ISO/IEC JTC1/SC7

Software Engineering

Secretariat: CANADA (SCC)

ISO/IEC JTC1 /SC7 N2416R

Date: 2002-03-15

Reference number of document: **ISO/IEC TR 9126-3**

Committee identification: ISO/IEC JTC1 /SC 7/WG 6

Secretariat: Japan

Software engineering –Product quality – Part 3: Internal metrics

Titre — Titre — Partie n: Titre

Document type: International technical report
Document subtype: if applicable
Document stage: (40) Enquiry
Document language: E

ISO Basic template Version 3.0 1997-02-03

ISO/IEC 9126-3: Software engineering - Product quality - Part 3: Internal metrics

ISO/IEC JTC 1/SC 7 N2416R

TR

ISO/IEC JTC 1/SC 7/WG 6

Date:15-03-2002 (Final editorial correction version of Approved DTR Balloted 7N2416 in 2001 for ISO/IEC publish)

Document type: Technical Report Type 2

Secretariat: ISO/IEC JTC 1/SC 7

Document subtype: Not applicable

Document language: E

Document stage: (20) Preparatory

ISO/IEC 9126-3: Software engineering – Product quality – Part 3: Internal metrics

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for world-wide 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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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 Technical Report ISO/IEC 9126-3 was prepared by the Joint Technical Committee ISO/IEC JTC1, Information Technology, Subcommittee SC7, Software Engineering

ISO/IEC 9126 consists of the following parts under the general title *Software Engineering - Product quality*

Part 1: Quality model

Part 2: External Metrics

Part 3: Internal Metrics

Part 4: Quality in use metrics

Annex A through annex E are for information only.

Introduction

This International Technical Report provides external metrics for measuring attributes of six external quality characteristics defined in ISO/IEC 9126-1. The metrics listed in this International Technical Report are not intended to be an exhaustive set. Developers, evaluators, quality managers and acquirers may select metrics from this technical report for defining requirements, evaluating software products, measuring quality aspects and other purposes. They may also modify the metrics or use metrics which are not included here. This report is applicable to any kind of software product, although each of the metrics is not always applicable to every kind of software product.

ISO/IEC 9126-1 defines terms for the software quality characteristics and how these characteristics are decomposed into subcharacteristics. ISO/IEC 9126-1, however, does not describe how any of these subcharacteristics could be measured. ISO/IEC 9126-2 defines external metrics, ISO/IEC 9126-3 defines internal metrics and ISO/IEC 9126-4 defines quality –in use metrics, for measurement of the characteristics or the subcharacteristics. Internal metrics measure the software itself, external metrics measure the behaviour of the computer-based system that includes the software, and quality in use metrics measure the effects of using the software in a specific context of use.

This International Technical Report is intended to be used together with ISO/IEC 9126-1. It is strongly recommended to read ISO/IEC 14598-1 and ISO/IEC 9126-1, prior to using this International Technical Report, particularly if the reader is not familiar with the use of software metrics for product specification and evaluation.

The clauses 1 to 7 and annexes A to D are common to ISO/IEC 9126-2 , ISO/IEC 9126-3 , and ISO/IEC 9126-4. The annex E is for ISO/IEC 9126-3 use.

Software engineering – Product quality –

Part 3: Internal metrics

1. Scope

This International Technical Report defines external metrics for quantitatively measuring external software quality in terms of characteristics and subcharacteristics defined in ISO/IEC 9126-1, and is intended to be used together with ISO/IEC 9126-1.

This International Technical Report contains:

I. an explanation of how to apply software quality metrics

II. a basic set of metrics for each subcharacteristic

III. an example of how to apply metrics during the software product life cycle

This International Technical Report does not assign ranges of values of these metrics to rated levels or to grades of compliance, because these values are defined for each software product or a part of the software product, by its nature, depending on such factors as category of the software, integrity level and users' needs. Some attributes may have a desirable range of values, which does not depend on specific user needs but depends on generic factors; for example, human cognitive factors.

This International Technical Report can be applied to any kind of software for any application. Users of this International Technical Report can select or modify and apply metrics and measures from this International Technical Report or may define application-specific metrics for their individual application domain. For example, the specific measurement of quality characteristics such as safety or security may be found in International Standard or International Technical Report provided by IEC 65 and ISO/IEC JTC1/SC27.

Intended users of this International Technical Report include:

Acquirer (an individual or organization that acquires or procures a system, software product or software service from a supplier);

Evaluator (an individual or organization that performs an evaluation. An evaluator may, for example, be a testing laboratory , the quality department of a software development organization, a government organization or an user);

Developer (an individual or organization that performs development activities, including requirements analysis, design, and testing through acceptance during the software life cycle process);

Maintainer (an individual or organization that performs maintenance activities);

Supplier (an individual or organization 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) when validating software quality at qualification test;

User (an individual or organization that uses the software product to perform a specific function) when evaluating quality of software product at acceptance test;

Quality manager (an individual or organization that performs a systematic examination of the software product or software services) when evaluating software quality as part of quality assurance and quality control.

2. Conformance

There are no conformance requirements in this TR.

Note: General conformance requirements for metrics are in ISO/IEC 9126-1 Quality Model.

3. References

1. ISO 8402: 1994, Quality management and quality assurance – Quality vocabulary
2. ISO/IEC 9126: 1991, Software engineering – Software product evaluation – Quality characteristics and guidelines for their use
3. ISO/IEC 9126-1(new): Software engineering – Product quality - Part 1: Quality model
4. ISO/IEC TR 9126-3(new): Software engineering – Product quality - Part 3: Internal metrics
5. ISO/IEC TR 9126-4(new): Software engineering – Product quality - Part 4: Quality in use metrics
6. ISO/IEC 14598-1: 1999, Information technology – Software product evaluation - Part 1: General overview
7. ISO/IEC 14598-2: 2000, Software engineering – Product evaluation - Part 2: Planning and management
8. ISO/IEC 14598-3: 2000, Software engineering - Product evaluation - Part 3: Process for developers
9. ISO/IEC 14598-4: 1999, Software engineering - Product evaluation - Part 4: Process for acquirers
10. ISO/IEC 14598-5: 1998, Information technology - Software product evaluation - Part 5: Process for evaluators
11. ISO/IEC 14598-6 (new): Software engineering - Product evaluation - Part 6: Documentation of evaluation modules
12. ISO/IEC 12207: 1995, Information technology – Software life cycle processes.
13. ISO/IEC 14143-1 1998, Functional size measurement Part 1.
14. ISO 2382-20:1990, Information technology, vocabulary
15. ISO 9241-10 (1996) , Ergonomic requirements for office work with visual display terminals (VDTs) – Part 10; Dialogue principles

4. Terms and Definitions

For the purposes of this ISO/IEC TR 9126-3 International Technical Report, the definitions contained in ISO/IEC 14598-1 and ISO/IEC 9126-1 apply. They are also listed in annex D.

5. Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this International Technical Report:

1. SQA - Software Quality Assurance (Group)
2. SLCP – Software Life Cycle Processes

6. Use of Software Quality Metrics

These International Technical Reports (ISO/IEC 9126-2 External metrics, ISO/IEC 9126-3 Internal metrics and ISO/IEC 9126-4 Quality in use metrics) provides a suggested set of software quality metrics (external, internal and quality in use metrics) to be used with the ISO/IEC 9126-1 Quality model. The user of these technical reports may modify the metrics defined, and/or may also use metrics not listed. When using a modified or a new metric not identified in these International Technical Reports, the user should specify how the metrics relate to the ISO/IEC 9126-1 quality model or any other substitute quality model that is being used.

The user of these International Technical Reports should select the quality characteristics and subcharacteristics to be evaluated, from ISO/IEC 9126-1; identify the appropriate direct and indirect measures, identify the relevant metrics and then interpret the measurement result in a objective manner. The user of these International Technical Reports also may select product quality evaluation processes during the software life cycle from the ISO/IEC 14598 series of standards. These give methods for measurement, assessment and evaluation of software product quality. They are intended for use by developers, acquirers and independent evaluators, particularly those responsible for software product evaluation (see Figure 1).

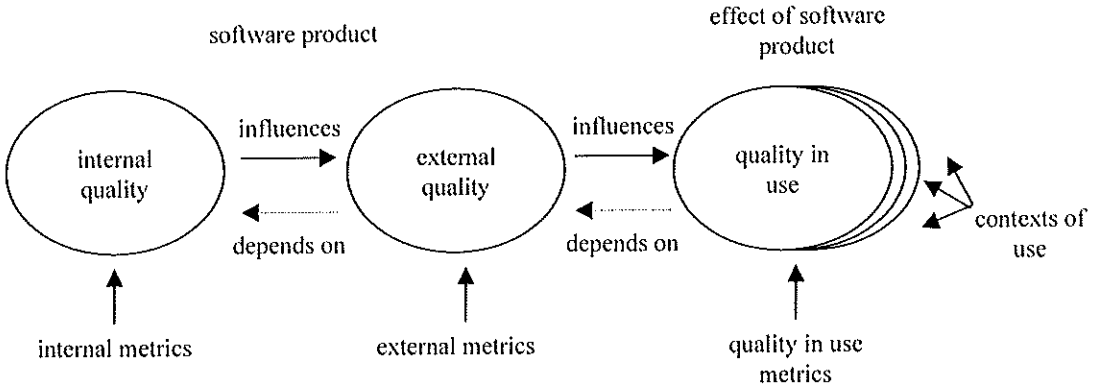

Figure 1 – Relationship between types of metrics

The internal metrics may be applied to a non-executable software product during its development stages (such as request for proposal, requirements definition, design specification or source code). Internal metrics provide the users with the ability to measure the quality of the intermediate deliverables and thereby predict the quality of the final product. This allows the user to identify quality issues and initiate corrective action as early as possible in the development life cycle.

The external metrics may be used to measure the quality of the software product by measuring the behaviour of the system of which it is a part. The external metrics can only be used during the testing stages of the life cycle process and during any operational stages. The measurement is performed when executing the software product in the system environment in which it is intended to operate.

The quality in use metrics measure whether a product meets the needs of specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in a specified context of use. This can be only achieved in a realistic system environment.

User quality needs can be specified as quality requirements by quality in use metrics, by external metrics, and sometimes by internal metrics. These requirements specified by metrics should be used as criteria when a product is evaluated.

