Information Technology Software product quality Part 1: Quality model

1 Scope

This part of ISO/IEC 9126 describes a model for software product quality which includes internal quality, external quality and quality in use. It specifies six characteristics for internal and external quality, which are further sub-divided into subcharacteristics. 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. This part of ISO/IEC 9126 does not elaborate the model for internal and external quality below the level of subcharacteristics.

This part of ISO/IEC 9126 also specifies four quality in use characteristics which are the combined effect of the software quality characteristics for the user. This part of ISO/IEC 9126 does not elaborate the model for quality in use below the level of characteristics.

The characteristics defined are applicable to every kind of software, including computer programs and data contained in firmware. The characteristics and subcharacteristics provide consistent terminology for software quality. They also provide a framework for specifying quality requirements for software, and making trade-offs between software product capabilities.

Normative Annex A provides recommendations and requirements for software product metrics and quality in use metrics. Examples of these metrics are contained in other parts of ISO/IEC 9126. These metrics are applicable when specifying quality requirements and design goals for software products, including intermediate products. An explanation of how this quality model can be applied in software product evaluation is contained in ISO/IEC 14598-1.

This part of ISO/IEC 9126 is intended for use by developers, acquirers, quality assurance staff and independent evaluators, particularly those responsible for specifying and evaluating software product quality. It enables software product quality to be specified and evaluated from different perspectives by those associated with acquisition, requirements, development, use, evaluation, support, maintenance, quality assurance and audit of software. Examples of uses of the quality model defined in this part of ISO/IEC 9126 are to:

- validate the completeness of a requirements definition;
- identify software requirements;
- identify software design objectives;
- identify software testing objectives;
- identify quality assurance criteria;
- identify user acceptance criteria for a completed software product.

NOTE 1 This part of ISO/IEC 9126 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.

NOTE 2 This part of ISO/IEC 9126 can be used in conjunction with ISO/IEC 12207 (which is concerned with the software lifecycle) to provide:

- a framework for software quality requirements definition in the primary lifecycle process;
- support for review, verification and validation in supporting life cycle processes.

NOTE 3 This part of ISO/IEC 9126 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 statement of conformance to this part of ISO/IEC 9126 shall explain how the appropriate characteristics and subcharacteristics from clauses 6 and 7 were selected , and the reasons for any exclusions.

Requirements for metrics are contained in A.4.

3 Normative references

The following normative document contains provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9126. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 9126 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.

ISO/IEC 14598-1:1998, *Information Technology - Software product evaluation - Part 1: General overview*

4 Terms and definitions

For the purposes of all parts of ISO/IEC 9126, the following definition and the definitions contained in ISO/IEC 14598-1 apply.

NOTE The relevant definitions are reproduced in informative annex B.

4.1
level of performance
the degree to which the needs are satisfied, represented by a specific set of values for the quality characteristics

5 Quality model framework

This clause describes a quality model which explains the relationship between different approaches to quality. A specific implementation of this quality model is given in clauses 6 and 7.

5.1 Approaches to quality

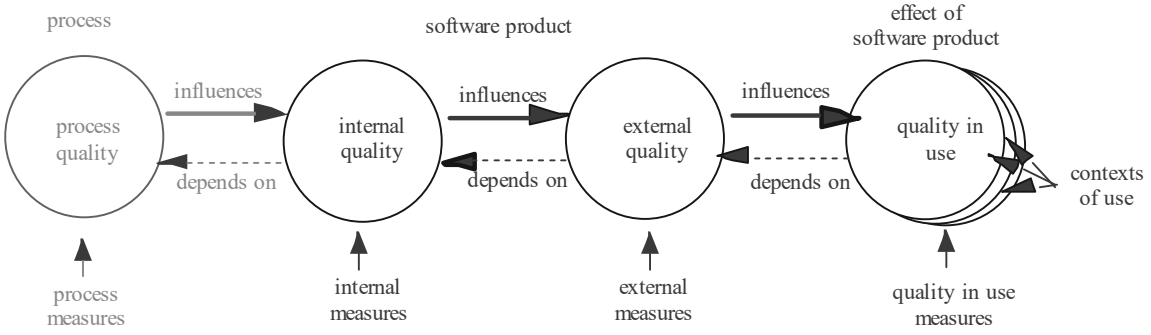

Figure 2 - Quality in the life-cycle

Evaluation of software products in order to achieve software product quality is one of the processes in the software development life-cycle. Software 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). The objective is for the product to have the required effect in a particular context of use (Figure 2).

Process quality (the quality of any of the life cycle processes defined in ISO/IEC 12207) 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 quality evaluation processes are required to support the measurement of quality during development. 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 2).

