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.

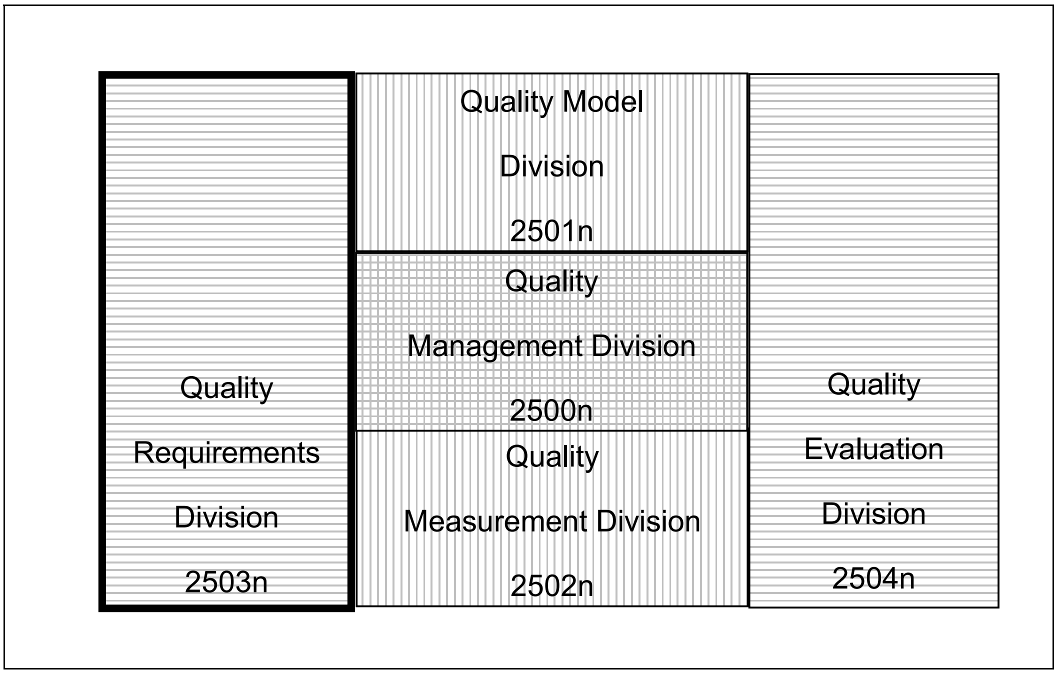

Figure 1 — Organisation of the ISO/IEC 25000 SQuaRE series of International Standards

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.