It is recommended to use internal metrics having a relationship as strong as possible with the target external metrics so that they can be used to predict the values of external metrics. However, it is often difficult to design a rigorous theoretical model that provides a strong relationship between internal metrics and external metrics. Therefore, a hypothetical model that may contain ambiguity may be designed and the extent of the relationship may be modelled statistically during the use of metrics.

Recommendations and requirements related to validity and reliability are given in ISO/IEC 9126-1, clause A.4. Additional detailed considerations when using metrics are given in Annex A of this International Technical Report.

7. How to read and use the metrics tables

The metrics listed in clause 8 are categorised by the characteristics and subcharacteristics in ISO/IEC 9126-1. The following information is given for each metric in the table:

a) **Metric name:** Corresponding metrics in the internal metrics table and external metrics table have similar names.

b) **Purpose of the metric:** This is expressed as the question to be answered by the application of the metric.

c) **Method of application:** Provides an outline of the application.

) **Measurement, formula and data element computations:** Provides the measurement formula and explains the meanings of the used data elements.

NOTE: In some situations more than one formula is proposed for a metric..

e) **Interpretation of measured value:** Provides the range and preferred values.

f) **Metric scale type:** Type of scale used by the metric. Scale types used are; Nominal scale, Ordinal scale, Interval scale, Ratio scale and Absolute scale.

NOTE: A more detailed explanation is given in annex C.

g) **Measure type:** Types used are; Size type (e.g. Function size, Source size) , Time type (e.g. Elapsed time, User time) , Count type (e.g. Number of changes, Number of failures).

NOTE: A more detailed explanation is given in Annex C.

h) **Input to measurement:** Source of data used in the measurement.

ı) **ISO/IEC 12207 SLCP Reference:** Identifies software life cycle process(es) where the metric is applicable.

j) **Target audience:** Identifies the user(s) of the measurement results.

8. Metrics Tables

The metrics listed in this clause are not intended to be an exhaustive set and may not have been validated. They are listed by software quality characteristics and subcharacteristics, in the order introduced in ISO/IEC 9126-1.

Metrics, which may be applicable, are not limited to these listed here. Additional specific metrics for particular purposes are provided in other related documents, such as functional size measurement or precise time efficiency measurement.

NOTE: It is recommended to refer a specific metric or measurement form from specific standards, technical reports or guidelines. Functional size measurement is defined in ISO/IEC 14143. An example of precise time efficiency measurement can be referred from ISO/IEC 14756.

Metrics should be validated before application in a specific environment (see Annex A).

NOTE: This list of metrics is not finalised, and may be revised in future versions of this International Technical Report. Readers of this International Technical Report are invited to provide feedback.

8.1 Functionality metrics

Internal functionality metrics are used for predicting if the software product in question will satisfy prescribed functional requirements and implied user needs.

8.1.1 Suitability metrics

Internal suitability metrics indicate a set of attributes for assessing explicitly functions to prescribed tasks, and for determining their adequacy for performing the tasks.

8.1.2 Accuracy metrics

Internal accuracy metrics indicate a set of attributes for assessing the capability of the software product to achieve correct or agreeable results.

8.1.3 Interoperability metrics

Internal Interoperability metrics indicate a set of attributes for assessing the capability of the software product's interaction with designated systems.

8.1.4 Security metrics

Internal security metrics indicate a set of attributes for assessing the capability of the software product to avoid illegal access to the system and/or data.

8.1.5 Functionality compliance metrics

Internal compliance metrics indicate a set of attributes for assessing the capability of the software product to comply to such items as standards, conventions or regulations of the user organisation in relation to functionality.

Table 8.1.1 Suitability metrics

Internal suitability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Sources of input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional adequacy	How adequate are the checked functions?	Count the number of implemented functions that are suitable for performing the specified tasks, then measure the ratio of it to functions implemented. The following may be measured; -all or parts of design specifications -completed modules/parts of software products	X=1-A/B A= Number of functions in which problems are detected in evaluation B= Number of functions checked	0 <= X <= 1 The closer to 1, the more adequate.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers
Functional implementation completeness	How complete is the functional implementation?	Count the number of missing functions detected in evaluation and compare with the number of function described in the requirement specifications	X=1-A/B A=Number of missing functions detected in evaluation. B=Number of functions described in requirement specifications ***NOTE**: Input to the measurement process is the updated requirement specifications. Any changes identified during life cycle must be applied to the requirement specifications before using in measurement process.*	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers

Internal suitability metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Sources of input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional implementation coverage	How correct is the functional implementation?	Count the number of incorrectly implemented or missing functions and compare with the number of functions described in the requirement specifications ***Note:*** *Review by functional item.*	X=1-A/B A= Number of incorrectly implemented or missing functions detected. B= Number of functions described in requirement specifications ***Note:*** *Input to the measurement process is the updated requirement specifications. Any changes identified during life cycle must be applied to the requirement specifications before using in measurement process.*	0 <= X <= 1 The closer to 1, the more correct.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers
Functional specification stability (volatility)	How stable is the functional specification during the development life cycle?	Count the number of functions changed (added, modified, or deleted) during development life cycle phase, then compare with the number of functions described in the requirement specifications.	X=1-A/B A=Number of functions changed during development life cycle phases B=Number of functions described in requirement specifications	0 <= X <= 1 The closer to 1 the more stable.	absolute	A=Count B=Count X=Count/Count	Requirement specifications Review report	6.5 Validation 6.3 Quality Assurance 5.3 Qualification testing 6.8 Problem Resolution 5.4 Operation	Developers Maintainers

Table 8.1.2 Accuracy metrics

Internal accuracy metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Computational Accuracy	How completely have the accuracy requirements been implemented?	Count the number of functions that have implemented the accuracy requirements and compare with the number of functions with specific accuracy requirements.	X=A/B A= Number of functions in which specific accuracy requirements had been implemented, as confirmed in evaluation. B= Number of functions for which specific accuracy requirements need to be implemented.	0 <= X <= 1. The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	Verification Joint review	Requirers Developers
Precision	How complete was the implementation of specific levels of precision for the data items?	Count the number of data items that meet the requirements of specific levels of precision and compare to the total number of data items with specific level of precision requirements.	X=A/B A= Number of data items implemented with specific levels of precision, confirmed in evaluation B= Number of data items that require specific levels of precision	0 <= X <= 1. The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	Verification Joint review	Requirers Developers

Table 8.1.3 Interoperability metrics

Internal interoperability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Data exchangeability (Data format based)	How correctly have the interface data formats been implemented?	Count the number of interface data formats that have been implemented correctly as in the specifications and compare to the number of data formats to be exchanged as in the specifications.	X=A/B A=Number of interface data formats that have been implemented correctly as in the specifications B=Number of data formats to be exchanged as in the specifications	0 <= X <= 1. The closer to 1, the more correct.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	Verification Joint review	Developers Requirers
Interface consistency (protocol)	How correctly have the interface protocols been implemented?	Count the number of interface protocols that were implemented correctly as in the specifications and compare with the number of interface protocols to be implemented as in the specifications.	X=A/B A=Number of interface protocols implementing consistent format as in the specification confirmed in review B=Number of interface protocols to be implemented as in the specifications	0 <= X <= 1 The closer to 1, the more consistent.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	Verification Joint review	Developers Requirers

Table 8.1.4 Security metrics

Internal security metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Access auditability	How auditable is access login?	Count the number of access types that are being logged correctly as in the specifications and compare with the number of access types that are required to be logged in the specifications.	X=A/B A= Number of access types that are being logged as in the specifications B= Number of access types required to be logged in the specifications	0 <= X <= 1 The closer to 1, the more auditable.	Absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers
Access controllability	How controllable is access to the system?	Count the number of access controllability requirements implemented correctly as in the specifications and compare with the number of access controllability requirements in the specifications.	X=A/B A= Number of access controllability requirements implemented correctly as in the specifications. B= Number of access controllability requirements in the specifications..	0 <= X <= 1 The closer to 1, the more controllable.	Absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation 6.6 Joint review	Requirers Developers

Internal security metrics

Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Data corruption prevention	How complete is the implementation of data corruption prevention?	Count the number of implemented instances of data corruption prevention as specified and compare with the number of instances of operations/ access specified in requirements as capable of corrupting/ destroying data.	X=A/B A= Number of implemented instances of data corruption prevention as specified confirmed in review. B= Number of instances of operation/access identified in requirements as capable of corrupting/destroying data Note: Consider security levels when using this metric.	0 <= X <= 1 The closer to 1, the more complete.	Absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation 6.6 Joint review	Developers
Data encryption	How complete is the implementation of data encryption?	Count the number of implemented instances of encryptable/decryptable data items as specified and compare with the number of instances of data items requiring data encryption/decryption facility as in specifications.	X=A/B A=Number of implemented instances of encryptable/decryptable data items as specified confirmed in review B= Number of data items requiring data encryption/decryption facility as in specifications ***NOTE:*** *Data encryption: e.g., data in open database, data in public communication facility*	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Requirement specification Design Source code Review report	6.5 Validation	Developers

Table 8.1.5 Functionality compliance metrics

Internal functionality compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Functional compliance	How compliant is the functionality of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification.	X=A/B A= Number of correctly implemented items related to functionality compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1. The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers
Intersystem standard compliance	How compliant are the interfaces to applicable regulations, standards and conventions	Count the number of interfaces that meet required compliance and compare with the number of interfaces requiring compliance as in the specifications ***Note:*** *All specified attributes of a standard must be checked*	X=A/B A= Number of correctly implemented interfaces as specified, confirmed in review B= Total number of interfaces requiring compliance	0 <= X <= 1. The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Req spec Design Source code Review report	Verification Joint review	Developers Requirers

8.2 Reliability metrics

Internal reliability metrics are used for predicting if the software product in question will satisfy prescribed reliability needs, during the development of the software product.

8.2.1 Maturity metrics

Internal maturity metrics indicate a set of attributes for assessing the maturity of the software.

8.2.2 Fault tolerance metrics

Internal fault tolerance metrics indicate a set of attributes for assessing the software products capability in maintaining a desired performance level in case of operational faults or infringement of its specified interface.

8.2.3 Recoverability metrics

Internal recoverability metrics indicate a set of attributes for assessing the software product's capability to re-establish an adequate level of performance and recover the data directly affected in case of a failure.

