific goals with effectiveness, productivity, safety and satisfaction in specific contexts of use

4.22 Quality measure

a measure that is used in the quality model.

4.23 Quality measure element

Measure, which is either a base measure or the mathematical function of a derived measure, that is used to construct a derive**d** measure

Note for the QME the combination of two based measures is exclude, it only include the transformation of a base measure using a mathematical function

4.24 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

4.25 Reproductibility (of results of measurement):

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

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

4.26 Rules

a single column through the condition and action entry parts of the decision table, defining a unique set of conditions to be satisfied and the actions to be taken in consequence. ISO 5806:1984 Information processing -- Specification of single-hit decision tables. 3.4.

NOTE A rule is satisfied if all conditions meet the condition entries of the rule.

4.27 Unit (of measurement)

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. 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.
NOTES
1 Units of measurement have conventionally assigned names and symbols.
2 Units of quantities of the same dimension may have the same names and symbols even when the quantities are not of the *same* kind.

5 Symbols (and abbreviated terms)

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

1) QME – *Quality measure element*

2) SPQM-RM - Software Product Quality Measurement Reference Model

6 Quality measure elements concept in SPQM-RM

Quality measure elements (QME) are used throughout the software product life cycle as an input for quality measures used in internal, external and quality in use measures listed in ISO/IEC TR 9126-2, ISO/IEC TR 9126-3 and ISO/IEC TR 9126-4.

SPQM-RM (Figure 4) shows the position of quality measure elements in the software product quality measures definition process. Any single quality measure element by itself will not indicate the quality measured of the entity. The quality measures of characteristics and sub characteristics are obtained by applying the results of quality measure elements.

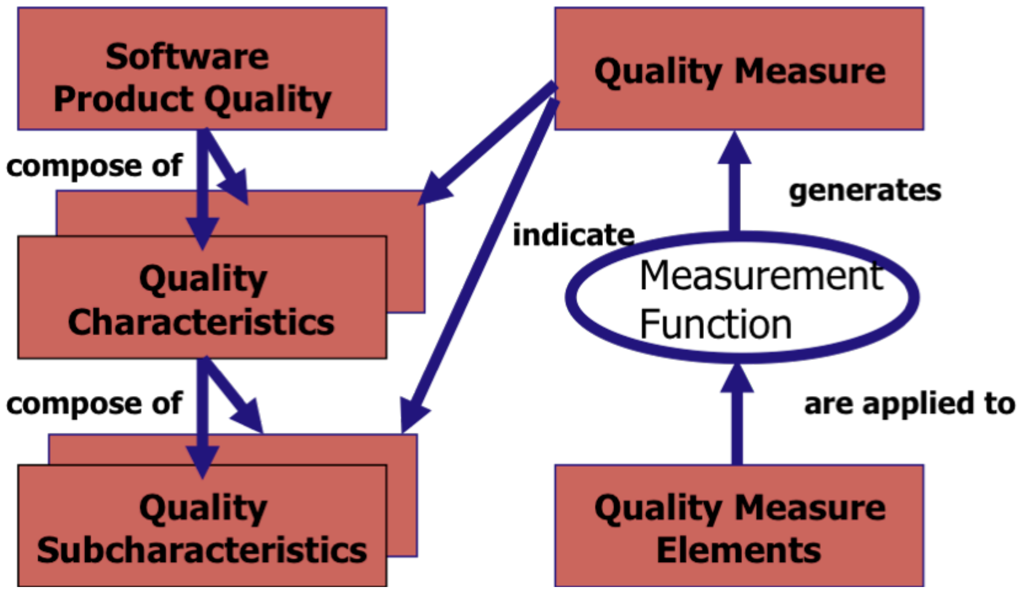


Figure 4 — Quality measure elements concept in SPQM-RM

A quality measure is generated from one or more quality measure elements (annex B). A quality measure element is an input to one or more quality measures. This relationship can be represented in the form of a two-dimensional cross-reference table (annex D), which allows a user of this document to select the proper quality measure elements that are necessary to generate a set of quality measures applying a measurement function (Annex B). The quality measures indicate the quality sub-characteristics and/or characteristics of the software product quality model.

7 Designing a Quality Measure Element (QME)

7.1 Introduction

Following the relations between attribute, measurement method and QME.

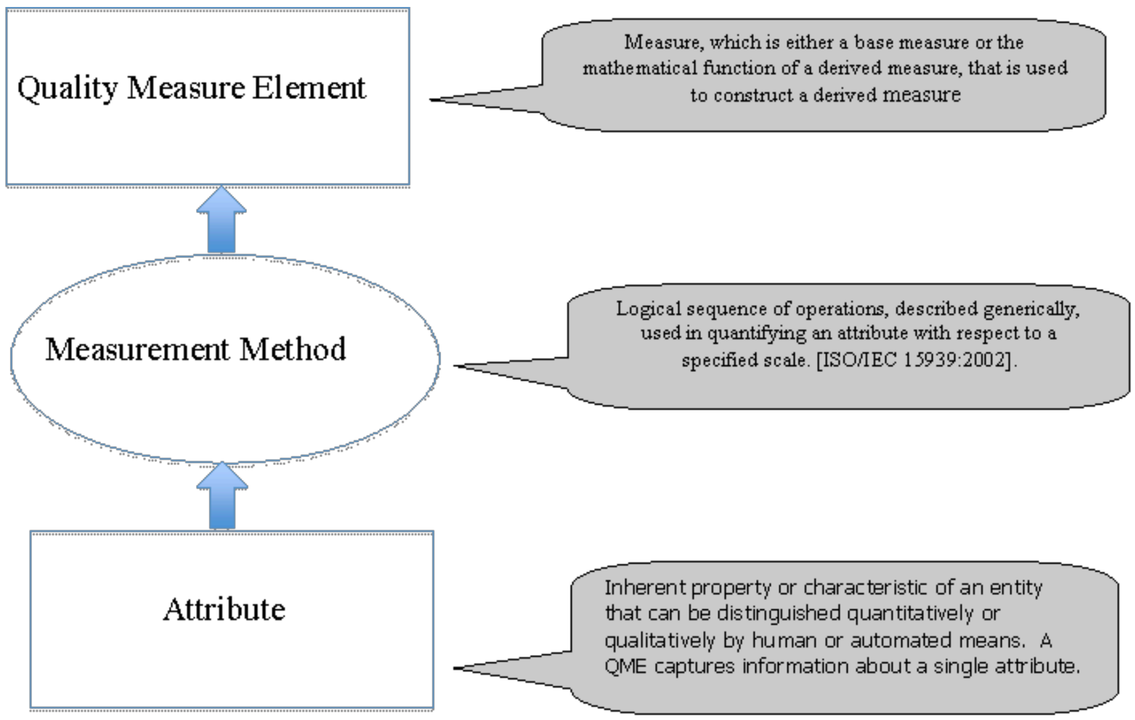

Quality Measure Element
Measure, which is either a base measure or the mathematical function of a derived measure, that is used to construct a derived measure
Measurement Method
Logical sequence of operations, described generically, used in quantifying an attribute with respect to a specified scale. [ISO/IEC 15939:2002].
Attribute
Inherent property or characteristic of an entity that can be distinguished quantitatively or qualitatively by human or automated means. A QME captures information about a single attribute.

