



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

**ISO/IEC JTC1/SC7 N1667**  
**1997-02-12**

**DOC TYPE:** CD Ballot

**TITLE:** CD 9126-1 - Information technology -  
Software quality characteristics and metrics Part 1 -  
Quality characteristics and sub-characteristics

**SOURCE:** JTC1/SC7/WG6

**PROJECT:** 07.13.01.1

**STATUS:** CD

**REFERENCES:** ISO 9126, SC7 N 1558, N1626 & N1666

**ACTION ID:** ACT

**DUE DATE:** 1997-05-26

**MAILING DATE:** 1997-02-12

**DISTRIBUTION:** P, O and L

**MEDIUM:** E-Mail - uuencoded Word, Diskette - Word file

**NO. OF PAGES:** 27

**DISK NUMBER:** 19

**NOTE:**

---

Address reply to: ISO/IEC JTC1/SC7 Secretariat  
Bell Canada - IT Procurement & Supplier Quality  
2265 Roland Therrien, Room 226, Longueuil (Québec) Canada J4N 1C5  
Tel.: +1 (514) 448-5100 Fax: +1 (514) 448-2090 or +1 (514) 647-3163  
sc7@qc.bell.ca





ISO/IEC JTC1/SC7 CD 9126-1	
Date <b>1997-02-12</b>	Reference number ISO/JTC 1/SC 7 N 1667
Supersedes document SC7 N 1558	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/JTC 1/SC 7 Committee Title <b>Software Engineering</b> Secretariat: Standards Council of Canada (SCC)	Circulated to P- and O-members, and to technical committees and organizations in liaison for: discussion at X comments by 1997-05-26 X voting by (P-members only)  <b>1997-05-26</b>  Please return all votes and comments in electronic form directly to the SC 7 Secretariat by the due date indicated.

ISO/IEC JTC1/SC7

Title: Information Technology - Software quality characteristics and metrics Part 1 - Quality characteristics and sub-characteristics.

Project: 07.13.01.1.

---

Introductory note: See comment disposition report in N1666 and Annex C of the attached document.

Medium: E-Mail - uuencoded Word, Diskette - Word file

No. of pages: 27

---

---

Address reply to: ISO/IEC JTC1/SC7 Secretariat  
Bell Canada - IT Procurement & Supplier Quality  
2265 Roland Therrien, Room 226, Longueuil (Québec) Canada J4N 1C5  
Tel.: +1 (514) 448-5100 Fax: +1 (514) 448-2090 or +1 (514) 647-3163  
sc7@qc.bell.ca



**Vote on Second Committee Draft ISO/IEC 9126-1**Date of circulation  
**1997-02-12**Reference number  
ISO/JTC 1/SC 7 N 1667Closing date  
**1997-05-26**ISO/JTC 1/SC 7  
Committee Title  
**Software Engineering**  
Secretariat:  
Standards Council of Canada (SCC)

Circulated to P-members of the committee for voting

Please return all votes and comments in electronic form directly to the SC 7 Secretariat by the due date indicated.

Title: Information Technology - Software quality characteristics and metrics Part 1 -  
Quality characteristics and sub-characteristics.

Project: 07.13.01.1

**Vote:**☐ APPROVAL OF THE DRAFT AS PRESENTED☐ APPROVAL OF THE DRAFT WITH COMMENTS AS GIVEN ON THE ATTACHED☐ general:

technical:

editorial:

☐ DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED☐ Acceptance of these reasons and appropriate changes in the text will change our vote to approval☐ ABSTENTION (FOR REASONS BELOW):

P-member voting:

National Body (Acronym)

Date:

YYCC-MM-DD

Submitted by:

Name

---

Address reply to: ISO/IEC JTC1/SC7 Secretariat  
Bell Canada - IT Procurement & Supplier Quality  
2265 Roland Therrien, Room 226, Longueuil (Québec) Canada J4N 1C5  
Tel.: +1 (514) 448-5100 Fax: +1 (514) 448-2090 or +1 (514) 647-3163  
sc7@qc.bell.ca



# ISO/IEC JTC1/SC7/WG6 N385

## Evaluation and Metrics

TITLE: ISO/IEC 9126-1:  
Information Technology - Software quality characteristics and metrics -  
**Part 1: Quality characteristics and sub-characteristics**

DATE: 12-Feb-97

SOURCE: JTC1/SC7/WG6

WORK ITEM: Project 7.13.01.1

STATUS: Version 6.3 based on feedback after the Curitiba meeting

DOCUMENT  
TYPE: **CD**

ACTION: For vote

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

DOCUMENT EDITOR: Nigel BEVAN  
National Physical Laboratory  
NPL Usability Services  
Teddington, Middx. TW11 0LW, United Kingdom  
FAX: +44 181 943 6306  
Nigel@hci.npl.co.uk

CO-EDITOR: Julie MCMULLAN  
Centre for Software Engineering Ltd  
Dublin City University Campus  
Dublin 9  
Ireland  
FAX: +353-1-7045605  
julie@cse.dcu.ie

