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

ISO/IEC JTC1/SC7 N4098

2008-07-17

Document Type	**CD**
Title	CD 25010.2, Software engineering-Software product Quality Requirements and Evaluation (SQuaRE)Quality model
Source	WG23
Project	**25010**
Status	2nd CD
References	N3803, N4008, N4097
Action ID	ACT
Due Date	2008-10-18
Start Date	2008-07-18
Distribution	SC7 AG
Medium	PDF
No. of Pages	42
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

ISO/IEC JTC1/SC7/WG6 N
17-Jul-08

TITLE: ISO/IEC CD 25010:
Software engineering –
Software product Quality Requirements and Evaluation (SQuaRE) – Software and quality in use models

DATE: 17-Jul-08

SOURCE: JTC1/SC7/WG6

WORK ITEM: Project

STATUS: Version 0.55

DOCUMENT TYPE: Text for second CD

ACTION: For comment

PROJECT EDITOR: Prof. Motoei AZUMA
Department of Industrial Eng. and Management Systems Eng.
Waseda University
3-4-1, Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
FAX: +81-3-3200-2567
azuma@azuma.mgmt.waseda.ac.jp

DOCUMENT EDITOR: Nigel BEVAN
nigel@nigelbevan.com

CO-EDITOR: Vipula Godamunne

CO-EDITOR: David Zubrow

CO-EDITOR: Yukio Tanitsu

CO-EDITOR: Markku Tukiainen

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

Date: 17-Jul-08

Reference number of document: **ISO/IEC CD 25010.2**

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

Secretariat: SC7

Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Software and quality in use models

Document type: International standard
Document subtype:
Document stage:
Document language: E

Content

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 25010 is a part of the SQuaRE series of standards and was prepared by Joint Technical Committee ISO/IEC JTC 1, *information technology*, Subcommittee SC 7, *Software and System Engineering*

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

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

Introduction

Computers are being used in an increasingly wide variety of application areas, and their correct operation is often critical for business success and/or human safety. Developing or selecting high quality software products is therefore of prime importance. Comprehensive specification and evaluation of software product quality is a key factor in ensuring adequate quality. This can be achieved by defining appropriate quality characteristics, taking account of the purpose of usage of the software product. It is important that every relevant software product quality characteristic is specified and evaluated, whenever possible using validated or widely accepted measures.

ISO/IEC 9126 (1991): Software product evaluation - Quality characteristics and guidelines for their use, which was developed to support these needs, defined six quality characteristics and described a software product evaluation process model.

ISO/IEC 9126 (1991) was replaced in 2001 by two related multipart standards: ISO/IEC 9126 (Software product quality) and ISO/IEC 14598 (Software product evaluation).

This International Standard is a revision of ISO/IEC 9126-1: 2001, and incorporates the same software quality characteristics with some amendments.

- *Security* has been added as a characteristic, rather than a subcharacteristic of functionality.
- Portability has been split into *transferability* and *compatibility* (including *interoperability*).
- The following subcharacteristics have been added: *robustness, helpfulness, technical accessibility, modularity, reusability,* and *portability*.
- Quality in use has been split into *usability in use, flexibility in use* and *safety*.
- Several characteristics and subcharacteristics have been given more accurate names.

Full details are in Annex A.

This International Standard is intended to be used in conjunction with the other parts of the SQuaRE series (ISO/IEC 25000 – ISO/IEC 25099) of standards, and with ISO/IEC 14598 until superseded by the ISO/IEC 2502n series of standards.

Figure 1 (adapted from ISO/IEC 25000) illustrates the organisation of the SQuaRE series representing families of standards, further called Divisions.

The Divisions within SQuaRE series are:

- **ISO/IEC 2500n - Quality Management Division**. The standards that form this division define all common models, terms and definitions referred further by all

other standards from SQuaRE series. The division provides also requirements and guidance for a supporting function that is responsible for the management of software product requirements specification and evaluation.

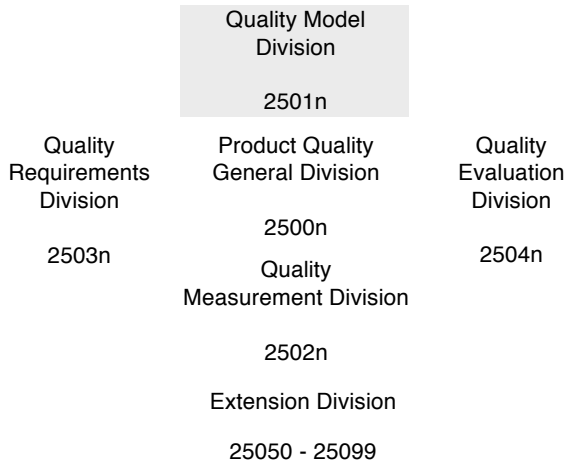


Figure 1 – Organization of SQuaRE series of standards

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

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

- **ISO/IEC 2503n - Quality Requirements Division**. The standards that form this division help specifying quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a software product to be developed or as input for an evaluation process.

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

- **ISO/IEC 25050 – 25099 SQuaRE extension standards**. These currently include Requirements for quality of Commercial Off-The-Self software and Common Industry Formats for usability reports.

Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) — Software and quality in use models

1 Scope

This International Standard defines:

a) A software product quality model composed of eight characteristics, which are further subdivided into subcharacteristics that can be measured internally or externally. These subcharacteristics are manifested externally when the software is used as a part of a computer system, and are a result of internal software attributes and computer system behaviour.
b) A system quality in use model composed of three characteristics, which are further subdivided into subcharacteristics that can be measured when a product is used in a realistic context of use. When used to specify or measure the effect of software quality in a particular context of use, quality in use may be influenced by any of the eight software product quality characteristics. Although quality in use is described in the context of software product quality, as it is a property of the whole system, it can also be used to evaluate other components of the system (including hardware, the user or the environment).

The characteristics of the models are applicable to every kind of software. The characteristics and subcharacteristics provide consistent terminology for software product quality. They also provide a set of quality characteristics against which stated quality requirements can be compared for completeness.

The quality models can be used to support specification and evaluation of software from different perspectives by those associated with acquisition, requirements, development, use, evaluation, support, maintenance, quality assurance and audit of software. They can for example be used by developers, acquirers, quality assurance staff and independent evaluators, particularly those responsible for specifying and evaluating software product quality. Activities during product development that can benefit from the use of the quality models include:

- identify software requirements;
- validate the comprehensiveness of a requirements definition;
- identify software design objectives;
- identify software testing objectives;
- identify quality assurance criteria;

- identify acceptance criteria for a completed software product.

ISO/IEC 25012 contains a model for data quality that is complementary to this model.

The quality models can be used in conjunction with the ISO/IEC 12207 processes associated with requirements definition, verification and validation with a specific focus on the specification and evaluation of quality requirements. Additionally, the models may be useful for evaluating a system in operation to characterize it in terms of quality characteristics.

