FINAL DRAFT

INTERNATIONAL STANDARD

ISO/IEC FDIS 25010

ISO/IEC JTC 1

Secretariat: **ANSI**

Voting begins on:
2010-10-14

Voting terminates on:
2010-12-14

Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

Ingénierie des systèmes et du logiciel — Exigences de qualité et évaluation des systèmes et du logiciel (SQuaRE) — Modèles de qualité du système et du logiciel

Please see the administrative notes on page iii

Reference number
ISO/IEC FDIS 25010:2010(E)

Contents

Page

Foreword

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

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

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

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

ISO/IEC 25010 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

This first edition of ISO/IEC 25010 cancels and replaces ISO/IEC 9126-1:2001, which has been technically revised.

ISO/IEC 25010 is a part of the SQuaRE series of International Standards, which consists of the following divisions:

- Quality Management Division (ISO/IEC 2500n),
- Quality Model Division (ISO/IEC 2501n),
- Quality Measurement Division (ISO/IEC 2502n),
- Quality Requirements Division (ISO/IEC 2503n),
- Quality Evaluation Division (ISO/IEC 2504n),
- SQuaRE Extension Division (ISO/IEC 25050 – ISO/IEC 25099).

Introduction

Software products and software-intensive computer systems are increasingly used to perform a wide variety of business and personal functions. Realization of goals and objectives for personal satisfaction, business success and/or human safety relies on high-quality software and systems. High-quality software products and software-intensive computer systems are essential to provide value, and avoid potential negative consequences, for the stakeholders.

Software products and software-intensive computer systems have many stakeholders including those who develop, acquire, use, or who are customers of businesses using software-intensive computer systems. Comprehensive specification and evaluation of the quality of software and software-intensive computer systems is a key factor in ensuring value to stakeholders. This can be achieved by defining the necessary and desired quality characteristics associated with the stakeholders' goals and objectives for the system. This includes quality characteristics related to the software system and data as well as the impact the system has on its stakeholders. It is important that the quality characteristics are specified, measured, and evaluated whenever possible using validated or widely accepted measures and measurement methods. The quality models in this International Standard can be used to identify relevant quality characteristics that can be further used to establish requirements, their criteria for satisfaction and the corresponding measures.

This International Standard is derived from ISO/IEC 9126:1991, *Software engineering — Product quality*, which was developed to support these needs. It defined six quality characteristics and described a software product evaluation process model.

ISO/IEC 9126:1991 was replaced by two related multipart standards: ISO/IEC 9126, *Software engineering — Product quality* and ISO/IEC 14598, *Software engineering — Product evaluation*.

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

- The scope of the quality models has been extended to include computer systems, and quality in use from a system perspective.
- Context coverage has been added as a quality in use characteristic, with subcharacteristics *context completeness* and *flexibility*.
- *Security* has been added as a characteristic, rather than a subcharacteristic of functionality, with subcharacteristics *confidentiality*, *integrity*, *non-repudiation*, *accountability* and *authenticity*.
- C*ompatibility* (including *interoperability* and *co-existence)* has been added as a characteristic.
- The following subcharacteristics have been added: *functional completeness, capacity, user error protection, accessibility, availability, modularity* and *reusability*.
- The compliance subcharacteristics have been removed, as compliance with laws and regulations is part of overall system requirements, rather than specifically part of quality.
- The internal and external quality models have been combined as the product quality model.
- When appropriate, generic definitions have been adopted, rather than using software-specific definitions.
- Several characteristics and subcharacteristics have been given more accurate names.

Full details of the changes are in Annex A.

This International Standard is intended to be used in conjunction with the other parts of the SQuaRE series of International Standards (ISO/IEC 25000 to ISO/IEC 25099), and with ISO/IEC 14598 until superseded by the ISO/IEC 2504n series of International Standards.

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

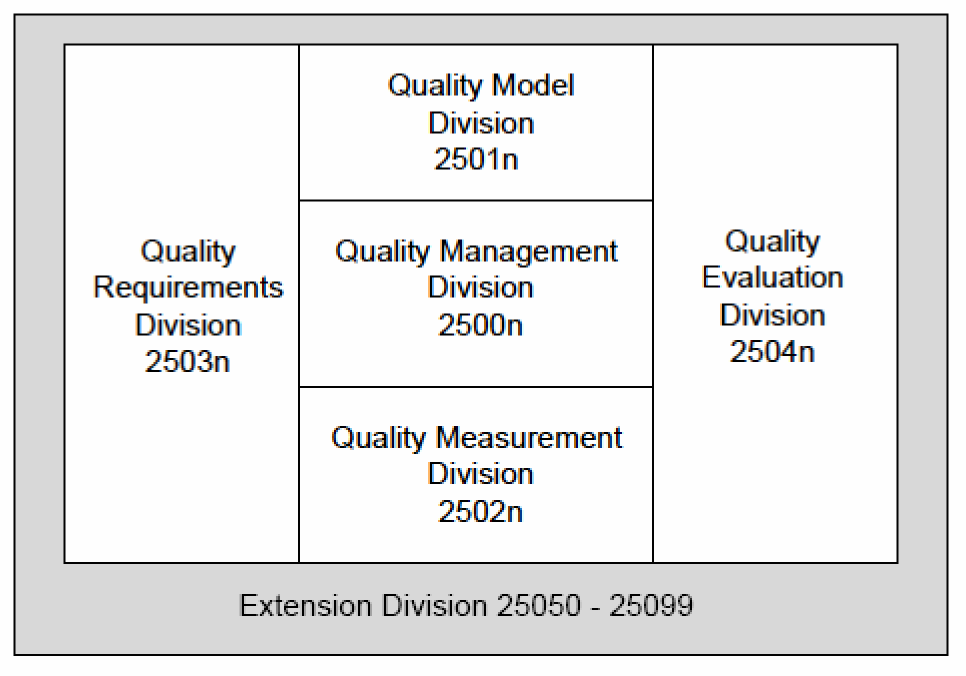

Quality Model Division 2501n
Quality Requirements Division 2503n
Quality Management Division 2500n
Quality Evaluation Division 2504n
Quality Measurement Division 2502n
Extension Division 25050 - 25099

Figure 1 — Organization of SQuaRE series of International Standards

The divisions within the SQuaRE series are:

- **ISO/IEC 2500n - Quality Management Division**. The International Standards that form this division define all common models, terms and definitions further referred to by all other International Standards from the SQuaRE series. The division also provides requirements and guidance for a supporting function that is responsible for the management of the requirements, specification and evaluation of software product quality.

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

- **ISO/IEC 2502n - Quality Measurement Division**. The International 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 are given of internal and external measures for software quality, and measures for quality in use. Quality Measure Elements (QME) forming foundations for these measures are defined and presented.

- **ISO/IEC 2503n - Quality Requirements Division**. The International Standards that form this division help specify quality requirements, based on quality models and quality measures. 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 International 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 Division**. These International Standards currently include requirements for quality of Commercial Off-The-Shelf software and Common Industry Formats for usability reports.

The quality models in this International Standard can be used in conjunction with ISO/IEC 12207 and ISO/IEC 15288, particularly the processes associated with requirements definition, verification and validation with a specific focus on the specification and evaluation of quality requirements. ISO/IEC 25030 describes how the quality models can be used for software quality requirements, and ISO/IEC 25040 describes how the quality models can be used for the software quality evaluation process.

This International Standard can also be used in conjunction with ISO/IEC 15504 (which is concerned with 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 organizational 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.

Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models

1 Scope

This International Standard defines:

a) A quality in use model composed of five characteristics (some of which are further subdivided into subcharacteristics) that relate to the outcome of interaction when a product is used in a particular context of use. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use.

b) A product quality model composed of eight characteristics (which are further subdivided into subcharacteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products.

The characteristics defined by both models are relevant to all software products and computer systems. The characteristics and subcharacteristics provide consistent terminology for specifying, measuring and evaluating system and software product quality. They also provide a set of quality characteristics against which stated quality requirements can be compared for completeness.

NOTE Although the scope of the product quality model is intended to be software and computer systems, many of the characteristics are also relevant to wider systems and services.

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

The scope of the models excludes purely functional properties (see C.6), but it does include functional suitability (see 4.2.1).

The scope of application of the quality models includes supporting specification and evaluation of software and software-intensive computer systems from different perspectives by those associated with their acquisition, requirements, development, use, evaluation, support, maintenance, quality assurance and control, and audit. The models can, for example, be used by developers, acquirers, quality assurance and control 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:

- identifying software and system requirements;
- validating the comprehensiveness of a requirements definition;
- identifying software and system design objectives;
- identifying software and system testing objectives;
- identifying quality control criteria as part of quality assurance;
- identifying acceptance criteria for a software product and/or software-intensive computer system;
- establishing measures of quality characteristics in support of these activities.