REVIEWERS V Godamunne, T Komiyama, D Natale

**Contents**

<b>1. SCOPE</b>	<b>5</b>
<b>2. NORMATIVE REFERENCES</b>	<b>5</b>
<b>3. CONFORMANCE</b>	<b>6</b>
<b>4. DEFINITIONS</b>	<b>6</b>
<b>5. QUALITY RELATIONSHIPS</b>	<b>6</b>
5.1 Quality and the life-cycle	6
5.2 Approaches to quality	8
5.3 Item to be evaluated	8
5.4 Quality model	9
<b>6. METRICS</b>	<b>9</b>
6.1 Attributes and characteristics	9
6.2 Internal metrics	10
6.3 External metrics	10
6.4 Relationship between external and internal metrics	10
6.5 Quality in use metrics	11
6.6 Choice of metrics	11
<b>7. SOFTWARE QUALITY CHARACTERISTICS</b>	<b>12</b>
7.1 Functionality	12
7.2 Reliability	13
7.3 Usability	13
7.4 Efficiency	14
7.5 Maintainability	14
7.6 Portability	14
<b>8. QUALITY IN USE</b>	<b>16</b>
<b>ANNEX A (INFORMATIVE) DEFINITIONS FROM OTHER STANDARDS</b>	<b>17</b>
<b>ANNEX B (INFORMATIVE) BIBLIOGRAPHY</b>	<b>20</b>
<b>ANNEX C (INFORMATIVE) HISTORY OF THE WORK</b>	<b>21</b>



## FOREWORD

ISO (the International Organisation for Standardisation) and IEC (the International Electrotechnical Commission) form the specialised system for worldwide standardisation. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of mutual interest. Other international organisations, 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. 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.

International Standard ISO/IEC 9126-1 was prepared by Joint Technical Committee ISO/IEC JTC1 *Information Technology*.

ISO/IEC 9126 consists of the following parts under the general title *Information Technology - Software quality characteristics and metrics*

*Part 1: Quality characteristics and sub-characteristics*

*Part 2: External Metrics*

*Part 3: Internal Metrics*

Annexes A, B and C of this part of ISO/IEC 9126 are for information.

## Introduction

As software applications have grown, so too has the importance of software quality. In order to specify and evaluate software product quality objectively and quantitatively, a framework for software quality is necessary. This part of ISO/IEC 9126 when used in conjunction with ISO/IEC 14598 provides that framework.

This part of ISO/IEC 9126 is a revision of ISO/IEC 9126 (1991), and retains the same software quality characteristics. The major changes have been to:

- introduce normative sub-characteristics;
- specify a quality model;
- remove the evaluation process (which is now contained in ISO/IEC 14598-1);
- ensure consistency with ISO/IEC 14598-1.

# Information Technology - Software Quality Characteristics and Metrics-

## Part 1: Quality characteristics and sub-characteristics

### 1. Scope

This part of ISO/IEC 9126 specifies a quality model which categorises software quality into six characteristics, which are further sub-divided into sub-characteristics. These sub-characteristics are manifested externally when the software is used as a part of a computer system, and are a result of internal software attributes. The combined result of the software quality characteristics for the end user is defined as quality in use. Other parts of ISO/IEC 9126 describe software quality metrics based on internal software attributes and external computer system behaviour. These types of metrics are applicable when specifying quality requirements and design goals for software products and intermediate products. An explanation of how this quality model can be applied in software product evaluation is contained in ISO/IEC 14598-1.

The characteristics defined are applicable to every kind of software, including computer programs and data contained in firmware.

This part of ISO/IEC 9126 is intended for those associated with acquisition, development, use, evaluate, support, maintenance, or audit of software. The quality model defined in this part of ISO/IEC 9126 can be used to:

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

### 2. Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9126. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9126 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

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

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

ISO 8402:(1994) Quality Vocabulary.

ISO/IEC 9126-2:(new) Information Technology - Software quality characteristics and metrics - Part 2: External metrics

ISO/IEC 9126-3:(new) Information Technology - Software quality characteristics and metrics - Part 3: Internal metrics

ISO 9241-10 (1996) Ergonomic requirements for office work with visual display terminals (VDT)s - Part 10: Dialogue principles

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

ISO/IEC 12207 (1995) Information Technology - Software life-cycle processes

ISO/IEC DIS 14598-1 (1996) Information Technology - Software product evaluation - Part 1: General overview

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

[Editor's Note - any of the above which are not referenced, or have not reached at least DIS stage at the time of publication, will be moved to Annex A or deleted.]

### 3. Conformance

Any specification of software quality conforming to this part of ISO/IEC 9126 shall include all relevant information of the following types:

1. Criteria for internal metrics for the characteristics and sub-characteristics in clause 7.

NOTE: Examples of appropriate metrics are given in ISO/IEC 9126-3.

2. Criteria for external metrics for the characteristics and sub-characteristics in clause 7.

NOTE: Examples of appropriate metrics are given in ISO/IEC 9126-2.

3. Criteria for quality in use metrics in clause 8.

NOTE: Examples of appropriate metrics are given in ISO/IEC 9126-2.

4. Compliance to any relevant standards, conventions or regulations in laws and similar prescriptions.

NOTE: Examples of appropriate metrics are given in ISO/IEC 9126-2 and ISO/IEC 9126-3.

### 4. Definitions

For the purpose of this part of ISO/IEC 9126, the following definitions and the definitions contained in ISO/IEC 14598-1 apply.

NOTE - the relevant definitions are reproduced in informative annex A.

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

### 5. Quality relationships

Software quality shall be evaluated using a defined quality model. A quality model shall be used when setting quality goals for software products and intermediate products. This part of ISO/IEC 9126 provides a recommended quality model which can be used as a checklist of issues related to quality (although other ways of categorising quality may be more appropriate in particular circumstances). When a quality model other than that in this part of ISO/IEC 9126 is used it shall be clearly described.