The requirements for software product quality will generally include criteria for internal quality, external quality and quality in use, to meet the needs of developers, maintainers, acquirers and end users.

5.2 Product quality and the life-cycle

Quality changes with the life-cycle of the software (e.g., required product quality at the start of the life-cycle differs from actual or delivered product quality). The assessment of quality is also a reflection of diverse points of view. It is necessary to define these diverse views of quality and changes in quality with the life-cycle, in order to manage quality properly at each stage of the life-cycle. The following are the different views of product quality at different stages in the software life-cycle (see ISO/IEC 14598-1:1998 Figure 4):

Goal Quality is the necessary and sufficient quality that reflects real user needs.

ISO 8402 defines quality in terms of the ability to satisfy stated and implied needs. However, needs stated by a user do not always reflect real user needs, because a user is often not aware of his real needs and needs may change after they are stated. So goal quality cannot be completely defined at the beginning of design. Yet, developers must keep this goal in mind and try to get closer to it. Goal quality does not necessarily mean perfect quality, but necessary and sufficient quality. Requirements for goal quality can be specified by quality in use metrics. It is often useful to document particular envisaged scenarios of use to more clearly understand the user needs and any potential conflicts. Goal quality can be evaluated by measuring quality in use when the product is complete, and requirements for quality in use should if possible be included in the quality requirements specification.

External Quality Requirements are what is actually stated in the quality requirements specification.

External quality requirements should be used as the target for initial validation. External quality requirements for all the quality characteristics defined in this part of ISO/IEC 9126 should be stated in the quality requirements specification using external metrics. Not only the optimal requirements, but also the minimum requirements, should be stated so that both the user and the developer can avoid unnecessary cost and schedule overruns.

Internal Quality Requirements specify the internal quality represented in the core parts or backbone of software design, including software architecture, program structure, and user interface design strategy. Specific internal quality requirements should be specified quantitatively using internal metrics.

Internal quality is a reflection of the design philosophy and strategy. It includes internal software properties which are mainly related to the programming environment, and also internal quality requirements for the quality characteristics defined in this part of ISO/IEC 9126. Details of software quality may be improved during code implementation and testing, but the fundamental nature of the software product quality represented by internal quality remains unchanged.

Estimated (or Predicted) External Quality is the external quality that is estimated or predicted for the end software product at each stage of development, and based on knowledge of the internal quality.

External quality may be estimated and predicted during development for each quality characteristic defined in this part of ISO/IEC 9126. For the purposes of prediction, technology should be developed to show the co-relation between external quality and internal quality.

External Quality is the quality of the delivered product, which is typically evaluated by testing in a simulated environment with simulated data using external metrics.

During testing, most faults should be discovered and eliminated. However, some faults may still remain after testing. As it is difficult to correct the software architecture or other fundamental design aspects of the software, the fundamental design usually remains unchanged throughout testing.

Quality in Use is the user's view of the quality of an environment containing software, and is measured from the results of using the software in the environment, rather than properties of the software itself (quality in use is defined in clause 7).

NOTE 'Users' refers to any type of intended users, including both operators and maintainers, and their requirements can be different.

The quality in the users' environment may be different from that in the developers' environment, because some functions may not be visible to a user, or may not be used by a user. The user evaluates only those attributes of software, which are visible during actual use. Sometimes, software attributes specified by an end user during the requirements analysis phase, no longer meet the user requirements

when the product is in use, because of changing user requirements and the difficulty of specifying implied needs.

5.3 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 metrics).

Software never runs alone, but always as part of a larger system consisting of other software products with which it has interfaces, hardware, human operators, and work flows. The completed software product can be evaluated by the levels of the chosen external metrics. These metrics describe its interaction with its environment, and are assessed by observing the software in operation. Quality in use (the capability of a product to meet stated and implied needs) can be measured by the extent to which a product used by specified users meets their needs to achieve specified goals with effectiveness, productivity, safety and satisfaction. This will normally be complemented by measures of more specific software quality characteristics, which is also possible earlier in the development process.

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 metrics. These metrics can be used to predict values of the external metrics. 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. 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 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.

5.4 Evaluation using a quality model

Software product quality should be evaluated using a defined quality model. The quality model should be used when setting quality goals for software products and intermediate products. Both software quality and quality in use should be hierarchically decomposed into a quality model composed of characteristics and subcharacteristics which can be used as a checklist of issues related to quality. Clauses 6 and 7 define a hierarchical quality model for software quality and quality in use (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. Resources for evaluation need to be allocated between the different types of measurement dependent on the business objectives and the nature of the product and design process.

