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

ISO/IEC JTC1/SC7 N2208

1999/09/29

Doc. Type	Combined WD & PDTR Registration Ballot
Title	WD 9126-4: Software Engineering - Product Quality - Part 4: Quality In Use Metrics.
Source	JTC1/SC7 WG6
Project	07.13.01.04
Status	WD
References	
Action ID	ACT
Due Date	1999/12/29
Mailing Date	1999/09/29
Distribution	SC7_AG; P, O & L Members
Medium	Adobe Acrobat
No. of Pages	37
Note	

Address reply to: ISO/IEC JTC1/SC7 Secretariat
Bell Canada - Quality Engineering & Research
1050 Beaver Hall Hill, 2nd Floor, Montréal (Québec) Canada H2Z 1S4
Tel.: +1 (514) 391-8286 Fax: +1 (514) 870-2246
sc7@qc.bell.ca

LETTER BALLOT

From the member body of:

*We support the proposal as presented

OR

*We support the proposal with the attached comments

OR

*We do not support the proposal for the technical reasons attached to this ballot

OR

*We abstain from voting
(“P” members have an obligation to vote)

***DELETE WHICHEVER DOES NOT APPLY**

TITLE:	ISO/IEC WD 9126-4: Information Technology - Software product quality - **Part 4: Quality in use metrics**
DATE:	29-Sep-99
SOURCE:	JTC1/SC7/WG6
WORK ITEM:	Project 7.13.01.1
STATUS:	Version 2.0
DOCUMENT TYPE:	WD
ACTION:	for WD and PDTR registration vote
PROJECT EDITOR:	Prof. Motoei AZUMA Department of Industrial Eng. and Management Waseda University 3-4-1, Okubo, Shinjuku-ku, Tokyo 169, Japan FAX: +81-3-3200-2567 azuma@azuma.mgmt.waseda.ac.jp
DOCUMENT EDITOR:	Nigel BEVAN Serco Usability Services Alderney House 4 Sandy Lane Teddington Middx TW11 0DU UK Fax: +44 181 614 3811 nbevan@usability.serco.com

Reference number of working document: **ISO/IEC JTC 1/SC 7 N2208**

Date: 29-Sep-99

Reference number of document: **ISO/IEC WD 9126-4**

Committee identification: ISO/IEC JTC 1/SC 7/WG 6

5 Secretariat: ANSI

Information Technology — Software Product Quality — Part 4 Quality in use metrics

Document type: Technical Report
Document subtype:
Document stage:
Document language: E

Contents

1 Scope 1

2 References ... 1

3 Terms and definitions ... 1

4 General intent of the quality metrics ... 1

4.1 Concept descriptions ... 1

4.2 Interpretation of measurement ... 3

4.3 Metrics desirable properties ... 4

5 Establish evaluation requirements ... 5

5.1 Establish purpose of evaluation ... 5

5.2 Identify types of products ... 6

5.3 Specify quality model ... 6

6 Specify the evaluation ... 7

6.1 Specify the context of evaluation ... 7

6.2 Select metrics ... 8

6.3 Establish criteria for assessment ... 10

6.4 Interpretation of measures ... 10

7 Design the evaluation ... 11

8 Execute the evaluation ... 11

8.1 Perform the user tests and collect data. ... 11

8.2 Produce a report ... 11

9 Basic use of Quality in use metrics for quality characteristics ... 1

Annex A (informative) Descriptions of the metrics tables

Annex B (informative) Remarks for better use of metrics

Annex C (informative) Bibliography ... 32

Foreword

ISO (the International Organisation for Standardisation) and IEC (the International Electrotechnical Commission) form the specialised system for world-wide standardisation. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, 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 Standard ISO/IEC 9126-1 was prepared by Joint Technical Committee ISO/IEC JTC1 *Information Technology.*

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

Part 1: Quality model

Part 2: External Metrics

Part 3: Internal Metrics

Part 4: Quality in use metrics

Introduction

This technical report presents quality in use metrics for quantitatively measuring software quality in terms of characteristics defined in ISO/IEC 9126-1 that an explanation of method for determining characteristics values from attributes values.

This technical report presents:

- a general intent of quality metrics (Clause 4)
- which measurable characteristics contribute to which software quality characteristics (including sub-characteristics) so that they can serve as metrics for these characteristics
- identify the metrics desirable properties
- an quality approach experimentation example

This report contains the terminology related to the metrics measurements, the usage of metrics in the life cycle process, and the introductory basic sets of quality in use metrics for each software quality characteristic. This report provides a guide to the user of metrics for planning evaluation, selecting metrics, designing metrics, applying metrics, and interpreting measurement data.

The 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 value, which does not depend on specific user needs but depends on generic factors, for example, human cognitive factors.

Users of this technical report include, as an example:

1. acquirer (an organization that acquires or procures a system, software product or software service from a supplier)
2. evaluator (an 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 a user)
3. developer (an organization that performs development activities, including requirements analysis, design, testing through acceptance during the software life cycle process)
4. maintainer (an organization that performs maintenance activities)
5. supplier (an organization that enter 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;
6. user (an individual that uses the software product to perform a specific function) when evaluating quality of software product at acceptance test;
7. quality managers (an organization that perform an systematic examination of the software product or software services) when evaluating software quality at qualification test

Information Technology - Software product quality - Part 4: Quality in use metrics

1 Scope

2 References

ISO/IEC 14598-1:1998, *Information Technology - Software product evaluation - Part 1: General overview*

3 Terms and definitions

For the purposes of ISO/IEC 9126-4, the following definition and the definitions contained in ISO/IEC 14598-1 apply.

NOTE The definitions contained in ISO/IEC 14598-1 are reproduced in informative annex B.