#### 5.1 Quality and the life-cycle

Quality changes with the life-cycle of the software, i.e., required quality at the start of the life-cycle differs from actual or delivered quality. Quality is also a reflection of diverse points of view. It is necessary to define these diverse views of quality and changes in quality with the life-cycle, in order to manage quality properly at each stage of the life-cycle. The following are the different views of quality at different stages in the life-cycle:

**Goal Quality (GQ)** means necessary and sufficient quality that reflects real user needs.

ISO 8402 defines quality in terms of the ability to satisfy stated and implied needs. However, needs stated by a user do not always reflect real user needs, because a user is often not aware of his real needs and needs may change after they are stated. So GQ is a conceptual entity which cannot be completely defined at the beginning of design. Yet, developers must keep this goal in mind and try to get closer to it. GQ does not necessarily mean perfect quality, but necessary and sufficient quality. Part of the requirements for GQ can be measured by quality in use (QIU) when the product is complete, and requirements for QIU may optionally be included in the quality requirements specification.

**Required Product Quality (RPQ)** means quality reflected by what is actually stated in the quality requirements specification.

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

**Design Quality (DQ)** means the quality represented in the core parts or backbone of software design, e.g., software architecture, program structure, and user interface design strategy.

DQ is the reflection of the design philosophy and strategy. Details of software quality may be improved during code implementation and testing, but the fundamental nature of the software product quality represented by DQ remains unchanged.

**Estimated (or Predicted) Product Quality (EPQ)** is quality that is estimated or predicted for the end software product quality at each stage of development, and which is based on DQ.

Product quality may be estimated and predicted during development for each quality characteristic defined in ISO/IEC 9126-1. For the purposes of prediction, technology should be developed to show the co-relation between DQ and EPQ.

**Delivered Product Quality (DPQ)** represents the quality of the delivered product, tested in a simulated environment with simulated data.

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

**Quality in Use (QIU)** is the quality that is perceived by users when the software is actually used in the users' environment. It can be measured by effectiveness, task efficiency and satisfaction.

The quality in the users' environment may be different from that in the developers' environment, because some functionality may not be visible to a user, or may not be used by a user. The user evaluates only those attributes of software, which are visible to him/her during actual use. Sometimes, software attributes specified by a user during the requirements analysis phase, are perceived as not preferable.

## 5.2 Approaches to quality

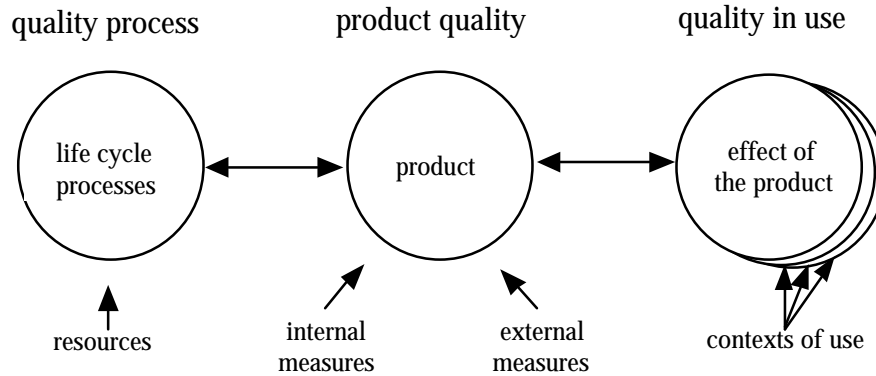


Figure 1. Quality in the life-cycle

Evaluation of software products in order to achieve software product quality is one of the processes in the software development life-cycle. Software product quality can be measured internally (typically by static measures of the code), or externally (typically by measuring the behaviour of the code when executed). The objective is for the product to have the required effect in a particular context of use.

Process quality contributes to improving product quality, and product quality contributes to improving quality in use. Therefore, assessing and improving a process is a mean to improve product quality, and evaluating and improving a product quality is one means of improving quality in use.

Appropriate quality processes are required to support the measurement of quality during development. Appropriate internal attributes of the software are a pre-requisite for achieving the required external behaviour, and appropriate external behaviour is a pre-requisite for achieving quality in use.

## 5.3 Item to be evaluated

Items can be evaluated by direct measurement, or indirectly by measuring their consequences. For example, a process may be assessed by measuring and evaluating its product, and a product may be evaluated by measuring the task performance of a user.

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

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

A further distinction can be made between the evaluation of a software product and the evaluation of the system in which it is executed. For example, the reliability of a system is assessed by observing all failures due to whatever cause (hardware, software, human error, etc.), whereas the reliability of the software product is assessed by extracting from the observed sample of failures only those that are due to design faults in the software. Also, where the boundary of the system is judged to depend upon the purpose of the evaluation, and upon who the users are. For example, if the users of an aircraft with a computer-based flight control system are taken to be the passengers, then the system upon which they depend includes the flight crew, the airframe, and the hardware and software in

the flight control system, whereas if the flight crew are taken to be the users, then the system upon which they depend consists only of the airframe and the flight control system.