2 Conformance

Any quality requirement, quality specification, or evaluation of quality that conforms to this International Standard shall either:

a) use the quality models defined in 4.1 and 4.2; or

b) tailor the quality model giving the rationale for any changes and provide a mapping between the tailored model and the *standard* model.

3 Quality model framework

3.1 Quality models

The quality of a system is the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value. These stated and implied needs are represented in the SQuaRE series of International Standards by quality models that categorize product quality into characteristics, which in some cases are further subdivided into subcharacteristics. (A subcharacteristic can sometimes be divided into sub-subcharacteristics.) This hierarchical decomposition provides a convenient breakdown of product quality. However, the set of subcharacteristics associated with a characteristic have been selected to be representative of typical concerns without necessarily being exhaustive.

The measurable quality-related properties of a system are called quality properties, with associated quality measures. To arrive at measures of the quality characteristic or subcharacteristic, it might be possible to either directly measure it, or it might be necessary to identify a collection of properties that together cover the characteristic or subcharacteristic, obtain quality measures for each, and combine them computationally to arrive at a derived quality measure corresponding to the quality characteristic or subcharacteristic (see Annex C). Figure 2 shows the relationship between quality characteristics and subcharacteristics, and quality properties.

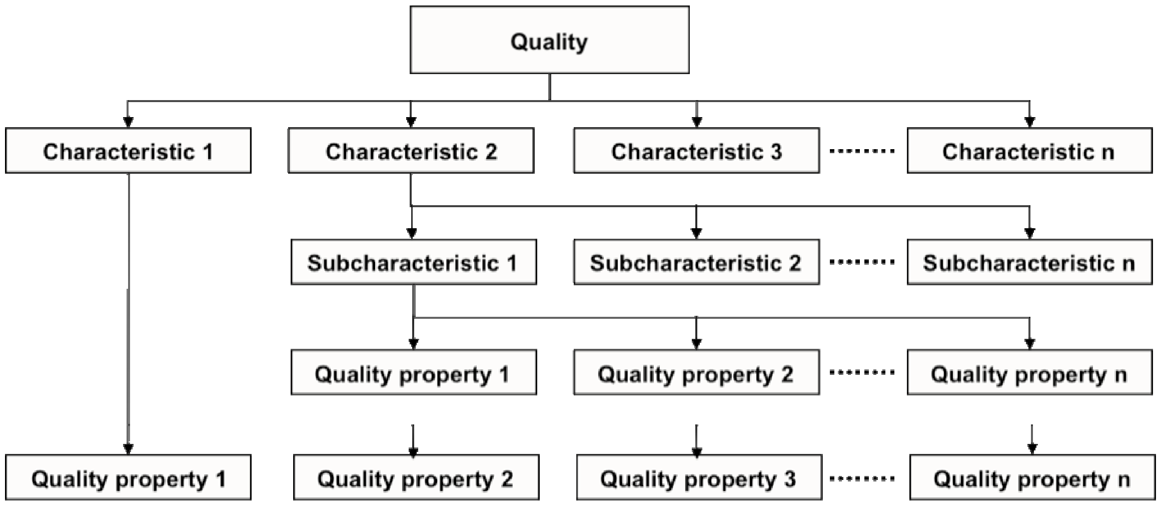


Figure 2 — Structure used for the quality models

Currently there are three quality models in the SQuaRE series: the quality in use model and the product quality model in this International Standard, and the data quality model in ISO/IEC 25012. The quality models together serve as a framework to ensure that all characteristics of quality are considered. These models provide a set of quality characteristics relevant to a wide range of stakeholders, such as: software developers, system integrators, acquirers, owners, maintainers, contractors, quality assurance and control professionals, and users.

The full set of quality characteristics across these models might not be relevant to all stakeholders. Nonetheless, each category of stakeholder should be represented in reviewing and considering the relevance of the quality characteristics in each model before finalizing the set of quality characteristics that will be used, for example to establish product and system performance requirements or evaluation criteria.

3.2 Quality in use model

The quality in use model in 4.1 defines five characteristics related to outcomes of interaction with a system: effectiveness, efficiency, satisfaction, freedom from risk, and context coverage (Figure 3 and Table 3). Each characteristic can be assigned to different activities of stakeholders, for example, the interaction of an operator or the maintenance of a developer.

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

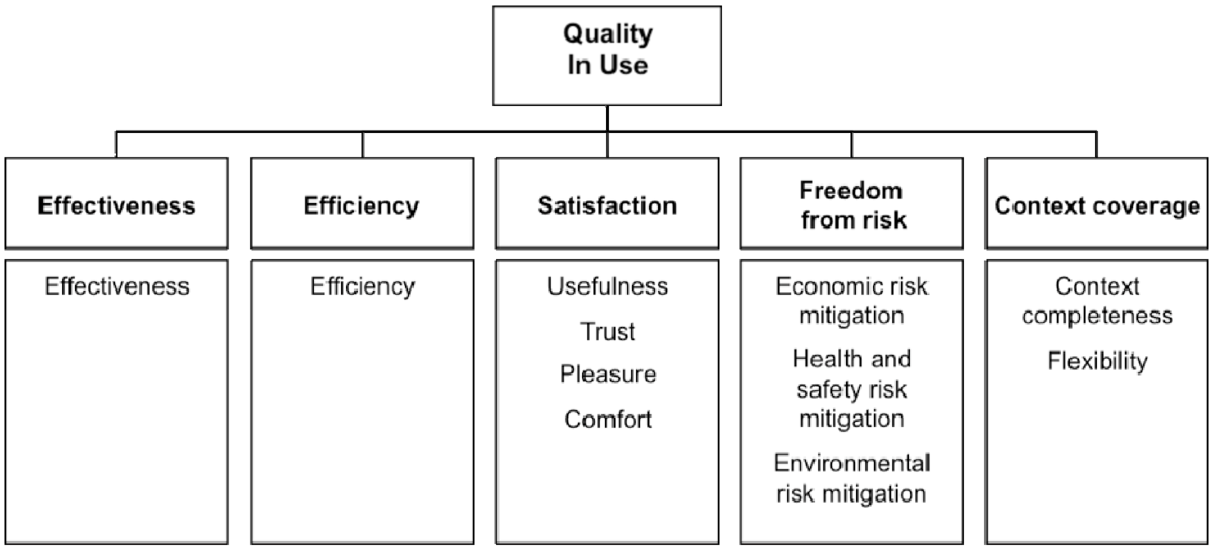


Figure 3 — Quality in use model

The quality in use of a system characterizes the impact that the product (system or software product) has on stakeholders. It is determined by the quality of the software, hardware and operating environment, and the characteristics of the users, tasks and social environment. All these factors contribute to the quality in use of the system.

Definitions and explanations of each quality characteristic for quality in use are given in 4.1.

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

3.3 Product quality model

The product quality model in 4.2 categorizes system/software product quality properties into eight characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. Each characteristic is composed of a set of related subcharacteristics (Figure 4 and Table 4).

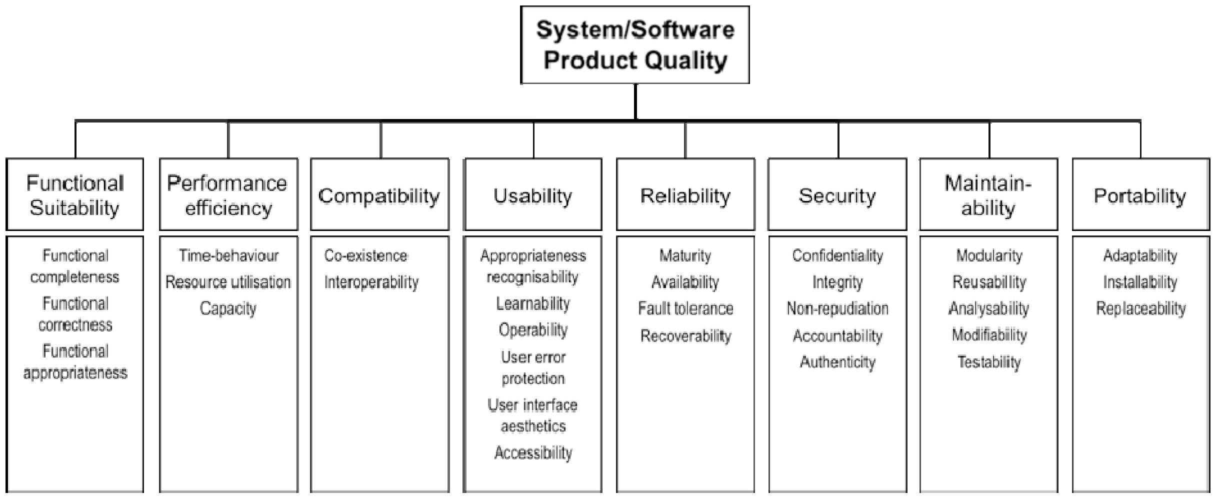


Figure 4 — Product quality model

NOTE The need for compliance with standards or regulations can be identified as part of requirements for a system, but these are outside the scope of the quality model.