Figure 5 — Relations between measurement function, attribute and QME

ISO/IEC 9126 series is compose of 4 documents list and describe characteristics, sub characteristics and quality measures that refer as the quality model. The QME is either a base measure or a derived measure (the later include only the derived measures with a mathematical function). This means that the combination of two or more based measures is exclude from the definition of a QME. Figure 5 shows that an attribute use a measurement method to create a Quality Measure Element (QME). This correspond to a portion of the ISO 15939 Measurement Information Model (ISO/IEC 15939). The attribute is by definition an inherent property or characteristic on an entity than can be distinguished quantitatively or qualitatively by human or automated means. Like a base measure, the QME captures information about a single attribute. The measurement method is a logical sequence of operations used in quantifying an attribute with respect to a specified scale. This document presents a procedure to design the measurement method.

As Figure 5 suggest, to quantify the QME, the designer of the measurement method shall identify and collect data related to the attribute of a QME. Depending of the context usage and objective(s) of the QME, a number of concepts shall be identified. These are the input of the measurement method. Those concepts shall be extract and define from the artefacts of the software (ex: software life cycle). The designer of the measurement method shall produce different outputs that are the identification of the concepts and constructs, the measurement principle and the description of its measurement method for the implementation of the numerical assignment rules.

This section describe 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 procedure.

Proposed steps to design QMEs:

a) identification of the QME (7.2)

b) identification of the attribute used by the identify QME and definition of the attribute (7.3)

c) definition of the concepts related to the attribute identified (7.4)

d) construction of the meta-model[2] of the attribute to be measured (7.5)

e) assignment of the unit of measurement (formula) and scale type (7.6)

Users of this document shall consider each steps listed in this section to design a measurement method for a specific Quality Measure Element (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 software product life cycle. In this document, the design procedure apply to implement a measurement method is independent of the QME and technology. However, considerations related to the objectives of the measurement shall 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 exclude the usage of a QME for different characteristics and subcharacteristics as long the measurement objective is not change.

Note 1: 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.

7.2 Identification of the QMEs

A list of QME were extract from the ISO/IEC 9126 series, part 2, 3 and 4. For each QME listed, an attribute was identified (Annex A). This list is not close. 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 attributes. 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 attribute use by the QME.

Consequently, the measurement design should contain the following descriptive information: name of the QME, context of usage (it is use by what quality measure), point of view, objective,

[2] Note: a **model** is a representation of a real world process, device, or concept. *IEEE 1233, 1998 Edition (R2002) IEEE Guide for Developing System Requirements Specifications. A* ***meta-model*** *is the* specification of the concepts, relationships and rules that are used to define a methodology. *ISO/IEC 24744:2007 Software Engineering--Metamodel for Development Methodologies.* 3.4. *Syn:* meta-model.

domain audience, definitions-abbreviations, constraints and references. About the objective, the designer should clarify whether the measurement of the attribute 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 concepts of the attribute. 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 (ref. ISO/IEC FCD 12207- 2006, Systems and Software Engineering - Software Life Cycle Processes).

7.3 Identification of the attribute used by the QME and definition of the attribute

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. To measure an attribute related to a QME the measurer shall take in account the existence of those elements. However the attribute needs to be identified and defined independently of those elements to avoid being technology depedent. <u>One</u> attribute is link to <u>one</u> QME. For example, the number of errors is the QME while the "error" is the attribute.

The definition of an attribute could be a part of an existing standard. In this document the objective is either to list, define and design most of the 80 attributes found in the 9126 series (Annex B). How to decide which attribute will be listed, defined or designed in this document? The following criteria were applied (see also annex A):

- <u>listed</u> attributes are those already defined and designed in other standards (ex: functional size). Effort and duration can also enter in this category because seconds, minutes, hours, days, weeks, months, etc. are already defined in metrology;

- <u>defined</u> attributes are those that normally occur only one or two times in the ISO/IEC 9126 series. Eventually they should be also designed;

- <u>designed</u> attributes are normally those that are present more than two times in the ISO/IEC 9126 series. Only a dozen of those attributes appears more than two times.

For the organisation, the choice of an attribute 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. For an organisation, each attribute that correspond to a needed QME shall be define by the organisation.

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 (annex A). The reference could be from an existing standard or from publish author's. For each identified attribute, what need to be measured shall be determined (ex: size, quality, etc.). An identified attribute is related to one or more concepts which in turn could have two or more sub concepts. The construction of a meta-model is then necessary.

7.4 Definition of the concepts related to the identified attribute

The identified attribute of the QME shall be decomposed into measurable concepts. An attribute could be decompose into one or more concepts. For example the attribute "USE CASE" can be decompose in three concepts "main scenario", "alternative paths" and "exceptions". The designer of the measurement method should make then a comprehensive literature review to find out how the attribute of the QME are defined and measured on previous researches. The designer should observe the similarities and the differences between the definition of the attribute in the quality model and other documents. This depend mainly of the objective and context usage of the QME (7.2). The results of the review should be compatible with the objective and usage of the QME.

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

There is an example of a decomposition of a concept. The attribute COSMIC Function Point (CFP) within ISO/IEC 19761 (COSMIC) is composed of the concept of “data movement” and some others concepts (layers, boundary, functional process) that help to understand and define “data movement". In this case we can also find the sub concepts (of data movement) identify as entry type, exit type, read type and write type. Later, those sub concepts will be useful (but not without rules) to construct a meta-model (7.5) and measure “data movement” to obtain the number of CFP (QME) (7.6).

7.5 Construction of the meta-model (ex: schema)

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

Describing a generic meta-model is good practice in designing measurement method (figure 6a). Meta-model should not be specific to any particular software and be independent of the specific context of the measurement, i.e. how the software is implemented (unless it is what we want to measure). For example, the data movement (a concept) in ISO/IEC 19761 (COSMIC) is a meta-model showing the relations between the sub concepts entry type, exit type, the read type and the write type (figure 6 b).