3.1
level of performance
the degree to which the needs are satisfied, represented by a specific set of values for the quality characteristics

4 General intent of the quality metrics

[editor's note: this clause based on ISO/IEC 9126-2/3 will be reviewed for relevance to ISO/IEC 9126-4]

This report provides a suggested set of quality metrics to be used with the 9126-1 quality model. The user of thi 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 this report, the user should specify how the metrics relate to the 9126-1 quality model or any others substitute quality model that is being used.

The user of this report should select the quality characteristic to be evaluated, identify the relevant metrics and then interpret the measurement result in a objective manner.

4.1 Concept descriptions

This sub section will reinforce the ISO/IEC 9126-1 concepts of internal, external and quality in use metrics before explaining how to use quality in use metrics.

The internal metrics may be applied to a non-executable software product during it s 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 detect quality issues and take corrective actions during the early stages of the development life cycle process.

The external metrics may be used to measure the quality of the software product by measuring the behavior of the system of which it is a part. The external metrics should only be used during the testing stages of the life cycle process and during any operational stages. This is achieved by executing the software product in the system environment that it is intended for.

The quality in use metrics measure the extent to which 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 environment.

What bind these three types of metrics (internal, external and quality in use) in an objective manner during an evaluation process are the quality criteria to be used for determining the quality of the software product. The metrics should be applied in a manner that supplements each other during the evaluation.

When the software quality requirements are defined, the software quality characteristics or sub-characteristics, which represent the quality requirements, are listed. Then, the appropriate external metrics and acceptable ranges are specified to quantify the quality criteria, which validate that the software meets the user needs. The internal quality attributes of the software are then defined and specified to plan to achieve the required external quality characteristics finally and to build them into the intermediate product during development. Appropriate internal metrics and acceptable range are specified to quantify the internal quality characteristics so that they can be used for verifying that the intermediate software meets the internal quality specifications during the development. Quality in use metrics measure the extent to which a product meets the needs of specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction. The relationship with others characteristics depends on the type of the user.

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

In relation to these definitions the next sub section will present the interpretation of these concepts.

4.1.1 External metrics

External metrics should be designed to:

1. represent software product quality which includes software quality characteristics and sub-characteristics defined in ISO/IEC 9126-1, during testing or operation;

2. validate that the software satisfies external quality requirements;

3. predict the actual quality in use;

4. describe the extent to which the software product keeps on satisfying user's stated or implied needs during the actual operation by user.

External metrics should be designed to employ the following measurements:

1. Measurements of software behavior in testing and operating, and in cooperation with other software, hardware, or system;

2. Measurements of user behavior.

NOTES:

1. Measurements of software behavior include measurements of incidents which are threats to human life and health, environmental natural resources, data destruction, inconsistency or misleading of information, broken security, degrade of service, advantages or profits in a market, gain or loss of economy etc.

2. The external metrics may be used to predict and to indicate the density of potential faults remaining in the software.

3. Internal measurements may be used to calculate external measures. For example, the number of program steps, which is an internal measure, may be used to normalize the external measure.

4.1.2 Internal metrics

Internal metrics provide users, evaluators, testers, developers, quality managers, or managers with benefits whereby they are able to evaluate software quality of intermediate and final products during software product is not executable.

Internal metrics should be designed to:

1. represent software quality of intermediate and final products for characteristics which includes software quality characteristics and sub-characteristics defined in ISO/IEC 9126-1, during software product is not executable;

2. guide to plan and implementation for improving designs, programs or processes, which effect intermediate and final software products;

3. verify that the intermediate and final software products satisfy internal quality requirements which are quality improvement plans for designs, programs or processes;

4. predict the external metrics or quality.

Internal metrics should be designed to employ following measurement:

Measurements of static software attributes which have appeared in the text of source code, in a graph or table representation of control, data flow, or state transition structure, or in documentation of the product itself. *Examples of metrics for quality in use are given in ISO/IEC 9126-3.*

4.1.3 Quality in use metrics

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

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

Quality in use may be influenced by any of the quality characteristics, and is thus broader than usability, which is only concerned with the ease of use and attractiveness. It can be measured by the extent to which the specified users can achieve their goals with effectiveness, task efficiency and satisfaction. Effectiveness can be measured by the accuracy and completeness with which users achieve specified goals, task efficiency by the resources expended in relation to task effectiveness, and satisfaction by attributes to the use of the product.

4.2 Interpretation of measurement

The purpose of this sub section is to procure to the reader the interpretation of the measurement concept according to the ISO/IEC 9126-1 and ISO/IEC 14598-1.

4.2.1 Direct measure for software

A direct measure is a measure of an attribute that does not depend upon a measure of any other attribute.

Some measurements of software attributes can be done without being influenced by factors such as the choice of the server where the software is executed, behavior of a user, or other external factor. Those measurements are referenced here as direct measures. Some examples are source code and executable, total number of menu options of an application or number of configuration items.

4.2.2 Indirect measure for software

An indirect measure is derived from (direct or indirect) measures of one or more attribute. For example, measure of response time is not only affected by the evaluated software itself, but also by the operating environment including but not restricted to computer hardware and operating system software. So, the response time of software is derived from the response time of the computing environment as well.

4.2.3 Indicators

Some measures can be estimated or predicted from other measures. Those measures are referenced here as indicators, and may be useful to estimate or predict attributes that cannot be measured directly, or can not be measured at all without a model. For example, the response time is not measurable while the software is still a non-executable intermediate product. Therefore, program path length may be measured and used as an indicator to predict the future response time before the software becomes executable.

As another example, in the case that the software is executed in in-house testing, the response time under a testing environment is measurable but may vary from the response time experienced by an actual user in the final operating environment. Therefore, the response time of the software in the final user environment can be predicted from the response time of the testing environment.

