INTERNATIONAL STANDARD

ISO/IEC 14598-4

First edition
1999-10-01

Software engineering — Product evaluation —

Part 4: Process for acquirers

Ingéniérie du logiciel — Évaluation du produit —

Partie 4: Procédé pour les acquéreurs

Reference number
ISO/IEC 14598-4:1999(E)

Contents

Foreword

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

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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14598 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14598-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software engineering*.

ISO/IEC 14598 consists of the following parts under the general title *Software engineering — Product evaluation*:

— *Part 1: General overview*

— *Part 2: Planning and management*

— *Part 3: Process for developers*

— *Part 4: Process for acquirers*

— *Part 5: Process for evaluators*

— *Part 6: Documentation of evaluation modules*

Annexes A to E of this part of ISO/IEC 14598 are for information only.

Introduction

Software has become increasingly pervasive. The demand for added software functionality and faultless software products has grown as more processes are automated to take advantage of the power of the computer. Today's modern systems are so complex, that they are unable to perform their functions without software. The use of commercially available "off-the-shelf" software products is accelerating as the variety of available products grows and the rapid evolution of software engineering technology reduces reliance on custom-coded software. The object-oriented development approach, which is based on the development of an application system through the extension of existing libraries of self-contained units, has also reduced requirements for custom-coded software. This has led to intense focus on concomitant software product quality or self-contained software unit quality.

Development of custom software is prone to rework as a result of failure to meet user requirements. The use of custom software may also require a larger than anticipated effort with respect to deployment, implementation, training, and maintenance support activities. Acquisition of commercial "off-the-shelf" software products, or, reuse of in-house existing software products, is also not without risk. Problems can be encountered because the "off-the-shelf" software products may require customizing; testing and analysis requirements may be large; product maintenance and support is doubtful when the product becomes obsolete or revised; it may be difficult to integrate software products into larger systems; and the quality of the product may not be consistent with the required quality of the target system.

Commercial "off-the-shelf" software products are extremely varied. They can be:

— a) used as stand-alone products (i.e., payroll, accounting software, consumer software or 'shrink-wrapped software' [i.e., word-processing software, spreadsheets]);

— b) integrated as components into a larger system which consists of other software and hardware components (i.e., operating system, relational data base management system, graphical users interface [GUI]);

— c) embedded in hardware (i.e., communication data link, programmable array logic [PAL]);

— d) embedded as part of a configurable software/hardware system that can be used for the development of a specific application (i.e., distributed control system);

— e) CASE tools used to support the software development and maintenance process (i.e., compilers, configuration management tools).

Errors in stand-alone software products can impact productivity, cause financial loss, or cause unnecessary rework. Software components can be difficult to integrate, affect the reliability of the overall system, or be incompatible with system objectives. CASE tools may introduce an error into a product under development or be difficult to use.

It is therefore essential to be able to evaluate the quality of software products during acquisition, or when making a decision on reusing an existing software product or component. Evaluation may be used to accept or reject a single product, or to select one product, from among alternative products, that meets the quality requirements established for the target application. The level of rigor of the evaluation process is necessarily commensurate with the integrity requirements for the product. The highest level of rigor is required when performing evaluation of software products that are mission critical.

Software engineering — Product evaluation — Part 4: Process for acquirers

1 Scope

This part of ISO/IEC 14598 contains requirements, recommendations and guidelines for the systematic measurement, assessment and evaluation of software product quality during acquisition of "off-the-shelf" software products, custom software products, or modifications to existing software products. It uses the software quality model described in ISO/IEC 9126-1; expands on the general process for evaluating software quality that is defined in ISO/IEC 14598-1; and uses the process for acquisition that is defined in ISO/IEC 12207. It can be used in conjunction with ISO/IEC 12119, ISO/IEC 14598-2 (new), ISO/IEC 14598-3 (new) and ISO/IEC 14598-6. The steps of the evaluation process are similar between this part of ISO/IEC 14598 and ISO/IEC 14598-5, but the context of use is quite different. In the case that acquirers entrust second or third parties with evaluations, ISO/IEC 14598-5 is required to be applied. In the case that acquirers require third party testing of software packages against the quality requirements for the package, ISO/IEC 12119 may be applied.

The evaluation process described in this part of ISO/IEC 14598 also helps to meet the objectives of deciding on the acceptance of a single product, or for selecting a product from among alternate products. The evaluation process may be tailored to the nature and integrity level of the application. It is also sufficiently flexible to accommodate the wide range of forms and uses of software products in a cost-effective manner.

This part of ISO/IEC 14598 is intended for, but not limited to, project managers, system engineers, development and maintenance software engineering staff and end users who plan to acquire software products, and also suppliers who provide such products.

The target software products of the evaluation process in this part of ISO/IEC 14598 can be integrated into a larger system as components or can be used stand-alone. They are classified as:

a) Commercial off-the-shelf software products;

b) Existing software products developed or acquired for other applications, or for a wide range of common applications;

c) Custom software products or modifications to existing software products.

The evaluation process that is defined in this part is also applicable to CASE tools. Because evaluation of CASE tools is specifically addressed in ISO/IEC 14102, CASE tools are considered out of scope of this part of ISO/IEC 14598.

ISO/IEC 14598-4 is designed to work in partnership with other standards. For systems with high integrity requirements, additional requirements may be included in the evaluation process described in ISO/IEC 14598-4, that are derived from sector-specific standards, e.g., IEC 880, DOA-167A , MOD-55, etc.