6 Software quality characteristics

The quality model in this part of ISO/IEC 9126 categorises software quality attributes into six characteristics (functionality, reliability, usability, efficiency, maintainability and portability), which are further sub-divided into subcharacteristics (Figure 3). The subcharacteristics can be measured by internal or external metrics.

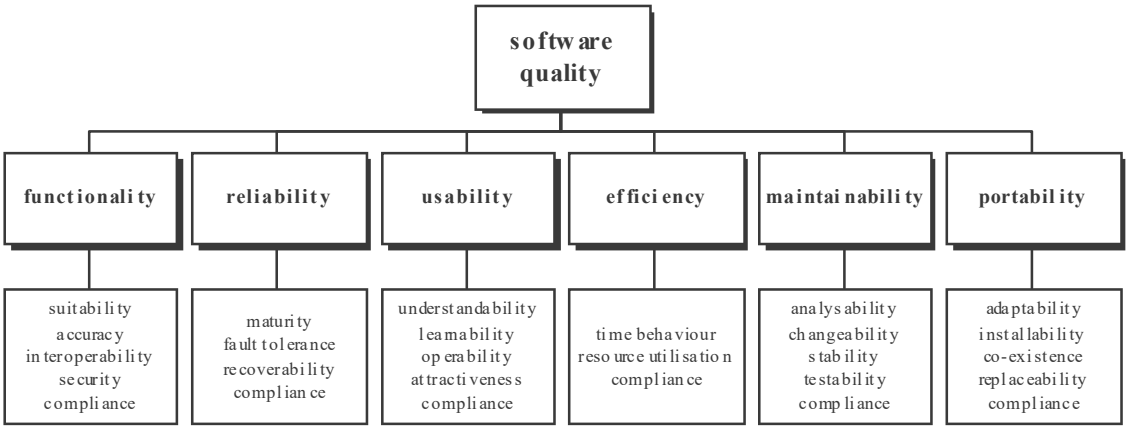

Figure 3 - Internal and external quality

Definitions are given for each quality characteristic and the subcharacteristics of the software which influence the quality characteristic. For each characteristic and subcharacteristic, the capability of the software is determined by a set of internal attributes which can be measured. Examples of internal metrics are given in ISO/IEC 9126-3. 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 metrics are given in ISO/IEC 9126-2.

NOTE Some of the characteristics in this part of ISO/IEC 9126 relate to dependability. Dependability characteristics are defined for all types of systems in IEC 50(191), and where a term in this part of ISO/IEC 9126 is also defined in IEC 50(191), the definition given is broadly compatible.

6.1 Functionality

The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.

NOTE 1 This characteristic is concerned with what the software does to fulfil needs, whereas the other characteristics are mainly concerned with when and how it fulfils needs.

NOTE 2 For the stated and implied needs in this characteristic, the note to the definition of quality applies, (see B.21).

NOTE 3 For a system which is operated by a user, the combination of functionality, reliability, usability and efficiency can be measured externally by quality in use (see clause 7).

6.1.1 Suitability

The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives.

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

NOTE 2 Suitability corresponds to suitability for the task in ISO 9241-10

NOTE 3 Suitability also affects operability.

6.1.2 Accuracy

The capability of the software product to provide the right or agreed results or effects.

NOTE This includes the needed degree of precision of calculated values.

6.1.3 Interoperability

The capability of the software product to interact with one or more specified systems.

NOTE Interoperability is used in place of compatibility in order to avoid possible ambiguity with replaceability (see 6.6.4).

6.1.4 Security

The capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them.

[ISO/IEC 12207: 1995]

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.

6.1.5 Compliance

The capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions.

6.2 Reliability

The capability of the software product to 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, functionality 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"

6.2.1 Maturity

The capability of the software product to avoid failure as a result of faults in the software.

6.2.2 Fault tolerance

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

NOTE 1 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.

NOTE 2 **Availability** is the capability of the software product to be in a state to perform a required function at a given point in time, under stated conditions of use. 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).

6.2.4 Compliance

The capability of the software product to adhere to standards, conventions or regulations relating to reliability.

6.3 Usability

The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.

NOTE 1 Some aspects of functionality, reliability and efficiency will also affect usability, but for the purposes of ISO/IEC 9126 are not classified as usability.

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. Usability should address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results.

6.3.1 Understandability

The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.

NOTE This will depend on the documentation and initial impressions given by the software.

6.3.2 Learnability

The capability of the software product to enable the user to learn its application.

NOTE The internal attributes correspond to suitability for learning as defined in ISO 9241-10.

6.3.3 Operability

The capability of the software product to enable the user to operate and control it.

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

NOTE 2 Operability corresponds to controllability, error tolerance and conformity with user expectations as defined in ISO 9241-10.

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

6.3.4 Attractiveness

The capability of the software product to be attractive to the user.