Finally, in the case that measuring efficiency of the software is dominantly dependent on time measurement, the response time may be used as an indicator to represent efficiency quantitatively in the software quality evaluation.

4.3 Metrics desirable properties

The accuracy and correctness of a quality evaluation relies strongly on the metrics used on that process. In order to improve metrics confidence, a list of validation requirements, which should be present on every metric applied in an evaluation, is given below.

Whenever a metric does not satisfy these validation requirements, the metric description should explain the associated constraint and, as far as possible, how that situation can be handled.

NOTE: Those properties are suited to the requirements on measurements of ISO/IEC 14598-1, and to the requirements on the evaluation of ISO/IEC 14598-5.

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

a) repeatability: repeated use of the metric in the same product to the same evaluation specification by the same evaluators, test users and environment should produce results that can be accepted as being identical,

b) reproducibility: use of the metric in the same product to the same evaluation specification by different evaluators, test users and environment should produce results that can be accepted as being identical.

Indicativeness: Capability of a metrics, which shows the part to be improved, when measure (value) is not sufficient to clear the criteria for the evaluation.

Availability: the selected or proposed metric should provide documented evidence of the availability of the metric for use.

Cost Effectiveness: user of the metric should provide documented evidence of the cost of applying the metric.

Correctness: This requirement regroups the properties associated with metrics objectivity, impartiality and precision

a) objectivity: the metric results and its data input should be factual, i.e. not colored by the feelings or the opinions of the evaluator, test users, etc.

NOTE: Since it is impossible to avoid subjective factors, especially on usability evaluation, it is recommended that procedure for assigning the number or category is enough well reviewed to be agreeable. This does not mean that it is not applicable such a metric with user sensitive testing based on questionnaire or interview.

Impartiality: the measurement should not be biased towards any particular result.

a) Precision: 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.

Meaningfulness: the measurement should produce meaningful results about the software behavior or quality characteristics. User of metrics will describe the goal or question the metric result aims at fulfilling.

5 Establish evaluation requirements

NOTE: Clauses 5, 6, 7 and 8 follow the structure of ISO/IEC 14598-1. They include material derived from ISO 9241-11.

5.1 Establish purpose of evaluation

Quality in use can be used to specify and measure the quality experienced by users in specific scenarios of usage.

5.1.1 Acquisition

Prior to development, an organisation seeking to acquire a product specifically adapted to its needs can use quality in use as a framework for specifying the quality in use requirements which the product should meet and against which acceptance testing may be carried out. Specific contexts in which quality in use is to be measured should be identified, measures of effectiveness, productivity, safety and satisfaction selected, and acceptance criteria based on these measures established.

5.1.2 Supply

A supplier can evaluate quality in use to ensure that the product meets the needs of specific types of users and usage environments. Providing the potential acquirer with quality in use results will help the acquirer judge whether the product meets their specific needs.

5.1.3 Development

A clear understanding of users' requirements for quality in use in different scenarios of usage will help a development team to orient design decisions towards meeting real user needs, and focus development objectives on meeting criteria for quality in use.

5.1.4 Operation

By measuring aspects of quality in use, the organisation operating a system can evauate the extent to which the system meets their needs, and assess what changes might be required in any futureversion.

5.1.5 Maintenance

For the person maintaining the software, the quality in use of the maintenance task can be measured, for the person porting the quality in use of the porting task can be measured.

[editor's note: this topic needs to be expanded]

5.2 Identify types of products

A stable working prototype or final product is required to evaluate quality in use.

5.3 Specify quality model

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

The components and the relationships between them are illustrated in figure 1.

