INTERNATIONAL STANDARD

**ISO/IEC 25030**

First edition
2007-06-01

# Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements

*Ingénierie du logiciel — Exigences de qualité et évaluation du produit logiciel (SQuaRE) — Exigences de qualité*



Reference number
ISO/IEC 25030:2007(E)



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

Published in Switzerland

 

# Contents

Page

# Foreword

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

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

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

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

ISO/IEC 25030 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

ISO/IEC 25030 is part of the ISO/IEC 25000 SQuaRE series of standards. The series consists of the following divisions under the general title *Software product Quality Requirements and Evaluation* (SQuaRE):

— ISO/IEC 2500n, *Quality Management Division*,

— ISO/IEC 2501n, *Quality Model Division*,

— ISO/IEC 2502n, *Quality Measurement Division*,

— ISO/IEC 2503n, *Quality Requirements Division*, and

— ISO/IEC 2504n, *Quality Evaluation Division*.

ISO/IEC 25050 to ISO/IEC 25099 are reserved to be used for SQuaRE extension International Standards, Technical Specifications, Publicly Available Specifications (PAS) and/or Technical Reports: ISO/IEC 25051 and ISO/IEC 25062 are already published.

# Introduction

It is important to identify and specify software quality requirements as part of specifying the requirements for a software product. Software is usually part of a larger system. System requirements and software requirements are closely related and software requirements can therefore not be considered in isolation. This International Standard focuses on software quality requirements, but takes a system perspective. Software quality requirements can be categorized by use of a quality model, for example the quality model defined in ISO/IEC 9126-1 [ISO/IEC 25010]. Measures of attributes of these characteristics and their subcharacteristics can be used to specify software quality requirements and evaluate the quality of a software product.

Software quality requirements address important issues of quality for software products. Software product quality requirements are needed for:

— specification (including contractual agreement and call for tender);

— planning (including feasibility analysis and translation of external software quality requirements into internal software quality requirements);

— development (including early identification of potential quality problems during development); and

— evaluation (including objective assessment and certification of software product quality).

If software quality requirements are not stated clearly, they may be viewed, interpreted, implemented and evaluated differently by different people. This may result in software which is inconsistent with user expectations and of poor quality; users, clients and developers who are unsatisfied; and time and cost overruns to rework software.

This International Standard aims to improve the quality of software quality requirements. It does this by providing requirements and recommendations for quality requirements, and guidance for the processes used to define and analyse quality requirements.

Application of this International Standard should help ensure that software quality requirements are:

— in accordance with stakeholder needs;

— stated clearly and precisely;

— correct, complete, and consistent; and

— verifiable and measurable.

This International Standard is intended to be used in conjunction with the other parts of the SQuaRE series of Standards (ISO/IEC 25000 – ISO/IEC 25049), and with ISO/IEC 14598 and ISO/IEC 9126, until superseded by the ISO/IEC 25000 series.

This International Standard complies with the technical processes defined in ISO/IEC 15288:2002 related to quality requirements definition and analysis.

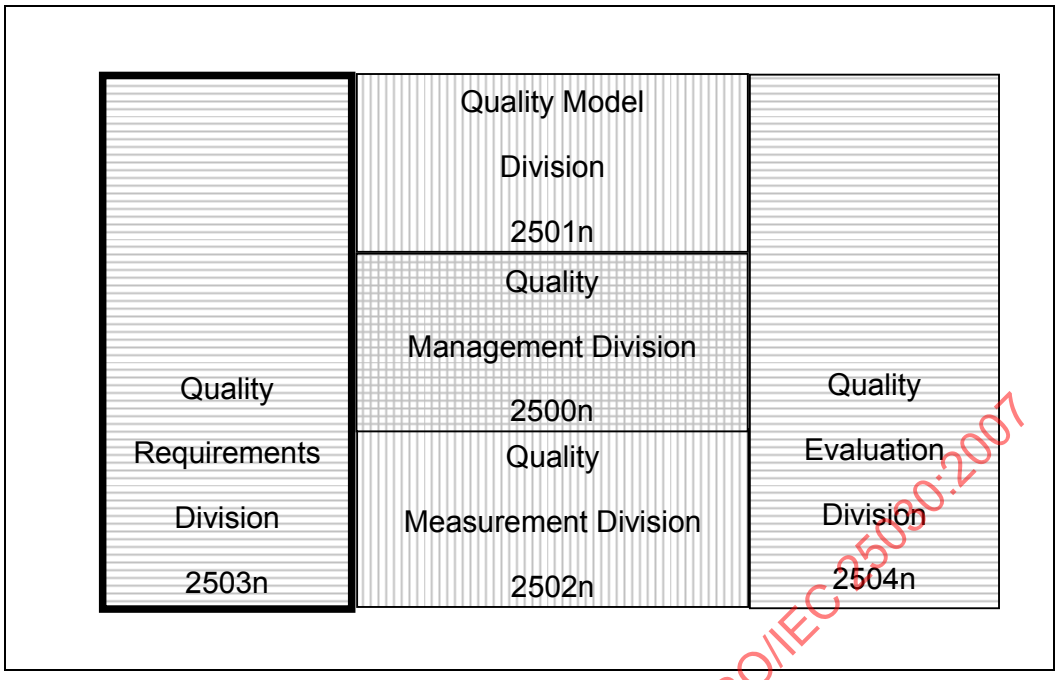


**Figure 1 — Organisation of the ISO/IEC 25000 SQuaRE series of International Standards**



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

The Divisions within SQuaRE model are:

— **ISO/IEC 2500n, Quality Management Division**. The International Standards that form this division define all common models, terms and definitions referred to further by all other International Standards from the SQuaRE series. Referring paths (guidance through SQuaRE documents) and high level practical suggestions in applying proper International Standards to specific application cases offer help to all types of users. The division also provides requirements and guidance for a supporting function which is responsible for the management of software product requirements specification and evaluation.

— **ISO/IEC 2501n, Quality Model Division**. The International Standard that forms this division presents a detailed quality model including characteristics for internal, external and quality in use. Furthermore, the internal and external software quality characteristics are decomposed into subcharacteristics. Practical guidance on the use of the quality model 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. Presented measures apply to internal software quality, external software quality and quality in use. Measurement primitives forming foundations for the latter measures are defined and presented.

— **ISO/IEC 2503n, Quality Requirements Division**. The International Standard that forms this division helps specify quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a software product to be developed or as input for an evaluation process. The requirements definition process is mapped to technical processes defined in ISO/IEC 15288.

— **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 presented.

ISO/IEC 25050 to ISO/IEC 25099 are reserved to be used for SQuaRE extension International Standards and/or Technical Reports.

# Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements

## 1 Scope

This International Standard provides requirements and recommendations for the specification of software product quality requirements.

This International Standard applies to organisations in their role as both acquirers and suppliers.

The quality model in ISO/IEC 9126-1 [ISO/IEC 25010] is used to categorize software quality requirements and to provide a basis for quantifying the quality requirements in terms of software quality measures.

This International Standard complies with the technical processes defined in ISO/IEC 15288:2002, which are relevant for identification of stakeholder product quality needs and for analysis of software product quality requirements.

This International Standard does not cover specification of other requirements (such as functional requirements, process requirements, business requirements, etc.).

This International Standard does not prescribe specific software quality measures nor does it prescribe any specific development process.



## 2 Conformance

Software quality requirements conform to this International Standard if they fulfil the requirements specified in Clause 6.

## 3 Normative references

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

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

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

1) ISO/IEC 9126-1:2001 will be cancelled and replaced by ISO/IEC 25010.

## 4 Terms and definitions

For the purposes of this International Standard, the terms and definitions given in ISO/IEC 25000 (repeated in Annex A for convenience) apply. There are no definitions specific to this International Standard.

## 5 Fundamental concepts for quality requirements

Clause 5 describes concepts related to software quality requirements that are used in this International Standard. This clause does not include requirements.