NOTE This refers to attributes of the software intended to make the software more attractive to the user, such as the use of colour and the nature of the graphical design.

6.3.5 Compliance

The capability of the software product to adhere to standards, conventions, style guides or regulations relating to usability.

6.4 Efficiency

The capability of the software product to provide 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 functionality, reliability, usability and efficiency can be measured externally by quality in use.

6.4.1 Time behaviour

The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.

NOTE Human resources are included as part of productivity (7.1.2).

6.4.2 Resource utilisation

The capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions.

6.4.3 Compliance

The capability of the software product to adhere to standards or conventions relating to efficiency.

6.5 Maintainability

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

6.5.1 Analysability

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

6.5.2 Changeability

The capability of the software product to enable a specified modification to be implemented.

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.

6.5.3 Stability

The capability of the software product to avoid unexpected effects from modifications of the software.

6.5.4 Testability

The capability of the software product to enable modified software to be validated.

6.5.5 Compliance

The capability of the software product to adhere to standards or conventions relating to maintainability.

6.6 Portability

The capability of the software product to be transferred from one environment to another.

NOTE The environment may include organisational, hardware or software environment.

6.6.1 Adaptability

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

NOTE 1 Adaptability includes the scalability of internal capacity (e.g. screen fields, tables, transaction volumes, report formats, etc.).

NOTE 2 If the software is to be adapted by the end user, adaptability corresponds to suitability for individualisation as defined in ISO 9241-10, and may affect operability.

6.6.2 Installability

The capability of the software product to be installed in a specified environment.

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

6.6.3 Co-existence

The capability of the software product to co-exist with other independent software in a common environment sharing common resources.

6.6.4 Replaceability

The capability of the software product to 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 is used in place of **compatibility** in order to avoid possible ambiguity with interoperability (see 6.1.3).

NOTE 3 Replaceability may include attributes of both installability and adaptability. The concept has been introduced as a subcharacteristic of its own because of its importance.

6.6.5 Compliance

The capability of the software product to adhere to standards or conventions relating to portability.

7 Quality in use characteristics

The attributes of quality in use are categorised into four characteristics: effectiveness, productivity, safety and satisfaction (Figure 4).

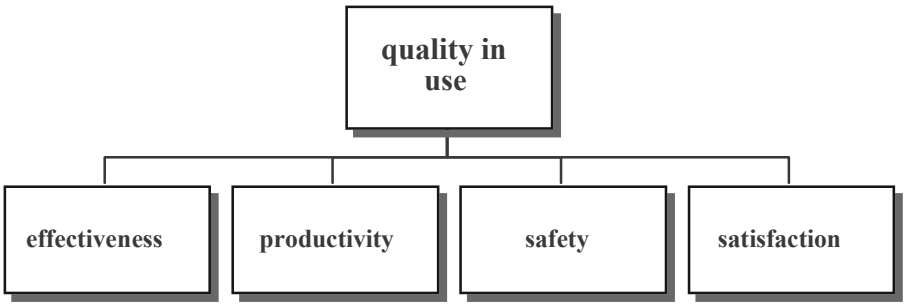

Figure 4 - Quality in use

Quality in use is the user's view of quality. Achieving quality in use is dependent on achieving the necessary external quality, which in turn is dependent on achieving the necessary internal quality (Figure 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. Examples of quality in use metrics are given in ISO/IEC 9126-4

7.1 Quality in use

The capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use.

NOTE 1 Quality in use is the user's view of the quality of an environment containing software, and is measured from the results of using the software in the environment, rather than properties of the software itself.

NOTE 2 Examples of metrics for quality in use are given in ISO/IEC 9126-4.

NOTE 3 The definition of quality in use in ISO/IEC 14598-1 (which is reproduced in Annex B) does not currently include the new characteristic of "safety".

NOTE 4 Usability is defined in ISO 9241-11 in a similar way to the definition of quality in use in this part of ISO/IEC 9126. Quality in use may be influenced by any of the quality characteristics, and is thus broader than usability, which is defined in this part of ISO/IEC 9126 in terms of understandability, learnability, operability, attractiveness and compliance.

7.1.1 Effectiveness

The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.

7.1.2 Productivity

The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

NOTE Relevant resources can include time, effort, materials or financial cost.

7.1.3 Safety

The capability of the software product to achieve acceptable levelsof risk of harm to people, software, equipmemt or the environment in a specified context of use.

NOTE 2 Risks to safety are usually a result of deficencies in the functionality, reliability, usability or maintainability.

7.1.4 Satisfaction

The capability of the software product to satisfy users in a specified context of use.

NOTE Psychometrically-valid questionnaires can be used to obtain reliable measures of satisfaction.