8.2.4 Reliability compliance metrics

Internal compliance metrics relating to reliability indicate a set of attributes for assessing the capability of the software product to comply to such items as standards, conventions or regulations of the user organisation in relation to reliability.

Table 8.2.1 Maturity metrics

Internal maturity metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Fault detection ***Note:*** *this metric should only be used for prediction during development.*	How many faults were detected in reviewed product?	Count the number of detected faults in review and compare it to the number of estimated faults to be detected in this phase.	X=A/B A=Absolute number of faults detected in review B=Number of estimated faults to be detected in review (using past history or reference model)	0 <= X A high value for X implies good product quality, while A=0 does not necessarily imply fault free status of the reviewed item. ***NOTE:*** *1. It is necessary to convert this value(X) to the <0,1> interval if making summarization of characteristics.*	Absolute	X=count/count A=count B=count	Value A comes from review report Value B comes from the organizatio n database.	Verification Joint review	Requirers Developers

Internal maturity metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Fault removal	How many faults have been corrected? What is the proportion of faults removed?	Count the number of faults removed during design/coding and compare it to the number of faults detected in review during design/coding.	X=A A=Number of corrected faults in design/coding Y=A/B A=Number of corrected faults design/coding B= Number of faults detected in review	0 <= X A high value of X implies, that less faults remain. 0 <= Y <= 1 The closer to 1, the better. (more faults removed) ***NOTE:*** *1. It is necessary to convert this value (X) to the <0,1> interval if making summarization of characteristics.*	ratio absolute	X=count A=count Y=count/count B=count	Value A comes from fault removal report. Value B comes from review report.	Verification Joint review	Requirers Developers
Test adequacy	How much of the required test cases are covered by the test plan?	Count the number of test cases planned and compare it to the number of test cases required to obtain adequate test coverage.	X=A/B A=Number of test cases designed in test plan and confirmed in review B= Number of test cases required	0 <= X Where X is greater the better adequacy	absolute	X=count/count A=count B=count	Value A comes from test plan Value B comes from requirements	QA Problem resolution Verification	Developers Maintainers

Table 8.2.2 Fault tolerance metrics

Internal fault tolerance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Failure avoidance	How many fault patterns were brought under control to avoid critical and serious failures?	Count the number of avoided fault patterns and compare it to the number of fault patterns to be considered	X=A/B A=Number of fault patterns having avoidance in design/code B=Number of fault patterns to be considered ***NOTE:*** *Fault pattern examples* *out of range data* *deadlock* ***NOTE:*** *Fault tree analysis technique may be used to detect fault patterns.*	0 <= X Where X is greater the better failure avoidance	absolute	X=count/count A=count B=count	Value A comes from review report Value B comes from requirement specification document.	Verification Validation Joint review Problem resolution	Developers Requirers Maintainers
Incorrect operation avoidance	How many functions are implemented with incorrect operations avoidance capability?	Count the number of implemented functions to avoid critical and serious failures caused by incorrect operations and compare it to the number of incorrect operation patterns to be considered. ***NOTE:*** *Also data damage in addition to system failure.*	X=A/B A=Number of functions implemented to avoid incorrect operation patterns. B=Number of incorrect operation patterns to be considered ***NOTE:*** *Incorrect operation patterns* *Incorrect data types as parameters* *Incorrect sequence of data input* *Incorrect sequence of operation* ***NOTE:*** *Fault tree analysis technique may be used to detect incorrect operation patterns.*	0 <= X Where X is greater the better incorrect operation avoidance.	absolute	X=count/count A=count B=count	Value A comes from review report Value B comes from requirement specification document.	Verification Validation Joint review Problem resolution	Developers Requirers Maintainers

Table 8.2.3 Recoverability metrics

Internal recoverability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Restorability	How capable is the product in restoring itself after abnormal event or at request?	Count the number of implemented restoration requirements and compare it to the number of restoration requirements in the specifications. Restoration requirement examples: database checkpoint, transaction checkpoint, redo function, undo function	X=A/B A=Number of implemented restoration requirements confirmed in review B=Number of restoration requirements in the specifications..	0 <= X <= 1 Where X is greater, the better restorability	Absolute	X=count/count A=count B=count	A comes from review document B comes from requirements or design document	Verification Joint review	Developers Maintainers
Restoration Effectiveness	How effective is the restoration capability?	Count the number of implemented restoration requirements meting target restoration time (by calculations or simulations) and compare it to the number of restoration requirements with specified target time.	X=A/B A=Number of implemented restoration requirements meeting target restore time B=Number of restoration requirements with specified target times	0 <= X <= 1 Where X is greater, the better effectiveness	Absolute	X=count/count A=count B=count	A comes from review document B comes from requirements or design document.	Verification Joint review	Developers Maintainers

Table 8.2.4 Reliability compliance metrics

Internal reliability compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Reliability compliance	How compliant is the reliability of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification..	X=A/B A= Number of correctly implemented items related to reliability compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1. The closer to 1, the more compliant.	Absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers

8.3 Usability Metrics

Internal usability metrics are used for predicting the extent to which the software in question can be understood, learned, operated, attractive and compliant with usability regulations and guidelines.

It should be possible for the measures taken to be used to establish acceptance criteria or to make comparisons between products. This means that the measures should be counting items of known value. Results should report the mean value and the standard error of the mean

8.3.1 Understandability metrics

Users should be able to select a software product which is suitable for their intended use. Internal understandability metrics assess whether new users can understand:

- whether the software is suitable
- how it can be used for particular tasks.

8.3.2 Learnability metrics

Internal learnability metrics assess how long users take to learn how to use particular functions, and the effectiveness of help systems and documentation.

Learnability is strongly related to understandability, and understandability measurements can be indicators of the learnability potential of the software.

8.3.3 Operability metrics

Internal operability metrics assess whether users can operate and control the software. Operability metrics can be categorised by the dialogue principles in ISO 9241-10:

- suitability of the software for the task
- self-descriptiveness of the software
- controllability of the software
- conformity of the software with user expectations
- error tolerance of the software
- suitability of the software for individualisation

The choice of functions to test will be influenced by the expected frequency of use of functions, the criticality of the functions, and any anticipated usability problems.

8.3.4 Attractiveness metrics

Internal attractiveness metrics assess the appearance of the software, and will be influenced by factors such as screen design and colour. This is particularly important for consumer products.

8.3.5 Usability compliance metrics

Internal compliance metrics assess adherence to standards, conventions, style guides or regulations relating to usability.

Table 8.3.1 Understandability metrics

Internal understandability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Completeness of description	What proportion of functions (or types of function) are described in the product description?	Count the number of functions which are adequately described and compare with the total number of functions in the product.	X= A/B A= Number of functions (or types of functions) described in the product description B= Total number of functions (or types of functions)	0<=X<=1 The closer to 1 the more complete	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
NOTE 1: This indicates whether potential users will understand the capability of the product after reading the product description. *NOTE 2: See also ISO/IEC 9127 Consumer software package.*									
Demonstration capablity	What proportion of functions requiring demonstration have demonstration capability?	Count the number of functions that are adequately demonstrable and compare with the total number of functions requiring demonstration capability	X=A/B A= Number of functions demonstrated and confirmed in review B= Total number of functions requiring demonstration capability	0<=X<=1 The closer to 1 the more capable.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
NOTE: Demonstrations step through the process showing how the product is used. This includes "wizards".									
Evident functions	What proportion of the product functions are evident to the user?	Count the number of functions that are evident to the user and compare with the total number of functions	X= A/B A= Number of functions (or types of functions) evident to the user B= Total number of functions (or types of functions)	0<=X<=1 The closer to 1 the better	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
NOTE: This indicates whether users will be able to locate functions by exploring the interface (e.g. by inspecting the menus)									
Function understandability	What proportion of the product functions will the user be able to understand correctly.	Count the number of user interface functions where purposes is understood by the user and compare with the number of user interface functions.	X= A/B A= Number of user interface functions whose purpose is understood by the user B= Number of user interface functions.	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers

Table 8.3.2 Learnability metrics

Internal learnability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Completeness of user documentation and/or help facility	What proportion of functions are described in the user documentation and/or help facility?	Count the number of functions implemented with help facility and/or documentation and compare with the total number of functions in product.	X= A/B A= Number of functions described B= Total of number of functions provided	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers

***NOTE:** : Three metrics are possible: completeness of the documentation, completeness of the help facility or completeness of the help and documentation used in combination.*

Table 8.3.3 Operability metrics

Internal Operability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Input validity checking	What proportion of input items provide check for valid data	Count the number of input items, which check for valid data and compare with the number of input items, which could check for valid data	X=A/B A=Number of input items which check for valid data B=Number of input items which could check for valid data	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
User operation cancellability	What proportion of functions can be cancelled prior to completion?	Count the number of implemented functions, which can be cancelled by the user prior to completion and compare it with the number of functions requiring the precancellation capability	X=A/B A=Number of implemented functions which can be cancelled by the user B= Number of functions requiring the precancellation capability	0 <= X <= 1 The closer to 1, the better cancellability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
User operation Undoability	What proportion of functions can be undone?	Count the number of implemented functions, which can be undone by the user after completion and compare it with the number of functions	X=A/B A=Number of implemented functions which can be undone by the user B= Number of functions.	0 <= X <= 1 The closer to 1, the better undoability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
NOTE: : Either single undoability or multiple undoability after several subsequent actions can be assessed									
Customisability	What proportion of functions can be customised during operation?	Count the number of implemented functions, which can be customized by the user during operation and compare it with the number of functions requiring the customization capability	X=A/B A=Number of functions which can be customised during operation B=Number of functions requiring the customization capability	0 <= X <= 1 The closer to 1, the better customisability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers