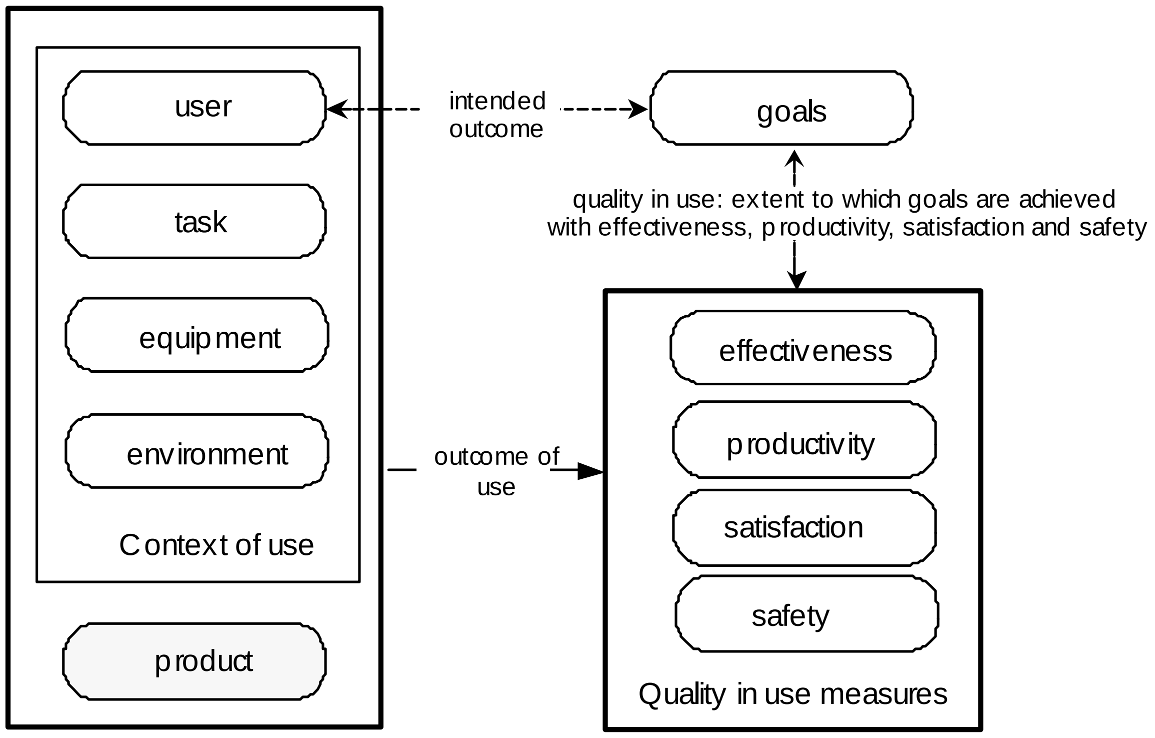

Figure 1: Quality in use model

Quality in use is the user’s view of quality. Achieving quality in use is dependent on achieving the necessary external quality, which in turn is dependent on achieving the necessary internal quality.

5.3.1 Effectiveness

The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.

5.3.2 Productivity

The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

NOTE Relevant resources can include time to complete the task, the user’s effort, materials or the financial cost of usage.

5.3.3 Safety

The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use.

NOTE Risks are usually a result of deficiencies in the functionality (including security), reliability, usability or maintainability.

5.3.4 Satisfaction

The capability of the software product to satisfy users in a specified context of use.

NOTE Satisfaction is the user’s response to interaction with the product, and includes attitudes towards use of the product.

5.3.5 Context of use

The users, tasks, equipment (hardware, software and materials), and the physical and social environments in which a product is used.

5.3.6 User

The person who interacts with the product.

5.3.7 Goal

An intended outcome.

5.3.8 Task

The activities required to achieve a goal.

NOTE 1: These activities can be physical or cognitive.

6 Specify the evaluation

6.1 Specify the context of evaluation

In order to specify or measure quality in use it is necessary to identify each component of the quality in use model:

- the goals of the user
- measures for each quality in use characteristic: effectiveness, productivity, safety, satisfaction and safety
- the context(s) of use: specification of the users, their tasks, and the environment of usage.

6.1.1 Goals

The goals of use of a product should be identified. Goals may be decomposed into sub-goals that specify components of an overall goal and the criteria that would satisfy that goal. For example, a telephone sales clerk might have the goal to "Maintain customer orders". This overall goal might then be decomposed into sub-goals such as:

- "Make accurate record of all orders placed by customers";
- "Provide information rapidly in response to customer inquiries about orders placed".

The level at which the overall goal is set is a function of the boundary of the work system which is under consideration and which provides the context of use. In the example above the work system under consideration consists of clerks taking telephone orders.

6.1.2 Context of use

Users

Relevant characteristics of the users need to be identified. These can include knowledge, skill, experience, education, training, physical attributes, and motor and sensory capabilities. It may be necessary to define the characteristics of different types of user, for example users having different levels of experience or performing different roles.

Tasks

Tasks are the activities undertaken to achieve a goal. Characteristics of tasks which may influence quality in use should be identified, e.g. the frequency and the duration of the task.

Any description of the activities and steps involved in performing the task should be related to the goals that are to be achieved. Tasks should not be identified solely in terms of the functions or features provided by a product or system.

For the purposes of evaluating quality in use, a set of key tasks will typically be selected to represent the significant aspects of the overall task.

Operating environment

Relevant characteristics of the operating environment need to be identified. The description of the hardware, software and materials may be in terms of a set of products (or system components), one or more of which may be the focus of quality in use specification or evaluation, or it may be in terms of a set of attributes or performance characteristics of the hardware, software and other materials.

Usage environments

Relevant characteristics of the physical and social environment need to be identified. Aspects which may need to be identified include attributes of the wider technical environment (e.g. the local area network), the physical environment (e.g. workplace, furniture), the ambient environment (e.g. temperature, humidity) and the social and cultural environment (e.g. work practices, organisational structure and attitudes).

6.2 Select metrics

6.2.1 Choice of measures

It is normally necessary to provide at least one measure for each of effectiveness, productivity, and satisfaction, and where relevant safety.

The choice of measures and the level of detail of each measure, is dependent on the objectives of the parties involved in the measurement. The relative importance of each measure to the goals should be considered. For example where usage is infrequent, high importance may be given to measures of learning and re-learning.

If it is not possible to obtain objective measures of effectiveness and productivity, subjective measures based on the user's perception can provide an indication of effectiveness and productivity.

Measures of quality in use should be based on data that reflect the results of users interacting with the product or work system. It is possible to gather data by objective means, such as the measurement of output, of speed of working or of the occurrence of particular events. Alternatively data may be gathered from the subjective responses of the users expressing feelings, beliefs, attitudes or preferences. Objective measures provide direct indications of effectiveness and productivity while subjective measures can be linked directly with satisfaction.

The validity of the data gathered to predict the level of quality in use achieved when a product is actually used will depend upon the extent to which the users, tasks and context of use are representative of the real situation and the nature of the measures chosen. At one extreme one may make measurements in the "field" using a real work situation as the basis for the evaluation of the quality in use of a product. At the other end of the continuum one may evaluate a particular aspect of the product in a "laboratory" setting in which those aspects of the context of use which are relevant are re-created in a representative and controlled way. The advantage of using the laboratory based approach is that it offers the opportunity to exercise greater control over the variables which are expected to have critical effects on the level of quality in use achieved, and more precise measurements can be made. The disadvantage is that the artificial nature of a laboratory environment can produce unrealistic results.

Evaluations can be conducted at different points along the continuum between the field and laboratory settings depending upon the issues that need to be investigated and the completeness of the product that is available for test. The choice of test environment and measures will depend upon the goals of the measurement activity and their relationship with the design cycle.