The product quality model can be applied to just a software product, or to a computer system that includes software, as most of the subcharacteristics are relevant to both software and systems.

Definitions and explanations of each quality characteristic for product quality are given in 4.2.

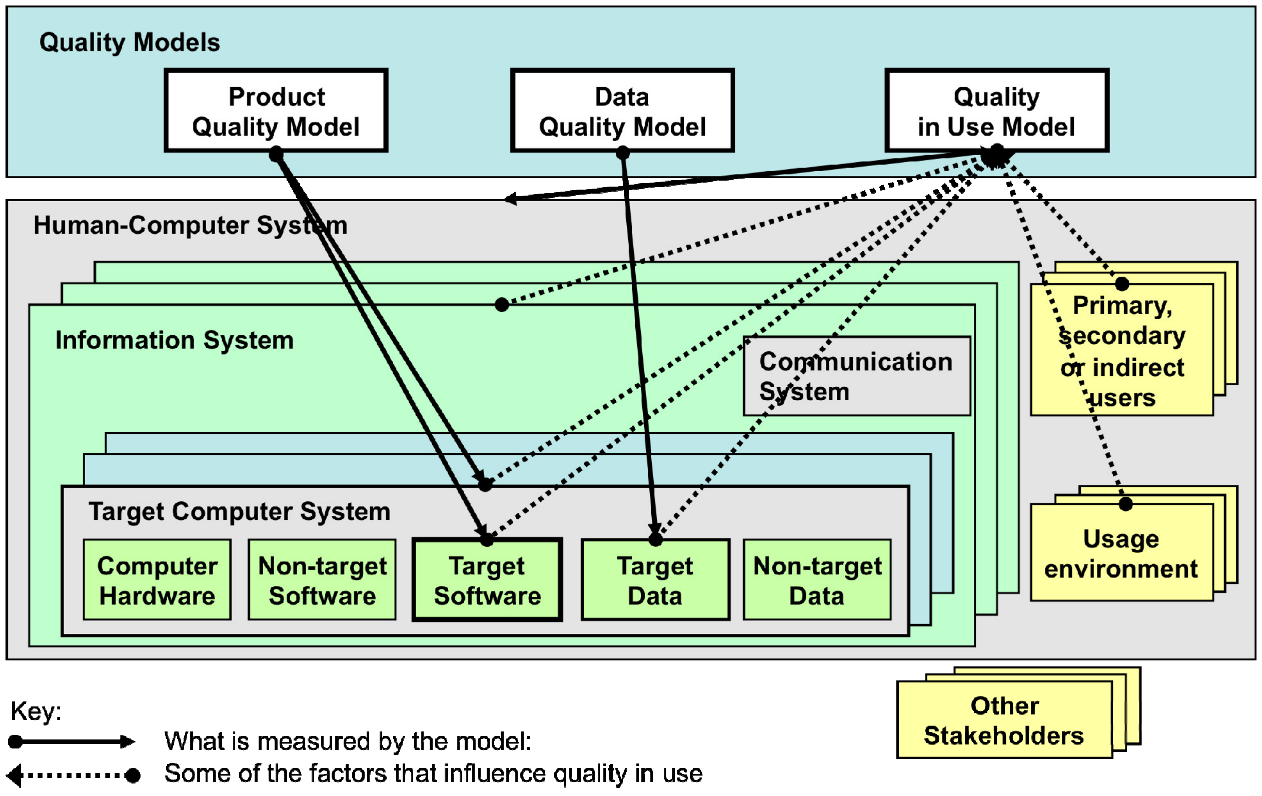
3.4 Targets of the quality models

Figure 5 illustrates the targets of the quality models and the related entities.

The product quality model focuses on the target computer system that includes the target software product, and the quality in use model focuses on the whole human-computer system that includes the target computer system and target software product. The target computer system also includes computer hardware, non-target software products, non-target data, and target data, which is the subject of the data quality model (see C.8). The target computer system is included in an information system that can also include one or more computer systems and communication systems, such as a local area network and the Internet. The information system is within a wider human-computer system (such as an enterprise system, embedded system or large-scale control system) and can include users and the technical and physical usage environment. Where the boundary of the system is judged to be, depends upon the scope of the requirements or evaluation, and upon who the users are.

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.

Other stakeholders, such as software developers, system integrators, acquirers, owners, maintainers, contractors, quality assurance and control professionals, will also be concerned with the quality.



NOTE This is conceptually the same as Figure 2 in ISO/IEC 25012 and Figure 5 in ISO/IEC 25030, but a different version that focuses on quality models.

Figure 5 — Targets of quality models

3.5 Using a quality model

The product quality and quality in use models are useful for specifying requirements, establishing measures, and performing quality evaluations (see Annex C). The defined quality characteristics can be used as a checklist for ensuring a comprehensive treatment of quality requirements, thus providing a basis for estimating the consequent effort and activities that will be needed during systems development. The characteristics in the quality in use model and product quality model are intended to be used as a set when specifying or evaluating computer system or software product quality.

It is not practically possible to specify or measure all subcharacteristics for all parts of a large computer system or software product. Similarly it is not usually practical to specify or measure quality in use for all possible user-task scenarios. The relative importance of quality characteristics will depend on the high-level goals and objectives for the project. Therefore the model should be tailored before use as part of the decomposition of requirements to identify those characteristics and subcharacteristics that are most important, and resources allocated between the different types of measure depending on the stakeholder goals and objectives for the product.

3.6 Quality from different stakeholder perspectives

The quality models provide a framework for collecting stakeholder needs. Stakeholders include the following types of user:

1. Primary user: person who interacts with the system to achieve the primary goals.
2. Secondary users who provide support, for example

a) content provider, system manager/administrator, security manager;

b) maintainer, analyzer, porter, installer.

3. Indirect user: person who receives output, but does not interact with the system.

Table 1 — Examples of user needs for quality in use and product quality

User needs	Primary user	Secondary users		Indirect user
		Content provider	Maintainer	
	Interacting	Interacting	Maintaining or porting	Using output
Effectiveness	How effective does the user need to be when using the system to perform their task?	How effective does the content provider need to be when updating the system?	How effective does the person maintaining or porting the system need to be?	How effective does the person using output from the system need to be?
Efficiency	How efficient does the user need to be when using the system to perform their task?	How efficient does the content provider need to be when updating the system?	How efficient does the person maintaining or porting the system need to be?	How efficient does the person using the output from the system need to be?
Satisfaction	How satisfied does the user need to be when using the system to perform their task?	How satisfied does the content provider need to be when updating the system?	How satisfied does the person maintaining or porting the system need to be?	How satisfied does the person using the output from the system need to be?
Freedom from risk	How risk free does using the system to perform their task need to be for the user?	How risk free does updating the content of the system need to be?	How risk free does making maintenance changes to the system or porting the system need to be?	How risk free does using the output from the system need to be?
Reliability	How reliable does the system need to be when the user uses it to perform their task?	How reliable does updating the system with new content need to be?	How reliable does maintaining or porting the system need to be?	How reliable does the output from the system need to be?
Security	How secure does the system need to be when the user uses it to perform their task?	How secure does the system need to be after the content provider updates it?	How secure does the system need to be after maintenance changes are made or when it is ported?	How secure does the output from the system need to be?
Context coverage	To what extent does the system need to be effective, efficient, risk free and satisfying in all the intended and potential contexts of use?	To what extent does providing content need to be effective, efficient, risk free and satisfying in all the intended and potential contexts of use?	To what extent does maintaining or porting the system need to be effective, efficient, risk free and satisfying in all the intended and potential contexts of use?	To what extent does using the output of the system need to be effective, efficient, risk free and satisfying in all the intended and potential contexts of use?
Learnability	To what extent does learning to use the system need to be effective, efficient, risk free and satisfying?	To what extent does learning to provide content need to be effective, efficient, risk free and satisfying?	To what extent does learning to maintain or port the system need to be effective, efficient, risk free and satisfying?	To what extent does learning to use the output from the system need to be effective, efficient, risk free and satisfying?
Accessibility	To what extent does the system need to be effective, efficient, risk free and satisfying to use for people with disabilities?	To what extent does providing content for the system need to be effective, efficient, risk free and satisfying for people with disabilities?	To what extent does maintaining or porting the system need to be effective, efficient, risk free and satisfying for people with disabilities?	To what extent does using the output of the system need be effective, efficient, risk free and satisfying for people with disabilities?

Each of these types of user has needs for quality in use and product quality in particular contexts of use, as illustrated for some examples of users and quality characteristics by the questions shown in Table 1.

NOTE The content provider will also have user needs for data quality.

The user needs in Table 1 provide examples of starting points for requirements, and can be used as a basis for measuring the impact of the quality of the system on use and maintenance.