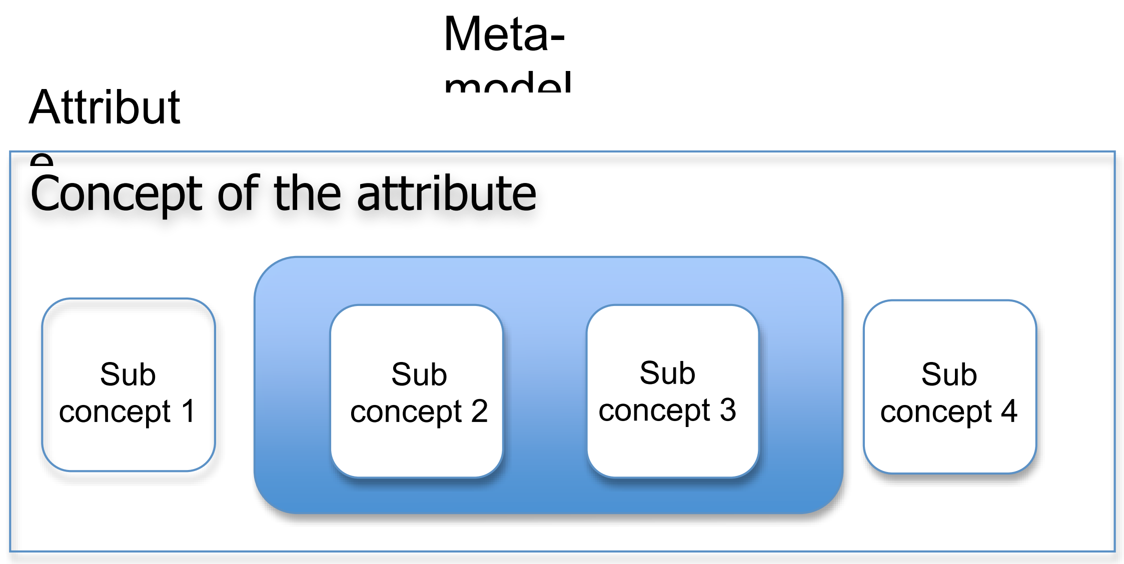


Figure 6a – General example of relations between concept and sub concepts in the meta-model

Figure 6a is a general example of relations between a concept and the sub concepts in a meta-model. The form of the figure could vary depending of the number of concepts and sub concepts with their relations. For an attribute it is possible to have more than one concept and each one have 0 and/or multiple sub concepts. For example the attribute "fault" could have two concepts like major faults and minor faults. Major faults (if related to requirements for example - see objective of the QME) could have sub concepts like faulty requirements and no requirements. Minor fault could be only a "syntax problem" (sub concept).

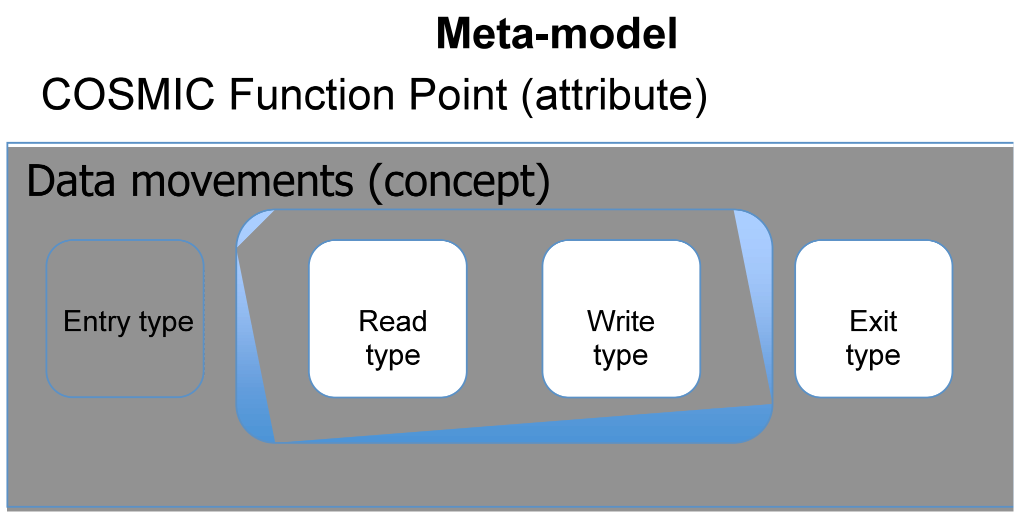


Figure 6b– COSMIC example of relations between the functional process and the types of data movement

Figure 6b presents a specific example of this meta-model for the COSMIC functional size in ISO/IEC 19761. The relations between the data movements that represent a part of a software constitute the COSMIC function point meta-model. The designen of the measurement method could decide at this point to call the measurement method "COSMIC function point" but only as a reference. This not change the measurement method. The COSMIC function point meta-model is composed of the following sub concepts: entry type, read type, write type and exit type. For this example, the main concept is the data movement and the attribute is COSMIC function point. Each one is a type of data movements. Each type of data movements within the COSMIC function point rules (ref. ISO/IEC 19761) is considered as an input (measurand) and taking into account while doing the measurement exercise.

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 concepts.

7.6 Assignment of the unit of measurement (formula) and scale type

The input source of data which is used to measure 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 concepts that need to be measured. For this reason, it is important to demonstrate that when adding the two concepts (or sub concepts) they are related by a common concept. For example adding apples and oranges could be justify when considering the concept 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. In measurement the cyclomatic complexity number is based on the relation between the concepts of edge and node (edge minus node) but there is no proof they are common concepts. The justification for the decision as to what unit to use in a measurement method should be provided (for example, by reference to a standard, to a theory, etc.). If this is not done, then the rationale for the interpretation of the unit is not provided.

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. Even with ratio value like in the “cyclomatic complexity” the value could be difficult to interpret if the mathematic relation is not defined. For example, 1 compare to 5 could be interpret as 5 times less complex if the mathematical relation is arithmetic. But what happen if the mathematical relation is geometric or logarithmic instead? There is no indication.

7.7 Measurement method table of information

From the proposed steps, a number of information is necessary for each attribute and QME. The following table identifies the necessary information and defines the content for each of them.