### 5.1 Software and systems

Software is the main focus of this International Standard. However, software usually appears as part of a larger system. Therefore it can be useful to take a system view. A system is defined as a combination of interacting elements organised to achieve one or more stated purposes. This definition allows a high degree of freedom to decide, what constitute a system and what the elements of the system are. The boundaries of a system will depend on the point of view.

Note 1 The boundary of a system depends on the point of view as illustrated by the following three examples. One example is the control system of an aircraft engine, the second example is the complete engine of an aircraft, and a third example is the complete aircraft. An aircraft can be considered as a combination of elements (the engines, the wings, etc.). These elements can also be considered systems on their own.

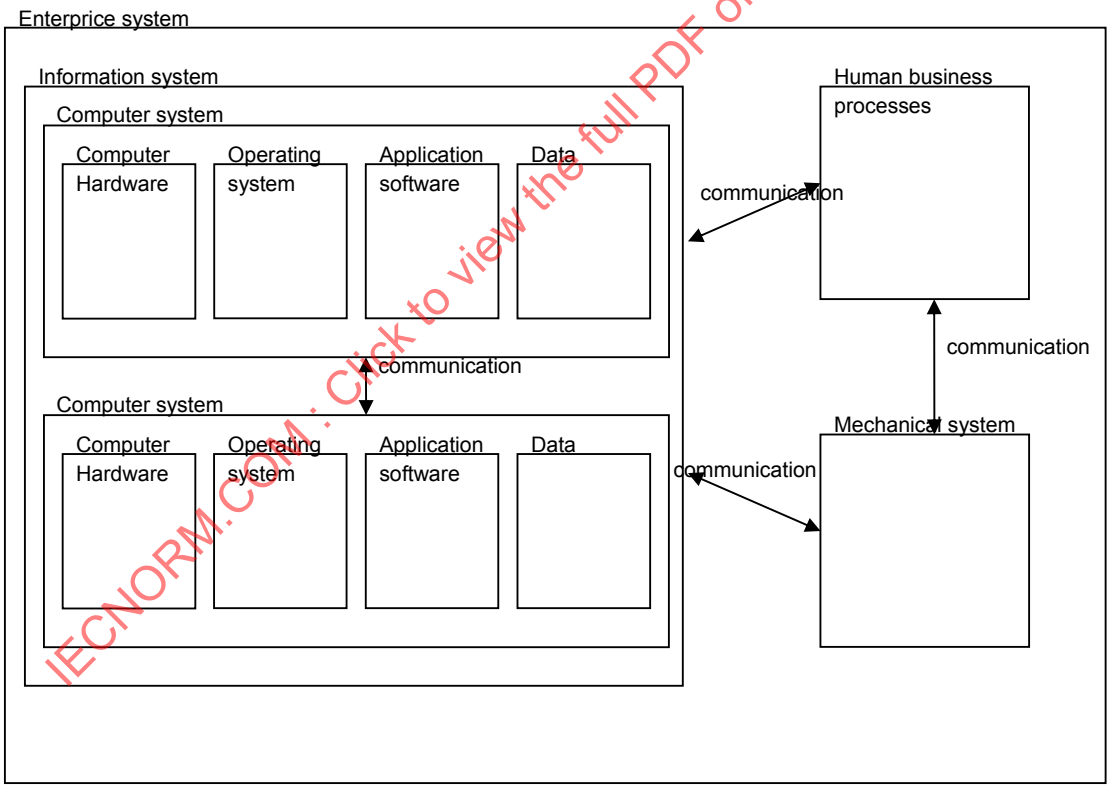


**Figure 2 – Example of a system model**

Figure 2 is an example of a system model showing hierarchies of systems including an information system, a mechanical system, human business processes and communication among them.

There may be several different appropriate ways of defining the elements of a system. Software may be considered one of the elements of a system. A computer system is an example of a system, which includes software. The elements of a computer system include the computer hardware, operating system and data necessary to apply the software. A computer system represents an applicable model when discussing single user software like a word processor. Client–server software or internet applications need a more complex system model like an information system which includes more communicating computer systems. E-commerce applications often include human business processes as well. Many devices include both computer systems and mechanical systems such as an antilock braking system (ABS) of a car. The luggage handling at an airport includes both computer systems, mechanical systems (such as conveyer belts), and human business processes. This example illustrates that humans can be part of a system.

## 5.2 Stakeholders and stakeholder requirements

Systems have a variety of stakeholders who have an interest in the system throughout its life cycle. The stakeholders of a system include all persons (for example end users), organisations (for example end user organisations or development organisations) and bodies (for example statutory and regulatory authorities or the general public) having a legitimate interest in the system. Stakeholders have different needs and expectations to the system. Their needs and expectations may change throughout the systems life cycle.

Stakeholder needs can be explicitly stated or only implied. Implied needs are often implied by the context where the software product is to be used and represent expectations based on similar software products or existing work routines, normal working procedures and operations of business, laws and regulations, etc. Stakeholders are sometimes not aware of all their needs. In many situations stakeholder needs only become evident when the software product and related business processes or tasks can be tried out. Scenarios, use cases, and prototypes are examples of methods that can be used to identify implied needs and other needs that stakeholders are not aware of (unaware needs) at an early stage in a development project. In some cases the real needs of some stakeholders are different from what they express.

The stakeholders' needs and expectations are identified through a requirements elicitation and definition process as illustrated in figure 3. The process takes all stakeholders' needs, wants, desires, and expectations into consideration. This includes the needs and requirements imposed by society, the constraints imposed by the acquirer, and the needs of the end users.

Different categories of stakeholders often have different needs and expectations. In some situations stakeholders have conflicting needs. Conflicts between stakeholder requirements could for example be between different end user perspectives, or between acquirer needs and available skills, experiences or resources in the developing organisation.

This International Standard does not prescribe any specific software life cycle processes, but it complies with the relevant system life cycle processes in ISO/IEC 15288:2002. The International Standard ISO/IEC 15288:2002 defines two technical processes aiming at defining stakeholder requirements and analysing system requirements:

a) Stakeholder Requirements Definition Process (ISO/IEC 15288:2002 clause 5.5.2)

b) Requirements Analysis Process (ISO/IEC 15288:2002 clause 5.5.3)

Note 1 ISO/IEC 15288:2002 is used in preference to ISO/IEC 12207:1995 as ISO/IEC 12207:1995 is less explicit about quality requirements.

Although these two processes form a logical sequence, they are often used iteratively. In Annex B of this International Standard the activities of the relevant processes are provided.

The two processes may be tailored as specified in ISO/IEC 15288:2002 Annex A to satisfy particular circumstances or factors that:

a) surround an organisation that is employing the International Standard in an agreement;

b) influence a project that is required to meet an agreement in which the International Standard is referenced;

c) reflect the needs of an organisation in order to supply products or services.

The two processes defined in ISO/IEC 15288:2002 are concerned with system requirements activities whereas this International Standard is concerned with software quality requirements. Although software quality requirements should be considered in a system perspective there may be activities prescribed in ISO/IEC 15288:2002 which are not or only marginally relevant from the point of view of software quality requirements.

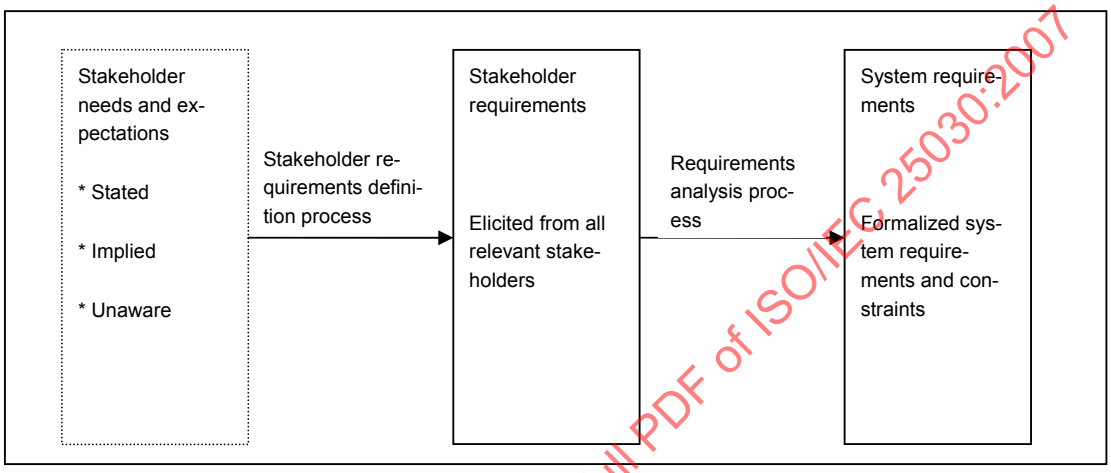


**Figure 3 – Stakeholder requirements definition and analysis**

The result of the definition process is called stakeholders requirements. The result of the analysis process is called system requirements.