Physical accessibility	What proportion of functions can be customised for access by users with physical handicaps	Count the number of implemented functions, which can be customised and compare it with the number of functions	X=A/B A=Number of functions which can be customised B=Number of functions	0 <= X <= 1 The closer to 1, the better physical accessibility	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
NOTE: Examples of physical accessibility are inability to use a mouse and blindness									
Operation status monitoring capability	What proportion of functions have operations status monitoring capability?	Count the number of implemented functions, which status can be monitored and compare it with the number of functions requiring the monitoring capability.	X=A/B A=Number of functions having status monitoring capability B=Number of functions that are required to have monitoring capability.	0 <= X <= 1 The closer to 1, the better monitoring capability	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
NOTE: : Status includes progress monitoring.									
Operational consistency	What proportion of operations behave the same way to similar operations in other parts of the system?	Count the number of instances of operations with inconsistent behaviour and compare it with the total number of operations	X=1 - A/B A=Number of instances of operations with inconsistent behaviour B=Total number of operations	0 <= X <= 1 The closer to 1, the more consistent	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
Message Clarity	What proportion of messages are self-explanatory?	Count the numbers of implemented messages with clear explanations and compare it with the total number of messages implemented.	X=A/B A=Number of implemented messages with clear explanations. . B=Number of messages implemented	0 <= X <= 1 The closer to 1, the more clear.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers
NOTE: : Clear error messages explain to the user what action to take to recover from the error									
Interface element clarity	What proportion of interface elements are self-explanatory?	Count the number of interface elements which are self explanatory and compare it with the total number of interface elements	X=A/B A=Number of interface elements which are self-explanatory. B=Total number of interface elements	0 <= X <= 1 The closer to 1, the more clear.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers

NOTE: : Elements are self explanatory when they use plain text or provide "hover-help" or "tool tips"

Operational error recoverability	What proportion of functions can tolerate user error?	Count the number of functions implemented with user error tolerance and compare it to the total number of functions requiring the tolerance capability	X=A/B A=Number of functions implemented with user error tolerance B=Total number of functions requiring the tolerance capability	0 <= X <= 1 The closer to 1, the more recoverable.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Requirers

Table 8.3.4 Attractiveness metrics

Internal attractiveness metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Attractive interaction	How attractive is the interface to the user?	Questionnaire to users	Questionnaire to assess the attractiveness of the interface to users, taking account of attributes such as colour and graphical design. NOTES: Issues that potentially contribute to attractiveness include: Alignment of items (vertical and Horizontal), Grouping, Use of colours, Appropriate and reasonable sized graphics, Use of whitespace/separators/borders, Animation, Typography, and 3D interface.	Assessment classification	Ordinal	X= Count (Count is a score)	Req spec Design Review report	Verification Joint review	Requirers Developers
***NOTE:** This could be based on screen sketches or mock-ups*									
User Interface appearance customisability	What proportion of user interface elements can be customised in appearance.	Inspection (by expert)	X=A/B A=Number of types of interface elements that can be customised. B=Total number of types of interface elements.	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Requirers Developers
NOTE:									

Table 8.3.5 Usability compliance metrics

Internal usability compliance metrics									
Metric name	Purpose	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Usability compliance	How compliant is the product to applicable regulations, standards and conventions for usability	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification..	X=A/B A= Number of correctly implemented items related to usability compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1 The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers
NOTE:									

8.4 Efficiency metrics

Internal efficiency metrics are used for predicting the efficiency of behavior of the software product during testing or operating. To measure efficiency, the stated conditions should be defined, i.e., the hardware configuration and the software configuration of a reference environment (which has to be defined in the software specifications) should be defined. When citing measured time behavior values the reference environment should be referred.

8.4.1 Time behaviour metrics

Internal time behavior metrics indicate a set of attributes for predicting the time behavior of the computer system including the software product during testing or operating.

8.4.2 Resource utilisation metrics

Internal resource utilization metrics indicate a set of attributes for predicting the utilization of hardware resources by the computer system including the software product during testing or operating.

8.4.3 Efficiency compliance metrics

Internal compliance metrics relating to efficiency indicate a set of attributes for assessing the capability of the software product to comply to such items as standards, conventions or regulations of the user organisation in relation to efficiency

Table 8.4.1 Time behaviour metrics

Internal time behaviour metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Response time	What is the estimated time to complete a specified task?	Evaluate the efficiency of the operating system and the application system calls. Estimate the response time based on this. The following may be measured, -all or parts of design specifications -test complete transaction path -test complete modules/parts of software product -complete software product during test phase	X=time (calculated or simulated)	The shorter the better.	ratio	X=time	Known operating system. Estimated time in system calls.	Verification Joint review	Developers Requirers
Throughput time	What is the estimated number of tasks that can be performed over a unit of time?	Evaluate the efficiency of handling resources in the system. Make a factor based upon the application calls to the system in handling the resources.	X=No of tasks per unit of time	The greater the better	ratio	X=count	Known operating system. Estimated time in system calls.	Verification Joint review	Developers Requirers

Internal time behaviour metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Turnaround time	What is the estimated time to complete a group of related tasks as a job lot?	Evaluate the efficiency of the operating system and the application system calls. Estimate the response time to complete a group of related tasks based on this. The following may be measured, -all or parts of design specifications -test complete transaction path -test complete modules/parts of software product -complete software product during test phase.	X=time (calculated or simulated)	The shorter the better.	ratio	X=time	Known operating system. Estimated time in system calls.	Verification Joint review	Developers Requirers
NOTE:									

Table 8.4.2 Resource utilisation metrics

Internal resource utilisation metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Referenc e	Target audience
I/O Utilization	What is the estimated I/O utilization to complete a specified task?	Estimate the I/O utilization requirement for the application.	X=number of buffers(calculated or simulated)	The shorter the better.	ratio	X=size	Source code	Verificati on	Developers
I/O Utilization Message Density	What is the density of messages relating to I/O utilization in the lines of code responsible in making system calls.	Count the number of errors pertaining to I/O failure and warnings and compare it to the estimated number of lines of code responsible in system calls..	X=A/B A=number of I/O related error messages. B=number of lines of code directly related to system calls.	The greater the better.	Absolute	X=count/co unt A=count B=count	Source code	Verificatio n	Developers
Memory utilization	What is the estimated memory size that the product will occupy to complete a specified task?	Estimate the memory requirement.	X=size in bytes (calculated or simulated)	The lesser the better.	ratio	X=size	Estimated size of memory utilization.	Verificatio n	Developers
Memory utilization message density	What is the density of messages relating to memory utilization in the lines of code responsible in making system calls?	Count the number of error messages pertaining to memory failure and warnings and compare it to the estimated number of lines of code responsible in system calls.	X=A/B A=Number of memory related error messages. B=Number of lines of code directly related to system calls.	The greater the better.	ratio	X=count/co unt A=count B=count	Source code	Verificatio n	Developers
Transmission Utilization	What is the estimated amount of transmission resources utilization?	Estimate the Transmission resource utilization requirements by estimating the transmission volumes	X=bits/time (calculated or simulated)	The lesser the better.	ratio	X=time	Known operating system. Estimated time in system calls.	Verificatio n	Developers

Table 8.4.3 Efficiency compliance metrics

Internal Efficiency compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Efficiency Compliance	How compliant is the efficiency of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification..	X=A/B A= Number of correctly implemented items related to efficiency compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1 The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers

8.5 Maintainability metrics

Internal maintainability metrics are used for predicting the level of effort required for modifying the software product.

8.5.1 Analysability metrics

Internal analyzability metrics indicate a set of attributes for predicting the maintainer's or user's spent effort or spent resources in trying to diagnose for deficiencies or causes of failure, or for identification of parts to be modified in the software product.

8.5.2 Changeability metrics

Internal Changeability metrics indicate a set of attributes for predicting the maintainer's or user's spent effort when rying to implement a specified modification in the software product.

8.5.3 Stability metrics

Internal stability metrics indicate a set of attributes for predicting how stable the software product would be after any modification.

8.5.4 Testability metrics

Internal testability metrics indicate a set of attributes for predicting the amount of designed and implemented autonomous test aid functions present in the software product

8.5.5 Maintainability compliance metrics

Internal compliance metrics relating to maintainability indicate a set of attributes for assessing the capability of the :oftware product to comply to such items as standards, conventions or regulations of the user organisation in relation to software maintainability.

Table 8.5.1 Analysability metrics

Internal analysability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Activity recording	How thorough is the recording of the system status.	Count the number of items logged in the activity log as specified and compare it to the number of items required to be logged.	X=A/B A=Number of implemented data login items as specified confirmed in review B=Number of data items to be logged defined in the specifications	0 <= X <= 1 The closer to 1, more data provided to record system status. ***NOTE:*** *1.It is necessary to convert this value to the <0,1> interval if making summarization of characteristics*	Absolute	X=count/count A=count B=count	Value A comes from review report. Value B comes from requirement specifications.	Verification Joint review	Maintainers Users
Readiness of diagnostic function	How thorough is the provision of the diagnostic functions.	Count the number of implemented diagnostic functions as specified and compare it to the number of diagnostic functions required in specifications. ***Note:*** *This metric is also used to measure failure analysis capability and causal analysis capability.*	X=A/B A=Number of implemented diagnostic functions as specified confirmed in review B=Number of diagnostic functions required	0 <= X The closer to 1, the better implementation of diagnostic functions. ***NOTE:*** *1.It is necessary to convert this value to the <0,1> interval if making summarization of characteristics.*	Absolute	X=count/count A=count B=count	Value A comes from review report. Value B comes from requirement specifications.	Verification Joint review	Maintainers Users

Table 8.5.2 Changeability metrics

Internal changeability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Change recordability	Are changes to specifications and program modules recorded adequately in the code with comment lines?	Record ratio of module change information	X=A/B A=Number of changes in functions/modules having change comments confirmed in review B=Total number of functions/modules changed from original code	0 <= X <= 1 The closer to 1, the more recordable. The change control 0 indicates poor change control or little changes, high stability.	absolute	X=count/count A=count B=count	Configuration control system Version logs Specifications	Verification Joint review	Developers Maintainers Requirers

Table 8.5.3 Stability metrics

Internal stability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Change impact	What is the frequency of adverse impacts after modification?	Count the number of detected adverse impacts after modification and compare it to the number of modifications performed.	X=1-A/B A=Number of detected adverse impacts after modifications B=Number of modifications made	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	A comes from review report B comes from review report	Joint review Verification	Developers Maintainers Requirers
Modification impact localization	How large is the impact of the modification on the software product?	Count the number of affected variables from a modification and compare it to the total number of variables in the product. ***NOTE:*** *1.Impacted variable is a) all variables in the instruction which was changed* *b) Variable which is in the same instruction with the variable defined by (a).*	X=A/B A=Number of affected variable data by modification, confirmed in review B=Total number of variables	0 <= X <= 1 The closer to 0, the lesser impact of modification.	absolute	X=count/count A=count B=count	A comes from review report B comes from review report	Joint review Verification	Developers Maintainers Requirers