Table 3 — Measurement method table of information

QME Name	Identification of the QME name. Most of the time it start by "number of… (ratio scale).
Attribute Name **(step 1)**	Identification of the attribute (a list is provide in annex A), Related to the name of the QME normally. Ex: number of errors is the QME and error is the attribute.
Measurement method name (optional)	The measurer can give a name to the measurement method to facilitate the distinction between QME name, Attribute name and Measurement method.
Domain audience **(step 1)**	Describe the application area of software which is applied to this QME i.e. MIS, Realtime, etc. This could affect the life cycle. More examples from ISO/IEC 14143-5: • Multi-User Operating System is 'Complex Process Control Software' • a PC Word Processor system is a special case of a 'Business Data Processing Application', • a washing machine's embedded control software and a traffic light control system software are examples of 'Simple Process Control Software' • an aircraft attitude control system is 'Complex Process Control Software', (it has to store some data indefinitely in the aircraft's 'black box') but with the addition of domain-specific algorithms, i.e. flight envelope control algorithms • 'Scientific/Engineering Computation Software' with the existence of Complex Scientific/Engineering algorithms.
Software Life Cycle **(step 1)**	The basic life cycle phases for ISO/IEC 12207 are requirements analysis, design, coding, testing, and maintenance (ref. ISO/IEC FCD 12207- 2006, Systems and Software Engineering - Software Life Cycle Processes). The software life cycle could change depending of the domain or the methodology use by the developer (ex: Agile). When answering the following question: which phase you intent to apply the measure? the measurer should also mention the methodology if the phases are different from ISO/IEC 12207. It is also possible that the intent is to apply the QME to more than one phase.
Constraints (if applied) **(step 1)**	These are constraints related to any part of the function method of the attribute. For example, if the intent is to count the number of cases, the cases that cannot be counted will not be considered. Another example, when counting the number of messages, the user profile should be taken in consideration.
Quality measure **(step 1)**	Reference to a specific quality measure as example from the actual 9126, not necessarily the only reference.

Point of view **(step 1)**	The point of view could be from the measurer, the requester, the user, the developer, etc. For example requirements are from the point of view of the user and the design from the point of view of the developer. Internal measure are generally from the point of view of the developer while the external measure from the point of view of the user.
Definition of the attribute **(step 2)**	Each attribute needs to be identified and defined. This definition could be a part of an existing standard. Each definition shall be referenced indicating the source of the definition. A new definition shall be mention as new definition. The attribute shall be defined.
Reference **(step 2)**	The definition should be taken from: 1-the actual ISO/IEC standards definition, 2- IEEE standards or 3- Other sources (Ex: IEEE journal[3]) A new definition should be avoided, except if there is no definition available or the actual definition is not clear enough to be used. Combined references are then useful to mention.
List of concepts and sub concepts related to the attribute **(step 3)**	For each identified attribute, what need to be measured shall be determined (ex: size, quality, duration). An identified attribute is related to different concepts. This relation between each concept that can be expressed in a schema or in a formula. This constitutes the meta-model. For example, within the COSMIC method, a functional process is one concept that can be expressed in a meta-model with different concepts (or sub concepts) like entry, read, write and exit. This can help to identify the attribute "data movement" related to functional size.
Definition of each concept and sub concept **(step 3)**	Each concept and sub concept shall be define.
Reference for each concept and sub concept **(step 3)**	Each concept and sub concept shall be reference,
Relations between the concepts and sub concepts **(step 4)**	A schema that shows the relations between concepts and subconcepts could be providing. This will help for a better understanding. Those relations shall be explain in text format.

[3] The date of the publication is an important factor to consider.

Source of the information for the measurand (data element) **(step 4)**	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 for the effort to represent the attributes of software like the ones any tangible product may have. Therefore any reference to make each sub concept more tangible should be referenced.
Mesurand (input for the measure) **(step 4)**	The input source of data, which shall be used to measure the QME, should be identified with the documentation used to obtain the quantitative information. For example, the measurer could identify in a data model the information to retrace the entity of a read type (data movement) in in COSMIC function point.
Unit of measurement for the QME **(step 5)**	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.
Numerical rules **(step 5)**	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 concepts that need to be measured. For this reason is important to demonstrate that when adding two entities they are related by a common concept. For example, adding faults will give the number of faults. But if there is a distinction between major and minor faults, a more precise measure will be obtain by adding separately the major and minor faults. The interpretation must consider the limit of the result apply to each concept and sub concept.
Scale type **(step 5)**	Examples of scale type are nominal, ordinal, interval and ratio (annex E)

8 Initial set of Quality Measure Elements (QMEs)

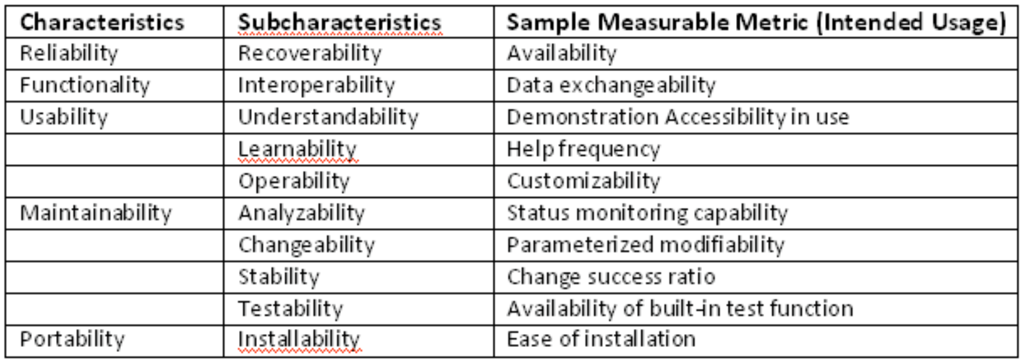
This part of the document applies the measurement method to different attributes to obtain QMEs

8.1 Use CASE

QME Name	Number of use cases actions (within ISO/IEC 9126-2 the word "case" is use)
Attribute Name **(step 1)**	Use case
Measurement method name (optional)	Measurement of number of use cases (MNUC)
Domain audience **(step 1)**	Applicable in data strong (MIS) and control strong systems (real time) especially if the organisation use UML approach, but limited to UML.
Software Life Cycle **(step 1)**	Can be applied from requirements analysis through maintenance depending of the objective of the measure.
Constraints (if applied) **(step 1)**	Use Cases that cannot be size (counting the number of actions) are exclude. Example: decision point because decision point has no action.

Quality measure

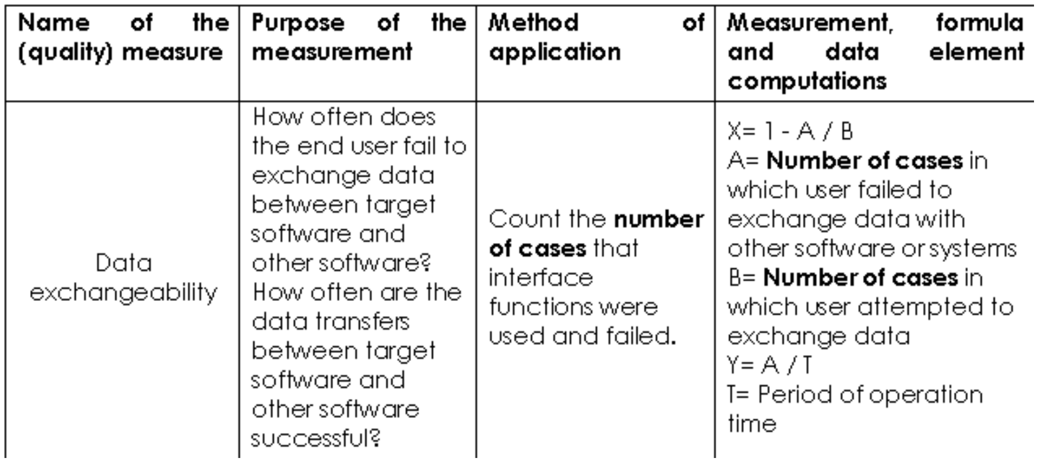
(step 1)