Prior to software development or acquisition, quality requirements should be defined from the perspective of stakeholders. Analysis of the in use requirements will result in derived functional and quality requirements needed for a product to achieve the in use requirements.

EXAMPLE Overall needs for system reliability can lead to specific requirements for software product maturity, availability, fault tolerance and recoverability. Reliability can also have an impact on overall system effectiveness, efficiency, freedom from risk and satisfaction.

3.7 Relationship between the models

The properties of the software product and computer system determine the product quality in particular contexts of use (Table 2).

The functional suitability, performance efficiency, usability, reliability and security will have a significant influence on the quality in use for primary users. Performance efficiency, reliability and security can also be specific concerns of other stakeholders who specialize in these areas.

Compatibility, maintainability and portability will have a significant influence on quality in use for secondary users who maintain the system.

Table 2 — Influence of the quality characteristics

Software product properties	Computer system properties	Product quality characteristic	Influence on quality in use for primary users	Influence on quality in use for maintenance tasks	Information system quality concerns of other stakeholders
➥	➥	Functional suitability	❋		
➥	➥	Performance efficiency	❋		❋
➥	➥	Compatibility		❋	
➥	➥	Usability	❋		
➥	➥	Reliability	❋		❋
➥	➥	Security	❋		❋
➥	➥	Maintainability		❋	
➥	➥	Portability		❋	

Key: ➥ These properties influence product quality.
❋ Product quality influences quality in use for these stakeholders.

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

NOTE Definitions of quality characteristics and subcharacteristics are given in 4.1 and 4.2, general definitions in 4.3 and the essential definitions from ISO/IEC 25000 are reproduced in 4.4.

4.1 Quality in use model

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

The properties of quality in use are categorized into five characteristics: effectiveness, efficiency, satisfaction, freedom from risk and context coverage (Figure 3 and Table 3).

Table 3 — Quality in use characteristics and subcharacteristics

Effectiveness
Efficiency
Satisfaction
Usefulness
Trust
Pleasure
Comfort
Freedom from risk
Economic risk mitigation
Health and safety risk mitigation
Environmental risk mitigation
Context coverage
Context completeness
Flexibility

NOTE Usability (4.2.4) is defined as a subset of quality in use consisting of effectiveness, efficiency and satisfaction, for consistency with its established meaning.

4.1.1
effectiveness
accuracy and completeness with which users achieve specified goals

[ISO 9241-11]

4.1.2
efficiency
resources expended in relation to the accuracy and completeness with which users achieve goals

[ISO 9241-11]

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

4.1.3
satisfaction
degree to which user needs are satisfied when a product or system is used in a specified context of use

NOTE 1 For a user who does not directly interact with the product or system, only purpose accomplishment and trust are relevant.

NOTE 2 Satisfaction is the user's response to interaction with the product or system, and includes attitudes towards use of the product.

4.1.3.1
usefulness
degree to which a user is satisfied with their perceived achievement of pragmatic goals, including the results of use and the consequences of use

4.1.3.2
trust
degree to which a user or other stakeholder has confidence that a product or system will behave as intended

4.1.3.3
pleasure
degree to which a user obtains pleasure from fulfilling their personal needs

NOTE Personal needs can include needs to acquire new knowledge and skills, to communicate personal identity and to provoke pleasant memories.

4.1.3.4
comfort
degree to which the user is satisfied with physical comfort

4.1.4
freedom from risk
degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment

NOTE Risk is a function of the probability of occurrence of a given threat and the potential adverse consequences of that threat's occurrence.

4.1.4.1
economic risk mitigation
degree to which a product or system mitigates the potential risk to financial status, efficient operation, commercial property, reputation or other resources in the intended contexts of use

4.1.4.2
health and safety risk mitigation
degree to which a product or system mitigates the potential risk to people in the intended contexts of use

4.1.4.3
environmental risk mitigation
degree to which a product or system mitigates the potential risk to property or the environment in the intended contexts of use

4.1.5
context coverage
degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified

NOTE Context of use is relevant to both quality in use and some product quality (sub)characteristics (where it is referred to as "specified conditions").

4.1.5.1
context completeness
degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in all the specified contexts of use

NOTE Context completeness can be specified or measured either as the degree to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, freedom from risk and satisfaction in all the intended contexts of use, or by the presence of product properties that support use in all the intended contexts of use.

EXAMPLE The extent to which software is usable using a small screen, with low network bandwidth, by a non-expert user; and in a fault-tolerant mode (e.g. no network connectivity).

4.1.5.2
flexibility
degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in contexts beyond those initially specified in the requirements

NOTE 1 Flexibility can be achieved by adapting a product (see 4.2.8.1) for additional user groups, tasks and cultures.

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

NOTE 3 If a product is not designed for flexibility, it might not be safe to use the product in unintended contexts.

NOTE 4 Flexibility can be measured either as the extent to which a product can be used by additional types of users to achieve additional types of goals with effectiveness, efficiency, freedom from risk and satisfaction in additional types of contexts of use, or by a capability to be modified to support adaptation for new types of users, tasks and environments, and suitability for individualization as defined in ISO 9241-110.

4.2 Product quality model

The product quality model categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability). Each characteristic is composed of a set of related subcharacteristics (Figure 4 and Table 4).

(Sub)Characteristic
Functional suitability
Functional completeness
Functional correctness
Functional appropriateness
Performance efficiency
Time behaviour
Resource utilization
Capacity
Compatibility
Co-existence
Interoperability
Usability
Appropriateness recognizability
Learnability
Operability
User error protection
User interface aesthetics
Accessibility
Reliability
Maturity
Availability
Fault tolerance
Recoverability
Security
Confidentiality
Integrity
Non-repudiation
Accountability
Authenticity
Maintainability
Modularity
Reusability
Analysability
Modifiability
Testability
Portability
Adaptability
Installability
Replaceability

4.2.1
functional suitability
degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions

NOTE Functional suitability is only concerned with whether the functions meet stated and implied needs, not the functional specification (see C.6).

4.2.1.1
functional completeness
degree to which the set of functions covers all the specified tasks and user objectives

4.2.1.2
functional correctness
degree to which a product or system provides the correct results with the needed degree of precision

4.2.1.3
functional appropriateness
degree to which the functions facilitate the accomplishment of specified tasks and objectives

EXAMPLE A user is only presented with the necessary steps to complete a task, excluding any unnecessary steps.

NOTE Functional appropriateness corresponds to suitability for the task in ISO 9241-110.

4.2.2
performance efficiency
performance relative to the amount of resources used under stated conditions

NOTE Resources can include other software products, the software and hardware configuration of the system, and materials (e.g. print paper, storage media).

4.2.2.1
time behaviour
degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements

4.2.2.2
resource utilization
degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements

NOTE Human resources are included as part of **efficiency** (4.1.2).

4.2.2.3
capacity
degree to which the maximum limits of a product or system parameter meet requirements

NOTE Parameters can include the number of items that can be stored, the number of concurrent users, the communication bandwidth, throughput of transactions, and size of database.

4.2.3
compatibility
degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment

NOTE Adapted from ISO/IEC/IEEE 24765.

4.2.3.1
co-existence
degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product

4.2.3.2
interoperability
degree to which two or more systems, products or components can exchange information and use the information that has been exchanged

NOTE Based on ISO/IEC/IEEE 24765.

4.2.4
usability
degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use

NOTE 1 Adapted from ISO 9241-210.

NOTE 2 Usability can either be specified or measured as a product quality characteristic in terms of its subcharacteristics, or specified or measured directly by measures that are a subset of quality in use.

4.2.4.1
appropriateness recognizability
degree to which users can recognize whether a product or system is appropriate for their needs

cf. **functional appropriateness** (4.2.1.3).

NOTE 1 Appropriateness recognizability will depend on the ability to recognize the appropriateness of the product or system's functions from initial impressions of the product or system and/or any associated documentation.

NOTE 2 The information provided by the product or system can include demonstrations, tutorials, documentation or, for a web site, the information on the home page.

4.2.4.2
learnability
degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use

NOTE Can be specified or measured either as the extent to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use, or by product properties corresponding to suitability for learning as defined in ISO 9241-110.

4.2.4.3
operability
degree to which a product or system has attributes that make it easy to operate and control

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

4.2.4.4
user error protection
degree to which a system protects users against making errors

4.2.4.5
user interface aesthetics
degree to which a user interface enables pleasing and satisfying interaction for the user

NOTE This refers to properties of the product or system that increase the pleasure and satisfaction of the user, such as the use of colour and the nature of the graphical design.

4.2.4.6
accessibility
degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use

NOTE 1 The range of capabilities includes disabilities associated with age.

NOTE 2 Accessibility for people with disabilities can be specified or measured either as the extent to which a product or system can be used by users with specified disabilities to achieve specified goals with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use, or by the presence of product properties that support accessibility.