This International Standard can be used in conjunction with ISO/IEC 15504 (which is concerned with the software process assessment) to provide:

- a framework for software product quality definition in the customer-supplier process;
- support for review, verification and validation, and a framework for quantitative quality evaluation, in the support process;
- support for setting organisational quality goals in the management process.

This International Standard can be used in conjunction with ISO 9001 (which is concerned with quality assurance processes) to provide:

- support for setting quality goals;
- support for design review, verification and validation.

2 Conformance

Any software product quality requirement, specification or evaluation that conforms to this International Standard shall either use the characteristics and subcharacteristics from clauses 6 and 7, giving the reasons for any exclusions, or describe its own categorisation of software product quality attributes and provide a mapping to the characteristics and subcharacteristics in clauses 6 and 7.

3 Normative references

The following normative document contains provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

None?

4 Terms and definitions

For the purposes of all parts of this International Standard, the terms and the definitions contained in ISO/IEC 25000 apply, except where indicated below.

NOTE The essential definitions from ISO/IEC 25000 are reproduced in Annex A.

4.1
asset
anything that has value to the organization

[ISO/IEC 13335-1:2004]

4.2
external software quality
degree to which a software product enables the behaviour of a system to satisfy stated and implied needs when the system including the software is used under specified conditions.

NOTE Attributes of the behaviour 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.

[ISO/IEC 25000:2005 definition, rephrased as “degree to which”]

4.3
internal software quality
degree to which a set of static attributes of a software product 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 Complexity measures and the number of faults found in a walk through are internal software quality measures made on the product itself.

[ISO/IEC 25000:2005 definition, rephrased as “degree to which”]

4.4
level of performance
set of criteria for measures of quality characteristics

4.5
software quality
degree to which the software product satisfies stated and implied needs when used under specified conditions

NOTE This definitions differs from the ISO 9000:2000 quality definition because the software quality definition refers to the satisfaction of stated and implied needs, while the ISO 9000 quality definition refers to the satisfaction of requirements.

[ISO/IEC 25000:2005 definition, rephrased as “degree to which”]

5 Quality model framework

5.1 Quality models

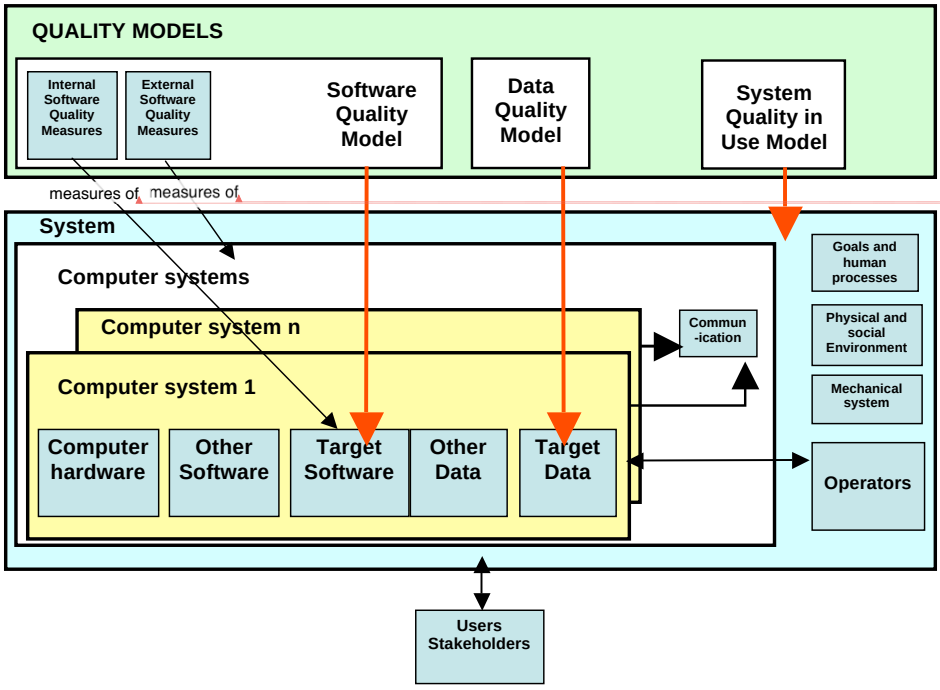
The quality of a system is the result of the quality of the system elements and their interaction. Software quality is the degree to which the software product satisfies stated and implied needs when used under specified conditions. The software quality model in clause 6 defines eight software quality characteristics: functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability and transferability.

The quality in use model in clause 7 defines three characteristics at the system level: usability in use, flexibility in use and safety in use that can be used to specify and evaluate requirements for the effect of software quality in specified contexts of use.

The quality characteristics have defined subcharacteristics and the standard allows for user-defined sub-subcharacteristics in a hierarchical structure. The defined quality characteristics can be used as a checklist for ensuring a comprehensive coverage of quality.

Quality in use is a measure of the overall quality of the system in its operational environment for specific users, for carrying out specific tasks. From a software perspective, quality in use can be used to measure the capability of the software to enable quality in use in its operational environment, for carrying out specific tasks by specific users.

External software quality provides a ‘black box’ view of the software and addresses properties related to the execution of the software on computer hardware and an operating system. Internal software quality provides a ‘white box’ view of software and addresses properties of the software product that typically are available during the development. Internal software quality is mainly related to static properties of the software. Internal software quality has an impact on external software quality, which again has an impact on quality in use. Figure 2 shows the different types of quality measures. Data quality is described in ISO/IEC 25012.



Nigel Bevan 08-5-24 11:36
Mis en forme: Police :8 pt

Nigel Bevan 08-5-24 11:36
Mis en forme: Police :8 pt

Figure 2 – Scope of quality measures

The quality models serve as a framework to ensure that all aspects of quality are considered from the internal, external, and quality in use point of view.

5.2 Software properties

Some software properties are inherent in the software product; some software properties are system dependent; some are assigned to the software product. The quality of a software product in a particular context of use is determined by its inherent properties.

NOTE 1 “Inherent” means existing in something, especially as a permanent characteristic or feature.

NOTE 2 Examples of inherent properties are number of lines of code and the accuracy of a numeric calculation provided by the software. Examples of assigned properties are the owner of a software product and the price of a software product.

Inherent properties can be classified as either functional properties or quality properties. Functional properties determine what the software is able to do. Quality properties determine how well the software performs. In other words, the quality properties show the degree to which the software is able to provide and maintain its specified services. Quality properties are inherent to a software product and associated system. An assigned property is therefore not considered to be a quality

characteristic of the software, since it can be changed without changing the software. Figure 3 illustrates this classification of software properties.