Characteristics	Subcharacteristics	Sample Measurable Metric (Intended Usage)
Reliability	Recoverability	Availability
Functionality	Interoperability	Data exchangeability
Usability	Understandability	Demonstration Accessibility in use
	Learnability	Help frequency
	Operability	Customizability
Maintainability	Analyzability	Status monitoring capability
	Changeability	Parameterized modifiability
	Stability	Change success ratio
	Testability	Availability of built-in test function
Portability	Installability	Ease of installation

Table of intended usage for cases within ISO/IEC 9126-2

Within 9126 many characteristics and subcharacteristics are used. Ten (10) intended usage were extract from ISO/IEC 9126-2 as represented in the table above. The right side of the table presents the quality measures.



Name of the (quality) measure	Purpose of the measurement	Method of application	Measurement, formula and data element computations
Data exchangeability	How often does the end user fail to exchange data between target software and other software? How often are the data transfers between target software and other software successful?	Count the **number of cases** that interface functions were used and failed.	X= 1 - A / B A= **Number of cases** in which user failed to exchange data with other software or systems B= **Number of cases** in which user attempted to exchange data Y= A / T T= Period of operation time

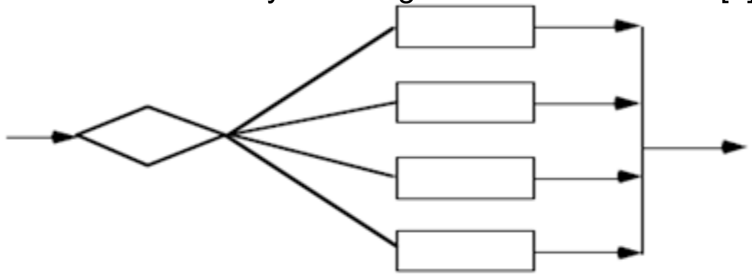
Example with data exchangeability from the table above

This example show that number of (use) cases is only one part of the equation. "Period of operation time" is another QME, It will be also necessary to define exchange data (failed to and attempted to). This is a part of a specific objective of the quality measure data exchangeability. However we can assume that the definition of "Number of (use) cases" is not dependent of the formula to find data exchangeability because "exchange data" and "Period of time" have their own definitions.

Point of view

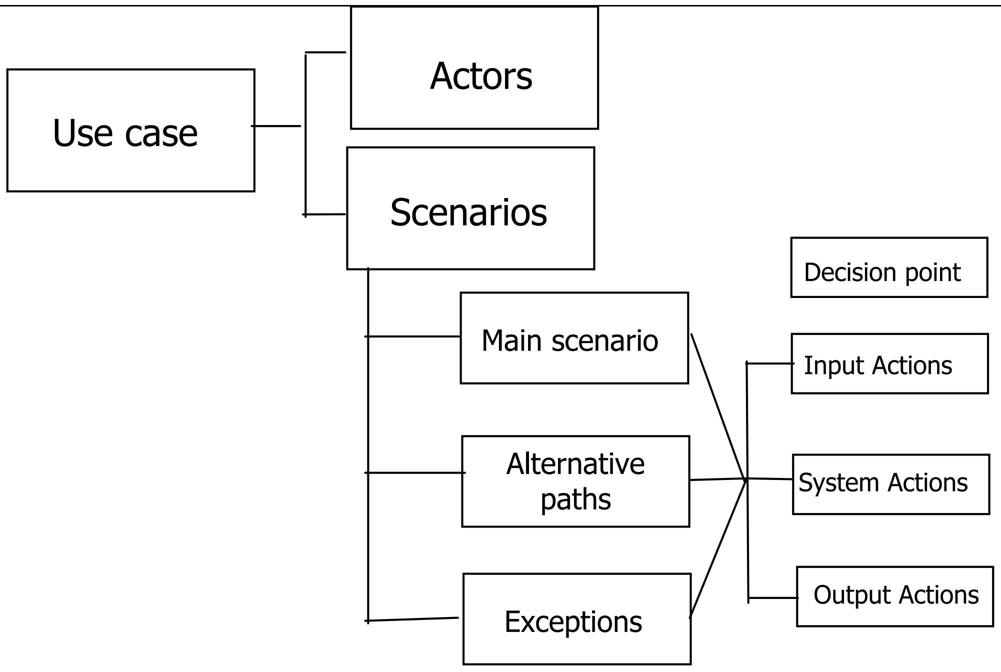
(step 1)

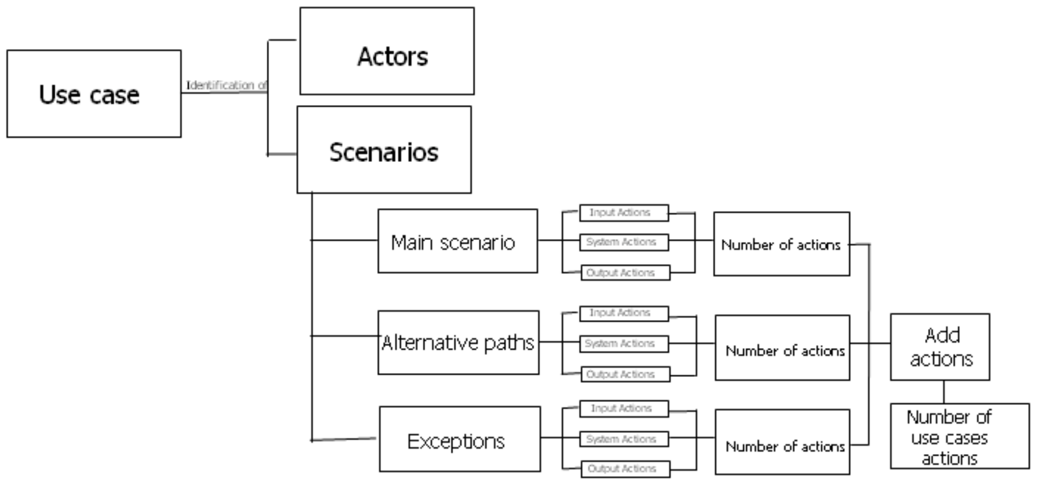
The method can be used by software developers, software managers and customers