6.2.2 Effectiveness

Measures of effectiveness relate the goals or sub-goals of the user to the accuracy and completeness with which these goals can be achieved.

For example if the desired goal is to accurately reproduce a 2-page document in a specified format, then accuracy could be specified or measured by the number of spelling mistakes and the number of deviations from the specified format, and completeness by the number of words of the document transcribed divided by the number of words in the source document.

To measure accuracy and completeness it is necessary to produce an operational specification of the criteria for successful goal achievement. This can be expressed in terms of the quality and quantity of output, for example, the specification of a required format for output documents together with the number and length of documents to be processed.

Accuracy can be measured by the extent to which the quality of the output corresponds to the specified criteria, and completeness can be measured as the proportion of the target quantity which has been achieved.

If a single measure of effectiveness is required, it is possible to combine measures of accuracy and completeness. For example, completeness and accuracy can be calculated as percentages and multiplied together to give a percentage value for effectiveness. In cases where it is not appropriate to trade accuracy off against completeness, the two measures should be considered independently.

6.2.3 Productivity

Measures of productivity relate the level of effectiveness achieved to the expenditure of resources. Relevant resources can include mental or physical effort, time, materials or financial cost. For example, human productivity could be measured as effectiveness divided by human effort, temporal productivity as effectiveness divided by time, or economic productivity as effectiveness divided by cost.

If the desired goal is to print copies of a report, then productivity could be specified or measured by the number of usable copies of the report printed, divided by the resources spent on the task such as labour hours, process expense and materials consumed.

6.2.4 Safety

[editor's note: this topic needs to be expanded]

6.2.5 Satisfaction

Satisfaction measures the extent to which users are free from discomfort and their attitudes towards the use of the product.

Satisfaction can be specified and measured by subjective rating on scales such as discomfort experienced, liking for the product, satisfaction with product use, or acceptability of the workload when carrying out different tasks, or the extent to which particular quality in use objectives (such as productivity or learnability) have been met. Other measures of satisfaction might include the number of positive and negative comments recorded during use. Additional information can be obtained from longer term measures such as rate of absenteeism, observation of overloading or underloading of the user’s cognitive or physical workload, or from health problem reports, or the frequency with which users request transfer to another job.

Subjective measures of satisfaction are produced by quantifying the strength of a user's subjectively expressed reactions, attitudes, or opinions. This process of quantification can be done in a number of ways, for example, by asking the user to give a number corresponding to the strength of their feeling at any particular moment, or by asking users to rank products in order of preference, or by using an attitude scale based on a questionnaire.

Attitude scales, when properly developed, have the advantage that they can be quick to use, have known reliabilities, and do not require special skills to apply. Attitude questionnaires which are developed using psychometric techniques will have known and quantifiable estimates of reliability and validity, and can be resistant to factors such as faking, positive or negative response bias, and social desirability. They also enable results to be compared with established norms for responses obtained in the past. See E.2.9, E.2.10 and E.2.12 for examples of questionnaires which measure satisfaction with computer-based systems.

6.3 Establish criteria for assessment

The choice of criterion values of measures of quality in use depends on the requirements for the product and the needs of the organisation setting the criteria. Quality in use objectives may relate to a primary goal (e.g. produce a letter) or a sub-goal (e.g. search and replace) or secondary goals (e.g. learnability or adaptability). Focusing quality in use objectives on the most important user goals may mean ignoring many functions, but is likely to be the most practical approach. Setting quality in use objectives for specific sub-goals may permit evaluation earlier in the development process.

It may be necessary to specify criteria both for the minimum acceptable level of quality in use and for the target level of quality in use.

When setting criterion values for a group of users, the criteria may be set as an average (e.g. average time for completion of a task to be no more than 10 minutes), for individuals (e.g. all users can complete the task within 10 minutes), or for a percentage of users (e.g. 90% of users are able to complete the task in 10 minutes).

When setting criteria, care should be taken that appropriate weight is given to each measurement item. For example, to set criteria based on errors, it may be necessary to assign weightings to reflect the relative importance of different types of error.

6.4 Interpretation of measures

Because the relative importance of characteristics of quality in use depends on the context of use and the purposes for which quality in use is being specified or evaluated, there is no general rule for how measures should be chosen or combined.

Care should be taken in generalising the results of any measurement of quality in use to another context which may have significantly different types of users, tasks or environments. If measures of quality in use are obtained over short periods of time the values may not take account of infrequent events which could have a significant impact on quality in use, for example intermittent system errors.

For a general-purpose product it will generally be necessary to specify or measure quality in use in several different representative contexts, which will be a subset of the possible contexts and of the tasks which can be performed. There may be differences between quality in use in these contexts.

7 Design the evaluation

The evaluation should be carried out in conditions as close as possible to those in which the product will be used. It is important that:

- Users are representative of the population of users who use the product
- Tasks are representative of the ones for which the system is intended
- Conditions are representative of the normal conditions in which the product is used

By controlling the context of evaluation, experience has shown that reliable results can be obtained with a sample of only eight participants.

8 Execute the evaluation

8.1 Perform the user tests and collect data.

When assessing quality in use it is important that the users work unaided, only having access to forms of assistance that would be available under normal conditions of use. As well as measuring effectiveness, productivity and satisfaction it is usual to document the problems users encounter, and to obtain clarification by discussing the problems with users at the end of the session. It is often useful to record the evaluation on video, which permits more detailed analysis, and production of video clips. It is also easier for users to work undisturbed if they are monitored remotely by video.

8.2 Produce a report

If a comprehensive report is required, the Common Industry Format provides a good comprehensive structure for reporting quality in use.

9 Basic use of Quality in use metrics for quality characteristics

The following are set metrics recommended to use as basic metrics which give measures or may be applied as checklists to represent software quality characteristics. Although, there some practical experiences in progress, these metrics are draft and need more validation and feedback. Therefore they are sorted out software quality characteristics and sub-characteristics.