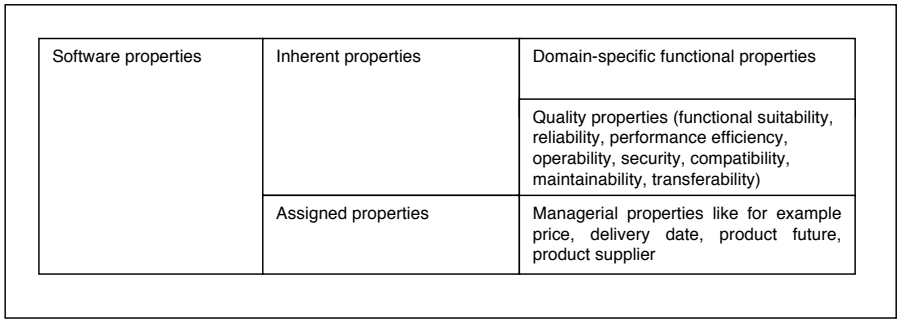


Figure 3 – Software properties

5.3 Structure used for the quality models

The SQuaRE quality models categorise product quality into characteristics which are further subdivided into subcharacteristics and quality attributes (Figure 4).

The SQuaRE quality model consists of two parts, the model for External and Internal Software Quality and the model for Quality in Use.

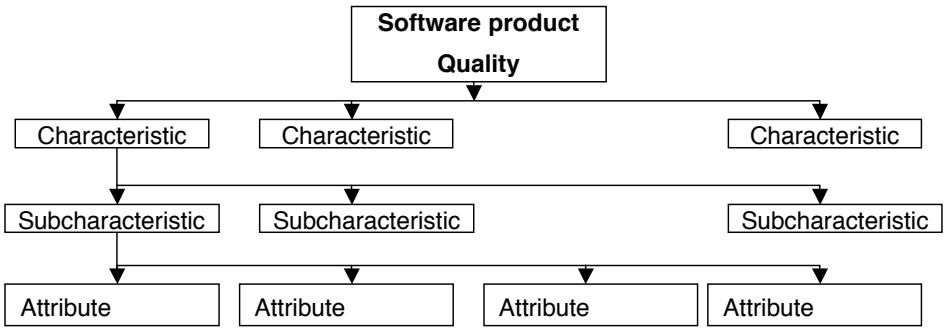


Figure 4 – Structure used for the quality model

5.4 Difference between internal, external and quality in use measures

Internal software quality measures can be used early in the system development process to predict external software quality measures. There are often internal and external measures for the same property, for example an internal measure to estimate the expected response time to predict the time measured externally.

Quality in use measures relate to users completing realistic tasks (either by user testing or in actual use). External operability measures relate to the behaviour of individual functions, and can be evaluated individually, or as part of wider user testing that may also be measuring overall usability in use.

The relationship of quality in use to the other software product quality characteristics depends on the type of user:

- the end user for whom quality in use is mainly a result of functional suitability, reliability, operability and performance efficiency;
- the person maintaining the software for whom quality in use is a result of maintainability;
- the person porting the software for whom quality in use is a result of portability.

5.5 Using a quality model

Software product quality should be evaluated using a defined quality model. The quality model should be used when setting quality requirements for software products and intermediate products. Software product quality should be hierarchically decomposed into a quality model composed of characteristics and subcharacteristics that can be used as a checklist of issues related to quality. Clauses 6 and 7 define hierarchical quality models (although other ways of categorising quality may be more appropriate in particular circumstances).

It is not practically possible to measure all internal and external subcharacteristics for all parts of a large software product. Similarly it is not usually practical to measure quality in use for all possible user-task scenarios. The relative importance of quality characteristics will depend on the product and application domain. So the model should be tailored before use, and resources for evaluation allocated between the different types of measurement dependent on the business objectives and the nature of the product and design processes.

NOTE 1 In a contractual environment, or in a regulated environment, such as the nuclear safety field, needs are specified, whereas in other environments, implied needs should be identified and defined (ISO 8402: 1994, note 1).

6 Software product quality model

This clause categorises software quality attributes into eight characteristics (functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability and transferability), which are further subdivided into subcharacteristics (Figure 5). The subcharacteristics can be measured by internal or external measures.

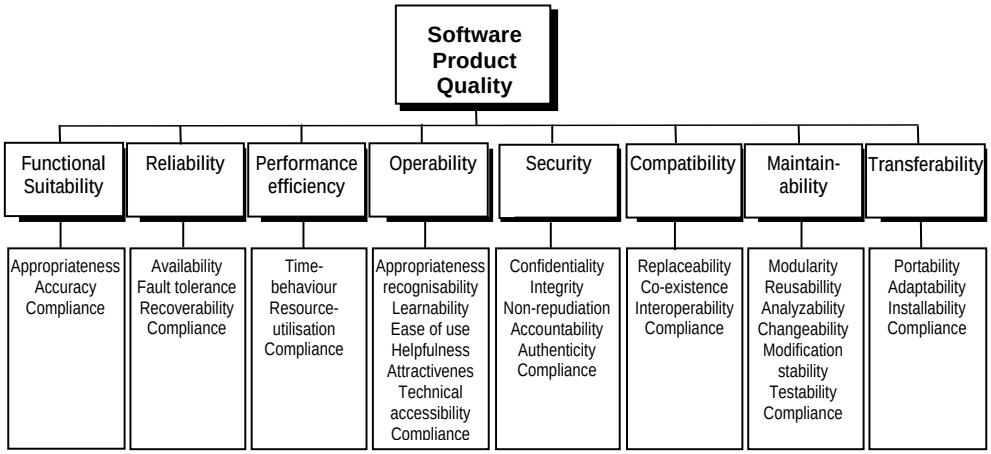


Figure 5 – Software product quality model

Definitions are given for each quality characteristic and the subcharacteristics of the software that influences the quality characteristic. For each characteristic and subcharacteristic, the capability of the software is determined by a set of internal attributes that can be measured. Examples of internal measures are given in ISO/IEC 9126-3 (to be replaced by ISO/IEC 25022). The characteristics and subcharacteristics can be measured externally by the extent to which the capability is provided by the system containing the software. Examples of external measures are given in ISO/IEC 9126-2 (to be replaced by ISO/IEC 25023).

NOTE 1 There is a compliance subcharacteristic for all characteristics, as the principles are generally applicable to all the internal and external quality characteristics.

NOTE 2 Some of the characteristics in this International Standard relate to dependability. Dependability characteristics are defined for all types of systems in IEC 50-191, and where a term in this International Standard is also defined in IEC 50-191, the definition given is broadly compatible.

6.1 Functional suitability

The degree to which the software product provides functions that meet stated and implied needs when the software is used under specified conditions.

6.1.1 Appropriateness

The degree to which the software product provides an appropriate set of functions for specified tasks and user objectives.

NOTE 1 Examples of appropriateness are task-oriented composition of functions from constituent subfunctions, and capacities of tables.

NOTE 2 Appropriateness corresponds to suitability for the task in ISO 9241-110.

NOTE 3 Appropriateness also affects operability.

6.1.2 Accuracy

The degree to which the software product provides the right or specified results with the needed degree of precision.

NOTE For the software to provide accuracy, the associated data needs to have Accuracy, Consistency and Precision (ISO/IEC 25012).

6.1.3 Functional suitability compliance

The degree to which the software product adheres to standards, conventions or regulations in laws and similar prescriptions relating to functional suitability.