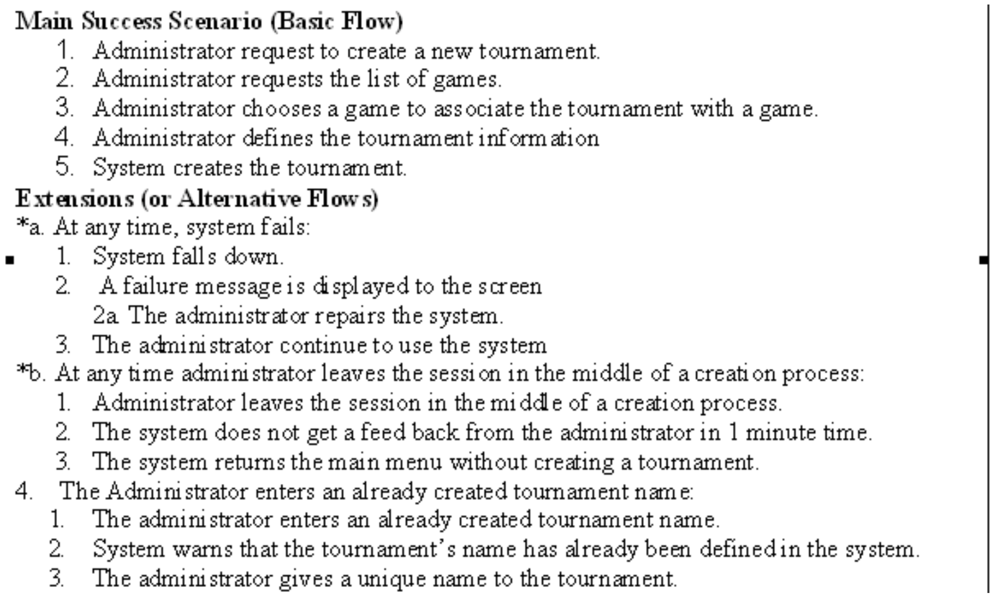
Definition of the attribute **(step 2)**	A case is a single-entry, single-exit multiple-way branch that defines a control expression, specifies the processing to be performed for each value of the control expression, and returns control in all instances to the statement immediately following the overall construct [1].  The definition and the figure refer to the representation of an algorithm or process. Rectangular boxes in the figure commonly refer to a processing step which is called activity or task. Diamond boxes refer to the decisions and arrows refer to the relationship and sequencing. A 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 [2]. In UML, a complete task of a system that provides a measurable result of value for an actor [3]. Note: when the definition of case was investigated, it has been realized that it is difficult to create a measurement method based on this definition. Because the definition is not instructive since the metrics which utilize *case* in Quality Model is concern with *the number of cases.* Therefore it is decided to take into account "use case" and later generate cases from the use cases actions, assuming an equivalency. Another advantage of taking use case instead of case reside in the fact that we can have them early in the life cycle (requirements phase).
Reference **(step 2)**	[1] ISO/IEC FCD 24765 Systems and software engineering – *[2]* Ivar Jacobson (1992). *Object-Oriented Software Engineering*. Addison Wesley Professional. ISBN 0-201-54435 *ISO/IEC 24765, Systems and Software Engineering Vocabulary* [3] Gunnar Overgaard and Karin Palmkvist "A Formal Approach to Use Cases and Their Relationships" J. Bézivin and P.-A. Muller (Eds.): «UML» '98, LNCS 1618, pp. 406-418, 1999

List of concepts and sub concepts related to the attribute **(step 3)**	The concepts related to a case are: task, activity, input (entry), output (exit) and decision point. A use case template present: actor, preconditions, main scenario, action, post-conditions, altenative paths, exceptions.
Definition of each concept and sub concept **(step 3)**	**ACTIVITY** is defined as (1) “a defined body of work to be performed, including its required input and output information” . (2) Set of cohesive tasks of a process [4] **INPUT (Entry)** “Any item, whether internal or external to the project that is required by a process before that process proceeds”. “Data received from an external source” **OUTPUT (Exit)** “Data transmitted to an external destination”. “A product, result, or service generated by a process.” **DECISION POINT** “The point in space and time where staff anticipates making a decision concerning a specific friendly course of action. A decision point is usually associated with a specific target area of interest” [6] 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)"

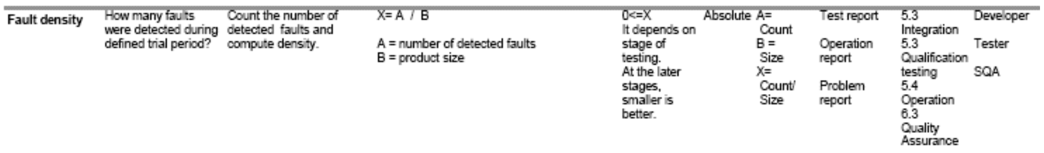
Definition of each concept and sub concept (continued) **(step 3)**	Within a use case template there are also a number of concepts **An Actor** is "someone or something outside the system that either acts on the system – a primary actor – or is acted on by the system – a secondary actor. An actor may 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)"

Relations between the concepts and sub concepts **(step 4)**	 Use case Actors Scenarios Main scenario Alternative paths Exceptions Decision point Input Actions System Actions Output Actions The concept of use case consists of actor and scenario. The sub concepts for scenario are main scenario, alternative paths and exceptions (all related to the concept of scenario). Those sub concepts can be measured using: input actions, system actions and output actions. The decision point is exclude because not measurable in term of actions.
Source of the information for the measurand (data element) **(step 4)**	Use case documents. The documents can be find at different phases of the life cycle.
Mesurand (input for the measure) **(step 4)**	The measurands are the action lines in the scenarios of the use case description base on rules to identify those lines.
Unit of measurement for the QME **(step 5)**	The action is the unit. The measure is use case action (UCA).

Numerical rules **(step 5)**	The model for numerical rules is the following:  Use case Identification of Actors Scenarios Main scenario Alternative paths Exceptions Input Actions System Actions Output Actions Number of actions Add actions Number of use cases actions This model shows that for each type of scenario ir is necessary to count the input, system and output actions.