Table 8.5.4 Testability metrics

Internal testability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Completeness of built-in test function *NOTE:*	How complete is the built-in test capability.	Count the number of implemented built-in test functions as specified and compare it to the number of built-in test functions in the requirements.	X=A/B A=Number of implemented built-in test function as specified confirmed in review B=Number of built-in test function required	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	A comes from review document B comes from requirements or design document	Verification Joint review	Developers Maintainers Requirers
Autonomy of testability *NOTE:*	How independently can the software be tested?	Count the number of dependencies on other systems for testing that have been simulated with stubs and compare it with the total number of test dependencies on other systems.	X=A/B A=Number of dependencies on other systems for testing that have been simulated with stubs B= Total number of test dependencies on other systems	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	A comes from review document B comes from requirements or design document	Verification Joint review	Developers Maintainers Requirers
Test progress observability	How complete are the built in test result displays during testing?	Count the number of implemented checkpoints as specified and compare it to the number specified checkpoints required by design.	X=A/B A=Number of implemented checkpoints as specified confirmed in review B=Number of designed checkpoints	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	A comes from review document B comes from design document	Verification Joint review	Developers Maintainers Requirers

Table 8.5.5 Maintainability compliance metrics

Internal maintainability compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Maintainability compliance	How compliant is the maintainability of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification..	X=A/B A= Number of correctly implemented items related to maintainability compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1 The closer to 1, the more compliant.	absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers
NOTE: .									

8.6 Portability metrics

Internal Portability metrics are used for predicting the effect the software product may have on the behavior of the implementor or system during the porting activity

8.6.1 Adaptability metrics

Internal adaptability metrics indicate a set of attributes for predicting the impact the software product may have on the effort of the user who is trying to adapt the software product to different specified environments

8.6.2 Installability metrics

Internal Installability metrics indicate a set of attributes for predicting the impact the software product may have on the effort of the user who is trying to install the software in a user specified environment.

8.6.3 Co-existence metrics

Internal Replaceability metrics indicate a set of attributes for predicting the impact the software product may have on the effort of the user who is trying to use the software in place of other specified software in a specified environment and context of use.

8.6.4 Replaceability metrics

Internal Co-existence metrics indicate a set of attributes for predicting the impact the software product may have on other software products sharing the same operational hardware resources.

8.6.5 Portability compliance metrics

Internal compliance metrics relating to portability indicate a set of attributes for assessing the capability of the software product to comply to such items as standards conventions or regulations of the user organisation in relation to portability.

Table 8.6.1 Adaptability metrics

Internal adaptability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Adaptability of data structures *NOTE:*	How adaptable is the product to the data structure changes.	Count the number of data structures, which are operable and has no limitation after adaptation and compare it to the total number of data structures requiring adaptation capability.	X=A/B A=Number of data structures which are operable and has no limitation after adaptation, confirmed in review B=Total number of data structures requiring adaptation capability	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Maintainers Requirers
Hardware environmental adaptability (adaptability to hardware devices and network facilities) *NOTE:*	How adaptable is the product to the H/W related environmental change.	Count the number of implemented functions which are capable of achieving required results in specified multiple H/W environments as specified and compare it to the number of functions with H/W environment adaptation capability requirements	X=A/B A=Number of implemented functions which are capable of achieving required results in specified multiple H/W environment as specified, confirmed in review B=Total number of functions with H/W environment adaptation capability requirements	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Maintainers Requirers
Organisational environment adaptability (Organisation adaptability to infrastructure of organisation) *NOTE:*	How adaptable is the product to organizational change?	Count the number of implemented functions which are capable of achieving required results in specified multiple organizational and business environments as specified and compare it to the number of functions with organizational environment adaptation capability requirements	X=A/B A=number of implemented functions which are capable of achieving required results in specified multiple organizational and business environment as specified, confirmed in review B=Total number of functions with organizational environment adaptation capability requirements	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Maintainers Requirers

Internal adaptability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Porting user friendliness	How effortless is it to perform porting operations on the product	Count the number of implemented functions which are capable of supporting ease-of-adaptation by user as specified and compare it to the number of functions with easy-to-adapt capability requirements	X=A/B A=Number of functions supporting ease-of-adaptation by user as specified, confirmed in review B=Total number of functions with ease-to-adapt capability requirements	0 <= X <= 1 The closer to 1, the more friendly.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Maintainers Requirers
System software environmental adaptability (adaptability to OS, network software and co-operated application software) ***NOTE:***	How adaptable is the product to system software related environmental changes	Count the number of implemented functions which are capable of achieving required results in specified multiple system software environments as specified and compare it to the number of functions with system software environment adaptation capability requirements	X=A/B A=Number of implemented functions which are capable of achieving required results in specified multiple system software environment as specified, confirmed in review B=Total number of functions with system software environment adaptation capability requirements B=Total number of functions with system software environment adaptation capability requirements	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Req spec Design Review report	Verification Joint review	Developers Maintainers Requirers

Table 8.6.2 Installability metrics

Internal installability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Ease of Setup Re-try	How easy is it to repeat setup operation?	Count the number of implemented setup retry operations and compare it to the number of setup retry operations required	X=A/B A=Number of implemented retry operations for setup, confirmed in review B=Total number of setup operations required	0 <= X <= 1 The closer to 1, the easier.	absolute	X=count/count A=count B=count	Review report	6.5 Validation	Developers
Installation effort	What level of effort is required for installation?	Count the number of implemented installation automated steps and compare it to the number of prescribed installation steps.	X=A/B A=Number of automated installation steps confirmed in review B=Number of installation steps required ***NOTE:*** *Prescribed: e.g., number of windows/commands/manual operation to reach target operation*	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Review report	6.5 Validation	Developers
Installation flexibility	How flexible and customizable is the installation capability?	Count the number of implemented customizable installation operations as specified and compare it to the number of installation operations with customization capability requirements	X=A/B A=Number of implemented customizable installation operation as specified confirmed in review B=Number of customizable installation operation required ***NOTE:*** *Customizable: e.g., nesting depth, number of panels*	0 <= X <= 1 The closer to 1, the more flexible.	absolute	X=count/count A=count B=count	Requirement specification Review report	6.5 Validation	Developers
NOTE:									

Table 8.6.3 Co-existence metrics

Internal co-existence metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Available co-existence	How flexible is the product in sharing its environment with other products without adverse impacts on other products.	Count the number of entities with which product can co-exist as specified and compare it to the number of entities in production environment that require co-existence	X=A/B A= Number of entities with which product can co-exist as specified B= Number of entities in production environment that require co-existence	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Requirements specification Review report Test report	Verification Joint review	Requirers Developers Maintainers

Table 8.6.4 Replaceability metrics

Internal replaceability metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Continued use of data *NOTE:*	What is the amount of original data that remain unchanged after replacement with this product?	Count the number of data items, that continue to be used after replacement as specified, and compare it to the total number of data items required to be used from the old data after software replacement.	X=A/B A=Number of software data items that continue to be used as specified after replacement, confirmed in evaluation. B=Number of old data items required to be used from old software	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Design Source code Review report Test report	Verification Joint review	Requirers Developers Maintainers
Function inclusiveness *NOTE:*	What's the amount of functions that remain unchanged?	Count the number of functions covered by new software that produces similar results and compare it to the number of function in the old software. .	X=A/B A=Number of functions covered by new software that produces similar results, confirmed in review B=Number of functions in old software	0 <= X <= 1 The closer to 1, the better.	absolute	X=count/count A=count B=count	Design Source code Review report Test report	Verification Joint review	Requirers Developers Maintainers

Table 8.6.5 Portability compliance metrics

Internal portability compliance metrics									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Portability compliance	How compliant is the portability of the product to applicable regulations, standards and conventions.	Count the number of items requiring compliance that have been met and compare with the number of items requiring compliance as in the specification.	X=A/B A= Number of correctly implemented items related to portability compliance confirmed in evaluation B= Total number of compliance items	0 <= X <= 1 The closer to 1, the more complete.	absolute	X=count/count A=count B=count	Specification of compliance and related standards, conventions or regulations. Design Source code Review report	Verification Joint review	Requirers Developers
NOTE:									

Annex A
(Informative)
Considerations When Using Metrics

A.1 Interpretation of measures

A.1.1 Potential differences between test and operational contexts of use

When planning the use of metrics or interpreting measures it is important to have a clear understanding of the intended context of use of the software, and any potential differences between the test and operational contexts of use. For example, the "time required to learn operation" measure is often different between skilled operators and unskilled operators in similar software systems. Examples of potential differences are given below.

a) Differences between testing environment and the operational environment

Are there any significant differences between the testing environment and the operational execution in user environment?

The following are examples:

- testing with higher / comparable / lower performance of CPU of operational computer;
- testing with higher / comparable / lower performance of operational network and communication;
- testing with higher / comparable / lower performance of operational operating system;
- testing with higher / comparable / lower performance of operational user interface.

b) Differences between testing execution and actual operational execution

Are there any significant differences between the testing execution and operational execution

in user environment?

The following are examples:

- coverage of functionality in test environment;
- test case sampling ratio;
- automated testing of real time transactions;
- stress loads;

- 24 hour 7 days a week (non stop) operation
- appropriateness of data for testing of exceptions and errors;
- periodical processing;
- resource utilisation.
- levels of interruption
- production preassures
- distractions

c) User profile under observation

Are there any significant differences between test user profiles and operational user profiles?

The following are examples:

- mix of type of users;
- user skill levels;
- specialist users or average users;
- limited user group or public users.

A.1.2 Issues affecting validity of results

The following issues may affect the validity of the data that is collected.

(a) procedures for collecting evaluation results:

- automatically with tools or facilities/ manually collected / questionnaires or interviews;

(b) source of evaluation results

- developers' self reports / reviewers' report / evaluator's report;

(c) results data validation

- developers' self check / inspection by independent evaluators.

A.1.3 Balance of measurement resources

Is the balance of measures used at each stage appropriate for the evaluation purpose?

It is important to balance the effort used to apply an appropriate range of metrics for internal, external and quality in use measures.