6.2 Reliability

The degree to which the software product can maintain a specified level of performance when used under specified conditions.

NOTE 1 Wear or ageing does not occur in software. Limitations in reliability are due to faults in requirements, design, and implementation. Failures due to these faults depend on the way the software product is used and the program options selected rather than on elapsed time.

NOTE 2 The definition of reliability in ISO/IEC DIS 2382-14:1994 is "The ability of functional unit to perform a required function...". In this document, functional suitability is only one of the characteristics of software quality. Therefore, the definition of reliability has been broadened to "maintain a specified level of performance..." instead of "...perform a required function"

NOTE 3 **Dependability** characteristics include availability and its inherent or external influencing factors, such as: reliability, fault tolerance, recoverability, integrity, security, maintainability, durability, and maintenance support. See Annex B.

6.2.1 Availability

The degree to which a software component is operational and available when required for use.

[Based on ISO/IEC 24765:2008]

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

6.2.2 Fault tolerance

The degree to which the software product can maintain a specified level of performance in cases of software faults or of infringement of its specified interface.

NOTE The specified level of performance may include fail-safe capability.

6.2.3 Recoverability

The degree to which the software product can re-establish a specified level of performance and recover the data directly affected in the case of a failure.

NOTE Following a failure, a software product will sometimes be down for a certain period of time, the length of which is assessed by its recoverability.

6.2.4 Reliability compliance

The degree to which the software product adheres to standards, conventions or regulations relating to reliability.

6.3 Performance efficiency

The degree to which the software product provides appropriate performance, relative to the amount of resources used, under stated conditions.

NOTE 1 Resources may include other software products, the software and hardware configuration of the system, and materials (e.g. print paper, diskettes).

NOTE 2 For a system which is operated by a user, the combination of functional suitability, reliability, operability and performance efficiency can be measured externally by quality in use.

6.3.1 Time behaviour

The degree to which the software product provides appropriate response and processing times and throughput rates when performing its function, under stated conditions.

6.3.2 Resource utilisation

The degree to which the software product uses appropriate amounts and types of resources when the software performs its function under stated conditions.

NOTE Human resources are included as part of efficiency in use (7.3.2).

6.3.3 Performance efficiency compliance

The degree to which the software product adheres to standards or conventions relating to performance efficiency.

6.4 Operability

The degree to which the software product can be understood, learned, used and attractive to the user, when used under specified conditions.

NOTE 1 Some aspects of functional suitability, reliability and performance efficiency will also affect operability, but for the purposes of ISO/IEC 25010 they are not classified as operability.

NOTE 2 Users may include operators, end users and indirect users who are under the influence of or dependent on the use of the software. Operability should address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results.

6.4.1 Appropriateness recognisability

The degree to which the software product enables users to recognise whether the software is appropriate for their needs.

NOTE 1 Appropriateness is defined in 6.1.1.

NOTE 2 Appropriateness recognisability will depend on the ability to recognise the appropriateness of the functions from initial impressions of the software and/or any associated documentation.

6.4.2 Learnability

The degree to which the software product enables users to learn its application.

NOTE The internal attributes correspond to suitability for learning as defined in ISO 9241-110.

6.4.3 Ease of use

The degree to which the software product makes it easy for users to operate and control it.

NOTE 1 Aspects of suitability, changeability, adaptability and installability may affect ease of use.

NOTE 2 Ease of use corresponds to controllability, (operator) error tolerance and conformity with user expectations as defined in ISO 9241-110.

NOTE 3 For a system which is operated by a user, the combination of functional suitability, reliability, operability and performance efficiency can be measured externally by quality in use.

NOTE 2 If the software is to be adapted by the end user, adaptability corresponds to suitability for individualisation as defined in ISO 9241-110, and may affect operability.

6.4.4 Helpfulness

The degree to which the software product provides help when users need assistance.

NOTE This includes help that is easy to find, comprehensive and effective.

6.4.5 Attractiveness

The degree to which the software product is attractive to the user.

NOTE This refers to attributes of the software that increase the pleasure and satisfaction of the user, such as the use of colour and the nature of the graphical design.

6.4.6 Technical accessibility

The degree of operability of the software product for users with specified disabilities.

NOTE This includes disabilities associated with ageing.

6.4.7 Operability compliance

The degree to which the software product adheres to standards, conventions, style guides or regulations relating to operability.

6.5 Security

The protection of system items from accidental or malicious access, use, modification, destruction, or disclosure.

[ISO/IEC 15026:1998]

NOTE 1 This also applies to data in transmission.

NOTE 2 **Safety** is defined as a characteristic of quality in use, as it does not relate to software alone, but to a whole system.

NOTE 3 **Survivability** (The degree to which the software product continues to fulfil its mission by providing essential services in a timely manner in spite of the presence of attacks) is covered by Recoverability (6.2.3)

NOTE 4 **Immunity** (the degree to which the software product is resistant to attack) is covered by Robustness (6.2.4)

6.5.1 Confidentiality

The degree to which the software product provides protection from unauthorized disclosure of data or information, whether accidental or deliberate.

6.5.2 Integrity

The degree to which the accuracy and completeness of assets are safeguarded.

[based on ISO/IEC 13335-1:2004]

6.5.3 Non-repudiation

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

[based on ISO 7498-2:1989]

6.5.4 Accountability

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

[based on ISO 7498-2:1989]

6.5.5 Authenticity

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

[based on ISO/IEC 13335-1:2004]

6.5.6 Security compliance

The degree to which the software product adheres to standards, conventions or regulations relating to security.

6.6 Compatibility

The ability of two or more software components to exchange information and/or to perform their required functions while sharing the same hardware or software environment.

[based on ISO/IEC 24765:2008]

6.6.1 Replaceability

The degree to which the software product can be used in place of another specified software product for the same purpose in the same environment.

NOTE 1 For example, the replaceability of a new version of a software product is important to the user when upgrading.

NOTE 2 Replaceability may include attributes of both installability and adaptability. The concept has been introduced as a subcharacteristic of its own because of its importance.

6.6.2 Co-existence

The degree to which the software product can co-exist with other independent software in a common environment sharing common resources without any detrimental impacts.

6.6.3 Interoperability

The degree to which the software product can be cooperatively operable with one or more other software products.

6.6.4 Compatibility compliance

The degree to which the software product adheres to standards, conventions or regulations relating to compatibility.

6.7 Maintainability

The degree to which the software product can be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

6.7.1 Modularity

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

[ISO/IEC 24765:2008]

6.7.2 Reusability

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

[IEEE 1517-1999]

6.7.3 Analysability

The degree to which the software product can be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.

NOTE Implementation can include providing mechanisms for the software product to analyse its own faults and report on the conditions prior to a failure or other event.

6.7.4 Changeability

The degree to which the software product enables a specified modification to be implemented. The ease with which a software product can be modified.

NOTE 1 Implementation includes coding, designing and documenting changes.

NOTE 2 If the software is to be modified by the end user, changeability may affect operability.