Numerical rules (continued)	The rules to find the different type of actions for different scenarios are the following:
(step 5)	a) use the sequential ordering of action descriptions (and hence their unique number identifiers) to indicate strict sequence between actions for the main scenario
	b) 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
	c) an action is atomic with only one clause. Each action description should start with a new line. Then one action per line in the main scenario and one action per line in the alternative paths and exceptions using all the possibilities.
	Example:
	
	Main Success Scenario (Basic Flow)
	1. Administrator request to create a new tournament.
	2. Administrator requests the list of games.
	3. Administrator chooses a game to associate the tournament with a game.
	4. Administrator defines the tournament information
	5. System creates the tournament.
	Extensions (or Alternative Flows)
	*a. At any time, system fails:
	1. System falls down.
	2. A failure message is displayed to the screen
	2a The administrator repairs the system.
	3. The administrator continue to use the system
	*b. At any time administrator leaves the session in the middle of a creation process:
	1. Administrator leaves the session in the middle of a creation process.
	2. The system does not get a feed back from the administrator in 1 minute time.
	3. The system returns the main menu without creating a tournament.
	4. The Administrator enters an already created tournament name:
	1. The administrator enters an already created tournament name.
	2. System warns that the tournament's name has already been defined in the system.
	3. The administrator gives a unique name to the tournament.
	In the main scenario there are 5 actions. In the alternative paths there are 10 actions including the repairs in 2a. There is no distinction between types of action in this count. This can be refined.
Scale type	A use case action (UCA) is a ratio. One action identified is one use case action (UCA).
(step 5)	

8.2 Software fault (note: this method was constructed using Monson and Nikora document [1]).

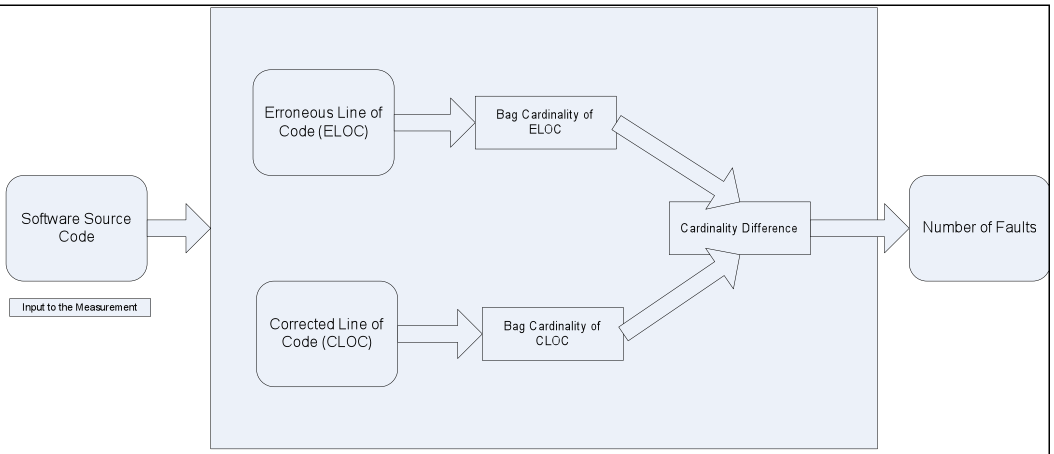
QME Name	Number of software faults
Attribute Name **(step 1)**	Software Fault
Measurement method name (optional)	Munson and Nikora method
Domain audience **(step 1)**	Management Information System and Real time
Software Life Cycle **(step 1)**	Coding and maintenance phase
Constraints (if applied) **(step 1)**	This method cover only the software faults from the source code. This measurement method can not be used if ELOC and CLOC have cardinal and token differences at the same time. (to revised)
Quality measure **(step 1)**	The quality measure fault density can be used to assess the quality characteristic Reliability and quality sub-characteristic Maturity. The quality measure use the number of faults detected during a defined trial period. For example, the number of faults detected the first month after implementation. The product size (ex: number of KLOC) is also a type of measure that can be used. Following is an extract from 9126. 

Fault density	How many faults were detected during defined trial period?	Count the number of detected faults and compute density.	X= A / B A = number of detected faults B = product size	0<=X It depends on stage of testing. At the later stages, smaller is better.	Absolute	A= Count B= Size X= Count/ Size	Test report Operation report Problem report	5.3 Integration 5.3 Qualification testing 5.4 Operation 6.3 Quality Assurance	Developer Tester SQA

Point of view **(step 1)**	User and developer. The user want to know if the product is reliable and mature, and the developer can estimate the effort in maintenance.

Definition of the attribute **(step 2)**	A fault, by definition, is a structural imperfection in a software system that may lead to the system's eventually failing [1]. Considering the constraint (method cover only the software faults from the source code) the definition is now: "A fault is an invalid source code that will cause a failure when the ompiled code that implement the source code is executed" (adaptation of [1]). Four (4) definitions can be found in ISO/IEC and IEEE standards fault. 1. 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. 2. 2. a manifestation of an error in software. IEEE 982.1-1988 IEEE Standard Dictionary of Measures to Produce Reliable Software. 2. 3. a defect in a hardware device or component. ISO/IEC 24765, Systems and Software Engineering Vocabulary. 4. an incorrect step, process, or data definition in a computer program. ISO/IEC 24765, Systems and Software Engineering Vocabulary. Syn: bug. NOTE A fault, if encountered, may cause a failure.
Reference **(step 2)**	[1] John C. Munson, Allen P. Nikora, 2002, Toward A Quantifiable Definition of Software Faults, IEEE) [2] The Common Software Measurement International Consortium (COSMIC): Guideline for Sizing Business Applications Software Using COSMIC-FFP, Version 1.0, 2005COSMIC Manual. [3] Xuemei Zhang; Xaolin Teng, Hoang Pham, 2003, Considering Fault Removal Efficiency in Software Reliability Assessment, IEEE [4] Desharnais J.-M., Abran A., Suryn W., Attributes Within ISO 9126: A Pareto Analysis, Quality Software Management 2009, British Computer Society, April 2009. [5] IEEE 982.1-1988 IEEE Standard Dictionary of Measures to Produce Reliable Software

List of concepts and sub concepts related to the attribute **(step 3)**	A fault is a manifestation of error in software. A fault, if encountered, may cause failure. Error ---) Fault ---) Failure A fault can have a source from error, specification requirements and source code. There are faults of commission (implementing code that is not part of the specification of design) or omission (a behavior specified in the design was not implemented) Software faults can be large or small. It is then necessary to measure the fault at a lower level of granularity from source code (executable statement). Each character use in a line of code are considered (call sometime token).
Definition of each concept and sub concept **(step 3)**	Executable Statements: The statements which can be categorised to labeled statements, expressions, selection statements, iteration statements and jump statements. Non-Executable Statements: The statements which can be categorised to declarations and declaration specifiers.
Definition of each concept and sub concept (continued) **(step 3)**	Erroneous Line of Code (ELOC): Line of code which contains fault expression or executable statement in. Corrected Line of Code (CLOC): Line of code which is purified from fault expression or executable statement. Token: Basic building blocks of a programming language which can be keywords, identifiers, operators, separators and constants. [8] [9] Ex: a=b+c; has 6 tokens a, =, b, +, c and ; Bag of Token: Each line of code in each version of the program can be seen as bag of tokens. Cardinality: Total number of tokens in an expression. Bag Difference: Token difference between two expressions. Note: all definitions were take from [1].