A.1.4 Correctness of specification

Are there significant differences between the software specification and the real operational needs?

Measurements taken during software product evaluation at different stages are compared against product specifications. Therefore, it is very important to ensure by verification and validation that the product specifications used for evaluation reflect the actual and real needs in operation.

A.2 Validation of Metrics

A.2.1 Desirable Properties for Metrics

To obtain valid results from a quality evaluation, the metrics should have the properties stated below. If a metric does not have these properties, the metric description should explain the associated constraint on its validity and, as far as possible, how that situation can be handled.

a) **Reliability (of metric):** Reliability is associated with random error. A metric is free of random error if random variations do not affect the results of the metric.

b) **Repeatability (of metric):** repeated use of the metric for the same product using the same evaluation specification (including the same environment), type of users, and environment by the same evaluators, should produce the same results within appropriate tolerances. The appropriate tolerances should include such things as fatigue,and learning effect

c) **Reproducibility (of metric):** use of the metric for the same product using the same evaluation specification (including the same environment), type of users, and environment by different evaluators, should produce the same results within appropriate tolerances.

NOTE: It is recommended to use statistical analysis to measure the variability of the results

d) **Availability (of metric):** The metric should clearly indicate the conditions (e.g. presence of specific attributes) which constrain its usage.

e) **Indicativeness (of metric):** Capability of the metric to identify parts or items of the software which should be improved, given the measured results compared to the expected ones.

NOTE: The selected or proposed metric should provide documented evidence of the availability of the metric for use, unlike those requiring project inspection only.

f) **Correctness (of measure):** The metric should have the following properties:

1) Objectivity (of measure): the metric results and its data input should be factual: i.e., not influenced by the feelings or the opinions of the evaluator, test users, etc. (except for satisfaction or attractiveness metrics where user feelings and opinions are being measured).

2) Impartiality (of measure): the measurement should not be biased towards any particular result.

3) Sufficient precision (of measure): Precision is determined by the design of the metric, and particularly by the choice of the material definition used as the basis for the metric. The metric user will describe the precision and the sensitivity of the metric.

g) **Meaningfulness (of measure):** the measurement should produce meaningful results about the software behaviour or quality characteristics.

   The metric should also be cost effective: that is, more costly metrics should provide higher value results.

A.2.2 Demonstrating the Validity of Metrics

The users of metrics should identify the methods for demonstrating the validity of metrics, as shown below:

(a) Correlation

The variation in the quality characteristics values (the measures of principal metrics in operational use) explained by the variation in the metric values, is given by the square of the linear coefficient.

An evaluator can predict quality characteristics without measuring them directly by using correlated metrics.

(b) Tracking

If a metric M is directly related to a quality characteristics value Q (the measures of principal metrics in operational use), for a given product or process, then a change value Q(T1) to Q(T2), would be accompanied by a change metric value from M(T1) to M(T2), in the same direction (for example, if Q increases, M increases).

An evaluator can detect movement of quality characteristics along a time period without measuring directly by using those metrics which have tracking ability.

(c) Consistency

If quality characteristics values (the measures of principal metrics in operational use) Q1, Q2,..., Qn, corresponding to products or processes 1, 2,..., n, have the relationship Q1 > Q2 > ...> Qn, then the correspond metric values would have the relationship M1 > M2 > ...> Mn.

An evaluator can notice exceptional and error prone components of software by using those metrics which have consistency ability.

(d) Predictability

If a metric is used at time T1 to predict a quality characteristic value Q (the measures of principal metrics in operational use) at T2, prediction error, which is {(predicted Q(T2) - actual Q(T2)) / actual Q(T2)}, would be within allowed prediction error range.

An evaluator can predict the movement of quality characteristics in the future by using these metrics, which measure predictability.

(e) Discriminative

A metric would be able to discriminate between high and low quality software.

An evaluator can categorise software components and rate quality characteristics values by using those metrics which have discriminative ability.

A.3 Use of Metrics for Estimation (Judgement) and Prediction (Forecast)

Estimation and prediction of the quality characteristics of the software product at the earlier stages are two of the most rewarding uses of metrics.

A.3.1 Quality characteristics prediction by current data

(a) Prediction by regression analysis

When predicting the future value (measure) of the same characteristic (attribute) by using the current value (data) of it (the attribute), a regression analysis is useful based on a set of data that is observed in a sufficient period of time.

For example, the value of MTBF (Mean Time Between Failures) that is obtained during the testing stage (activities) can be used to estimate the MTBF in operation stage.

(b) Prediction by correlation analysis

When predicting the future value (measure) of a characteristic (attribute) by using the current measured values of a different attribute, a correlation analysis is useful using a validated function which shows the correlation.

For example, the complexity of modules during coding stage may be used to predict time or effort required for program modification and test during maintenance process.

A.3.2 Current quality characteristics estimation on current facts

(a) Estimation by correlation analysis

When estimating the current values of an attribute which are directly unmeasurable, or if there is any other measure that has strong correlation with the target measure, a correlation analysis is useful.

For example, because the number of remaining faults in a software product is not measurable, it may be estimated by using the number and trend of detected faults.

Those metrics which are used for predicting the attributes that are not directly measurable should be estimated as explained below:

- Using models for predicting the attribute;
- Using formula for predicting the attribute;
- Using basis of experience for predicting the attribute;
- Using justification for predicting the attribute.

Those metrics which are used for predicting the attributes that are not directly measurable may be validated as explained below:

- Identify measures of attributes which are to be predicted;
- Identify the metrics which will be used for prediction;
- Perform a statistical analysis based validation;
- Document the results;

- Repeat the above periodically;

A.4 Detecting deviations and anomalies in quality problem prone components

The following quality control tools may be used to analyse deviations and anomalies in software product components:

(a) process charts (functional modules of software)

(b) Pareto analysis and diagrams

(c) histograms and scatter diagrams

(d) run diagrams, correlation diagrams and stratification

(e) Ishikawa (Fishbone) diagrams

(f) statistical process control (functional modules of software)

(g) check sheets

The above tools can be used to identify quality issues from data obtained by applying the metrics.

A.5 Displaying Measurement Results

(a) Displaying quality characteristics evaluation results

The following graphical presentations are useful to display quality evaluation results for each of the quality characteristic and subcharacteristic.

Radar chart; Bar chart numbered histogram, multi-variates chart, Importance Performance Matrix, etc.

(b) Displaying measures

There are useful graphical presentations such as Pareto chart, trend charts, histograms, correlation charts, etc.

Annex B
(Informative)
Use of Quality in Use, External & Internal Metrics (Framework Example)

B.1 Introduction

This framework example is a high level description of how the ISO/IEC 9126 Quality model and related metrics may be used during the software development and implementation to achieve a quality product that meets user's specified requirements. The concepts shown in this example may be implemented in various forms of customization to suit the individual, organisation or project. The example uses the key life cycle processes from ISO/IEC 12207 as a reference to the traditional software development life cycle and quality evaluation process steps from ISO/IEC 14598-3 as a reference to the traditional Software Product Quality evaluation process. The concepts can be mapped on to other models of software life cycles if the user so wishes as long as the underlying concepts are understood.

B.2 Overview of Development and Quality Process

Table B.1 depicts an example model that links the Software Development life cycle process activities (activity 1 to activity 8) to their key deliverables and the relevant reference models for measuring quality of the deliverables (i.e., Quality in Use, External Quality, or Internal Quality).

Row 1 describes the software development life cycle process activities. (This may be customized to suit individual needs). Row 2 describes whether an actual measure or a prediction is possible for the category of measures (i.e., Quality in Use, External Quality, or Internal Quality). Row 3 describes the key deliverable that may be measured for Quality and Row 4 describes the metrics that may be applied on each deliverable at each process activity.

Table B.1 Quality Measurement Model

	Activity 1	Activity 2	Activity 3	Activity 4	Activity 5	Activity 6	Activity 7	Activity 8
Phase	Requirement analysis (Software and systems)	Architectural design (Software and systems)	Software detailed design	Software coding and testing	Software integration and software qualification testing	System integration and system qualification testing	Software installation	Software acceptance support
9126 series model reference	Required user quality, Required internal quality, Required external quality	Predicted quality in use, Predicted external quality, Measured internal quality	Predicted quality in use, Predicted external quality, Measured internal quality	Predicted quality in use, Measured external quality, Predicted external quality,	Predicted quality in use, Measured external quality, Predicted external quality,	Predicted quality in use, Measured external quality, Measured internal quality	Predicted quality in use, Measured external quality, Measured internal quality	Measured quality in use, Measured external quality, Measured internal quality

				Measured internal quality	Measured internal quality			
Key deliverables of activity	User quality requirements (specified), External quality requirements (specified), Internal quality requirements (specified)	Architecture design of Software / system	Software detailed design	Software code, Test results	Software product, Test results	Integrated system, Test results	Installed system	Delivered software product
Metrics used to measure	Internal metrics (External metrics may be applied to validate specifications)	Internal metrics	Internal metrics	Internal metrics External metrics	Internal metrics External metrics	Internal metrics External metrics	Internal metrics External metrics	Quality in use metrics Internal metrics External metrics

B.3 Quality Approach Steps

B.3.1 General

Evaluation of the Quality during the development cycle is divided into following steps. Step 1 has to be completed during the Requirement Analysis activity. Steps 2 to 5 have to be repeated during each process Activity defined above.

B.3.2 Step #1 Quality requirements identification

For each of the Quality characteristics and subcharacteristics defined in the Quality model determine the User Needs weights using the two examples in Table B.2 for each category of the measurement. (Quality in Use, External and Internal Quality). Assigning relative weights will allow the evaluators to focus their efforts on the most important sub characteristics.