## 5.4 Quality model

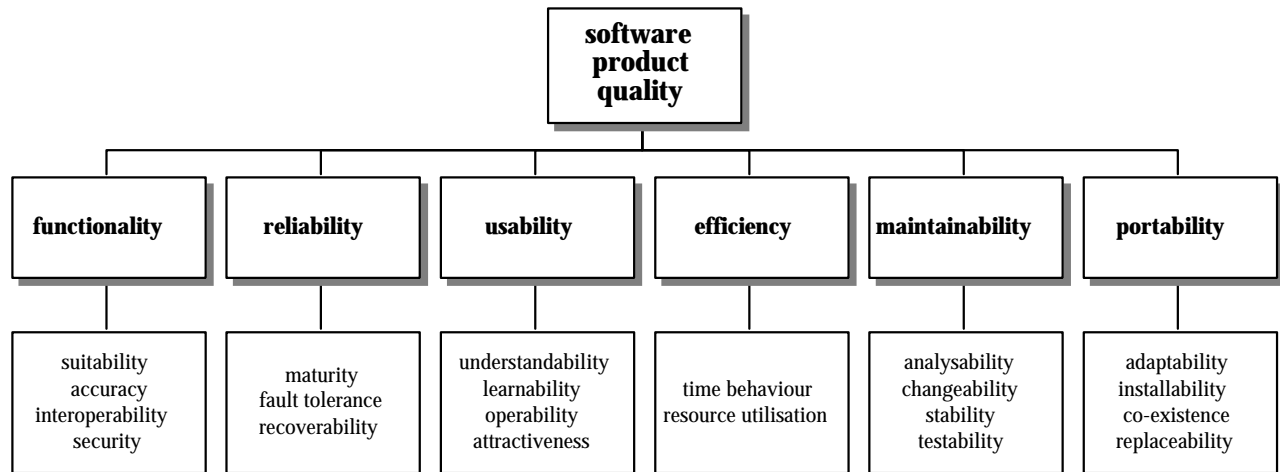


Figure 2. Software product quality

This part of ISO/IEC 9126 categorises the attributes of software quality into six characteristics (functionality, reliability, usability, efficiency, maintainability and portability), which are further sub-divided into sub-characteristics (see clause 7). Sub-characteristics can either be measured by internal metrics or by external metrics. Pure internal metrics (such as program size) are measures of the software which are not normally used alone as software quality metrics, but are used in combination with other measures to create metrics.

Quality in use is the user's view of quality. Achieving quality in use is dependent on meeting criteria for external measures of the relevant quality sub-characteristics, which in turn is dependent on achieving related criteria for the associated internal measures. Measures are normally required at all three levels, as meeting criteria for internal measures is not usually sufficient to ensure achievement of criteria for external measures, and meeting criteria for external measures of sub-characteristics is not usually sufficient to ensure achieving criteria for quality in use.

It is not practically possible to measure all sub-characteristics internally or externally for all parts of a large software product. Similarly it is not usually practical to measure quality in use for all possible user-task scenarios. Resources for evaluation need to be allocated between the different types of measurement dependent on the business objectives and the nature of the product and design process.

## 6. Metrics

### 6.1 Attributes and characteristics

The levels of certain internal attributes have been found to influence the levels of some external measures, so that there is both an external aspect and an internal aspect to most characteristics. For example, reliability may be measured externally by observing the number of failures in a given period of execution time during a trial of the software, and internally by inspecting the detailed specifications and source code to assess the level of fault tolerance. The internal attributes are said to be indicators of the external characteristics. One internal attribute may influence one or more characteristics as well as one characteristic may be influenced by more than one attribute (figure 3).

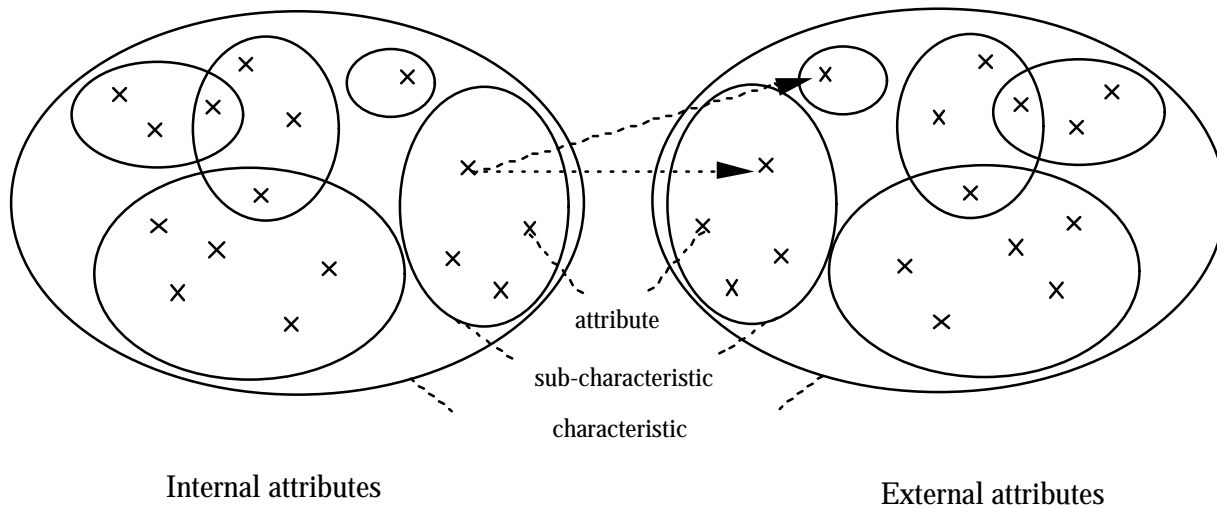


Figure 3: Quality characteristics, subcharacteristics and attributes

Note that the correlation between internal attributes and external measures is never perfect however, and the effect that a given internal attribute has upon an associated external measure will be determined by experience, and will depend on the particular context in which the software is used.