## 5.3 Stakeholder requirements and system requirements

An analysis process transforms stakeholder requirements into a technical view of system requirements that can be used to realise the desired system, see figure 3. The technical view of requirements is called system requirements. System requirements are verifiable and will state which characteristics the system is to possess in order to satisfy stakeholder requirements.

A system will often be composed of different elements, each with specific characteristics and serving different purposes in the whole system. In order to be operational, system requirements have to be formulated as requirements for the different system elements. As different elements interact to offer the system capabilities, requirements for different system elements cannot be seen in isolation, but only in a broader view including requirements for other system elements.

Stakeholder requirements may imply requirements for, for example, software, but it is not always the case that a stakeholder requirement implies a software requirement. Stakeholder requirements can be implemented in alternative ways, for example, either in hardware or in software or as a business process (for example as a manual process). Such implementation decisions are part of a high level design process. Figure 4 shows the requirements hierarchy based on system design decisions applying the system model shown in figure 2.

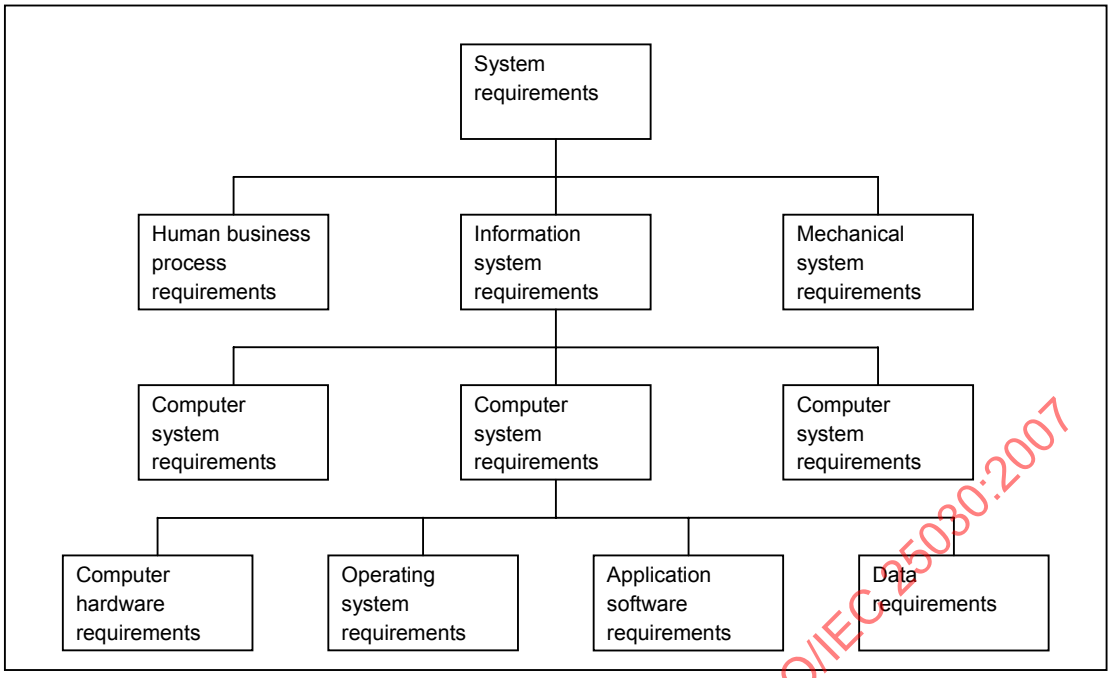


**Figure 4 – System and software requirements hierarchy**



## 5.4 Software quality model

The quality of a system is the result of the quality of the system elements and their interaction. This International Standard focuses on the quality of the software as part of a system.

Software quality is the capability of the software product to satisfy stated and implied needs when used under specified conditions. The software product quality model provided in ISO/IEC 9126-1 [ISO/IEC 25010] defines six quality characteristics:

- Functionality: The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.
- Reliability: The capability of the software product to maintain a specified level of performance when used under specified conditions
- Usability: The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.
- Efficiency: The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.
- 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.
- Portability: The capability of the software product to be transferred from one environment to another.

The standard defines an additional quality characteristic aiming at the system level:

- 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.

The quality characteristics have defined subcharacteristics and the standard allows for user defined sub-subcharacteristics in a hierarchical structure. The defined quality characteristics cover all quality aspects of interest for most software products and as such can be used as a checklist for ensuring a complete coverage of quality.

The quality model defines three different views of quality:

- Software quality in use
- External software quality
- Internal software quality

The software quality in use view is related to application of the software in its operational environment, for carrying out specific tasks by specific users. External software quality provides a 'black box' view of the software and addresses properties related to the execution of the software on computer hardware and applying an operating system. Internal software quality provides a 'white box' view of software and addresses properties of the software product that typically are available during the development. Internal software quality is mainly related to static properties of the software. Internal software quality has an impact on external software quality, which again has an impact on quality in use. Figure 5 shows the interaction between the different quality models and systems. The ISO/IEC 25000 SQuaRE series of International Standards covers the software quality models and the data quality model, but not the other quality models shown in the figure.



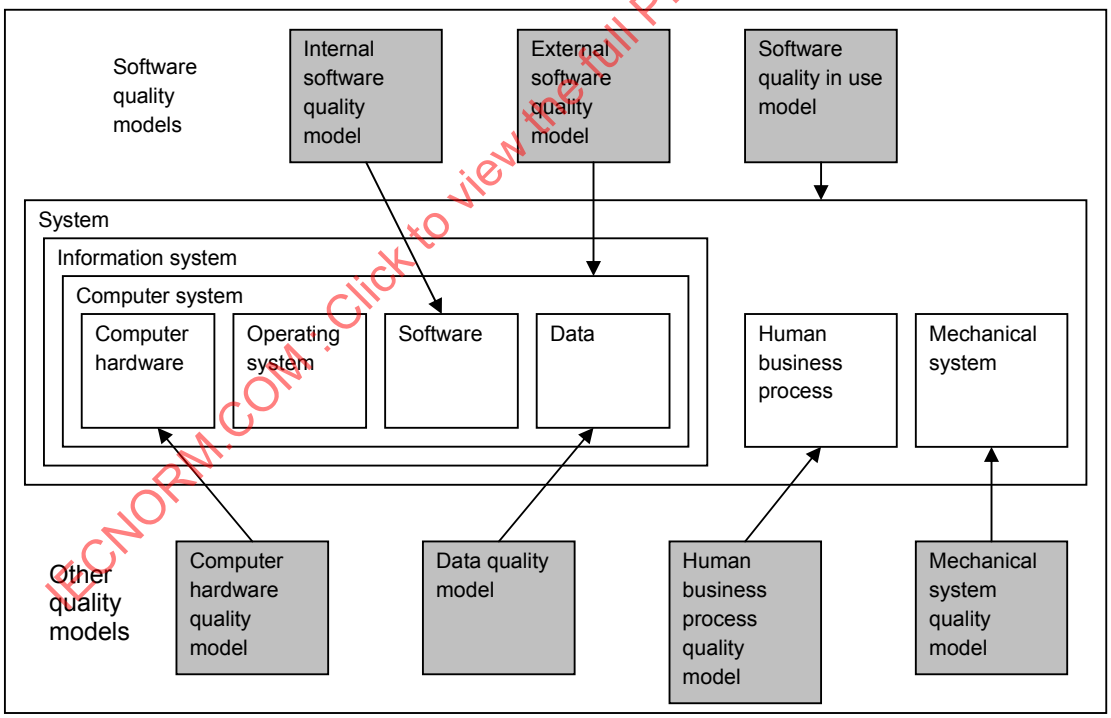


**Figure 5 – Example of system model and quality models**

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

## 5.5 Software properties

Some software properties are inherent in the software product; some are assigned to the software product. The capabilities of a software product are determined by its inherent properties.