6.7.5 Modification stability

The degree to which the software product can avoid unexpected effects from modifications of the software.

6.7.6 Testability

The degree to which the software product enables modified software to be validated.

6.7.7 Maintainability compliance

The degree to which the software product adheres to standards or conventions relating to maintainability.

6.8 Transferability

The degree to which the software product can be transferred from one environment to another.

6.8.1 Portability

The ease with which a system or component can be transferred from one hardware or software environment to another

[ISO/IEC 24765:2008]

6.8.2 Adaptability

The degree to which the software product can be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.

NOTE Adaptability includes the scalability of internal capacity (e.g. screen fields, tables, transaction volumes, report formats, etc.).

6.8.3 Installability

The degree to which the software product can be successfully installed and uninstalled in a specified environment.

NOTE If the software is to be installed by an end user, installability can affect the resulting suitability and operability.

6.8.4 Transferability compliance

The degree to which the software product adheres to standards or conventions relating to portability.

7 System quality in use model

7.1 Quality in use

Quality in use is the degree to which a product used by specific users meets their needs to achieve specific goals with effectiveness in use, efficiency in use, flexibility in use, safety and satisfaction in use in specific contexts of use.

Quality in use is a measure of the quality of the system in a real or simulated operational environment. It is determined by the quality of the software, hardware, operating environment, and the characteristics of the users, tasks and social environment. All these factors contribute to quality in use. Quality in use can be used to assess the quality of software in a specific context of use.

The attributes of quality in use are categorised into three characteristics: usability in use, flexibility in use, and safety (Figure 6).

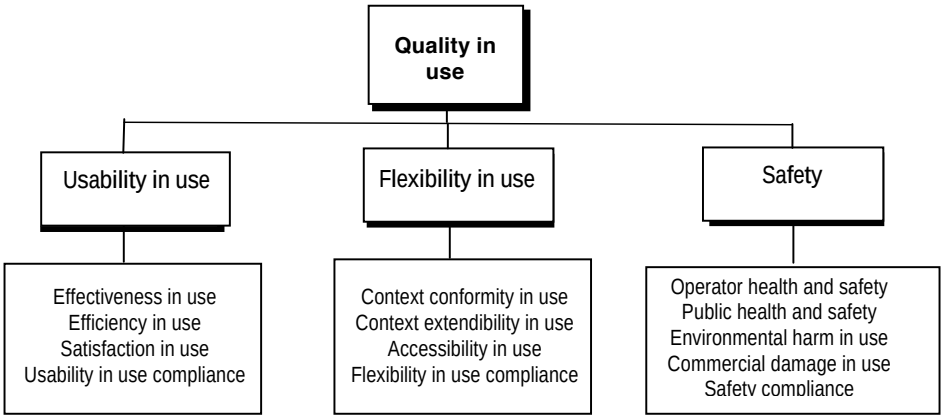


Figure 6 - Quality model for quality in use

Examples of quality in use measures are given in ISO/IEC TR 9126-4 (to be replaced by ISO/IEC 25024).

Achieving quality in use is dependent on achieving the necessary external quality, which in turn is dependent on achieving the necessary internal quality (Figure C.3). Measures are normally required at all three levels, as meeting criteria for internal measures is not usually sufficient to ensure achievement of criteria for external measures, and meeting criteria for external measures of subcharacteristics is not usually sufficient to ensure achieving criteria for quality in use.

The basic measures of quality in use are effectiveness in use, efficiency in use and satisfaction in use (collectively called usability in use).

NOTE 1 Before the product is released, quality in use can be specified and measured in a test environment for the intended users, goals and contexts of use. Once in use, it can be measured for actual users, goals and contexts of use. The actual needs of users may not be the same as those anticipated in requirements, so actual quality in use may be different from quality in use measured earlier in a test environment.

7.2 Usability in use

The degree to which specified users can achieve specified goals with effectiveness in use, efficiency in use and satisfaction in use in a specified context of use.

[based on ISO 9241-11]

7.2.1 Effectiveness in use

The degree to which specified users can achieve specified goals with accuracy and completeness in a specified context of use.

[based on ISO 9241-11]

7.2.2 Efficiency in use

The degree to which specified users expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

[based on ISO 9241-11]

NOTE Relevant resources can include time to complete the task, materials, or the financial cost of usage.

7.2.3 Satisfaction in use

The degree to which users are satisfied in a specified context of use.

Satisfaction is further subdivided into sub-subcharacteristics:

- Likability (cognitive satisfaction)
- Pleasure (emotional satisfaction)
- Comfort (physical satisfaction)
- Trust

NOTE Satisfaction is the user's response to interaction with the product, and includes attitudes towards use of the product.

7.2.4 Usability in use compliance

Adherence to standards or conventions relating to usability in use.

7.3 Flexibility in use

The degree to which the product is usable in all potential contexts of use.

7.3.1 Context conformity in use

The degree to which usability in use meets requirements in all the intended contexts of use.

7.3.2 Context extendibility in use

The degree of usability in use in contexts beyond those initially intended.

NOTE 1 Context extendibility can include providing usability for additional user groups, tasks and cultures.

NOTE 2 This enables products to take account of circumstances, opportunities and individual preferences that may not have been anticipated in advance.

7.3.3 Accessibility in use

The degree of usability in use for users with specified disabilities.

NOTE This includes disabilities associated with ageing.

7.3.4 Flexibility in use compliance

Adherence to standards or conventions relating to flexibility in use.

7.4 Safety

Acceptable levels of risk of harm to people, business, data, software, property or the environment in the intended contexts of use.

7.4.1 Operator health and safety

Acceptable levels of risk of harm to the operator in the intended contexts of use.

7.4.2 Public health and safety

Acceptable levels of risk of harm to the public in the intended contexts of use.

7.4.3 Environmental harm in use

Acceptable levels of risk of harm to property or the environment in the intended contexts of use.

7.4.4 Commercial damage in use

Acceptable levels of risk of a failure that would lead to commercial damage or reputation damage in the intended contexts of use.

NOTE Risks are usually a result of deficiencies in the functionality (including security), reliability, usability or maintainability.

7.4.5 Safety compliance

The degree of conformance to standards, conventions or regulations relating to safety.

Annex A
(informative)
Comparison with the quality model in ISO/IEC 9126-1

Table A.1 – Comparison with the previous model in ISO/IEC 9126-1:2001.