Table B.2 User Needs Characteristics & Weights

(a)

Quality in Use		
	CHARACTERISTIC	WEIGHT (High/Medium/Low)
	Effectiveness	H
	Productivity	H
	Safety	L
	Satisfaction	M

(b)

External & Internal Quality		
CHARACTERISTIC	SUBCHARACTERISTIC	WEIGHT (High/Medium/Low)
Functionality	Suitability	H
	Accuracy	H
	Interoperability	L
	Security	L
	Compliance	M
Reliability	Maturity (hardware/software/data)	L
	Fault tolerance	L
	Recoverability (data, process, technology)	H
	Compliance	H
Usability	Understandability	M
	Learnability	L
	Operability	H
	Attractiveness	M
	Compliance	H
Efficiency	Time behaviour	H
	Resource utilization	H
	Compliance	H
Maintainability	Analyzability	H
	Changeability	M
	Stability	L
	Testability	M
	Compliance	H
Portability	Adaptability	H
	Installability	L

	Co-existence	H
	Replaceability	M
	Compliance	H

Note: Weights can be expressed in the High/Medium/Low manner or using the ordinal type scale in the range 1-9 (e.g.: 1-3 = low, 4-6 = medium, 7-9 = high).

B.3.3 Step #2 Specification of the evaluation

This step is applied during every development process activity.

For each of the Quality subcharacteristics defined in the Quality model identify the metrics to be applied and the required levels to achieve the User Needs set in Step 1 and record as shown in the example in Table B.3.

Basic input and directions for the content formulation can be obtained from the example in Table B1 that explains what can be measured at this stage of the development cycle.

NOTE: It is possible, that some of the rows of the tables would be empty during the specific activities of the development cycle, because it would not be possible to measure all of the sub characteristics early in the development process.

Table B.3 Quality Measurement Tables

(a)

Quality in Use Measurement Category				
	CHARACTERISTIC	METRICS	REQUIRED LEVEL	ASSESSMENT ACTUAL RESULT
	Effectiveness			
	Productivity			
	Safety			
	Satisfaction			

(b)

External Quality Measurement Category				
CHARACTERISTIC	SUBCHARACTERISTIC	METRICS	REQUIRED LEVEL	ASSESSMENT ACTUAL RESULT
Functionality	Suitability			
	Accuracy			
	Interoperability			
	Security			
	Compliance			
Reliability	Maturity (hardware/software/data)			
	Fault tolerance			
	Recoverability (data, process, technology)			
	Compliance			
Usability	Understandability			
	Learnability			
	Operability			
	Attractiveness			
	Compliance			
Efficiency	Time behaviour			
	Resource utilisation			
	Compliance			
Maintainability	Analyzability			
	Changeability			
	Stability			
	Testability			
	Compliance			

Portability	Adaptability			
	Instability			
	Co-existence			
	Replaceability			
	Compliance			

(c)

Internal Quality Measurement Category				
CHARACTERISTIC	SUBCHARACTERISTIC	METRICS	REQUIRED LEVEL	ASSESSMENT ACTUAL RESULT
Functionality	Suitability			
	Accuracy			
	Interoperability			
	Security			
	Compliance			
Reliability	Maturity (hardware/software/data)			
	Fault tolerance			
	Recoverability (data, process, technology)			
	Compliance			
Usability	Understandability			
	Learnability			
	Operability			
	Attractiveness			
	Compliance			
Efficiency	Time behaviour			
	Resource utilisation			
	Compliance			
Maintainability	Analyzability			
	Changeability			
	Stability			
	Testability			
	Compliance			
Portability	Adaptability			
	Instability			
	Co-existence			
	Replaceability			
	Compliance			

B.3.4 Step #3 Design of the evaluation

This step is applied during every development process activity.

Develop a measurement plan (similar to example in Table B.4) containing the deliverables that are used as input to the measurement process and the metrics to be applied.

Table B.4 Measurement Plan

SUBCHARACTERISTIC	DELIVERABLES TO BE EVALUATED	INTERNAL METRICS TO BE APPLIED	EXTERNAL METRICS TO BE APPLIED	QUALITY IN USE METRICS TO BE APPLIED
1. Suitability	1. 2. 3.	1. 2. 3.	1. 2. 3.	(Not Applicable)
2. Satisfaction	1. 2. 3.	(Not Applicable)	(Not Applicable)	1. 2. 3.
3.				
4.				
5.				
6.				

B.3.5 Step #4 Execution of the evaluation

This step is applied during every development process activity.

Execute the evaluation plan and complete the column as shown in the examples in Table B.3. ISO-IEC 14598 series of standards should be used as a guidance for planning and executing the measurement process.

B.3.6 Step #5 Feedback to the organization

This step is applied during every development process activity.

Once all measurements have been completed map the results into Table B.1 and document conclusions in the form of a report. Also identify specific areas where quality improvements are required for the product to meet the user needs.

Annex C
(Informative)
Detailed explanation of metric scale types and measurement types

C.1 Metric Scale Types

One of the following measurement metric scale types should be identified for each measure, when a user of metrics has the result of a measurement and uses the measure for calculation or comparison. The average, ratio or difference values may have no meaning for some measures. Metric scale types are: Nominal scale, Ordinal scale, Intervals scale, Ratio scale, and Absolute scale. A scale should always be defined as M'=F(M), where F is the admissible function. Also the description of each measurement scale type contains a description of the admissible function (if M is a metric then M'=F(M) is also a metric).

(a) Nominal Scale

M'=F(M) where F is any one-to-one mapping.

This includes classification , for example, software fault types (data, control, other). An average has a meaning only if it is calculated with frequency of the same type. A ratio has a meaning only when it is calculated with frequency of each mapped type. Therefore, the ratio and average may be used to represent a difference in frequency of only the same type between early and later cases or two similar cases. Otherwise, they may be used to mutually compare the frequency of each other type respectively.

Examples:Town transport line identification number , Compiler error message identification number

Meaningful statements are Numbers of different categories only.

(b) Ordinal Scale

M'=F(M) where F is any monotonic increasing mapping that is, M(x)>=M(y) implies M'(x)>=M'(y).

This includes ordering, for example, software failure by severity (negligible, marginal, critical, catastrophic). An average has a meaning only if it is calculated with frequency of the same mapped order. A ratio has a meaning only when it is calculated with the frequency of each mapped order. Therefore, the ratio and the average may be used to represent a difference in frequency of only the same order between early and later cases or two similar cases. Otherwise, they may be used to compare mutually the frequency of each order.

Examples:School exam.result (excellent, good, acceptable, not acceptable),

Meaningful statements: Each will depend on its position in the order , for example the median.

(c) Interval Scale

M'=aM+b (a>0)

This includes ordered rating scales where the difference between two measures has an empirical meaning. However the ratio of two measures in an interval scale may not have the same empirical meaning.

Examples: Temperature (Celsius, Fahrenheit, Kalvin), difference between the actual computation time and the time predicted

Meaningful statements: An arithmetic average and anything that depends on an order

(d) Ratio Scale

M'=aM (a>0)

This includes ordered rating scales, where the difference between two measures and also the proportion of two measures have the same empirical meaning. An average and a ratio have meaning respectively and they give actual meaning to the values.

Examples: Length, Weight, Time, Size, Count

Meaningful statements: Geometrical mean, Percentage

(e) Absolute Scale

M'=M they can be measured only in one way.

Any statement relating to measures is meaningful. For example the result of dividing one ratio scale type measure by another ratio scale type measure where the unit of measurement is the same is absolute. An absolute scale type measurement is in fact one without any unit.

Example: Number of lines of code with comments divided by the total lines of code

Meaningful statements: Everything

C.2 Measurement Types

C.2.0 General

In order to design a procedure for collecting data, interpreting fair meanings, and normalizing measures for comparison, a user of metrics should identify and take account of the measure type of measurement employed by a metric.

C.2.1 Size Measure Type

C.2.1.0 General

A measure of this type represents a particular size of software according to what it claims to measure within its definition.

NOTE: software may have many representations of size (like any entity can be measured in more than one dimension - mass, volume, surface area etc.).

Normalizing other measures with a size measure can give comparable values in terms of units of size. The size measures described below can be used for software quality measurement.

C.2.1.1 Functional Size Type

Functional size is an example of one type of size (one dimension) that software may have. Any one instance of software may have more than one functional size depending on, for example:

(a) the purpose for measuring the software size (It influences the scope of the software included in the measurement);

(b) the particular functional sizing method used (It will change the units and scale).

The definition of the concepts and process for applying a functional size measurement method (FSM Method) is provided by the standard ISO/IEC 14143--1.

In order to use functional size for normalization it is necessary to ensure that the same functional sizing method is used and that the different software being compared have been measured for the same purpose and consequently have a comparable scope.

Although the following often claim that they represent functional sizes, it is not guaranteed they are equivalent to the functional size obtained from applying a FSM Method compliant with ISO/IEC 14143--1. However, they are widely used in software development:

1. **number of spread sheets;**

2. **number of screens;**

3. **number of files or data sets which are processed;**

4. **number of itemized functional requirements described in user requirements specifications.**

C.2.1.2 Program size type

In this clause, the term 'programming' represents the expressions that when executed result in actions, and the term 'language' represents the type of expression used.

1. **Source program size**

The programming language should be explained and it should be provided how the non executable statements, such as comment lines, are treated. The following measures are commonly used:

a Non-comment source statements (NCSS)

Non-comment source statements (NCSS) include executable statements and data declaration statements with logical source statements.

NOTE

1. New program size

A developer may use newly developed program size to represent development and maintenance work product size.

2. Changed program size

A developer may use changed program size to represent size of software containing modified components.

3. Computed program size

Example of computed program size formula is new lines of code + 0.2 x lines of code in modified components (NASA Goddard).

It may be necessary to distinguish a type of statements of source code into more detail as follows:

i. Statement Type

Logical Source Statement (LSS). The LSS measures the number of software instructions. The statements are irrespective of their relationship to lines and independent of the physical format in which they appear.

Physical Source Statement (PSS). The PSS measures the number of software source lines of code.

ii. Statement attribute

Executable statements;

Data declaration statements;

Compiler directive statements;

Comment source statements.

iii. Origin

Modified source statements;

Added source statements;

Removed source statements;

- Newly Developed source statements: (= added source statements + modified source statements);
- Reused source statements: (= original - modified - removed source statements);

2. Program word count size

The measurement may be computed in the following manner using the Halstead's measure:

Program vocabulary = n1+n2; Observed program length = N1+N2, where:

- n1: Is the number of distinct operator words which are prepared and reserved by the program language in a program source code;
- n2: Is the number of distinct operand words which are defined by the programmer in a program source code;
- N1: Is the number of occurrences of distinct operators in a program source code;
- N2: Is the number of occurrences of distinct operands in a program source code.

3. Number of modules

The measurement is counting the number of independently executable objects such as modules of a program.