Note 1 “Inherent”, as opposed to “assigned”, means existing in something, especially as a permanent characteristic or feature.

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

Inherent properties can be classified as either functional properties or quality properties. Functional properties determine what the software is able to do. Quality properties determine how well the software performs. In other words, the quality properties show the degree to which the software is able to provide and maintain its specified services. Quality is inherent to a software product. 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 6 illustrates this classification of software properties.

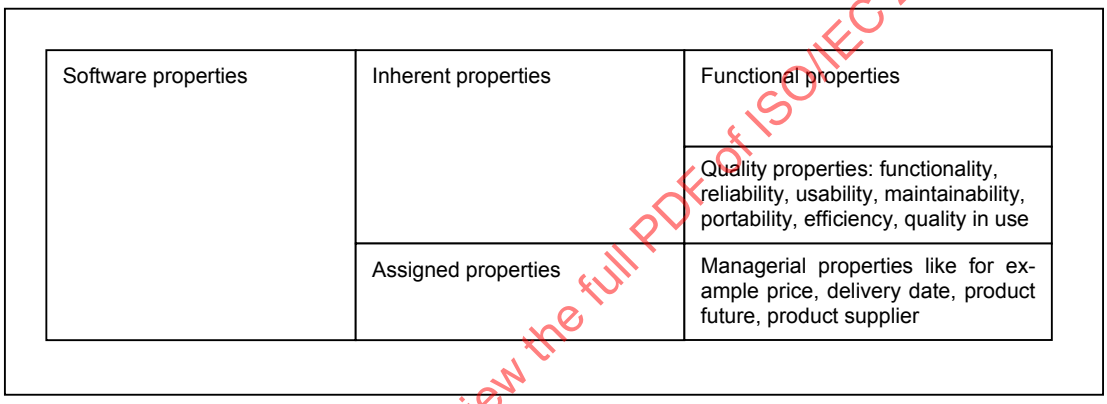


**Figure 6 – Software properties**



## 5.6 Software quality measurement model

Inherent software properties, that can be distinguished quantitatively or qualitatively, are called attributes. Quality attributes are categorised into one or more quality characteristics and subcharacteristics.

Quality attributes are measured by applying a measurement method. A measurement method is a logical sequence of operations used to quantify an attribute with respect to a specified scale. The result of applying a measurement method is called a base measure. 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 7 copied 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.

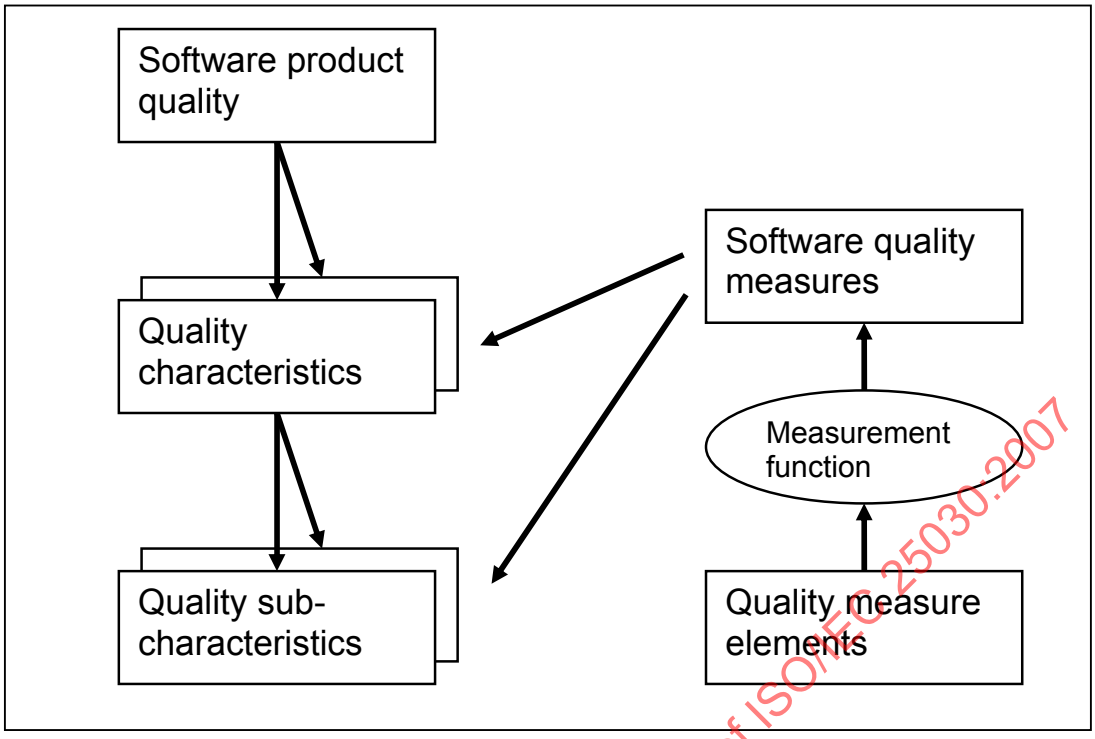


**Figure 7 – Software product quality measurement reference model**



### 5.7 Software quality requirements

A measurement function provides an interpretation of a software quality property, i.e. it assigns a value to a property. A target value of the software quality measure represents a software quality requirement, i.e. what is the required value of the property. Similarly, the actual value of the quality measurement represents the observed quality of the software.

Software quality requirements as well as all other requirements cannot be seen in isolation, but must be viewed in a broader context. Software quality requirements have a particularly close relation to functional requirements. Although functional requirements are not the topic of this International Standard, they play an important role in specifying software quality requirements.

Note 1 Functionality is one of the six characteristics for internal and external quality in ISO/IEC 9126-1 [ISO/IEC 25010]. Functionality requirements should not be confused with functional requirements. Functionality is the capability of the software to provide functions which meet its functional requirements. Functionality requirements are refined into requirements for the software product to be suitable, accurate, interoperable, secure and compliant with relevant functional standards and regulations.

In some situations it is meaningful to specify a software quality requirement for the complete software product, whereas in other situations a software quality requirement only applies to a portion of the software product. For example, some functions are only relevant for specific users and have specific software quality requirements, which are different from quality requirements for other functions intended for other purposes and other users. It is therefore important to specify which portion of a software product is relevant for a software quality requirement. In other words, a software quality requirement applies to a portion of the software product (a set of functions). For example, some functions may be intended for general end users and may hence require low error tolerance, whereas another group of functions may be intended for specialists and thus permit greater error tolerance. In both cases the error tolerance mechanism and the degree of error tolerance required should be rigorously specified.

Software quality requirements may result in additional functional requirements that support the software quality requirements. For example, a usability requirement can result in an additional function providing online help for the user.

Figure 8 shows how software quality requirements are derived as part of the requirements processes defined in ISO/IEC 15288. The definition process focuses on stakeholder requirements for the system. The analysis process assumes some architectural decisions, which makes it possible to identify requirements relevant for the software elements of the system. When focusing on stakeholder quality needs the quality model defined in ISO/IEC 25010 is useful for defining stakeholder quality requirements. Similarly the (software quality) measures defined in ISO/IEC TR 9126-2, ISO/IEC TR 9126-3 and ISO/IEC TR 9126-4 [ISO/IEC 2502n] are useful for formalising the stakeholder software quality requirements.