	This International Standard	ISO/IEC 9126-1	Notes
6	**Software product quality**	**Internal and External**	Quality in use is now a system quality
6.1	**Functional suitability**	**Functionality**	New name is more accurate, and avoids confusion with other meanings of “functionality”
6.1.1	Appropriateness	Suitability	Renamed
6.1.2	Accuracy	Accuracy	
		Interoperability	Moved to Compatibility
		Security	Now a characteristic
6.2	**Reliability**	**Reliability**	
6.2.1	Availability	Maturity	Availability is more important than Maturity. Maturity as previously defined was closely related to fault tolerance, but has other meanings.
6.2.2	Fault tolerance	Fault tolerance	
6.2.3	Recoverability	Recoverability	
6.2.4	Robustness		New subcharacteristic
6.3	**Performance efficiency**	**Efficiency**	Renamed to avoid conflicting with the definition of efficiency in 25062
6.3.1	Time behaviour	Time behaviour	
6.3.2	Resource utilisation	Resource utilisation	
6.4	**Operability**	**Usability**	Renamed to avoid conflicting with the definition of usability in 25062
6.4.1	Appropriateness recognisability	Understandability	New name is more accurate
6.4.2	Learnability	Learnability	
6.4.3	Ease of use	Operability	Renamed
6.4.4	Helpfulness		New subcharacteristic
6.4.5	Attractiveness	Attractiveness	
6.4.5	Technical accessibility		New subcharacteristic

6.5	**Security**	Security	Previously a sub-characteristic
6.5.1	Confidentiality		
6.5.2	Integrity		
6.5.3	Non-repudiation		
6.5.4	Accountability		
6.5.5	Authenticity		
6.5.6	Immunity		
6.5.7	Survivability		
6.6	**Compatibility**		Some subcharacteristics of Portability were not logically part of “transfer from one environment to another”
6.6.1	Replaceability	Replaceability	
6.6.2	Co-existence	Co-existence	
6.6.3	Interoperability		Moved from Functionality
6.8	**Maintainability**	**Maintainability**	
6.8.1	Modularity		New subcharacteristic
6.8.2	Reusability		New subcharacteristic
6.8.3	Analysability	Analysability	
6.8.4	Changeability	Changeability	
6.8.5	Modification stability	Stability	New name is more accurate
6.8.6	Testability	Testability	
6.9	**Transferability**	**Portability**	The previous Portability characteristic did not include a subcharacteristic for portability, and co-existence did not fit.
	Adaptability	Adaptability	
	Portability		Original characteristic had no subcharacteristic measuring “transfer from one environment to another”
	Installability	Installability	
		Co-existence	Moved to Compatibility
		Replaceability	Moved to Compatibility
7.2	**Quality in use**	**Quality in use**	
7.3	**Usability in use**		This characteristic is aligned with usability in 25062
7.3.1	Effectiveness in use	Effectiveness	

7.3.2	Efficiency in use	Productivity	Aligned with efficiency in 25062
7.3.3	Satisfaction in use	Satisfaction	
7.4	**Flexibility in use**		New characteristic
7.4.1	Context conformity in use		It is important that a product is usable in all required contexts of use
7.4.2	Context extendibility in use		
7.4.3	Accessibility in use		Accessibility is an important perspective on usability in use
7.5	**Safety**	Safety	
7.5.1	Operator health and safety		New subcharacteristic
7.5.2	Public health and safety		New subcharacteristic
7.5.3	Environmental harm in use		New subcharacteristic
7.5.4	Commercial damage in use		New subcharacteristic

Annex B
(informative)
Example of mapping to dependability

The North Texas Net Centric Systems Consortium uses the following definition of dependability.

> The **dependability** of a system is its ability to deliver specified services to the end users so that they can justifiably rely on and trust the services provided by the system. **Dependability** has several attributes, including reliability, availability, maintainability, confidentiality, integrity, and safety.

- **Availability**. The availability of a system for a period (0,t) is the probability that the system is available for use at any random time in (0,t).
- **Reliability**. The reliability of a system for a period (0,t) is the probability that the system is continuously operational (i.e., does not fail) in time interval (0,t) given that it is operational at time 0.
- **Maintainability**: The maintainability of a system is a measure of the ability of the system to undergo maintenance or to return to normal operation after a failure.
- **Confidentiality**: The confidentiality of a system is a measure of the degree to which the system can ensure that an unauthorized user will not be able to understand protected information in the system.
- **Integrity and Trustworthiness**. The integrity of a system is the probability that errors or attacks will not lead to damages to the state of the system, including data, code, etc.
- **Safety**. The safety of a system for a period (0,t) is the probability that the system will not incur any catastrophic failures in time interval (0,t).

This definition of dependability maps onto the parts of the ISO/IEC 25010 quality model shown in Table B.1.

Table B.1 – Mapping of dependability

Clause	ISO/IEC 25010	Dependability
6.1	**Functional suitability**	*
6.2	**Reliability**	Reliability
6.2.1	Availability	Availability
6.3	**Performance efficiency**	*
6.4	**Operability**	*
6.5	**Security**	

6.5.1	Confidentiality	Confidentiality
6.5.2	Integrity	Integrity
6.6	**Compatibility**	*
6.8	**Maintainability**	Maintainability
6.9	**Transferability**	*
7.3	**Usability in use**	*
7.4	**Flexibility in use**	*
7.5	**Safety**	Safety

So if this definition of Dependability was being used as part of a wider assessment of software quality, it would also be necessary to consider functional suitability, performance efficiency, operability, compatibility, transferability, usability in use and flexibility in use.

Annex C
(informative)
Using the quality model for measurement

C.1 General

The information in this Annex may be moved to a future revision on ISO/IEC 25000.

C.2 Software quality measurement model

Inherent software properties, that can be distinguished quantitatively or qualitatively, are called attributes. Quality attributes are inherent properties of the software that contribute to quality. Quality attributes are categorised into one or more (sub)-characteristics.

Quality attributes are measured by applying a measurement method. A measurement method is a logical sequence of operations used to quantify an attribute with respect to a specified scale. The result of applying a measurement method is called a quality measure element. The quality characteristics and subcharacteristics can be quantified by applying measurement functions. A measurement function is an algorithm used to combine quality measure elements. The result of applying a measurement function is called a software quality measure. In this way software quality measures become quantifications of the quality characteristics and subcharacteristics. More than one software quality measure may be used to measure a quality characteristic or subcharacteristic.

Figure C.1 from ISO/IEC 25020 shows the relations between the ISO/IEC 25010 quality model, the measure in ISO/IEC 2502n, and the measurement model suggested in ISO/IEC 15939.