In the same way, external properties (such as suitability, accuracy, fault tolerance or time behaviour) will influence the observed quality. A failure in quality in use (e.g. the user cannot complete the task) can be traced to external quality (e.g. suitability or operability) and the associated internal attributes which have to be changed.

## 6.2 Internal metrics

Internal metrics can be applied to a non executable software product (such as a specification or source code) during designing and coding. When developing a software product the intermediate products should be evaluated using internal metrics which measure intrinsic properties derived from simulated behaviour. The primary purpose of these internal metrics is to ensure that the required external quality is achieved: examples are given in ISO/IEC 9126-3. Internal metrics provide users, evaluators, testers, and developers with the benefit that they are able to evaluate software product quality and address quality issues early before the software product becomes executable.

Internal metrics measure internal attributes or indicate external attributes by analysis of the static properties of the intermediate or deliverable software products. The measurements of internal metrics use numbers or frequencies of software composition elements which appear on source code statements, the control graph, data flow and state transition representations.

NOTE - Documentation can also be evaluated using internal metrics.

## 6.3 External metrics

External metrics use measures of a software product derived from measures of the behaviour of the system of which it is a part, by testing, operating and observing the executable software or system. Before acquiring or using a software product it should be evaluated using metrics based on business objectives related to the use, exploitation and management of the product in a specified organisational and technical environment. These are primarily external metrics: examples are given in ISO/IEC 9126-2. External metrics provide users, evaluators, testers, or developers with the benefit that they are able to evaluate software product quality during testing or operation.

## 6.4 Relationship between external and internal metrics

When the software quality requirements are defined, the software quality characteristics or sub-characteristics which represent the quality requirements are listed. Then, the appropriate external metrics and acceptable ranges are specified to quantify the quality criteria which validate that the software meets the user needs. The internal quality



attributes of the software are then defined and specified to plan to achieve the required external quality characteristics finally and to build them into the intermediate product during development. Appropriate internal metrics and acceptable range are specified to quantify the internal quality characteristics so that they can be used for verifying that the intermediate software meets the internal quality specifications during the development.

It is recommended that the internal metrics are used which have as strong a relation as possible with the target external metrics, so that they can be used to predict the values of external metrics. However, it is generally difficult to design a rigorous theoretical model which provides a strong relationship between internal metrics and external metrics.

## 6.5 Quality in use metrics

The objective of software quality is to achieve quality in use. For systems with end users this means that specified types of users should be able to carry out specified types of tasks to a required level of productivity and user satisfaction in specified environments. Evaluating quality in use validates software quality in specific user-task scenarios.

Quality in use is the user's view of the quality of a system containing software, and is measured in terms of the result of using the software, rather than properties of the software itself. Quality in use is the combined effect of the software quality characteristics for the end user.

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

- the end user for whom quality in use is a result of functionality, reliability, usability and efficiency
- the person maintaining the software for whom quality in use is a result of maintainability
- the person porting the software for whom quality in use is a result of portability

Quality in use may be influenced by any of the quality characteristics, and is thus broader than usability, which is only concerned with the ease of use and attractiveness.

NOTE - Usability is defined in ISO 9241-11 in a similar way to the definition of quality in use in this part of ISO/IEC 9126. For the purposes of this part of ISO/IEC 9126 usability refers only to the ease of use and attractiveness of the software.

## 6.6 Choice of metrics

The basis on which the metrics are selected will depend on the business priorities for the product and the needs of the evaluator. The model in this part of ISO/IEC 9126 supports a variety of evaluation requirements, for example:

- a user or a user's business unit could evaluate the suitability of a software product using metrics for quality in use;
- an acquirer could evaluate a software product against criterion values of external measures of functionality, reliability, usability and efficiency;
- a maintainer could evaluate a software product using metrics for maintainability;
- a person responsible for implementing the software in different environments could evaluate a software product using metrics for portability;
- a developer could evaluate a software product against criterion values using internal measures of any of the quality characteristics.

## 7. Software quality characteristics

The quality model in clause 5 categorises software product quality into six characteristics (functionality, reliability, usability, efficiency, maintainability and portability). These are further sub-divided into the sub-characteristics listed below. In addition there are pure internal metrics (see ISO/IEC 9126-3) and pure external metrics (quality in use, see clause 8).

Definitions are given for each quality characteristic and the sub-characteristics of the software which influence the quality characteristic. For each characteristic and sub-characteristic, the capability of the software is determined by a set of internal attributes which can be measured. Examples of internal metrics are given in ISO/IEC 9126-3. The characteristics and sub-characteristics can be measured externally by the extent to which the capability is provided by the system containing the software. Examples of external metrics are given in ISO/IEC 9126-2.

It is also necessary to define metrics which assess the capability of the software to comply with any relevant standards, conventions or regulations in laws and similar prescriptions which may exist for each characteristic, e.g.

- the capability of the software to adhere to application related standards, conventions or regulations in laws and similar prescriptions:
- the capability of the software to adhere to standards, style guides or regulations relating to ergonomics and ease of use;
- the capability of the software to adhere to standards or conventions relating to portability.