4.2.5
reliability
degree to which a system, product or component performs specified functions under specified conditions for a specified period of time

NOTE 1 Adapted from ISO/IEC/IEEE 24765.

NOTE 2 Wear does not occur in software. Limitations in reliability are due to faults in requirements, design and implementation, or due to contextual changes.

NOTE 3 Dependability characteristics include availability and its inherent or external influencing factors, such as availability, reliability (including fault tolerance and recoverability), security (including confidentiality and integrity), maintainability, durability, and maintenance support.

4.2.5.1
maturity
degree to which a system meets needs for reliability under normal operation

NOTE The concept of maturity can also be applied to other quality characteristics to indicate the degree to which they meet required needs under normal operation.

4.2.5.2
availability
degree to which a system, product or component is operational and accessible when required for use

[ISO/IEC/IEEE 24765]

NOTE Externally, availability can be assessed by the proportion of total time during which the system, product or component 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).

4.2.5.3
fault tolerance
degree to which a system, product or component operates as intended despite the presence of hardware or software faults

NOTE Adapted from ISO/IEC/IEEE 24765.

4.2.5.4
recoverability
degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system

NOTE Following a failure, a computer system will sometimes be down for a period of time, the length of which is determined by its recoverability.

4.2.6
security
degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization

NOTE 1 As well as data stored in or by a product or system, security also applies to data in transmission.

NOTE 2 Survivability (the degree to which a product or system continues to fulfil its mission by providing essential services in a timely manner in spite of the presence of attacks) is covered by **recoverability** (4.2.5.4).

NOTE 3 Immunity (the degree to which a product or system is resistant to attack) is covered by **integrity** (4.2.6.2).

NOTE 4 Security contributes to **trust** (4.1.3.2).

4.2.6.1
confidentiality
degree to which a product or system ensures that data are accessible only to those authorized to have access

4.2.6.2
integrity
degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data

[ISO/IEC/IEEE 24765]

4.2.6.3
non-repudiation
degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later

NOTE Adapted from ISO 7498-2:1989.

4.2.6.4
accountability
degree to which the actions of an entity can be traced uniquely to the entity

NOTE Adapted from ISO 7498-2:1989.

4.2.6.5
authenticity
degree to which the identity of a subject or resource can be proved to be the one claimed

NOTE Adapted from ISO/IEC 13335-1:2004.

4.2.7
maintainability
degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers

NOTE 1 Modifications can include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.

NOTE 2 Maintainability includes installation of updates and upgrades.

NOTE 3 Maintainability can be interpreted as either an inherent capability of the product or system to facilitate maintenance activities, or the quality in use experienced by the maintainers for the goal of maintaining the product or system.

4.2.7.1
modularity
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/IEEE 24765]

4.2.7.2
reusability
degree to which an asset can be used in more than one system, or in building other assets

NOTE Adapted from IEEE 1517-2004.

4.2.7.3
analysability
degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified

NOTE Implementation can include providing mechanisms for the product or system to analyse its own faults and provide reports prior to a failure or other event.

4.2.7.4
modifiability
degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality

NOTE 1 Implementation includes coding, designing, documenting and verifying changes.

NOTE 2 **Modularity** (4.2.7.1) and **analysability** (4.2.7.3) can influence modifiability.

NOTE 3 Modifiability is a combination of changeability and stability.

4.2.7.5
testability
degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met

NOTE Adapted from ISO/IEC/IEEE 24765.

4.2.8
portability
degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another

NOTE 1 Adapted from ISO/IEC/IEEE 24765.

NOTE 2 Portability can be interpreted as either an inherent capability of the product or system to facilitate porting activities, or the quality in use experienced for the goal of porting the product or system.

4.2.8.1
adaptability
degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments

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

NOTE 2 Adaptations include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.

NOTE 3 If the system is to be adapted by the end user, adaptability corresponds to suitability for individualization as defined in ISO 9241-110.

4.2.8.2
installability
degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment

NOTE If the product or system is to be installed by an end user, installability can affect the resulting functional appropriateness and operability.

4.2.8.3
replaceability
degree to which a product can be replaced by another specified software product for the same purpose in the same environment

EXAMPLE The replaceability of a new version of a software product is important to the user when upgrading.

NOTE 1 Replaceability can include attributes of both installability and adaptability. The concept has been introduced as a subcharacteristic of its own because of its importance.

NOTE 2 Replaceability will reduce lock-in risk: so that other software products can be used in place of the present one, for example by the use of standardized file formats.

4.3 General

4.3.1
asset
anything that has value to a person or organization

NOTE 1 Adapted from ISO/IEC 13335-1:2004.

NOTE 2 In this International Standard assets are work products such as requirements documents, source code modules, measurement definitions, etc.

4.3.2
benchmark
standard against which results can be measured or assessed

[ISO/IEC/IEEE 24765]

4.3.3
component
entity with discrete structure, such as an assembly or software module, within a system considered at a particular level of analysis

[ISO/IEC 15026:1998]

4.3.4
direct user
person who interacts with the product

NOTE 1 This includes primary and secondary users.

NOTE 2 This is the ISO 9241-11 definition of “user”.

4.3.5
external measure of software quality
measure of the degree to which a software product enables the behaviour of a system to satisfy stated and implied needs for the system including the software to be used under specified conditions

NOTE 1 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 measure of software quality related to the number of faults present in the computer system. The two measures are not necessarily identical since testing might not find all faults, and a fault can give rise to apparently different failures in different circumstances.

NOTE 2 Adapted from the definition of external software quality in ISO/IEC 25000:2005.

4.3.6
indirect user
person who receives output from a system, but does not interact with the system

4.3.7
internal measure of software quality
measure of the degree to which a set of static attributes of a software product satisfies stated and implied needs for the software product to be 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, simulation and/or automated tools.

EXAMPLE Complexity measures and the number, severity, and failure frequency of faults found in a walk through are internal software quality measures made on the product itself.

NOTE 3 Based on the ISO/IEC 25000:2005 definition of internal software quality.

4.3.8
quality in use
degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use

4.3.9
quality property
measurable component of quality

4.3.10
quality measure
measure that is defined as a measurement function of two or more values of quality measure elements

[ISO/IEC TR 25021]

4.3.11
quality measure element
measure defined in terms of an attribute and the measurement method for quantifying it, including optionally the transformation by a mathematical function

[ISO/IEC TR 25021]

4.3.12
risk
function of the probability of occurrence of a given threat and the potential adverse consequences of that threat's occurrence

[ISO/IEC 15026:1998]

4.3.13
software quality
degree to which a software product satisfies stated and implied needs when used under specified conditions

NOTE 1 This definition differs from the definition of quality in ISO 9000:2000 in that it refers to the satisfaction of stated and implied needs, while the ISO 9000 quality definition refers to the satisfaction of requirements.

NOTE 2 Adapted from ISO/IEC 25000:2005.

4.3.14
stakeholder
individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations

4.3.15
user
individual or group that interacts with a system or benefits from a system during its utilization

NOTE Primary and secondary users interact with a system, and primary and indirect users can benefit from a system (see 3.6).

4.4 Terms and definitions from ISO/IEC 25000

4.4.1
attribute
inherent property or characteristic of an entity that can be distinguished quantitatively or qualitatively by human or automated means

NOTE 1 Adapted from 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]

4.4.2
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]

4.4.3
end user
individual person who ultimately benefits from the outcomes of the system

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

[ISO/IEC 25000:2005]

4.4.4
implied needs
needs that might 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]

4.4.5
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 15939:2007]

4.4.6
measure, verb
make a measurement

[ISO/IEC 14598-1:1999]

4.4.7
measurement
set of operations having the object of determining a value of a measure

[ISO/IEC 15939:2007]

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

4.4.8
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]

4.4.9
software product
set of computer programs, procedures, and possibly associated documentation and data

[ISO/IEC 12207:1998]

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.

4.4.10
software quality requirement
requirement that a software quality attribute be present in software

4.4.11
software quality characteristic
category of software quality attributes that bears on software quality

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

[ISO/IEC 25000:2005]

4.4.12
system
combination of interacting elements organized to achieve one or more stated purposes

NOTE 1 A system can 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 can be substituted simply by a context-dependent synonym, e.g. aircraft, though this might then obscure a system principles perspective.

[ISO/IEC 15288:2008]

4.4.13
user
individual or group that benefits from a system during its utilization

[ISO/IEC 15939:2007]

4.4.14
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 can be necessary in earlier stages.

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

4.4.15
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 activity to determine conformity with the stated requirement for that activity.