More general external metrics are usable as quality in use metrics during operational testing or operation as well as other external metrics for quality characteristics.

Table 6.1.1 Quality in use metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Measurement scale	Scale type	Measure type	Input to measurement	Perspective Beneficer
Task effectiveness			**task effectiveness M1 = A x B** A= Quantity (completeness) = the proportion of task goals represented in the output of task B= Quality (correctness) = the degree to which the task goals represented in the output have been achieved NOTE: It is recommended for evaluator to previously specify several patterns of user's output and error for corresponding to each achieved level of task goals. That helps evaluator to assess quantity (completeness) and quality (correctness) of tasks which examined user achieve.	0<= M1 <=1 The closer to 1.0 is the better.	M1= Abs. A= Abs. B= Abs.	A=Count B=Count M1= Count x Count		

Task productivity	**task productivity M2 = M1 / T**
	
	A = task effectiveness
	T = task time
		0<= M2
		
		The larger is the better.
			M2=
			
			Ratio
				T= Time
				
				M2= (Count x Count)/ Time

Productive proportion	**productive proportion M3 = Ta / Tb**
	
	Ta = productive time = task time - help time - error time - search time
	Tb = task time
		0<= M3 <=1
		
		The closer to 1.0 is the better.
			M3 = Abs	Ta=Time
				Tb=Time
				M3= Time/ Time

Relative user productivity	**Relative user productivity M4 = A / B**
	
	A = ordinary user's task productivity = M2
	B = expert user's task productivity = M2
		0<= M4 <=1
		
		The closer to 1.0 is the better.
			M3 = Abs	M4= M2 / M2

Attitude	**SUMI scale X = A**
	A = questionnaire producing psychometric scales
		population average is 50 for each scale	Ord.	A= Count
				X= Count

Annex A
(informative)

Descriptions of the metrics tables

[editor's note: this annex from ISO/IEC 9126-2/3 will be reviewed for relevance to ISO/IEC 9126-4]

The purpose of these descriptions is to clarify the reader comprehension for the quality metrics formulas proposal in the clause 5 .

A.1 Metrics name

Metrics name characterizes measurable attribute of software and represents a unique or group of measurements.

Metrics name has the same or similar name in internal, external and quality in use metrics, when they are intended to be mutually corresponded respectively.

A.2 Purpose of the metrics

This helps to identify what user of metric can know by using the metric.

NOTE: This is described as a questionary style to look up easily in accordance with Goal / Question / Metric framework.

A.3. Method of application

This helps to understand what way are useful and recommended to apply metrics.

A.4. Measurement, formula and data element computations

This helps to understand what kind of measurement, formula and data element are used to compute measure.

A.5. Interpretation of measured value

This helps to understand the range of measured value and the interpreted better range.

A.6. Scale types

The following measurement scale types should be identified for each measure, when a user of metrics has the result of a measurement and use measure to calculate together or compared with other. The average, ratio or difference values may have no meaning for some measures. Such scale types are: Nominal scale, Ordinal scale, Intervals scale, Ratio scale, Absolute scale. M'=F(M), where F is the admissible function, explain what the admissible function is (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 compare mutually frequency of each other type respectively.

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 frequency of each mapped order. Therefore, the ratio and 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 mutual frequency of each order.

c) **Intervals scale**

M'=aM+b (a>0)

This includes artificial rating scales, for example, rating scales of sensitive questionnaire for asking about usability. These rating scales are comparable and an average has meaning only if it is calculated with the same rated scales for the same asking, but a ratio of such rating scales has no meaning. Even when double scores, it is pointed out only that a score is twice as much as the another, but not that anything is double.

d) **Ratio scale**

M'=aM (a>0)

This includes time interval, for example, time between occurred software failures. An average and a ratio has meaning respectively and they give actual meaning to the values. When the value is double, it is pointed out it takes as two times long as the another.

e) **Absolute scale**

M'=M

This includes density and frequency, for example, detected software fault density. An average has no meaning, while a ratio has meaning. In a 2 Kstep program, 20 faults was detected and in another 8Kstep, 160 faults was detected, resulting 10 faults / Kstep and 20 faults / Kstep respectively. It may seems that the average measure is 15 faults / Kstep, but actual average is 180 faults for 10 kilo step code, that is, 18 faults / Kstep.

A.7. Measurements types

For designing procedure for collecting data, interpreting fair meanings, and normalizing measures for comparison, a user of metrics should identify and take account of measure type of measurement employed by a metric.

A.7.1. Size measure type

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

NOTE: that 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 in this report can be used for software quality metrics.

A.7.1.1. 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:

- the purpose for measuring the software size (It influences the scope of the software included in the measurement);
- 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-Part1.

In order to use Functional Size for normalization a quality assessor needs 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 an FSM Method and compliant with ISO/IEC 14143-Part1. 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.

A.7.1.2. Program size type

In this section, the terms programming should represent the number of execution resulting an action state and the language represent the type of expression used

1) Program source size

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

a) Non-comment source statements (NCSS)

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

b) New program size

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

c) Language use

For a same executable statement, the program size depend on the language use.

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

NOTE: 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 needed to distinguish a type of statements of source code into more detail like followings.

1) Statement type

- **Logical Source Statement(LSS).** The LSS measures the number of software instructions. The statements are irrespective of its 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.

2) Statement attribute

- Executable statements;

- Data declaration statements;

- Compiler directive statements;

- Comment source statements.

3) 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 by the following so-called Halstead's measure:

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

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

3) Number of modules

The measurement is counting the number of modules of a program.