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

### 7.1 Functionality

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

#### NOTES

- 1 This characteristic is concerned with what the software does to fulfil needs, whereas the other characteristics are mainly concerned with when and how it does.
- 2 For the stated and implied needs in this characteristic, the note to the definition of quality applies, (see A.1.1).
- 3 For a system which is operated by a user, the combination of functionality, usability and efficiency can be measured externally by quality in use (see clause 8).

**7.1.1 Suitability:** The capability of the software to provide an appropriate set of functions for specified tasks and user objectives.

#### NOTES

- 1 Examples of appropriateness are task oriented composition of functions from constituent sub-functions, capacities of tables.
- 2 Suitability corresponds to suitability for the task in ISO 9241-10, and is a pre-requisite for operability.

**7.1.2 Accuracy:** The capability of the software to provide the right or agreed results or effects.

NOTE - This includes the expected data with the needed degree of precision of calculated values.

**7.1.3 Interoperability:** The capability of the software to interact with one or more specified systems.

NOTE - Interoperability is used in place of compatibility in order to avoid possible ambiguity with replaceability (see 7.6.4).

**7.1.4 Security:** The capability of the software to prevent unintended access and resist deliberate attacks intended to gain unauthorised access to confidential information, or to make unauthorised modifications to information or to the program so as to provide the attacker with some advantage or so as to deny service to legitimate users.

## NOTES

- 1 This also applies to data in transmission.
- 2 **Safety** is not defined as a sub-characteristic, as it does not relate to software alone, but to a whole system.

## 7.2 Reliability

**reliability:** The capability of the software to maintain the level of performance of the system when used under specified conditions

## NOTES

- 1 Wear or ageing does not occur in software. Limitations in reliability are due to faults in requirements, design, and implementation. Failures due to these faults depend on the way the software product is used and the program options selected rather than on elapsed time.
- 2 The definition of reliability in ISO/IEC DIS 2382-14:1994 is "The ability of functional unit to perform a required function...". In this document, functionality is only one of the characteristics of software quality. Therefore, the definition of reliability has been broadened to "maintain its level of performance..." instead of "...perform a required function"

**7.2.1 Maturity:** The capability of the software to avoid failure as a result of faults in the software.

**7.2.2 Fault tolerance:** The capability of the software to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.

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

**7.2.3 Recoverability:** The capability of the software to re-establish its level of performance and recover the data directly affected in the case of a failure.

## NOTES

- 1 Following a failure, a software product will sometimes be down for a certain period of time, the length of which is assessed by its recoverability.
- 2 **Availability** is the capability of the software to be in a state to perform a required function at a given point in time, under stated conditions of use. Externally, availability can be assessed by the proportion of total time during which the software product is in an up state. Availability is therefore a combination of maturity (which governs the frequency of failure) and recoverability (which governs the length of down time following each failure).

## 7.3 Usability

**usability:** The capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.

## NOTES

1. Some aspects of functionality, reliability and efficiency will also affect usability, but for the purposes of ISO/IEC 9126 are not classified as usability.
2. Users may include operators, end users and indirect users who are under the influence of or dependent on the use of the software. Usability should address all of the different user environments that the software may affect, which may include preparation for usage and evaluation of results.

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

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

**7.3.2 Learnability:** The capability of the software product to enable the user to learn its application.

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

**7.3.3 Operability:** The capability of the software product to enable the user to operate and control it.

## NOTES

- 1 Aspects of changeability, adaptability and installability may be pre-requisites for operability.
- 2 Operability corresponds to controllability, error tolerance and conformity with user expectations as defined in ISO 9241-10.
- 3 For a system which is operated by a user, the combination of functionality, usability and efficiency can be measured externally by quality in use.

**7.3.4 Attractiveness:** The capability of the software product to be liked by the user.

NOTE - This refers to attributes of the software intended to make the software more attractive to the user.

## 7.4 Efficiency

**efficiency:** The capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.

### NOTES

- 1 Resources may include other software products, hardware facilities, materials, (e.g. print paper, diskettes).
- 2 For a system which is operated by a user, the combination of functionality, operability and efficiency can be measured externally by quality in use.

**7.4.1 Time behaviour:** The capability of the software to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.

**7.4.2 Resource utilisation:** The capability of the software to use appropriate resources in an appropriate time when the software performs its function under stated conditions.

## 7.5 Maintainability

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

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

**7.5.2 Changeability:** The capability of the software product to enable a specified modification to be implemented.

### NOTES

- 1 Implementation includes coding, designing and documenting changes.
- 2 If the software is to be modified by the end user, changeability may be a pre-requisite for operability.

**7.5.3 Stability:** The capability of the software to minimise unexpected effects from modifications of the software.

**7.5.4 Testability:** The capability of the software product to enable modified software to be validated.

NOTE - Values of this sub-characteristic may be altered by the modifications under consideration.

## 7.6 Portability

**portability:** The capability of software to be transferred from one environment to another.

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

**7.6.1 Adaptability:** The capability of the software to be modified for different specified environments without applying actions or means other than those provided for this purpose for the software considered.

### NOTES

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

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

**7.6.2 Installability:** The capability of the software to be installed in a specified environment.

NOTE - If the software is to be installed by an end user, installability will be a pre-requisite for operability.

**7.6.3 Co-existence:** The capability of the software to co-exist with other independent software in a common environment sharing common resources.

**7.6.4 Replaceability:** The capability of the software to be used in place of other specified software in the environment of that software.