Relations between the concepts and sub concepts **(step 4)**	 Software Source Code Input to the Measurement Erroneous Line of Code (ELOC) Bag Cardinality of ELOC Corrected Line of Code (CLOC) Bag Cardinality of CLOC Cardinality Difference Number of Faults Software source code is used as an input for Software Fault Measurement Method. The project source code which is used to measurement is examined line by line, that is LOC. Erroneous executable statements within that lines are found and the corrected statements in response to erroneous one are specified as well. After this step, one of three options could be followed: 1. Bag cardinality of erroneous and corrected line of codes is calculated. The difference of them is found. Final solution gives **number of faults**. 2. Bag of tokens of erroneous and corrected line of codes are inspected and token difference is found. Number of different tokens gives **number of faults**. There might be both cardinality and token difference between two LOC. That is a constraint in this proposal and studies on this issue is carried on
Source of the information for the measurand (data element) **(step 4)**	Source code
Mesurand (input for the measure) **(step 4)**	Bag of tokens and changes in different versions of source line of code.
Unit of measurement for the QME **(step 5)**	Tokens within an executable statement (LOC) or bag of tokens when considering all the token in an executable statement. A fault is then an invalid token or bag of tokens in the source code that will cause the failure when the compiled code that implementes the source code is executed.

Numerical rules **(step 5)**	Example from [1]. Each line of text in each version of the program can be seen as a bag of tokens. New tokens may be added, invalid token may be moved and the sequence of tokens may be changed. Example of a count: Consider the following line of C code. (1) a = b + c; There are five tokens on this line of code. They are B1 = {<a>, <=>, <b>, <+>, <c>} where B1 is the bag representing this token sequence. Now let us suppose that the design, in fact, required that the difference between b and c be computed: (2) a = b - c; There will again be five tokens in the new line of code. This will be the bag B2 = {<a>, <=>, <b>, <->, <c>}. The bag difference is B1 - B2 = {<+>, <-> }. The cardinality of B1 and B2 is the same. There are two tokens in the difference. Clearly, one token has changed from one version of the module to another. There is one fault. Now let us suppose that the new problem introduced by the code in statement (2) is that the order of the operations is incorrect. It should read: (3) a = c - b; The new bag for this new line of code will be B3 = {<a>, <=>, <c>, <->, <b>}. The bag difference between (2) and (3) is B2 - B3 = { }. The cardinality of B2 and B3 is the same. This is a clear indication that the tokens are the same but the sequence has been changed. There is one fault representing the incorrect sequencing of tokens in the source code.
Scale type **(step 5)**	Ratio

8.3 Error message

QME Name	Number of error messages (to be revised)

Attribute Name	Error Message
Domain audience	Management Information System, Real time
Constraints	Counting the error message is useless, what count is the effectiveness of error message. From there the boundary between derive measure (mathematical function) and quality measure (combination of different QME) should be clarified. In the actual proposal we consider the error message effectiveness as mathematical function. Need to be discuss.
Software Life Cycle **(if apply)**	Coding, testing and maintenance
Definition of the attribute	Error message is a message that the application gives when incorrect data is entered or when another processing error occurs.
Reference	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.
List of concepts related to the attribute	Message appearance, message content.
Definition of each concept	
Reference for each concept	
Relations between the concepts	
Mesurand (input for the measure)	
Unit of measurement for the QME	

Numerical rule	To measure the error message effectiveness of the whole system, we need to calculate the effectiveness of all error messages separately and calculate the mean effectiveness of that system then. The final result obtained will be error message effectiveness. It will be measured in Effectiveness Percentage. We will discuss later the procedure or different steps that should be done in order to calculate this value. Effectiveness = Attractiveness + Format Compliance + User Interactiveness + Clearness + Completeness
Scale type	Ordinal and ratio
Quality measure (optional)	
List of qualifiers (optional and related to quality measure)	
Characteristics and sub characteristics	Usability Operability Understandability Learnability

Annex A – Justification of the selection of the initial QMEs

Reference: 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, British Computer Society, April 2009.

Annexe B Procedure for the measurement function (informative)

The measurement function is the link between the QME and the quality measures.

Each of the ISO 9126-proposed derived measures (there are over 250 of them) is defined at a fairly high level as a formula composed of QMEs. It must be observed that the result of the mathematical operations must also lead to the combination of the measurement units of its corresponding QME. Here is an example of a ratio (not a model):

QME 1 (B1): Number of detected failures.

QME 2 (B2): Number of performed test cases.

Quality Measure: B1 / B2, with the following measurement units:

B1 =	Number of detected failures
B2 =	Number of performed test cases

It must be noted that a quality measure is descriptive. It does not explain a relationship, nor does it say anything about the strength of such a relationship. Therefore, if a derived measure is designed bottom-up, the name assigned to this combination of units should correspond to the concept representing the particular combination of measurable attributes.

The accuracy of a derived measure (together with the corresponding measurement errors) is directly related to:

- the accuracy of each of its base measures, and
- how these base measures are mathematically combined

Consequently, the qualities of the corresponding measuring instrument(s) of the base measures impact the quality of the derived measures.

For example a combinaison of two QMEs implies the the two QMEs have some mathematical relations.

Annex C Cross-reference table representing the Life cycle phases and the use of QMEs

Annex D 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 attributes of the quality measure elements (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 quality measure element and suggested extension to all the quality measures where its usage applies; by column, all the crosses indicate the quality measure elements that are required for the quality measures plus their qualifier in that column.

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

	Quality measure 1 plus qualifier	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 1 should be constructed from the list of quality measure elements and quality measures (Annex B). The list was extracted from Quality Measures Elements related to quality in ISO/IEC TR 9126 - 2,3,4. Note that some quality measure elements 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 1 should use the list of attributes of QME with the qualifiers of the quality measure (Annex A). Annex C presents the ISO/IEC 9126 series cross-reference table representing the relationship between quality measures, characteristics and sub characteristics.

Annex D gives some information about measurement scale types and Annex E provides the bibliography.

Table 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 quality measure elements 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 quality measure elements 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

Absolute scale type - The purpose of absolute scale type measures in a set of QME is to measure a proportion between measures with the same units. Absolute scale quality measure elements include measures resulting from dividing one ratio scale measure to another ratio scale measure where the measurement unit is the same in both cases. The ratio or average calculated from two absolute measures does not have any empirical meaning.

EXAMPLE Number of comment lines divided by total lines of code.

Annex F Bibliography

ISO/IEC 25000 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – General overview, reference models and guide

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

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

ISO/IEC 25022 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Measurement of internal quality

ISO/IEC 25023 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Measurement of external quality

ISO/IEC 25024 Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Measurement of quality in use

ISO/IEC 25041 - Software engineering: Software product Quality Requirements and Evaluation (SQuaRE) – Evaluation modules

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, 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]

ISO/IEC TR 15939

Book Adison Wesley

Pareto measures