A.7.1.3. Utilized resource size measure type

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

a) **Amount of memory** ex.) amount of disk memory occupied temporally or stable during the software execution;

b) **I/O load** ex.) bit size of communication data (meaningful for backup tools on a network);

c) **CPU load** ex.) percentage of occupied CPU instruction sets per second (meaningful for CPU utilization and efficiency of process distribution in multi-thread software's running on concurrent/parallel systems);

d) **Files and data records** ex.) bit size of file or record;

e) **Documents** ex.) number of document pages.

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

A.7.1.4. Specified operating procedure step type

This type identifies static steps of procedure 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 procedure.

A.7.1.5. Time measure type

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

The user of metrics should be aware that there are many ways in which time can be measured as a unit, including following:

a) Real time unit

This is 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 site but all the sites running, because whole sites are involved into one system. This unit is usually used for describing system reliability, for example, system failure rate.

A.7.1.6. System operation time type

System operation time type provides a basis for measuring time of 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 time measurement is done just on the periods the software is active (this is obviously extended to continuous operation).

a) Elapsed time

When use 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 power and applying a correction factor.

A.7.1.7. 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 should be computed. The execution under the specific conditions, particularly overloaded condition, should be examined. Execution time measure type is mainly used for efficiency evaluation.

A.7.1.8. User time type

User time type is measured upon time periods spent by individual user 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 behavior 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) **User operating time (task time)**

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

c) **User time**

Time spent by an individual user to use the software from the getting started to now. (Approximately, it is how many hours or days user uses the software from beginning.)

A.7.1.9. 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 developer, maintainer, or operator to work to complete a specified task. Individual effort assumes productive hours only according to a certain number of productive hour per day.

b) Task effort

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

A.7.1.10. Time interval of events types

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

9.1.1 A.7.2. Count measure type

This measure type identifies number of counted number, turn, event or incident with investigation activity.

Investigation activity includes reviewing, testing and operating, and using from view of human-engineering and ergonomics. This measure type should be attached informative description on the context including periods, and number and profile of experimented site and users during counting are performed, because the measure strongly depends on those. The followings are belong to static count type.

A.7.2.1. Number of detected fault type

The measurement is to count the detected faults during reviewing, testing, correcting, operating or maintaining. Severity level may be used to categorize them for taking account of impacts.

A.7.2.2. Program structural complexity number type

The measurement is to count program structural complexity. Examples are number of distinct paths or McCabe's cyclomatic number.

A.7.2.3. Number of detected inconsistent type

This measure is to count detected inconsistent items which are prepared for investigation.

a)Number of failed conforming items

Examples:

- Conformance to specified items of requirements specifications;
- Conformance to rule, regulation, or standard;
- Conformance to protocol, data format, character code, or other media format.

b)Number of failed user expectation

The measurement is to count satisfied/unsatisfied items which are listed up and describing the gaps between user's reasonable expectation and software product performance.

The measurement use a questionnaires for asking tester, customer, operator, or end user which kinds of deficiencies are discovered.

The followings are example items for specified user callable function:

- Actually available or not;
- Actually operable effectively or not;
- Actually operable to user's specific intended use;
- Actually Expected/needed or not.

A.7.2.4. Number of change type

This type identifies software configuration items which are detected they have been changed. Example is number of changed source line of code

The followings are belong to kinetic count type.

A.7.2.5. Number of detected failure type

The measurement is to count the detected failures during the life cycle development of the product, the testing, operating or maintaining. Severity level may be used to categorize them for taking account of impacts.

A.7.2.6. Number of attempt (trial) type

This measure is to count check points, questionnaires, or test cases for investigation during reviewing, testing, correcting, operating or maintaining.   Examples are number of design anomalies during the review, number of misunderstanding requirements at the specification level, number of actually tried test cases, number of cancellation operation and so on.

A.7.2.7. Stroke of human operating procedure type

This type identifies kinetic steps of procedure, when user is operating the software. This measure type provides values quantifying the both of an effort and an ergonomics and this is used usability metrics. Examples are number of strokes to perform an action; number of eye movements; etc.

A.7.2.8. Score type

This type identifies the score or arithmetic calculation result. Score may includes count and calculation of checking on/off on check lists.   Examples: Score of check list; score of questionnaire; Delphi method; etc.

A.8. Input to measurement

This helps to understand what kinds of information or documents are generally required to do measurement.

A.9. Perspective/Beneficer

This helps to understand whose view and benefit are strogly related with the metric, though any other personel and party related to the software are also receive benefit.

Annex B
(Informative)

Remarks for better use of metrics

[editor's note: this annex from ISO/IEC 9126-2/3 will be reviewed for relevance to ISO/IEC 9126-4]

B.1 Take accounts of constraints of applied metrics

The measures may not be interpret adequately and not understood sufficiently without context describing situation and condition during data are gathered and metrics are applied. For examples, 1)"time required to learn operation" measure is often different between skillful operators in similar software systems and unskillful operators. 2) If testing is not so much intensive, measures acquired as testing result may be different so much from the true value.

Therefore, it is recommended to take accounts the following contexts to avoid to interpret inadequately and to misunderstand the measures of metrics.

a) Differences between testing environment and actual user operation one

Are there any remarkable differences between testing environment and actual user operation one?

The followings are examples:

-testing with higher / comparable / lower performance of CPU of users' computer;

-testing with higher / comparable / lower performance of network and communication;

-testing with higher / comparable / lower performance of operating system;

-testing with higher / comparable / lower performance of user interface type provided by operating system.

b) Differences between testing execution and actual user operational execution

Are there any remarkable difference between testing execution and actual user operational execution?