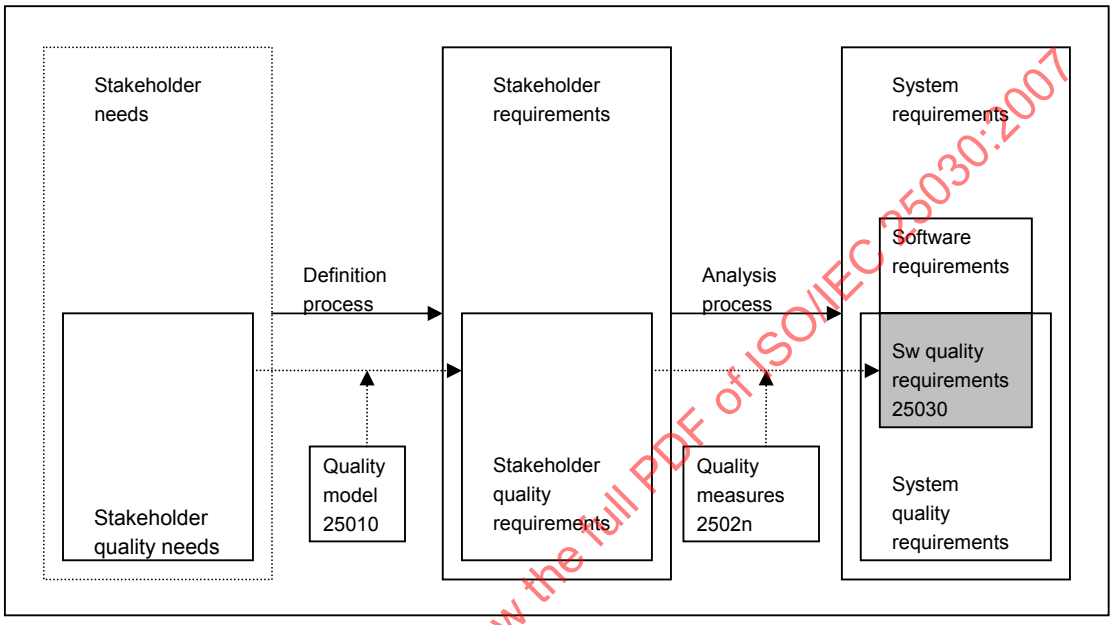


**Figure 8 – Software quality requirements definition and analysis**

Note 2 In figure 8 it is emphasised that stakeholder requirements and stakeholder quality requirements addresses the whole system in question and not only the software.

## 5.8 System requirements categorisation

A system consists of a number of interacting elements. They can be defined and categorised in different ways. Figure 2 provides one possible categorisation. Figure 9 provides a similar categorisation of system requirements. System requirements can for example include requirements for software, computer hardware, data, mechanical system, human business organisation, etc., and may come from a variety of stakeholders including end users, organisations, and official bodies. As this International Standard mainly focuses on software, figure 9 emphasises software requirements.

Software requirements address either the software product or the software development process.

Software product requirements include functional requirements, quality requirements and managerial requirements. Functional requirements include the application domain specific requirements as well as functional requirements that support quality requirements. Quality requirements may also imply architectural and structural requirements.

Software development process requirements may for example include requirements for artefacts, processes, project, development organisation, and developers. There will often be dependencies

between software development requirements and software product requirements. Possible dependencies are not shown in figure 9.

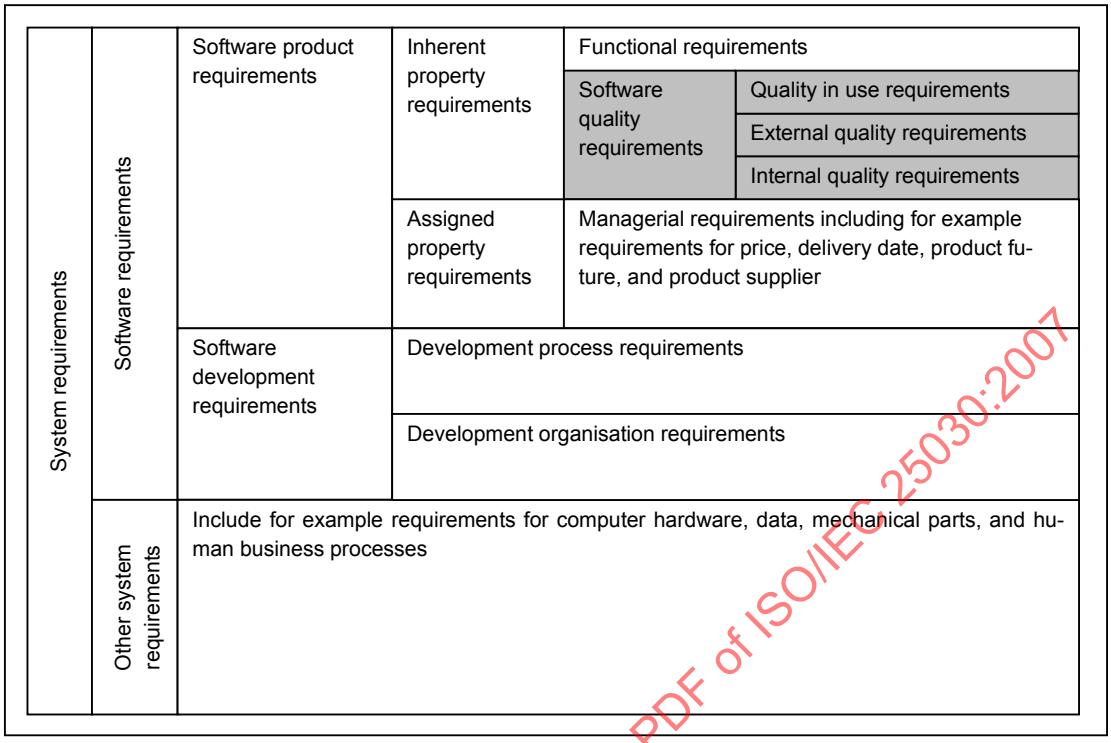


**Figure 9 – System requirements categorisation**

## 5.9 Quality requirements life cycle model

Stakeholder requirements come from many sources. Figure 10 illustrates the situation where a similar system already exists and is in use. In that case the properties and the users of the existing system will be a major source of input to the requirements of the new system. Stakeholder requirements can then largely be identified based on experiences from actual use of the existing system. When the system in question is completely new, i.e. no similar systems exist, it may be more difficult to identify the real needs of stakeholders.

There are three views of software quality as described in clause 5.4. These different views give rise to three types of software quality requirements:

- Software quality in use requirements
- External software quality requirements
- Internal software quality requirements

Quality in use requirements are typically derived from stakeholder requirements such as a) business requirements (company policy, competitors, etc.), b) functional requirements, and c) application domain specific requirements. Quality in use requirements are normally used for software validation (is the software fit for its intended purpose). External software quality requirements are typically derived from a number of sources including a) stakeholder requirements, b) legal requirements, c) standards and guidelines for the relevant application, d) quality in use requirements, e) functional requirements, f) application domain specific requirements, and g) security requirements, which may be derived from risk analysis. External software quality requirements are used for software validation and verification (is the software built according to specifications). Internal software

quality requirements are typically derived from a number of sources including a) external software quality requirements, b) company policy, c) development policy and limitations, and d) best practice guidelines. Internal software quality requirements are normally used for quality monitoring and control during development.

Software quality in use requirements may imply external software quality requirements and similarly external software quality requirements may imply internal software quality requirements. The software implementation process realises the software quality requirements. The quality of the new system can be used as input to another new system again, thereby completing the cycle indicated in figure 10.

Software requirements can be specified as part of an iterative development process, where one version of the software is used as a basis for deciding the requirements for the next version. In general it is the functional requirements that evolve in this way, whereas the software quality requirements are fixed. However, quality requirements can also evolve across versions.