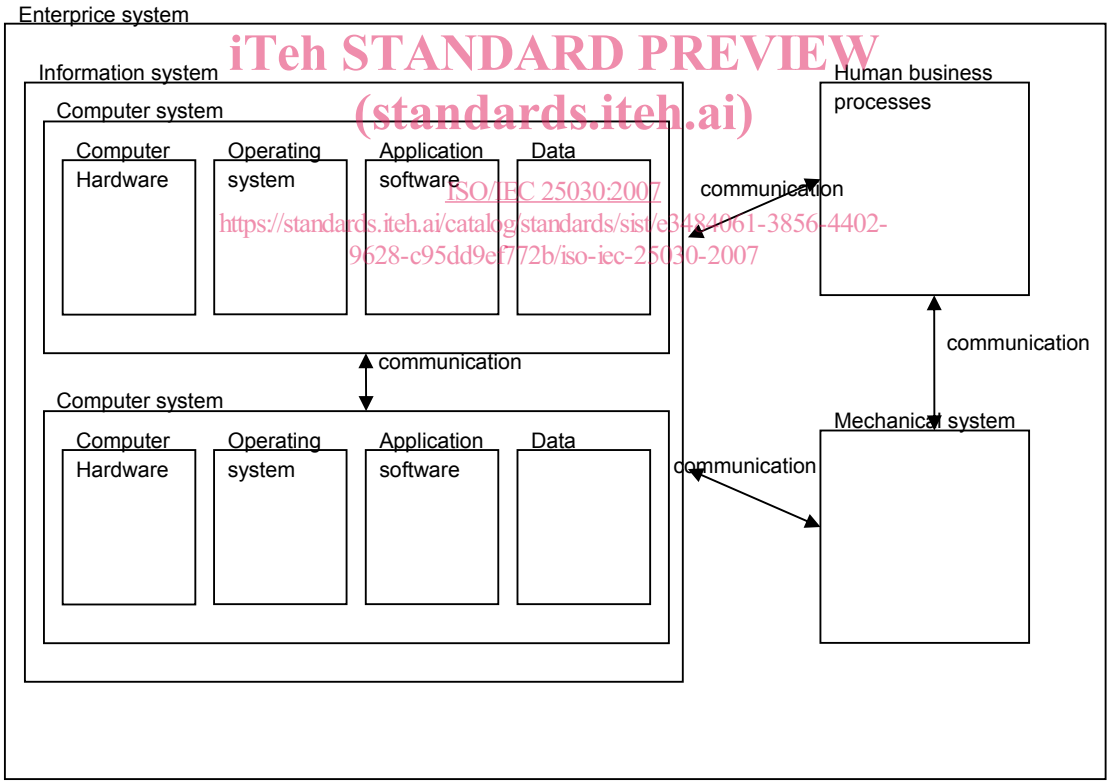

Figure 2 – Example of a system model

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.

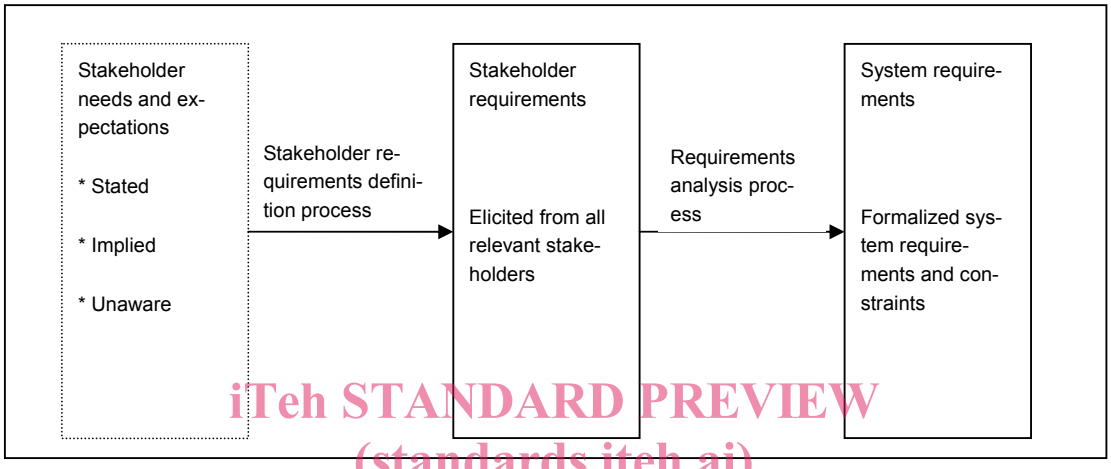

Figure 3 – Stakeholder requirements definition and analysis

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.

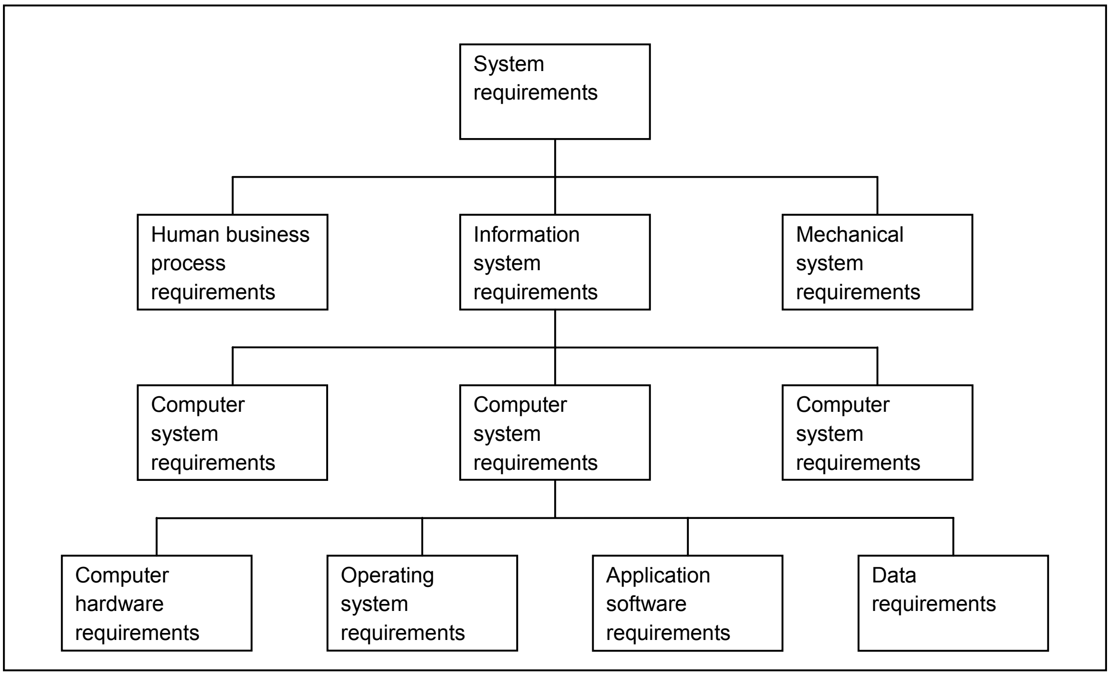

Figure 4 – System and software requirements hierarchy

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.