C.2.1.3 Utilized resource measure type

This type identifies resources utilized by the operation of the software being evaluated. Examples are:

(a) **Amount of memory**, for example, amount of disk or memory occupied temporally or permanently during the software execution;

(b) **I/O load**, for example, amount of traffic of communication data (meaningful for backup tools on a network);

(c) **CPU load**, for example, percentage of occupied CPU instruction sets per second (This measure type is meaningful for measuring CPU utilization and efficiency of process distribution in multi-thread software running on concurrent/parallel systems);

(d) **Files and data records**, for example, length in bytes of files or records;

(e) **Documents**, for example, number of document pages.

It may be important to take note of peak (maximal), minimum and average values, as well as periods of time and number of observations done.

C.2.1.4 Specified operating procedure step type

This type identifies static steps of procedures which are specified in a human-interface design specification or a user manual.

The measured value may differ depending on what kinds of description are used for measurement, such as a diagram or a text representing user operating procedures.

C.2.2 Time measure type

C.2.2.0 General

The user of metrics of time measure type should record time periods, how many sites were examined and how many users took part in the measurements.

There are many ways in which time can be measured as a unit, as the following examples show.

(a) Real time unit

This is a physical time: i.e. second, minute, or hour. This unit is usually used for describing task processing time of real time software.

(b) Computer machinery time unit

This is computer processor's clock time: i.e. second, minute, or hour of CPU time.

(c) Official scheduled time unit

This includes working hours, calendar days, months or years.

(d) Component time unit

When there are multiple sites, component time identifies individual site and it is an accumulation of individual time of each site. This unit is usually used for describing component reliability, for example, component failure rate.

(e) System time unit

When there are multiple sites, system time does not identify individual sites but identifies all the sites running, as a whole in one system. This unit is usually used for describing system reliability, for example, system failure rate.

C.2.2.1 System operation time type

System operation time type provides a basis for measuring software availability. This is mainly used for reliability evaluation. It should be identified whether the software is under discontinuous operation or continuous operation. If the software operates discontinuously, it should be assured that the time measurement is done on the periods the software is active (this is obviously extended to continuous operation).

(a) Elapsed time

When the use of software is constant, for example in systems operating for the same length of time each week.

(b) Machine powered-on time

For real time, embedded or operating system software that is in full use the whole time the system is operational.

(c) Normalized machine time

As in "machine powered-on time", but pooling data from several machines of different "powered-on-time" and applying a correction factor.

C.2.2.2 Execution time type

Execution time type is the time which is needed to execute software to complete a specified task. The distribution of several attempts should be analyzed and mean, deviation or maximal values should be computed. The execution under the specific conditions, particularly overloaded condition, should be examined. Execution time type is mainly used for efficiency evaluation.

C.2.2.3 User time type

User time type is measured upon time periods spent by individual users on completing tasks by using operations of the software. Some examples are:

(a) Session time

Measured between start and end of a session. Useful, as example, for drawing behaviour of users of a home banking system. For an interactive program where idling time is of no interest or where interactive usability problems only are to be studied.

(b) Task time

Time spent by an individual user to accomplish a task by using operations of the software on each attempt. The start and end points of the measurement should be well defined.

(c) User time

Time spent by an individual user using the software from time started at a point in time. (Approximately, it is how many hours or days user uses the software from beginning).

C.2.2.4 Effort type

Effort type is the productive time associated with a specific project task.

(a) Individual effort

This is the productive time which is needed for the individual person who is a developer, maintainer, or operator to work to complete a specified task. Individual effort assumes only a certain number of productive hours per day.

(b) Task effort

Task effort is an accumulated value of all the individual project personnel: developer, maintainer, operator, user or others who worked to complete a specified task.

C.2.2.5 Time interval of events type

This measure type is the time interval between one event and the next one during an observation period. The frequency of an observation time period may be used in place of this measure. This is typically used for describing the time between failures occurring successively.

C.2.3 Count measure type

If attributes of documents of the software product are counted, they are static count types. If events or human actions are counted, they are kinetic count types.

C.2.3.1 Number of detected fault type

The measurement counts the detected faults during reviewing, testing, correcting, operating or maintaining. Severity levels may be used to categorize them to take into account the impact of the fault.

C.2.3.2 Program structural complexity number type

The measurement counts the program structural complexity. Examples are the number of distinct paths or the McCabe's cyclomatic number.

C.2.3.3 Number of detected inconsistency type

This measure counts the detected inconsistent items which are prepared for the investigation.

(a) Number of failed conforming items

Examples:

- Conformance to specified items of requirements specifications;
- Conformance to rule, regulation, or standard;
- Conformance to protocols, data formats, media formats, character codes

(b) Number of failed instances of user expectation

The measurement is to count satisfied/unsatisfied list items, which describe gaps between user's reasonable expectation and software product performance.

The measurement uses questionnaires to be answered by testers, customers, operators, or end users on what deficiencies were discovered.

The following are examples:

- Function available or not;
- Function effectively operable or not;
- Function operable to user's specific intended use or not;
- Function is expected, needed or not needed.

C.2.3.4 Number of changes type

This type identifies software configuration items which are detected to have been changed. An example is the number of changed lines of source code.

C.2.3.5 Number of detected failures type

The measurement counts the detected number of failures during product development, testing, operating or maintenance. Severity levels may be used to categorize them to take into account the impact of the failure.

C.2.3.6 Number of attempts (trial) type

This measure counts the number of attempts at correcting the defect or fault. For example, during reviews, testing, and maintenance.

C.2.3.7 Stroke of human operating procedure type

This measure counts the number of strokes of user human action as kinetic steps of a procedure when a user is interactively operating the software. This measure quantifies the ergonomic usability as well as the effort to use. Therefore, this is used in usability measurement. Examples are number of strokes to perform a task, number of eye movements, etc.

C.2.3.8 Score type

This type identifies the score or the result of an arithmetic calculation. Score may include counting or calculation of weights checked on/off on checklists. Examples: Score of checklist; score of questionnaire; Delphi method; etc.

Annex D
(Informative) Term(s)

D.1 Definitions

Definitions are from ISO/IEC 14598-1 and ISO/IEC 9126-1 unless otherwise indicated.

D.1.1 Quality

External quality: The extent to which a product satisfies stated and implied needs when used under specified conditions.

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

NOTES:

The term "internal quality", used in this technical report to contrast with "external quality", has essentially the same meaning as "quality" in ISO 8402.

The term "attribute" is used (rather than the term "characteristic" used in 3.1.3) as the term "characteristic" is used in a more specific sense in ISO/IEC 9126 series.

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).

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

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

NOTE: The definition of quality in use in ISO/IEC 14598-1 does not currently include the new characteristic of "safety".

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

D.1.2 Software and user

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.

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.

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.

D.1.3 Measurement

Attribute: A measurable physical or abstract property of an entity.

Direct measure: A measure of an attribute that does not depend upon a measure of any other attribute.

External measure: An indirect measure of a product derived from measures of the behaviour of the system of which it is a part.

NOTES:

The system includes any associated hardware, software (either custom software or off-the-shelf software) and users.

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 identify the faults in the code.

External measures can be used to evaluate quality attributes closer to the ultimate objectives of the design.

Indicator: A measure that can be used to estimate or predict another measure.

NOTES:

The measure may be of the same or a different characteristic.

Indicators may be used both to estimate software quality attributes and to estimate attributes of the production process. They are indirect measures of the attributes.

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.

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.

Measure (noun): The number or category assigned to an attribute of an entity by making a measurement.

Measure (verb): Make a measurement.

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.

Metric: A measurement scale and the method used for measurement.

NOTE: Metrics can be internal or external.

Metrics includes methods for categorizing qualitative data.

Annex E
(Informative) Pure Internal Metrics

E.1 Pure Internal Metrics

Pure Internal metrics are used to measure certain attributes of the software design and code of the software product that will influence the same or all of the overall software characteristics and sub-characteristics

Table E.1.1 Pure Internal Metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric Scale type	Measure type	Input to measurement	ISO/IEC 12207 reference	Target Audiences
Coherence									
Traceability	To measure effectiveness of documentation and design structure and code of software product in mapping functions from requirements to implementation.		X=A/B A=Number of traceable items confirmed in review B=Number of items checked	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count			
Cyclomatic number	To measure the level of complexity of the software design and coding structure		e-n+2p e: # of sides n: # of edges p: # of adjacent components						
Information Flow Complexity	To measure complexity of design control structure. (refer to IEEE 982.1)		IFC(Information Flow Complexity) =(fanin x fanout)2						
Self-descriptiveness									
Modularity	To measure the easiness to update and generalize the functional knowledge base on program function/data,		X1=A1/B1 where A1=the number of modules that are functionally associated with each other, and B1=the number of modules						

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric Scale type	Measure type	Input to measurement	ISO/IEC 12207 reference	Target Audiences
	sequence of execution, and hierarchy of control flow.		X2=A2/B2 Where A2=the number of modules that are associated with each other in data structure, and B2=the number of modules						
Self-containedness									
Program size	To measure the program scale.		(N1+N2)log2(n1+n2) N1: operator occurrences N2: operand occurrences n1: total # of operators n2: total # of operands						
Conditional statement	To measure the complexity level of coded modules		X=A A= Number of conditional statements	0 <= X		X=size A=size			
Unified data reference	To measure the data unification		X=A/B A=Number of data references with unified name confirmed in review B=Total number of data references	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count			
Adequacy of variable names	To measure the variable names adequacy		X=A/B A=Number of variables with adequate names confirmed in review B=Total number of variables	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count			
Data-coupled module ratio	To measure the data-coupled module ratio		X=A/B A=Number of data-coupled modules confirmed in review B=Total number of all modules	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count			
Program statements	To measure the program source statement		X=A A=Total number of program statements	0 <= X		X=size A=size			
Average module size	To measure the average module size		X=A/B A=Total lines of source statements in all modules B=Total number of all modules	0 <= X	Absolute	X=size A=size			

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric Scale type	Measure type	Input to measurement	ISO/IEC 12207 reference	Target Audiences
Function-coupled module ratio	To measure the function-coupled module ratio		X=A/B A=Number of function-coupled modules confirmed in review B=Total number of all modules	0 <= X <= 1 The closer to 1, the better.	Absolute	X=count/count A=count B=count			