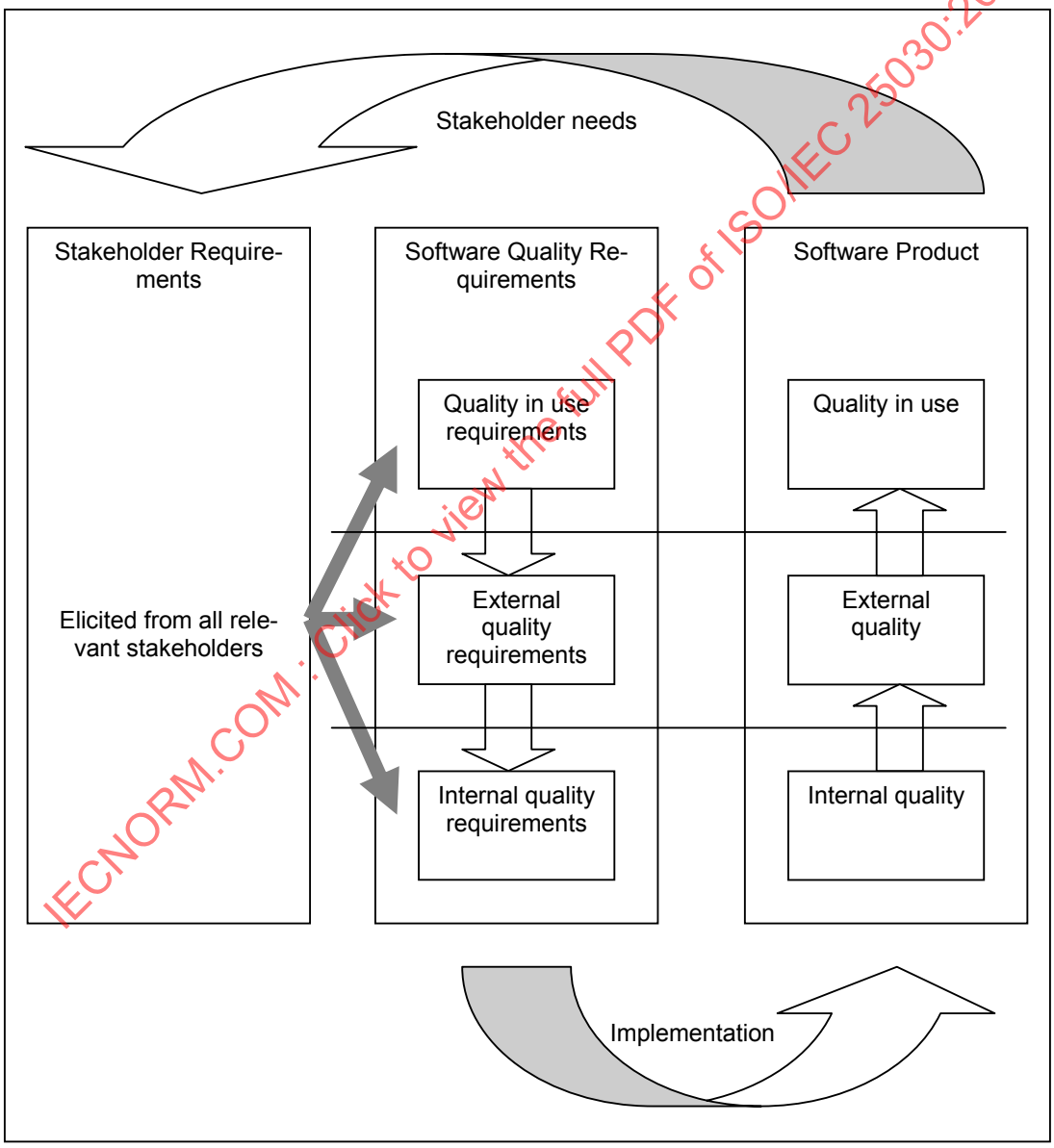




**Figure 10 – Quality requirements life cycle**

# 6 Requirements for quality requirements

Clause 6 provides requirements and recommendations for quality requirements for a software product.

## 6.1 General requirements and assumptions

Software is often part of a larger system. Architectural decisions made at a higher level of system hierarchal structure define boundaries and interfaces to the software.

Note 1 Architectural decisions about which parts of a system will actually be implemented in software may not or only partly be made at the time when stakeholder quality requirements are specified. Therefore it may be impossible to decide at an early time in the system life cycle process whether or not a quality requirement is related to software.

Note 2 Quality requirements cannot be seen in isolation, but only in connection with other types of requirements.

Note 3 ISO 9001:2000 requires that quality requirements for a product are determined. More specifically, clause 7.2.1 in 9001:2000 states the following requirements concerning determination of requirements related to a product: The organisation shall determine a) requirements specified by the customer, including the requirements for delivery and post-delivery activities, b) requirements not stated by the customer but necessary for specified or intended use, where known, c) statutory and regulatory requirements related to the product, and d) any additional requirements determined by the organisation.

This International Standard assumes that a software quality model with a structure similar to the ISO/IEC 25010 quality model is used. The applied software quality model shall be documented.

Note 4 The quality model defined in ISO/IEC 25010 is suggested. This model may be used in many situations, and is the basis for this International Standard. However, in some cases it may be appropriate to use another quality model emphasising specific quality aspects.

No specific development model or measures are prescribed or assumed.

Note 5 This International Standard applies the generic system life cycle processes framework defined in ISO/IEC 15288:2002. The stakeholder requirement definition process and the requirement analysis process are most relevant for this International Standard.

Note 6 There are many similarities between processes and activities involved with requirements in general and software product quality requirements in particular. In many cases the activities are combined.

## 6.2 Stakeholder requirements

Stakeholder requirements are related to a system. Some stakeholder requirements may not be relevant for software. When system architecture decisions are made it can be decided which stakeholder requirements influence the software. Clause 6.2 takes a system view and is not specific for software.

Note ISO/IEC 15288:2002 clause 5.5.2 describes stakeholder requirements definition process activities. See annex B of this International Standard for more details.

### 6.2.1 System boundaries

The requirements and recommendations in this clause aim to ensure that system boundaries and constraints relevant for the system quality are documented.

The boundaries of the system shall be documented.

Note 1 Clause 5.2 provides an explanation of the system concept.

The intended purpose of the system shall be documented. The documentation should include usage, the benefits, system specified life time, system criticality and risk dependencies like safety or hazard issues.

System constraints shall be documented.

Note 2 System constraints are unavoidable consequences of existing agreements, management decisions and technical decisions. These may result from 1) instances or areas of stakeholder-defined solution 2) implementation decisions made at higher levels of system hierarchical structure 3) required use of defined enabling systems, resources and staff.

Specific concepts and terms used should be defined or explained, in order to avoid misunderstandings of the system quality requirements.

### 6.2.2 Stakeholder quality requirements

The requirements and recommendations in this clause aim to ensure that the documented quality requirements take all stakeholder needs and expectations into consideration.

The purpose of specifying the stakeholder quality requirements shall be documented.

Note 1 The purpose may be new development, enhancement of existing system, modification of existing system, or evaluation of system.

All relevant stakeholders shall be listed.

Note 2 Clause 5.3 explains concepts related to stakeholders and their requirements.

The listed stakeholders' roles and interests shall be documented.

Note 3 The information needed about stakeholders depends on the situation. In some cases age is very important, in other cases for example nationality, occupation, sex, education, experience are important.

It shall be documented whether or not a stakeholder is taken into consideration when identifying stakeholder quality requirements. If a stakeholder is not taken into consideration the rationale should be documented.

Note 4 A commercially successful software product may have excluded specific stakeholder requirements from its requirements. For example, it may be decided to fulfil requirements of decision makers and neglect some end user requirements. It is important to be explicit about such decisions. A consequence is that some stakeholders may not be satisfied with the software product.

Identified but not stated stakeholder quality requirements shall be documented.