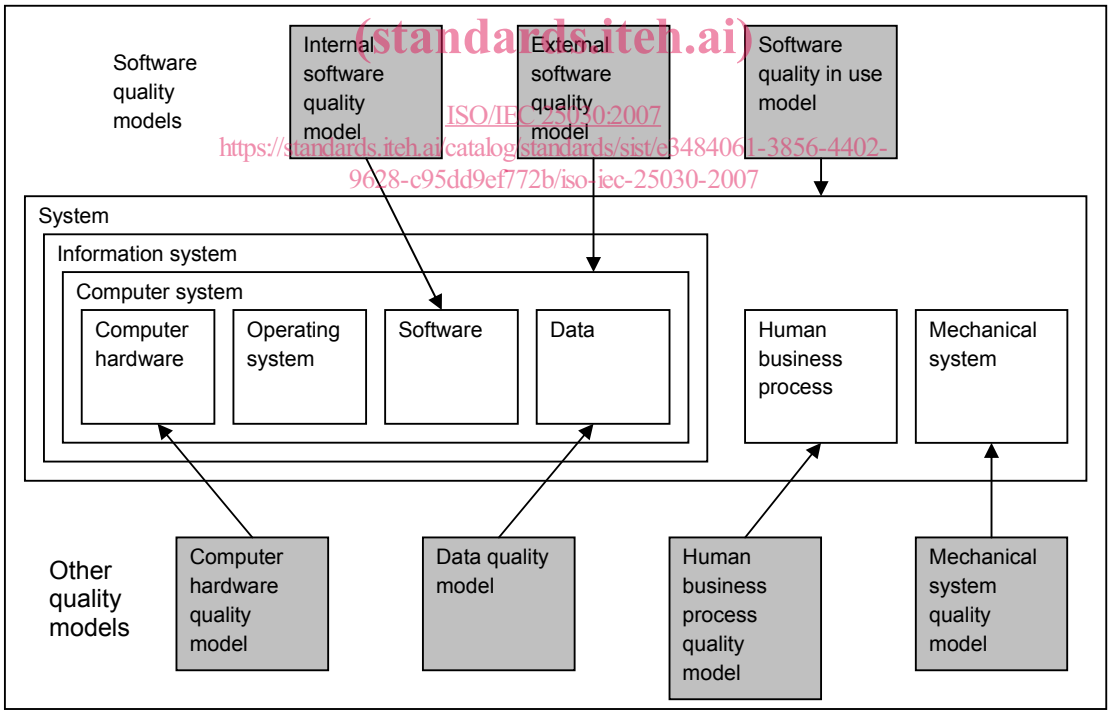

Figure 5 – Example of system model and quality models

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.

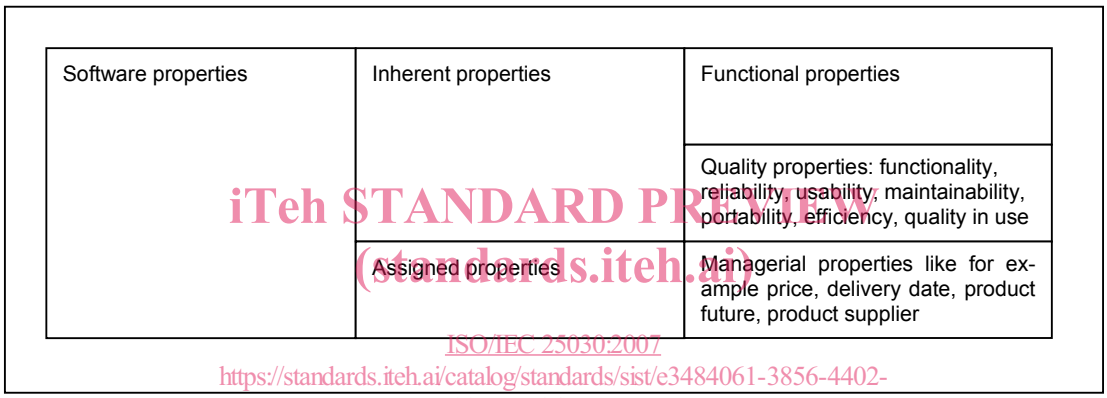

Figure 6 – Software properties

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.