#### NOTES

- 1 Replaceability is used in place of **compatibility** in order to avoid possible ambiguity with interoperability (see 7.1.3).
- 2 Replaceability does not imply that this software is able to replace the software under consideration.
- 3 Replaceability may include attributes of both installability and adaptability. The concept has been introduced as a sub-characteristic of its own because of its importance.

## 8. Quality in use

Quality in use is the user's view of the quality of a system containing software, and is measured in terms of the result of using the software, rather than properties of the software itself.

**8.1 quality in use:** the extent to which a product can be used by specified users to meet their needs to achieve specified goals with effectiveness, productivity and satisfaction in a specified context of use.

**8.1.1 effectiveness:** the accuracy and completeness with which users achieve specified goals

**8.1.2 productivity:** the resources expended in relation to task effectiveness

**8.1.3 satisfaction:** attitudes to the use of the product

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

NOTE - Quality in use is an external measure of the combination of functionality, usability and efficiency.

## Annex A (informative)

### Definitions from other standards

Definitions are from ISO/IEC 14598-1 unless otherwise stated.

#### A.1 Quality

**A.1.1 quality:** The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs. [ISO 8402]

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

**A.1.2 implied needs:** Needs that may not have been stated but are actual needs when the entity is used in particular conditions.

NOTE - Implied needs are real needs which may not have been documented.

**A.1.3 quality model:** The set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality.

**A.1.4 external quality:** The extent to which a product satisfies stated and implied needs when used under specified conditions.

**A.1.5 internal quality:** The totality of attributes of a product that determine its ability to satisfy stated and implied needs when used under specified conditions.

NOTE - The term “internal quality”, used in this standard to contrast with “external quality”, has essentially the same meaning as “quality” in ISO 8402. The term “attribute” is used as the term “characteristic” is used in a more specific sense in ISO/IEC 9126.

**A.1.6 quality in use:** The extent to which a product can be used by specified users to meet their needs to achieve specified goals with effectiveness, task efficiency and satisfaction in a specified context of use.

#### A.2 Software and user

**A.2.1 software:** All or part of the programs, procedures, rules, and associated documentation of an information processing system. (ISO/IEC 2382-1 : 1993)

NOTE - Software is an intellectual creation that is independent of the medium on which it is recorded.

**A.2.2 software product:** The set of computer programs, procedures, and possibly associated documentation and data designated for delivery to a user. [ISO/IEC 12207]

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

**A.2.3 fault:** An incorrect step, process or data definition in a computer programme. [IEEE 610.12-1990]

**A.2.4 user:** An individual that uses the software product to perform a specific function.

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

#### A.3 Measurement

**A.3.1 attribute:** A measurable physical or abstract property of an entity.

**A.3.2 measurement:** The process of assigning a number or category to an entity to describe an attribute of that entity.

NOTE - "category" is used to denote qualitative measures of attributes. For example, some important attributes of software products, e.g. the language of a source program (ADA, C, COBOL, etc.) are qualitative.

**A.3.3 measure (verb):** Make a measurement.

**A.3.4 measure (noun)** The number or category assigned to an attribute of an entity by making a measurement.

**A.3.5 direct measure:** A measure of an attribute that does not depend upon a measure of any other attribute.

**A.3.6 indirect measure:** A measure of an attribute that is derived from measures of one or more other attributes.

NOTE - An external measure of an attribute of a computing system (such as the response time to user input) is an indirect measure of attributes of the software as the measure will be influenced by attributes of the computing environment as well as attributes of the software.

**A.3.7 internal measure:** A measure derived from the product itself, either direct or indirect; it is not derived from measures of the behaviour of the system of which it is a part.

NOTE - Lines of code, complexity, the number of faults found in a walk through and the Fog Index are all internal measures made on the product itself.

**A.3.8 external measure:** An indirect measure of a product derived from measures of the behaviour of the system of which it is a part.

#### NOTES

1. The system includes any associated hardware, software and users.
2. The number of faults found during testing is an external measure of the number of faults in the program because the number of faults are counted during the operation of a computer system running the program to find the number of faults in the code.
3. External measures can be used to evaluate quality attributes closer to the ultimate objectives of the design.

**A.3.9 measurement scale:** A scale that constrains the type of data analysis that can be performed on it.

NOTE - Examples 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 possess an absolute zero.

**A.3.10 rating:** The action of mapping the measured value to the appropriate rating level. Used to determine the rating level associated with the software for a specific quality characteristic.

**A.3.11 rating level:** A scale point on an ordinal scale which is anchored to a range of values on another ordinal, interval or ratio scale measure.

#### NOTES

1. The rating level enables software to be classified (rated) in accordance with the stated or implied needs.
2. Appropriate rating levels may be associated with the different views of quality i.e. Users', Managers' or 'Developers'.
3. These rating levels are different from the "grades" defined in ISO 8402.

**A.3.12 indicator:** An indirect measure that can be used to estimate or predict another measure.

#### NOTES

1. The measure may be of the same or a different characteristic.
2. Indicators may be used both to estimate software quality attributes and to estimate attributes of the production process.

**A.3.13 metric:** A measurement scale and the method used for measurement.

#### NOTES

1. Metrics can be internal or external
2. Metrics include methods for categorising qualitative data.

#### A.4 Evaluation

**A.4.1 quality evaluation:** Systematic examination of the extent to which an entity is capable of fulfilling specified requirements. [ISO 8402]

**A.4.2 verification:** Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled.

##### NOTES

1. In design and development, verification concerns the process of examining the result of a given activity to determine conformity with the stated requirement for that activity.
2. "Verified" is used to designate the corresponding status.