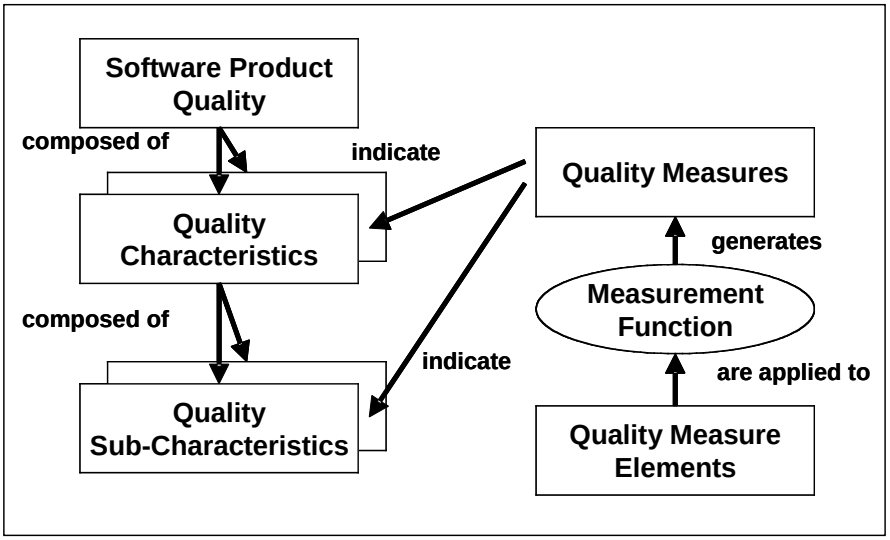


Figure C.1 – Software product quality measurement reference model

C.3 Approaches to quality

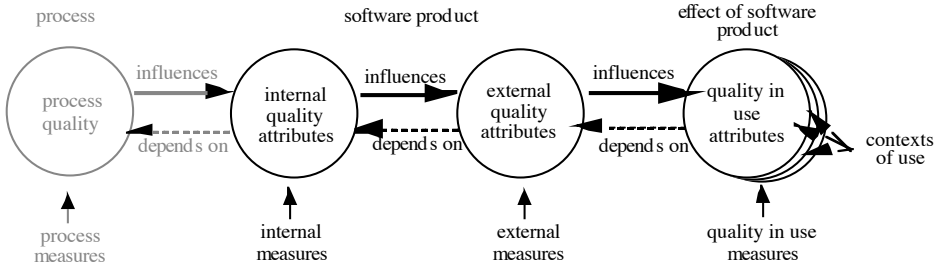


Figure C.2 - Quality in the lifecycle

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

Software product quality can be evaluated by measuring internal attributes (typically static measures of intermediate products), or by measuring external attributes (typically by measuring the behaviour of the code when executed), or by measuring quality in use attributes (when the product is in real or simulated use) (Figure C.2).

Improving process quality (the quality of any of the lifecycle processes defined in ISO/IEC 12207 and ISO/IEC 15288) contributes to improving product quality, and product quality contributes to improving quality in use. Therefore, assessing and improving a process is a means to improve product quality, and evaluating and improving product quality is one means of improving quality in use. Similarly,

evaluating quality in use can provide feedback to improve a product, and evaluating a product can provide feedback to improve a process.

Appropriate internal attributes of the software are a pre-requisite for achieving the required external behaviour, and appropriate external behaviour is a pre-requisite for achieving quality in use (Figure C.2).

C.4 Software product quality life cycle model

The software product quality life cycle model (Figure C.3) addresses software product quality in three principal phases of software product life cycle: product under development, product testing and product in use.

- The phase of a product under development is the subject of internal software quality
- The phase of product testing is the subject of external software quality, and
- The phase of a product in use is the subject of quality in use.

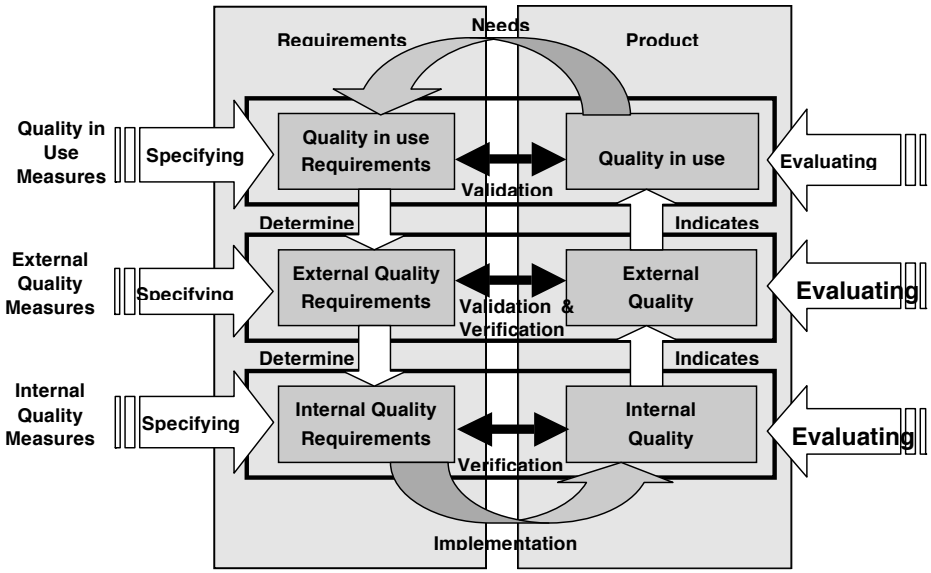


Figure C.3 – Software Product Quality Life Cycle Model

The software product quality life cycle model also indicates that achieving acceptable levels of software quality should be an integral part of the software development

process for each type of quality including: requirements, implementation and validation of the results.

Quality in use requirements specify the required level of quality from the end user's point of view. These requirements are derived from user and other stakeholder needs. Quality in use requirements are used as the target for validation of the software product by the user. Requirements for quality in use characteristics should be stated in the quality requirements specification using quality in use measures and used as criteria when a product is evaluated.

NOTE 1 Quality in use requirements contribute to identifying and to defining external software quality requirements.

EXAMPLE 1 Specified types of uses can achieve a specified task in a specified time.

External software quality requirements specify the required level of quality from the external view. They include requirements derived from stakeholder quality requirements, including quality in use requirements. External software quality requirements are used as the target for technical verification and validation of the software product. Requirements for external software quality characteristics should be stated quantitatively in the quality requirements specification using external measures and used as criteria when a product is evaluated.

NOTE 2 External software quality requirements contribute to identifying and to defining internal software quality requirements.

NOTE 3 External software quality evaluation can be used to predict quality in use.

EXAMPLE 2 Users respond appropriately to error messages and successfully undo errors.

Internal software quality requirements specify the level of required quality from the internal view of the product. They include requirements derived from external software quality requirements. Internal software quality requirements are used to specify properties of intermediate software products (specifications, source code, etc.). Internal software quality requirements may also be used to specify properties of deliverable, non-executable software products such as documentation and manuals. Internal software quality requirements can be used as targets for verification at various stages of development. They can also be used for defining strategies of development and criteria for evaluation and verification during development. Internal quality requirements should be specified quantitatively in terms of internal measures.

NOTE 4 Internal software quality evaluation can be used to predict external software quality.

EXAMPLE 3 All error messages specify corrective action, and all user inputs can be undone.

ISO/IEC 25030 describes software quality requirements, and ISO/IEC 25040 describes the software quality evaluation process.

C.5 Items to be evaluated

Items can be evaluated by direct measurement, or indirectly by measuring their consequences. For example, a process may be assessed indirectly by measuring

and evaluating it's product, and a product may be evaluated indirectly by measuring the task performance of a user (using quality in use measures).

Software never runs alone, but always as part of a larger system typically consisting of other software products with which it has interfaces, and of hardware, human operators, and workflows. The completed software product can be evaluated by the levels of the chosen external measures. These measures describe its interaction with its environment, and are assessed by observing the software in operation. Quality in use can be measured by the extent to which a product used by specified users meets their needs to achieve specified goals with effectiveness, productivity, satisfaction, safety and flexibility.