Scenarios and user interactions should be used to identify stakeholder quality requirements when appropriate. Scenarios should include a representative set of activity sequences to identify all required services that correspond to anticipated operational and support scenarios and environments. The interaction between users and the system should be documented. If possible, stakeholder quality requirements should be traceable to scenarios and user interaction descriptions.

Each stakeholder quality requirement shall be uniquely identified.

Note 5 Individual stakeholders may have unclear, ambiguous, unrealistic or conflicting requirements and there may be conflicts between different stakeholders' requirements. The stakeholder quality requirements represent a consolidated set of requirements where all such issues have been resolved. Clause 5.3 provides information about stakeholder requirements.

It should be documented that health, safety, security, environment and other stakeholder requirements and functions that relate to critical qualities are considered.

Note 6 Risk analysis can be applied for ensuring coverage of critical aspects.

Each stakeholder quality requirement shall be traceable to individual stakeholders or classes of stakeholders.

Note 7 ISO 9001:2000 clause 5.2 requires that top management shall ensure that customer requirements are determined and are met with the aim of enhancing customer satisfaction.

Note 8 ISO 9001:2000 clause 7.2.1 classifies requirements (user needs) as a) requirements specified by the customer, b) requirements not stated by the customer but necessary for specified use, c) statutory and regulatory requirements, and d) any additional requirements determined by the development organisation.

### 6.2.3 Validation of stakeholder quality requirements

The requirements and recommendations in this clause aim to ensure the quality of the stakeholder quality requirements.

The stakeholder quality requirements shall be validated and approved.

It shall be documented who has validated and approved the stakeholder quality requirements.

## 6.3 Software requirements

Clause 6.3 assumes that high level architectural decisions have been made, which identify the software elements of the system.

Note ISO/IEC 15288:2002 clause 5.5.3 describes the requirements analysis process activities. See annex B of this International Standard for more details.

### 6.3.1 Software boundaries

The intended purpose of the software shall be documented. The documentation should describe the role of the software in the system of which the software is an element.

Note 1 Software quality requirements may be part of a contractual agreement between acquirer and developer or may be input for a software product quality evaluation.

The functional boundaries of the software in terms of functional behaviour and properties to be provided shall be documented. The documentation should include architectural decisions that are allocated from design at higher levels in the structure of the system about which functions will be implemented as part of the software.

The system solution constraints on the software quality requirements shall be documented. The rationale and sources for system solution constraints should be documented as far as possible.

Implementation constraints relevant for the software quality requirements shall be documented.

Note 2 Implementation constraints are introduced by stakeholder requirements or are unavoidable solution limitations. This includes the implementation decisions that are allocated from design at higher levels in the structure of the system.

### 6.3.2 Software quality requirements

The requirements and recommendations in this clause aim to ensure that software quality requirements are documented according to the chosen quality model.

Software quality requirements shall be uniquely identified.

Software quality requirements shall be traceable to stakeholder requirements.

Software quality requirements shall take all relevant stakeholder quality requirements into consideration.

Note 1 Some stakeholder quality requirements may be realised by other means than software.

Software quality requirement shall be associated with quality characteristics (or subcharacteristics) as defined in the applied quality model.

Note 2 The quality model can be used as a checklist to ensure coverage of all quality aspects.

If no software quality requirements address a specific characteristic (or subcharacteristic) of the quality model, this shall be documented.

Software quality requirements should be categorised according to the quality model as one of the following:

- Software quality in use requirement;
- External software quality requirement;
- Internal software quality requirement.

A software quality requirement shall be specified in terms of a software quality measure and associated target value.

The software quality measures used shall be documented according to requirements in ISO/IEC 25020.

Note 3 It is recommended to develop and maintain a list of quality measures for use. Applying the same set of measures for different projects makes it possible to create an experience base as described in ISO/IEC 14598-3 [ISO/IEC 25042]. This approach makes estimation more reliable. Alternatively the GQM (Goal-Question-Metric) or similar approaches may be applied to identify appropriate measures for specific situations.

Note 4 The target values for the measure are the values which are acceptable for fulfilling the software quality requirements. The target values may be a single value or a range of values.

Note 5 ISO/IEC TR 9126-2, ISO/IEC TR 9126-3 and ISO/IEC 9126-4 [ISO/IEC 2502n] provide many measures, which may be used as they are or adapted to specific organisational needs.

Note 6 A quality requirement may be specified in terms of a set of other quality requirements. ISO/IEC 15408 'Common Criteria (CC) and Common Methodology for Information Technology Security Evaluation' provides an example of this approach. In this example an external software quality requirement (related to security) is specified as a set of internal software quality requirements.

It shall be documented which required functions of the software a software quality requirement is applicable for.

Note 7 See clause 5.7 for more information about the relation between software functions and software product quality requirements.

The criteria applied for selecting software quality measures shall be documented.

Note 8 Criteria may include relevance, feasibility and ease of data collection, protection of privacy, ease of interpretation, and life cycle stage applicability. ISO/IEC 25020 provides more detailed criteria for selecting among alternative measures.

The operational profile for a software quality requirement shall be specified when relevant.

Note 9 The operational profile specifies the workload, user profile and types of transactions under which the software must be able to maintain a specified level of service. Different operational profiles may result in dif-

ferent measurement results and hence, without a specification of the operational profile, the requirement is not uniquely specified. Context of use is an alternative term for this concept.

Note 10 This requirement applies primarily to software quality requirements referring to external quality attributes and quality in use attributes. Internal software quality attributes measures are often static; in this case the operational profile is not relevant.

When specifying the target value of a software quality requirement the acceptable tolerance shall be documented.

Note 11 The informative Annex A.2 of ISO/IEC 25020 provides a list of issues affecting the measurement reliability of quality measure elements.

### 6.3.3 Verification of software quality requirements

The requirements and recommendations in this clause aim to ensure the quality of the software quality requirements.

The software quality requirements shall be verifiable.

When special tools, techniques, or other resources such as effort or time are needed for verification this shall be documented. The time and effort needed to verify a software quality requirement should be documented. If special resources or competences are required in order to verify a software quality requirement, such demands should be documented.

Identified conflicts between software quality requirements shall be documented.

Note 1 In order to understand if two or more software quality requirements are achievable in a development project, a deep understanding of the underlying quality model is often mandatory. Practical experiences indicate that for example a high reliability requirement, a high maintainability requirement and a high efficiency requirement may be difficult to fulfil simultaneously. The ISO/IEC 9126-1 [ISO/IEC 25010] quality model provides no information suggesting such relationships. Requirements may be consistent, but difficult to fulfil in practice. What is achievable depends largely on the abilities of the software developers, and also on the methods, tools and techniques utilised.

Note 2 The process of keeping consistency between requirements may involve establishing precedence and priorities as well as using different requirements for different contexts of use.

Identified conflicts between software quality requirements and implementation constraints shall be documented.

Additional or changed software quality requirements that solve identified issues with software quality requirements shall be traceable to original software quality requirements. Software quality requirements that have been replaces or deleted shall be marked as such.

The software quality requirements shall be reviewed and approved.

It shall be documented who has reviewed and approved the software quality requirements.

Note 3 Software quality requirements may be part of a contractual agreement.

Note 4 Approval of the software quality requirements by the developing organisation implies that the organisation has the ability (technically, managerially, and financially) to meet the quality requirements.

Note 5 ISO 9001:2000 clause 7.2.2 provides requirements for product requirements review.

The software quality requirements shall be documented in a format such that they can be managed according to a configuration and change management system.

Note 6 The configuration management process is described in ISO/IEC 15288 clause 5.4.7

Note 7 Software quality requirements maintenance can be done by some or all of the following means: a) versioning each individual requirement whenever it is changed, b) indicating date of last change, c) specifying author of last change, d) giving the rationale of changes, and e) specifying the authorised change instance (owner of a specification).

(Blank page)

# Annex A
(normative)

# Terms and definitions

For the purposes of this document, the following terms and definitions given in ISO/IEC 25000 (repeated here for convenience) apply.