The followings are examples:

-coverage of functional specification or program by test cases;

-test case sampling ratio;

-high speed real time processing testing;

-over loaded data processing testing;

-non-stop operation testing;

-abnormal, exception, or fault injected data input testing;

-frequency of use, such are daily, weekly, monthly, or only at emergency;

-combination of considerable other software, devices, equipment, and systems.

c) User profile under observation

Are there any remarkable different user profile between tested and actual? The followings are examples:

-heavy, moderate, less or temporal user;

-trained level, e.g., skillful operator or first entry operator;

-expert user or ordinal user;

-office user or home user.

d) Data validation level

Are there any problems come from what kinds of data collection procedures and level of data validation?
The followings are examples.

-procedures of data collection:
automatically with tools or facilities/ manually collected / questionnaires or interviews;

-data source report level:
developers' self reports / reviewers' report / inspected report by evaluator;

-data validation activity:

developers' self check / inspection by independent evaluators.

e) Balance of the extent of the investigation performance and measures

Are there remarkable problems in investigation performance?

It is important to take consider to keep balance between the extent of the investigation performance and measures (detected results) in fair range during testing;

Investigation includes reviewing and testing performance.

Most of the external metrics use measurement value derived from testing cases and results of problem detection in a validation testing or operation testing. Most of internal metrics also use derived from review. Therefore, the measured value of external metrics are affected by the extent of the testing effort, that is, investigation performance. Internal metrics also may be similarly affected the extent of the reviewing, that is, investigation performance.

The user of metrics should identify the extent of the justification of measured values depends on the extent of the investigation performance.

The extent of investigation to detect problems as input affects the extent of detected problems (failures, faults, miss-matching, etc.) and the extent of residual covered problems as outputs.

f) Balance of the extent of the specification clearance and conformance testing

Are there remarkable problems in specification clearance? Metrics often use measurements which count problems by comparing specification with testing results and by determining whether they are consistently matching or not, that is, whether the software conform to its specification or not. However, specification may not be enough mature to do that appropriately. Therefore, it is recommended to conduct to review and to improve the specification from a view of software product fulfilling its specific intended use in actual, during specification based matching testing case are designed and tested.

B.2 Validity demonstration and use of metrics

The user of metrics should identify the methods for metrics validity demonstration, which are following 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, which is given by the square of the linear coefficient.

An evaluator can predict quality characteristics without measure directly by using these metrics which have correlation ability.

b) **Tracking**

If a metric M is directly related to a quality characteristics values 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), which is the same direction (for example, if Q increase, M increase). An evaluator can detect movement of quality characteristics along time preriod without measure directly by using these 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 process 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 these metrics which have consistent ability.

d) **Predictability**

If a metric is used at time T1 to predict a quality characteristics values 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 with allowed prediction error.

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

e) **Discriminative**

A metric would be able to discriminate between high quality software components and low quality software components.
An evaluator can categorize software components and rate quality characteristics values by using these metrics which have discriminative ability.

B.3 Prediction use of metrics

Early detection and prediction of quality of the software product is a one of the most effective use of metrics.

a) Future prediction

- Future measure prediction

To estimate the future values of the same measure by using the current measured values, it is estimated based on trend along with time.

For example, the measured value trend of mean time between failures during testing can be used to estimate the value of mean time in actual operation.

- Future invisible other measure prediction

To estimate the future values of the invisible other measure by using the current measured values, it is estimated based on correlative relations.

For example, the complexity of modules during coding may be used to predict time or effort of program change and test during maintenance.

b) Current fact finding prediction

- Invisible other measure prediction

To estimate the current values of other measure which is supposed to be strongly related mutually by using the current measured values, it is estimated based on correlative relations.

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

The metrics which are designed for predicting the attributes should be documented with explanations including below:

-models for predicting the attribute;

-formula for predicting the attribute;

-experience for predicting the attribute;

-justification for predicting the attribute.

The metrics which are designed for predicting the attributes may be sophisticated with the metrics validation procedure containing below:

-identify the samples of measures of attributes which are to be predicted

-identify the metrics which are supposed to be capable for prediction

-perform a statistical analysis

-document the results

-iterate above periodically

B.4 Detect quality problem prone components

The following methods are usually employed to detect quality problem prone components:

-extremely deviated in distribution;

-exceeding boundaries of adequate pair of upper or lower limit in trend;

-extremely deviated or exceeding boundaries of adequate pair of upper or lower limits correlation.

It is recommended to use chrats such Pareto, trend along the time or histograms.

B.5 Displatying measurement results

a) Displaying quality characteristics evaluation result

For example,following graphical presentations are useful to display quality evaluation result for each quality characteristics and sub-characteristics:
Radar chart; Bar chart and so on.

b) Displaying measures

They are useful grafical presentations such Pareto chart, trend chart along the time, histograms, correlation chart and so on.

Annex C
(informative)

Bibliography

Common Industry Format for usability reports. http://www.nist.gov/iusr

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

IEEE 610.12-1990 Standard Glossary of Software Engineering Terminology

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

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

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

ISO 8402: 1994, Quality -Vocabulary .

ISO 9001:1994, Model for quality assurance in design, development, production, installation and servicing;

ISO/IEC PDTR 9126-2:new, Information Technology - Software product quality - Part 1: External metrics

ISO/IEC PDTR 9126-3:new, Information Technology - Software product quality - Part 1: Internal metrics

ISO/IEC PDTR 9126-4:new, Information Technology - Software product quality - Part 1: Quality in use metrics

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

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

ISO/IEC 12207:1995, Information Technology - Software lifecycle processes.

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

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

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

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

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

ISO/IEC PDTR 15504:1996, Information Technology - Software Process Assessment

Kirakowski J (1996) The software usability measurement inventory: background and usage. In: P Jordan, B Thomas, & B Weerdmeester, Usability Evaluation in Industry. Taylor & Frances, UK.