Annex A
(informative)

Comparison with the quality model in ISO/IEC 9126-1

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

- Context coverage has been added as a quality in use characteristic, with subcharacteristics *context completeness* and *flexibility*.
- *Security* has been added as a characteristic, rather than a subcharacteristic of functionality, with subcharacteristics *confidentiality*, *integrity*, *non-repudiation*, *accountability* and *authenticity*.
- C*ompatibility* (including *interoperability a*nd *co-existence)* has been added as a characteristic.
- The following subcharacteristics have been added to existing product quality characteristics: *functional completeness, capacity, user error protection, accessibility, availability, modularity* and *reusability*.
- Compliance with standards or regulations that were subcharacteristics in ISO/IEC 9126-1 are now outside the scope of the quality model as they can be identified as part of requirements for a system.
- When appropriate, generic definitions have been adopted to extend the scope to computer systems, rather than using software-specific definitions.
- To comply with ISO/IEC Directives, definitions have been based on existing ISO/IEC when possible, and terms defined in this International Standard have been worded to represent the general meaning of the term.
- Several characteristics and subcharacteristics have been given more accurate names.

Table A.1 lists the differences between the characteristics and subcharacteristics in this International Standard, and ISO/IEC 9126-1:2001.

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

Clause	ISO/IEC 25010	ISO/IEC 9126-1	Notes
4.1	**Quality in use**	**Quality in use**	Quality in use is now a system quality
4.1.1	**Effectiveness**	Effectiveness	
4.1.2	**Efficiency**	Productivity	Name aligned with efficiency in ISO/IEC 25062 and ISO 9241-11
4.1.3	**Satisfaction**	Satisfaction	
4.1.3.1	Usefulness		No previous subcharacteristics
4.1.3.2	Trust		”
4.1.3.3	Pleasure		”
4.1.3.4	Comfort		”
4.1.4	**Freedom from risk**	Safety	
4.1.4.1	Economic risk mitigation		No previous subcharacteristics
4.1.4.2	Health and safety risk mitigation		”

Clause	ISO/IEC 25010	ISO/IEC 9126-1	Notes
4.1.4.3	Environmental risk mitigation		”
4.1.5	**Context coverage**		Implicit quality issue made explicit
4.1.5.1	Context completeness		New subcharacteristic (it is important that a product is usable in all required contexts of use)
4.1.5.2	Flexibility		New subcharacteristic (enables a product to be used in new contexts of use)
4.2	**Product quality**	**Internal and external quality**	Internal and external quality combined as product quality
4.2.1	**Functional suitability**	**Functionality**	New name is more accurate, and avoids confusion with other meanings of “functionality”
4.2.1.1	Functional completeness		Coverage of the stated needs
4.2.1.2	Functional correctness	Accuracy	More general than accuracy
4.2.1.3	Functional appropriateness	Suitability	Coverage of the implied needs
		Interoperability	Moved to Compatibility
		Security	Now a characteristic
4.2.2	**Performance efficiency**	**Efficiency**	Renamed to avoid conflicting with the definition of efficiency in ISO/IEC 25062
4.2.2.1	Time behaviour	Time behaviour	
4.2.2.2	Resource utilization	Resource utilization	
4.2.2.3	Capacity		New subcharacteristic (particularly relevant to computer systems)
4.2.3	**Compatibility**		New characteristic
4.2.3.2	Co-existence	Co-existence	Moved from Portability
4.2.3.3	Interoperability		Moved from Functionality
4.2.4	**Usability**		Implicit quality issue made explicit
4.2.4.1	Appropriateness recognizability	Understandability	New name is more accurate
4.2.4.2	Learnability	Learnability	
4.2.5.3	Operability	Operability	
4.2.4.4	User error protection		New subcharacteristic (particularly important to achieve freedom from risk)
4.2.4.5	User interface aesthetics	Attractiveness	New name is more accurate
4.2.4.6	Accessibility		New subcharacteristic
4.2.5	**Reliability**	**Reliability**	
4.2.5.1	Maturity	Maturity	
4.2.5.2	Availability		New subcharacteristic
4.2.5.3	Fault tolerance	Fault tolerance	
4.2.5.4	Recoverability	Recoverability	
4.2.6	**Security**	Security	No previous subcharacteristics
4.2.6.1	Confidentiality		”

Clause	ISO/IEC 25010	ISO/IEC 9126-1	Notes
4.2.6.2	Integrity		”
4.2.6.3	Non-repudiation		”
4.2.6.4	Accountability		”
4.2.6.5	Authenticity		”
4.2.7	**Maintainability**	**Maintainability**	
4.2.7.1	Modularity		New subcharacteristic
4.2.7.2	Reusability		New subcharacteristic
4.2.7.3	Analysability	Analysability	
4.2.7.4	Modifiability	Stability	More accurate name combining changeability and stability
4.2.7.5	Testability	Testability	
4.2.8	**Portability**	**Portability**	
4.2.8.1	Adaptability	Adaptability	
4.2.8.2	Installability	Installability	
		Co-existence	Moved to Compatibility
4.2.8.3	Replaceability	Replaceability	

Annex B
(informative)

Example of mapping to dependability

This annex gives an example of how an organization could map its own model of software quality to the ISO/IEC 25010 model.

Dependability is defined in IEC 60050-191 as the "ability to perform as and when required". One example of an alternative categorization of product qualities based on dependability [1] is:

- **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.
- **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 damage to the state of the system, including data, code, etc.
- **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.
- **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).

To conform with this International Standard, this definition of dependability could be mapped 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**
4.1.1	**Effectiveness**	*
4.1.2	**Efficiency**	*
4.1.3	**Satisfaction**	*
4.1.4	**Freedom from risk**	Safety
4.1.5	**Context coverage**	*
4.2.1	**Functional suitability**	*
4.2.2	**Performance efficiency**	*
4.2.3	**Compatibility**	*
4.2.4	**Usability**	*
4.2.5	**Reliability**	Reliability
4.2.5.2	Availability	Availability
4.2.6	**Security**	
4.2.6.1	Confidentiality	Confidentiality
4.2.6.2	Integrity	Integrity
4.2.7	**Maintainability**	Maintainability
4.2.8	**Portability**	*

If this definition of Dependability was being used as part of a wider assessment of software quality, it would also be necessary to consider effectiveness, efficiency, satisfaction, usability, context of use, functional suitability, performance efficiency, compatibility, and portability.

Annex C
(informative)

Using the quality model for measurement

C.1 General

Information in this annex may be moved to future revisions of other standards in the ISO/IEC 25000 series.

C.2 Software quality measurement model

Quality properties are inherent properties of the software that contribute to quality. Quality properties can be categorized into one or more (sub)-characteristics.

Quality properties are measured by applying a measurement method. A measurement method is a logical sequence of operations used to quantify properties 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.