At the earliest stages of development, only resources and process can be measured. When intermediate products (specifications, source code, etc.) become available, these can be evaluated by the levels of the chosen internal measures. These measures can be used to predict values of the external measures. They may also be measured in their own right, as essential pre-requisites for external quality.

A further distinction can be made between the evaluation of a software product and the evaluation of the system in which it is executed.

NOTE 1 For example, the reliability of a system is assessed by observing all failures due to whatever cause (hardware, software, human error, etc.), whereas the reliability of the software product is assessed by extracting from the observed failures only those that are due to faults (originating from requirements, design or implementation) in the software.

Also, where the boundary of the system is judged to be, depends upon the purpose of the evaluation, and upon who the users are.

NOTE 2 For example, if the users of an aircraft with a computer-based flight control system are taken to be the passengers, then the system upon which they depend includes the flight crew, the airframe, and the hardware and software in the flight control system, whereas if the flight crew are taken to be the users, then the system upon which they depend consists only of the airframe and the flight control system.

Annex D
(normative)
Terms and definitions

D.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 15939:2002.

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.

[ISO/IEC 25000:2005]

D.2
context of use

users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used

[ISO 9241-11:1998]

D.3
end user

individual person who ultimately benefits from the outcomes of the system

NOTE The end user may be a regular operator of the software product or a casual user such as a member of the public.

[ISO/IEC 25000:2005]

D.4
implied needs

needs that may not have been stated but are actual needs

NOTE Some implied needs only become evident when the software product is used in particular conditions.

EXAMPLE Implied needs include: needs not stated but implied by other stated needs and needs not stated because they are considered to be evident or obvious.

[ISO/IEC 25000:2005]

D.5
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, derived measures, and indicators.

[ISO/IEC 15939:2002]

D.6
measure (verb)

make a measurement

[ISO/IEC 14598-1:1999]

D.7
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.).

D.8
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

[ISO/IEC 25000:2005]

D.9
quality model

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

[ISO/IEC 25000:2005]

D.10
requirements

expression of a perceived need that something be accomplished or realized

NOTE The requirements may be specified as part of a contract, or specified by the development organisation, as when a product is developed for unspecified users, such as consumer software, or the requirements may be more general, as when a user evaluates products for comparison and selection purpose.

[ISO/IEC 25000:2005]

D.11
software product

set of computer programs, procedures, and possibly associated documentation and data

[ISO/IEC 12207:1995]

NOTE 1 Products include intermediate products, and products intended for users such as developers and maintainers.

NOTE 2 In SQuaRE standards software quality has the same meaning as software product quality.

D.12
software quality

degree to which the software product satisfies stated and implied needs when used under specified conditions

NOTE This definitions differs from the ISO 9000:2000 quality definition mainly because the software quality definition refers to the satisfaction of stated and implied needs, while the ISO 9000 quality definition refers to the satisfaction of requirements.

[ISO/IEC 25000:2005]

D.13
software quality requirement

a requirement that a software quality attribute be present in software

D.14
software quality characteristic

category of software quality attributes that bears on software quality

NOTE Software quality characteristics may be refined into multiple levels of subcharacteristics and finally into software quality attributes.

[ISO/IEC 25000:2005]

D.15
system

a combination of interacting elements organised to achieve one or more stated purposes

NOTE 1 A system may be considered as a product or as the services it provides.

NOTE 2 In practice, the interpretation of its meaning is frequently clarified by the use of an associative noun, e.g. aircraft system. Alternatively the word system may be substituted simply by a context dependent synonym, e.g. aircraft, though this may then obscure a system principles perspective.

[ISO/IEC 15288:2008]

D.16
user

individual or organisation that uses the system to perform a specific function

NOTE Users may include operators, recipients of the results of the software, or developers or maintainers of software.

[ISO/IEC 15939:2002]

D.17
validation

confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled

NOTE 1 "Validated" is used to designate the corresponding status.

[ISO 9000:2000]

NOTE 2 In design and development, validation concerns the process of examining a product to determine conformity with user needs.

NOTE 3 Validation is normally performed on the final product under defined operating conditions. It may be necessary in earlier stages.

NOTE 4 Multiple validations may be carried out if there are different intended uses.

D.18
verification

confirmation, through the provision of objective evidence, that specified requirements have been fulfilled

NOTE 1 "Verified" is used to designate the corresponding status.

[ISO 9000:2000]

NOTE 2 In design and development, verification concerns the process of examining the result of a given

Annex E
(informative)

Bibliography

IEC 50-(191) *International Electrotechnical vocabulary - Dependability and quality of service*

IEEE 610.12-1990 Standard Glossary of Software Engineering Terminology

IEEE 1517-1999 (R2004) IEEE Standard for Information Technology - Software Life Cycle Processes - Reuse Processes

ISO/IEC 2382-1:1993 Data processing - Vocabulary - Part 1: Fundamental terms

ISO/IEC 2382-14:1997 *Reliability, maintainability and availability*

ISO/IEC 2382-20:1990, Information technology -- Vocabulary - Part 20 : Systems development.

ISO 7498-2:1989, Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture

ISO 9001:2000, Quality management systems -- Requirements

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

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

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

ISO 9241-11:1997, Ergonomic requirements for office work with visual display terminals (VDT)s - Part 11: Guidance on usability.

ISO 9241-110:2006, Ergonomics of human-system interaction -- Part 110: Dialogue principles

ISO/IEC 12207:2008, Systems and software engineering -- Software life cycle processes.

ISO/IEC 13335-1:2004, Information technology -- Security techniques -- Management of information and communications technology security -- Part 1: Concepts and models for information and communications technology security management

ISO 13407:1999, Human centred design processes for interactive systems.

ISO/IEC 14598-2:2000, Information Technology - Software product evaluation - Part 2: Planning and management

ISO/IEC 14598-3:2000, Information Technology - Software product evaluation - Part 3: Process for developers

ISO/IEC 14598-4:1999, Information Technology - Software product evaluation - Part 4: Process for acquirers

ISO/IEC 14598-5:1998, Information Technology - Software product evaluation - Part 5: Process for evaluators

ISO/IEC 14598-6:2001, Information Technology - Software product evaluation - Part 6: Documentation of evaluation modules

ISO/IEC 15026:1998, Information technology -- System and software integrity levels

ISO/IEC 15504 (parts 1 to 5) Information Technology - Process Assessment

ISO/IEC 15288:2008 Systems and software engineering -- System life cycle processes

ISO/IEC FCD 24765:2008, Systems and software engineering vocabulary

ISO/IEC 25000:2005 Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE.

ISO/IEC FCD 25012:2008 Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model

ISO/IEC 25020:2007 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Measurement reference model and guide

North Texas Net Centric Systems Consortium (2008). Dependability definitions. http://csrl.unt.edu/~kavi/NetCentric/Dependability-Defn.doc Retrieved 2 June 2008.