[ISO 8402: 1994, 2.17]

**A.4.3 validation:** Confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled.

##### NOTES

1. In design and development, validation concerns the process of examining a product to determine conformity with user needs.
2. Validation is normally performed on the final product under defined operating conditions. It may be necessary in earlier stages.
3. "Validated" is used to designate the corresponding status.
4. Multiple validations may be carried out if there are different intended uses.

[ISO 8402: 1994, 2.18]



## **Annex B (informative)**

### **Bibliography**

IEEC 610.12-1990

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

ISO 8402 : 1994, *Quality -Vocabulary .*

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

ISO/IEC 14598-2:(new) Information Technology - Software product evaluation - Part 2: Planning and management

ISO/IEC 14598-3:(new) Information Technology - Software product evaluation - Part 3: Process for developers

ISO/IEC 14598-4:(new) Information Technology - Software product evaluation - Part 4: Process for acquirers

ISO/IEC 14598-5:(new) Information Technology - Software product evaluation - Part 5: Process for evaluators

ISO/IEC 14598-6:(new) Information Technology - Software product evaluation - Part 6: Documentation of evaluation modules

## **Annex C (informative)**

### **History of the work**

#### **C.1 Background**

The software industry is entering a period of some maturing, while at the same time software is becoming a crucial component of many of today's products. This pervasive aspect of software makes it a major new factor in trade. Furthermore, with new global demands for safety and quality, the need for international agreements on software quality assessment procedures is becoming important.

There are essentially two approaches that can be followed to ensure product quality, one being assurance of the process by which the product is developed, and the other being the evaluation of the quality of the end product. Both avenues are important and both require the presence of a system for managing quality. Such a system identifies the management commitment to quality, and states its policies, as well as the detailed steps that must be in place.

To evaluate the quality of a product through some quantitative means, a set of quality characteristics that describe the product and form the basis for the evaluation is required. This part of ISO/IEC 9126 defines these quality characteristics for software products.

#### **C.2 History**

State of the art in software technology does not yet present a well established and widely accepted description scheme for assessing the quality of software product. Much work has been done since about 1976 by a number of individuals to define a software quality framework. Models by McCall, Boehm, the US Air Force, and others have been adopted and enhanced over the years. However, today it is difficult for a user or consumer of software products to understand or compare the quality of software.

For a long time, reliability has been the only way to gauge quality. Other quality models have been proposed and submitted for use. While studies were useful, they also caused confusion because of the many quality aspects offered. Thus, the need for one standard model came about.

It is for this reason that the ISO/IEC JTC1 began to develop the required consensus and encourage standardization worldwide.

First considerations originated in 1978, and in 1985 the development of ISO/IEC 9126 was started. The models proposed initially introduced properties of software that depend on application or implementation aspects (or both), to describe the quality of software.

The first step of the ISO technical committee to arrange these properties systematically failed for lack of definitions. Terms were interpreted in different ways by experts. All structures discussed were, therefore, of an arbitrary nature, without a common basis.

As a result it was decided that the best chance for establishing an International Standard was to stipulate a set of characteristics based on a definition of quality that that was subsequently used in ISO 8402. This definition is accepted for all kinds of products and services. It starts with the user's needs.

#### **C.3 Six ISO software quality characteristics**

The requirements for choosing the characteristics described in ISO/IEC 9126 were as follows:

- To cover together all aspects of software quality resulting from the ISO quality definition.
- To describe the product quality with a minimum of overlap.
- To be as close as possible to the established terminology.

- To form a set of not more than six to eight characteristics for reason of clarity and handling.
- To identify areas of attributes of software products for further refinement.

The work of the technical committee resulted in the above set of characteristics.

However, a pure terminology standard, containing definitions of characteristics would not have provided sufficient support to users in assessing software quality. Therefore, a description on how to proceed with evaluating the quality of a software product was included.

Evaluating product quality in practice requires characteristics beyond the set at hand, and requires metrics for each of the characteristics. The state of art at present did not permit standardization in this area. Waiting for enhancements would have delayed the publication of ISO/IEC 9126 substantially.

For this reason, the technical committee issued the 1991 version of ISO/IEC 9126 to harmonize further development.

#### **C.4 Revision of ISO/IEC 9126**

In 1994 it was felt that other Standards being produced in the area of product quality evaluation necessitated the revision of ISO/IEC 9126. The revision retains the same 6 quality characteristics, but clarifies their relationship to internal and external metrics. The relationship between the characteristics and quality in use is also explained.

Quality is defined in ISO 8402 in terms of "Totality of characteristics of an entity that bear on ...". Note 4 in this definition states that "The term quality should not be used as a single term to express a degree of excellence in a comparative sense". For this reason the terms "internal quality" and "external quality" have been defined in ISO/IEC 14598-1 to refer to aspects of quality which can be measured. The wording of the definitions of the quality characteristics has been changed from: "A set of attributes that bear on" to: "The capability of the software to ..." so they can be interpreted in terms which enable both internal and external quality to be measured.

Sub-characteristics have been introduced, based on those in the informative annex of the previous version of ISO/IEC 9126. Conformance and compliance were removed as subcharacteristics, as the principles are generally applicable to all the software characteristics.

The evaluation process model has been moved to ISO/IEC 14598-1. Two new technical reports are being prepared as parts 2 and 3 of ISO/IEC 9126, giving examples of external and internal metrics.