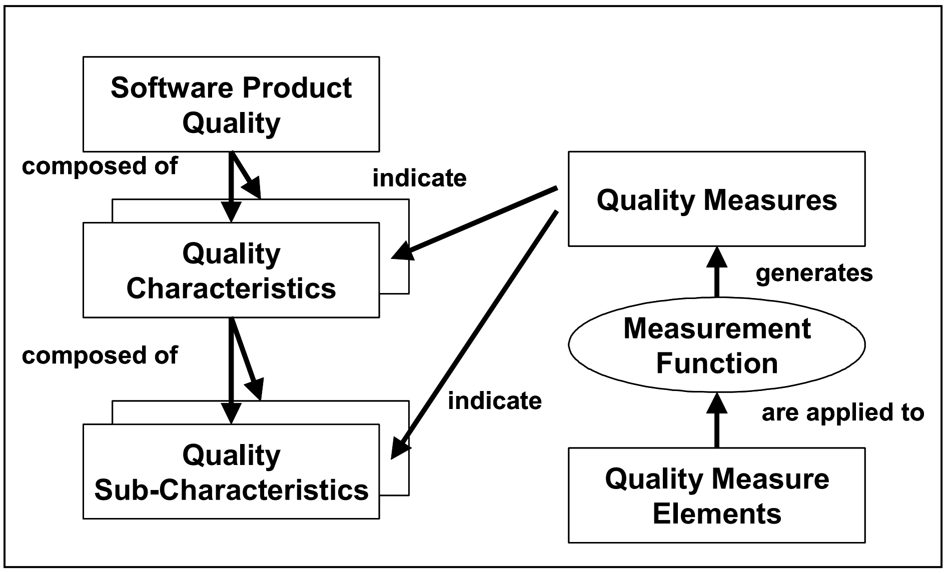


Figure C.1 — Software product quality measurement reference model

C.3 Approaches to quality

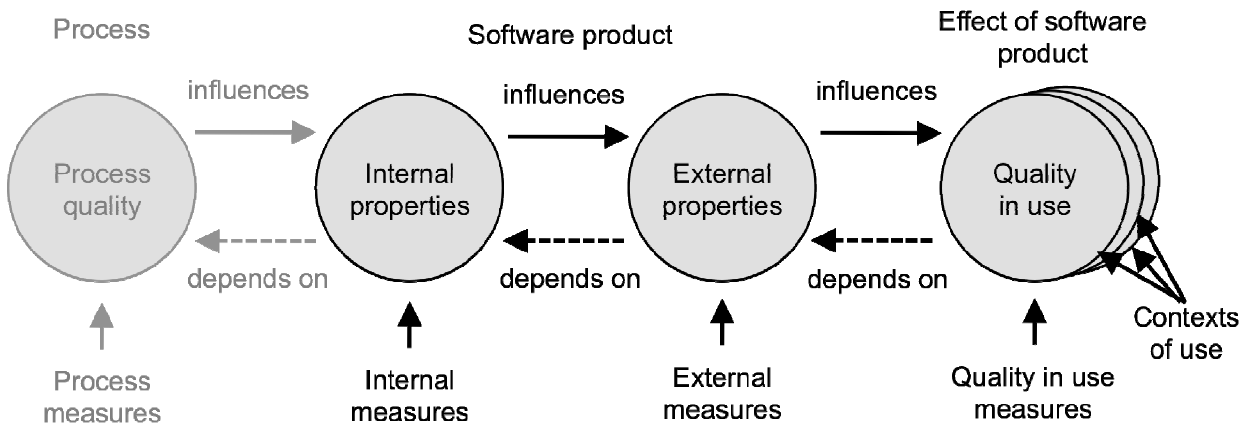


Figure C.2 — Quality in the lifecycle

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

Software product quality can be evaluated by measuring internal properties (typically static measures of intermediate products), or by measuring external properties (typically by measuring the behaviour of the code when executed), or by measuring quality in use properties (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 system 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 the system quality in use. Similarly, evaluating system quality in use can provide feedback to improve a product, and evaluating a product can provide feedback to improve a process.

Appropriate internal properties 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 Quality influences

Figure C.3 illustrates the relationships among target entities of the quality model. The software lifecycle processes (such as the quality requirements process, design process and testing process) influence the quality of the software product and the system. The quality of resources, such as human resources, software tools and techniques used for the process, influence the process quality, and consequently, influence the product quality.

Software product quality, as well as the quality of other components of a system, influences the quality of the system. The system quality has various influences (effects) depending on the contexts of use. The context of use can be defined by a set of a user, a task, and the environment. Some examples of context of use are shown in Table 1 (see 3.6).

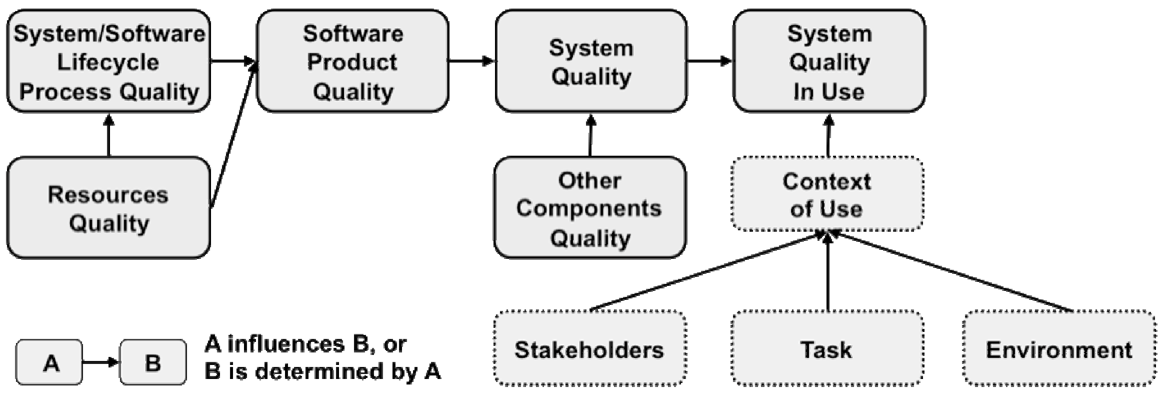


Figure C.3 — Target entities of quality model and their relationship

C.5 Quality life cycle model

ISO/IEC 25030 explains the quality requirement process using a diagram (Figure C.4). The “stakeholder needs” in this figure can be collected as needs for quality in use and product quality, and then transformed and specified as quality requirements (stakeholder requirements).

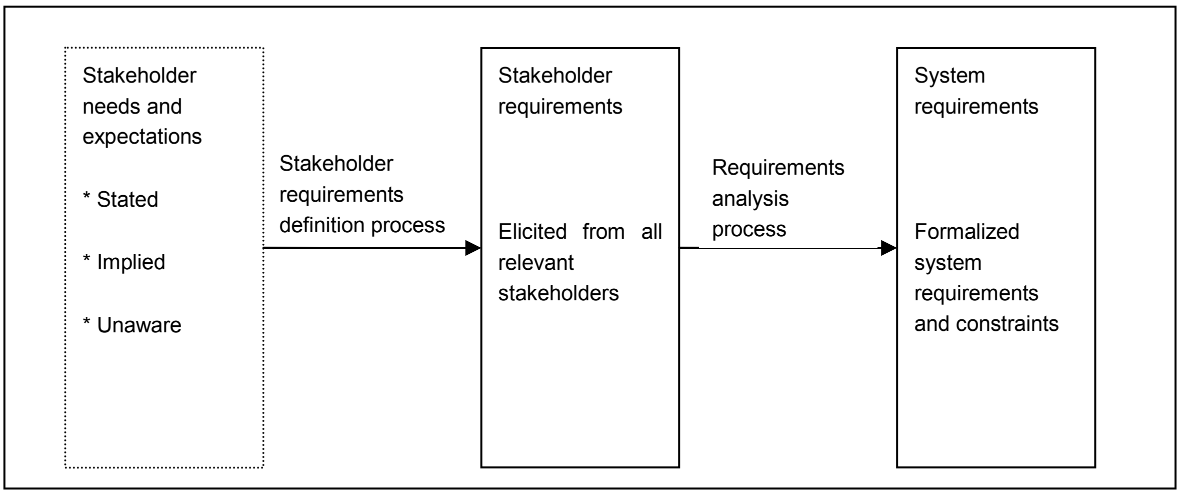

Stakeholder needs and expectations
* Stated
* Implied
* Unaware
Stakeholder requirements definition process
Stakeholder requirements
Elicited from all relevant stakeholders
Requirements analysis process
System requirements
Formalized system requirements and constraints

Figure C.4 — Stakeholder requirements definition and analysis

The quality life cycle model (Figure C.5) addresses quality in three principal phases of the software product life cycle:

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

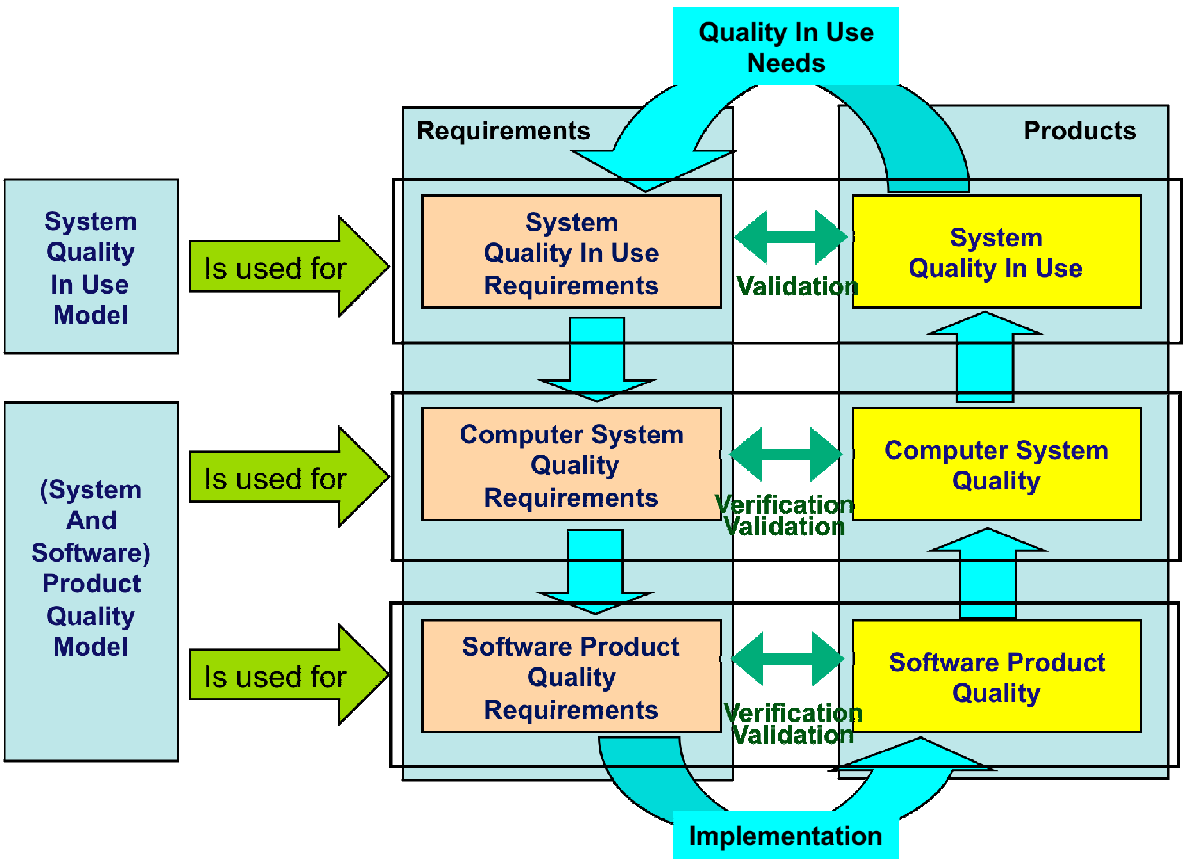


Figure C.5 — System/Software Quality Life Cycle Model

The system/software quality life cycle model also indicates that achieving acceptable levels of quality should be an integral part of the development process for each type of quality including: requirements, implementation and validation of the results.