Note The definitions are common to all parts of the ISO/IEC 25000 SQuaRE series of International Standards.

## *A.1*
## acquirer

individual or organisation that acquires or procures a system, software product or software service from a supplier

Note Based on the definition in ISO/IEC 12207:1995.

## *A.2*
## analysis model

algorithm or calculation combining one or more base and/or derived measures with associated decision criteria

## *A.3*
## attribute

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

Note 1 based on ISO/IEC 15939:2002.

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

## *A.4*
## attribute for quality measure

Attribute that relates to software product itself, to the use of the software product or to its development process

Note Attributes for quality measure are used in order to obtain measurement primitives.

## *A.5*
## base measure

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

Note A base measure is functionally independent of other measures.

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

### *A.6*
### commercial-off-the-shelf software product

software product defined by a market-driven need, commercially available, and whose fitness for use has been demonstrated by a broad spectrum of commercial users

### *A.7*
### 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]

### *A.8*
### custom software

software product developed for a specific application from a user requirements specification

### *A.9*
### data

collection of values assigned to base measures, derived measures and/or indicators

[ISO/IEC 15939:2002]

### *A.10*
### decision criteria

thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result.

[ISO/IEC 15939:2002]

### *A.11*
### derived measure

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

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

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

### *A.12*
### developer

individual or organisation that performs development activities (including requirements analysis, design, testing through acceptance) during the software life cycle process

Note Based on the definition in ISO/IEC 12207:1995

### *A.13*
### division of standards

division forms a family of standards serving complementary purposes

### *A.14*
### end user

individual person who ultimately benefits from the outcomes of the system

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

### *A.15*
### entity

object that is to be characterised by measuring its attributes

EXAMPLE An object can be a process, product, project, or resource.

[ISO/IEC 15939:2002]

### *A.16*
### evaluation method

procedure describing actions to be performed by the evaluator in order to obtain results for the specified measurement applied to the specified product components or on the product as a whole

### *A.17*
### evaluation module

package of evaluation technology for measuring software quality characteristics, subcharacteristics or attributes

Note The package includes evaluation methods and techniques, inputs to be evaluated, data to be measured and collected and supporting procedures and tools.

### *A.18*
### evaluator

individual or organisation that performs an evaluation

### *A.19*
### external software quality

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

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

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

### *A.20*
### failure

termination of the ability of a product to perform a required function or its inability to perform within previously specified limits

Note Based on the definition in IEEE 610.12-1990.

### *A.21*
### fault

incorrect step, process or data definition in a computer program

[IEEE 610.12-1990]

### *A.22*
### functional requirement

requirement that specifies a function that a system or system component must be able to perform

[IEEE 610.12-1990]

Note The quality characteristic “functionality” can be used to specify or evaluate the suitability, accuracy, interoperability, security and compliance of a function (see ISO/IEC 9126-1 [ISO/IEC 25010]).

### *A.23*
### implied needs

needs that may not have been stated but are actual needs

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

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

### *A.24*
### indicator

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

[ISO/IEC 15939:2002]

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

### *A.25*
### information need

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

[ISO/IEC 15939:2002]

### *A.26*
### information product

one or more indicators and their associated interpretations that address information need

EXAMPLE A comparison of a measured defect rate to planned defect rate along with an assessment of whether or not the difference indicates a problem.

[ISO/IEC 15939:2002]

### *A.27*
### information system needs

needs that can be specified as quality requirements by external measures and sometimes by internal measures

### *A.28*
### intermediate software product

product of the software development process that is used as input to another stage of the software development process

EXAMPLE Intermediate software products can include static and dynamic models, other documents and source code.

### *A.29*
### intermediate software product needs

needs that can be specified as quality requirements by internal measures

### *A.30*
### internal software quality

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

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

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

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

### *A.31*
### maintainer

individual or organisation that performs maintenance activities

Note Based on the definition in ISO/IEC 12207: 1995

### *A.32*
### measure (noun)

variable to which a value is assigned as the result of measurement

Note The term “measures” is used to refer collectively to base measures, derived measures, and indicators.

[ISO/IEC 15939:2002]

### *A.33*
### measure (verb)

make a measurement

[ISO/IEC 14598-1:1999]

### *A.34*
### measurement

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

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

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

### *A.35*
### measurement function

algorithm or calculation performed to combine two or more base measures

[ISO/IEC 15939:2002]

### *A.36*
### measurement method

logical sequence of operations, described generically, used in quantifying an attribute with respect to a specified scale

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

### *A.37*
### measurement procedure

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

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

### *A.38*
### measurement process

process for establishing, planning, performing and evaluating software measurement within an overall project or organisational measurement structure

[ISO/IEC 15939:2002]

### *A.39*
### observation

instance of applying a measurement procedure to produce a value for a base measure

[ISO/IEC 15939:2002]

### *A.40*
### operator

individual or organisation that operates the system

Note Based on the definition in ISO/IEC 12207:1995.

### *A.41*
### process

system of activities, which use resources to transform inputs into outputs

[ISO 9000:2000]

### *A.42*
### quality in use (measure)

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

### *A.43*
### quality measure element

Measure, which is either a base measure or a derived measure that is used for constructing software quality measures

Note The software quality characteristic or subcharacteristic of the entity is derived afterwards by calculating a software quality measure.

### *A.44*
### quality model

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

### *A.45*
### rating

action of mapping the measured value to the appropriate rating level. Used to determine the rating level associated with the software product for a specific quality characteristic

### *A.46*
### rating level

scale point on an ordinal scale, which is used to categorise a measurement scale

Note 1 The rating level enables software product to be classified (rated) in accordance with the stated or implied needs.

Note 2 Appropriate rating levels may be associated with the different views of quality i.e. Users', Managers' or Developers'.

### *A.47*
### requirements

expression of a perceived need that something be accomplished or realized

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

### *A.48*
### scale

ordered set of values, continuous or discrete, or a set of categories to which the attribute is mapped

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

EXAMPLE Types of scales are: a nominal scale which corresponds to a set of categories; an ordinal scale which corresponds to an ordered set of scale points; an interval scale which corresponds to an ordered scale with equidistant scale points; and a ratio scale which not only has equidistant scale point but also possesses an absolute zero. Measures using nominal or ordinal scales produce qualitative data, and measures using interval and ratio scales produce quantitative data.

### *A.49*
### software product

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

[ISO/IEC 12207:1995]

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

Note 2 In SQuaRE Internaional Standards software quality has the same meaning as software product quality.

### *A.50*
### software product evaluation

technical operation that consists of producing an assessment of one or more characteristics of a software product according to a specified procedure

### *A.51*
### software quality

capability of software product to satisfy stated and implied needs when used under specified conditions

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

### *A.52* software quality characteristic

category of software quality attributes that bears on software quality

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

### *A.53*
### software quality evaluation

systematic examination of the extent to which a software product is capable of satisfying stated and implied needs

### *A.54*
### software quality in use

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

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

### *A.55*
### Software quality measure

measure of internal software quality, external software quality or software quality in use

Note Internal software quality, external software quality and software quality in use are described in the quality model in ISO/IEC 9126-1 [ISO/IEC 25010].

### *A.56*
### stakeholder

a party having a right, share or claim in a system or in its possession of characteristics that meet that party's needs and expectations

[ISO/IEC 15288:2002]

Note Stakeholders include, but are not limited to, end users, end user organisations, supporters, developers, producers, trainers, maintainers, disposers, acquirers, supplier organisations and regulatory bodies

### *A.57*
### supplier

individual or organisation that enters into a contract with the acquirer for the supply of a system, software product or software service under the terms of the contract

[ISO/IEC 12207:1995]

### *A.58*
### system

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

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

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

[ISO/IEC 15288:2002]

### *A.59*
### target of process

software product or task executed by software product to which measurement or evaluation process is applied

### *A.60*
### unit of measurement

particular quantity defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity

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

### *A.61*
### user

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

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

[ISO/IEC 15939:2002]

### *A.62*
### validation

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

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

[ISO 9000:2000]

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

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

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