Requirements for quality in use specify the required levels of quality from the users' point of view. These requirements are derived from the needs of users and other stakeholders (such as software developers, system integrators, acquirers, or owners). 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 criteria for quality in use measures that are used when a product is evaluated.

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

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

Requirements for external measures of computer system quality specify the required levels 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 measures of quality should be stated quantitatively in the quality requirements specification using criteria for external measures that are used when a product is evaluated.

NOTE 2 Requirements for external measures of quality contribute to identifying and to defining requirements for internal measures of software quality.

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

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

Requirements for internal software quality measures specify the level of required quality from the internal view of the product. They include requirements derived from external quality requirements. Requirements for internal software quality measures 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. Requirements internal software quality measures 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.

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

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.

During development, the models and associated measures can be used to manage design and implementation activities to ensure quality objectives are met. A key application of the quality models and associated measures is to gain early insight into software quality. This insight can be used to manage quality throughout the life cycle and to predict whether in use quality objectives are likely to be met.

NOTE 5 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, 2.1, Note 1).

C.6 Software properties

Some software properties are inherent in the software product; 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 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, a warranty 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. 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 C.6 illustrates this classification of software properties.

Software properties	Inherent properties	Domain-specific functional properties
		Quality properties (functional suitability, reliability, performance efficiency, operability, security, compatibility, maintainability, portability,)
	Assigned properties	Managerial properties like for example price, delivery date, product future, product supplier

Figure C.6 — Software properties

NOTE 2 Domain-specific functional properties are generally concerned with transformation of input data to output data. Additional functions may be needed to implement quality properties and constraints

C.7 Internal, external and quality in use measures

For each subcharacteristic, the capability of the software is determined by a set of static internal properties that can be measured. Examples of internal measures are given in ISO/IEC TR 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 TR 9126-2 (to be replaced by ISO/IEC 25023).

External measures of system/software quality provide a “black box” view of the system/software and address properties related to the execution of the software on computer hardware and an operating system. Internal measures of software quality provide a “white box” view of software and address static properties of the software product that typically are available to be evaluated during the development. The quality of software measured internally has an impact on the quality of system/software measured externally, which again has an impact on the quality in use of the system.

EXAMPLE Operability measured internally by the degree of conformance to menu interface design guidelines in ISO 9241-14 would contribute to operability measured externally by the extent to which users could successfully manipulate the menus, which would contribute to the effectiveness, efficiency and satisfaction when carrying out tasks (quality in use).

Internal measures based on inspecting static properties can be used to measure inherent properties of a software work product (see Table C.1). Static analysis methods include inspection and automated analysis tools. Work products include requirements and design documents, source code, and test procedures.

External measures of dynamic properties can be used to measure inherent properties of a computer system (the target computer system in Figure 5), and system-dependent properties of a software product.

Quality in use measures (derived from testing or observing the results of real or simulated use) measure intrinsic properties of a system that can include software, hardware, communications and users, and system dependent properties of a software-intensive computer system or of a software product. Quality in use measures relate to the impact of the system on stakeholders.

Table C.1 — Differences between internal quality measures, external quality measures and quality in use measures

Type of properties measured	**Software product properties**	**Computer system behaviour properties**	**Human-computer system impact properties**
Type of quality measure	Internal: inspection of static properties	External: test or modelling of dynamic properties	Quality in use: test or observation of results of real or simulated use
Type of properties of software product	Inherent	Computer system-dependent	Human-computer system-dependent
Type of properties of computer system		Inherent	Human-computer system-dependent
Type of properties of human-computer system			Inherent

Internal measures of software can be used at an early stage of the system/software development process to predict external measures of system/software quality. 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.

Examples of software product quality measures are given in ISO/IEC TR 9126-2 and ISO/IEC TR 9126-3 (to be replaced by ISO/IEC 25023 and ISO/IEC 25022 respectively).

C.8 Relationship between product quality and data quality

The data quality model in ISO/IEC 25012 is complementary to the product quality model.

- Inherent data quality (Table C.2) and internally measured software quality (Table C.1) both contribute to the overall quality of the computer system.
- Measures of system-dependent data quality and measures of external software quality assess similar aspects of the computer system. The difference is that measures of system-dependent data quality focus on the contribution that data makes to the quality of the computer system, while measures of external software quality focus on the contribution of the software. But what is actually measured in both cases is properties of the computer system.

Table C.2 — Relationship between intrinsic properties of data and computer system properties

Type of properties measured	**Intrinsic properties of data**	**Computer system properties**
Type of quality measure	Inherent data quality	System-dependent data quality
Type of properties of data	Inherent	Computer system-dependent
Type of properties of computer system		Inherent

Bibliography

[1] IEC 60050-191 Ed.2.0, *International Electrotechnical Vocabulary — Part 191: Dependability*[1)]

[2] IEEE 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*

[3] IEEE 1517-1999(R2004), *IEEE Standard for Information Technology — Software Life Cycle Processes — Reuse Processes*

[4] ISO/IEC 2382-1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*

[5] ISO/IEC 2382-14:1997, *Information technology — Vocabulary — Part 14: Reliability, maintainability and availability*

[6] ISO/IEC 2382-20:1990, *Information technology — Vocabulary — Part 20: System development*

[7] ISO 7498-2:1989, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture*

[8] ISO 9001:2000, *Quality management systems — Requirements*

[9] ISO/IEC 9126-1:2001, *Software engineering — Product quality — Part 1: Quality model*

[10] ISO/IEC TR 9126-2:2003, *Software engineering — Product quality — Part 2: External metrics*

[11] ISO/IEC TR 9126-3:2003, *Software engineering — Product quality — Part 3: Internal metrics*

[12] ISO/IEC TR 9126-4:2004, *Software engineering — Product quality — Part 4: Quality in use metrics*

[13] ISO 9241-11:1997, *Ergonomic requirements for office work with visual display terminals (VDTs) — Part 11: Guidance on usability*

[14] ISO 9241-14:1997, *Ergonomic requirements for office work with visual display terminals (VDTs) — Part 14: Menu dialogues*

[15] ISO 9241-110:2006, *Ergonomics of human-system interaction — Part 110: Dialogue principles*

[16] ISO/IEC 12207:2008, *Systems and software engineering — Software life cycle processes*

[17] 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*[2)]

[18] ISO 13407:1999, *Human-centred design processes for interactive systems*[2)]

[19] ISO/IEC 14598-2:2000, *Software engineering — Product evaluation — Part 2: Planning and management*

[20] ISO/IEC 14598-3:2000, *Software engineering — Product evaluation — Part 3: Process for developers*

[21] ISO/IEC 14598-4:1999, *Software engineering — Product evaluation — Part 4: Process for acquirers*

1) To be published.

2) Withdrawn.

[22] ISO/IEC 14598-5:1998, *Information technology — Software product evaluation — Part 5: Process for evaluators*

[23] ISO/IEC 14598-6:2001, *Software engineering — Product evaluation — Part 6: Documentation of evaluation modules*

[24] ISO/IEC 15026:1998, *Information technology — System and software integrity levels*

[25] ISO/IEC 15504 (parts 1 to 5), *Information technology — Process assessment*

[26] ISO/IEC 15288:2008, *Systems and software engineering — System life cycle processes*

[27] ISO/IEC/IEEE 24765:—[3], *Systems and software engineering — Vocabulary*

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

[29] ISO/IEC 25012:2008, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model*

[30] ISO/IEC 25020:2007, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Measurement reference model and guide*

[31] ISO/IEC 25030:2007, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements*

[32] ISO/IEC 25040:—[3], *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process*

[33] ISO/IEC TR 25021:2007, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality measure elements*

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

3) To be published.

ICS 35.080

Price based on 34 pages