2 Conformance

Because of the freedom of choice afforded to the user by the general nature of its recommendations, a simple claim of compliance with this part of ISO/IEC 14598 is not valid. Any organization imposing this part of ISO/IEC 14598 as a condition of trade is responsible for specifying and making public the evaluation process that meets the mandatory objectives specified in 6.1.1. The specified evaluation process constitutes the terms for compliance for a given application of this part of ISO/IEC 14598. All activities of clauses 6 and 7 shall be considered for applicability.

Requirements on the evaluation process can also be established contractually during execution of the acquisition process. Compliance with the evaluation process described in this part of ISO/IEC 14598 is then easily established.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14598. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 14598 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 9126-1, *Information technology - Software quality characteristics and metrics – Part 1: Quality characteristics and subcharacteristics.*[1)]

ISO/IEC 12207:1995, *Information technology - Software life cycle processes.*

ISO/IEC 14598-1:1999, *Information technology – Software product evaluation - Part 1: General overview.*

ISO/IEC 14598-5:1998, *Information technology – Software product evaluation - Part 5: Process for evaluators.*

ISO/IEC 15026:1998, *Information technology – System and software integrity levels.*

4 Terms and definitions

For the purposes of this part of ISO/IEC 14598, the following definitions apply. Key definitions from other standards used in this part of ISO/IEC 14598 are reproduced in Annex A for convenient reference.

4.1
commercial-off-the-shelf software (COTS)
software defined by a market-driven need, commercially available, and whose fitness for use has been demonstrated by a broad spectrum of commercial users

NOTE See also definition in IEEE Std 1062-1993.

4.2
custom software
software developed for a specific application from a user requirements specification

4.3
existing software
software that is already developed and available; is usable either "as is" or with modifications; and which is provided by the supplier, acquirer, or a third party

NOTE See also definition of "off-the-shelf product" [ISO/IEC 12207: 1995].

5 Software product evaluation - General considerations

5.1 Correlation between evaluation and acquisition processes

The acquisition process activities (defined in ISO/IEC 12207) summarized below, are combined with the general evaluation process activities (defined in ISO/IEC 14598-1) in Clauses 6 and 7. Clause 6 focuses on the application of the evaluation of end product quality during acquisition of COTS products. Clause 7 focuses on the application of the evaluation process during acquisition of custom software or modifications to existing software.

1) To be published. Until this part is published ISO/IEC 9126 :1991 should be used.

a) Initiation - identification of software requirements for the product to be acquired, the acquisition plan, and the acceptance strategy and criteria;

b) Request-for-proposal (-tender) preparation - specification and documentation of acquisition requirements;

c) Contract preparation and update - supplier selection, contract preparation and negotiation, and contract change control;

d) Supplier monitoring - evaluation activities performed during contract execution leading to acceptance and delivery of the software product;

e) Acceptance and completion - activities performed during product acceptance and delivery of the final software product.

Note, that, the general evaluation process defined in ISO/IEC 14598-1, is not defined as a process in ISO/IEC 12207, but as an elementary function equivalent to the "check" portion of the plan-do-check-act (PDCA) cycle that is implemented by each life-cycle process. However, the general evaluation process may be implemented in any ISO/IEC 12207 process (i.e., development, maintenance, acquisition, validation); it is therefore at a different level of abstraction from the sense of "process" used in ISO/IEC 12207.

This distinction is important when implementing the general evaluation process. The acquirer needs to define both the evaluation process and the acquisition process that he/she will follow to achieve the evaluation requirements during acquisition. In the context of larger system development, the acquisition and evaluation activities to be followed need to be integrated with other development and integration activities, and identified in a project measurement plan as specified in ISO/IEC 14598-2 (in preparation); i.e., specific acquisition implementation considerations for evaluation include the following considerations:

- a software requirements specification required for evaluation can form the basis for acquisition requirements required for the request-for-proposal (-tender);
- separate preliminary evaluation activities may be needed to preselect software products and suppliers;
- supplier and product information requirements for evaluation need to be specified in the acquisition requirements or during contract preparation;
- evaluation activities can be executed as part of proposal evaluation, during monitoring of contract execution, as part of product development, as part of formal product acceptance, or after product delivery.

5.2 Inputs to the evaluation process

5.2.1 System requirements

The starting point for determining evaluation requirements for target software begins with the overall system requirements. The system requirements identify the user, user goals, tasks and characteristics, including the environment in which the product is to be used, in addition to functional and other requirements for the product or system. They form the basis for subsequent system architecture design, specification of software requirements, and software architecture design. Relevant legal and regulatory requirements need to be identified at this stage as they impact on the rigor and formality of the acquisition and evaluation processes.

During system requirements decomposition and design, system requirements are allocated to hardware and software configuration items, and to user operations including system procedures. Design activities during a system development life-cycle result in subsequent decisions to acquire or reuse "off-the-shelf" software products. Some of the evaluation work is actually a part of these design activities, since it plays a role in the decision making process. Evaluation of software products to be acquired is performed separately. During system integration and testing of the end product, software configuration items are integrated with other software, and with the hardware configuration items (Refer to ISO/IEC 12207). Figure 1 shows the larger systems engineering context for evaluation and acquisition.

Candidates for use and acquisition in this part of ISO/IEC 14598 are software products which can be integrated into a larger system as components or which can be used stand-alone. They are classified as:

a) Commercial-off-the-shelf software products;

b) Existing software products developed or acquired for other applications, or for a wide range of common applications;

c) Custom software products or modifications to existing software products.

In the case of software configuration items that are to be integrated into a larger system, software requirements need to be defined for each item. In other cases, the system and the software configuration items coincide, and may be considered equivalent.

Hardware configuration items to be acquired may contain software such as an operating system resident in firmware (i.e. ROM, PROM). When the existing software forms an integral part of the hardware in this fashion it usually needs to be evaluated with the hardware configuration item.

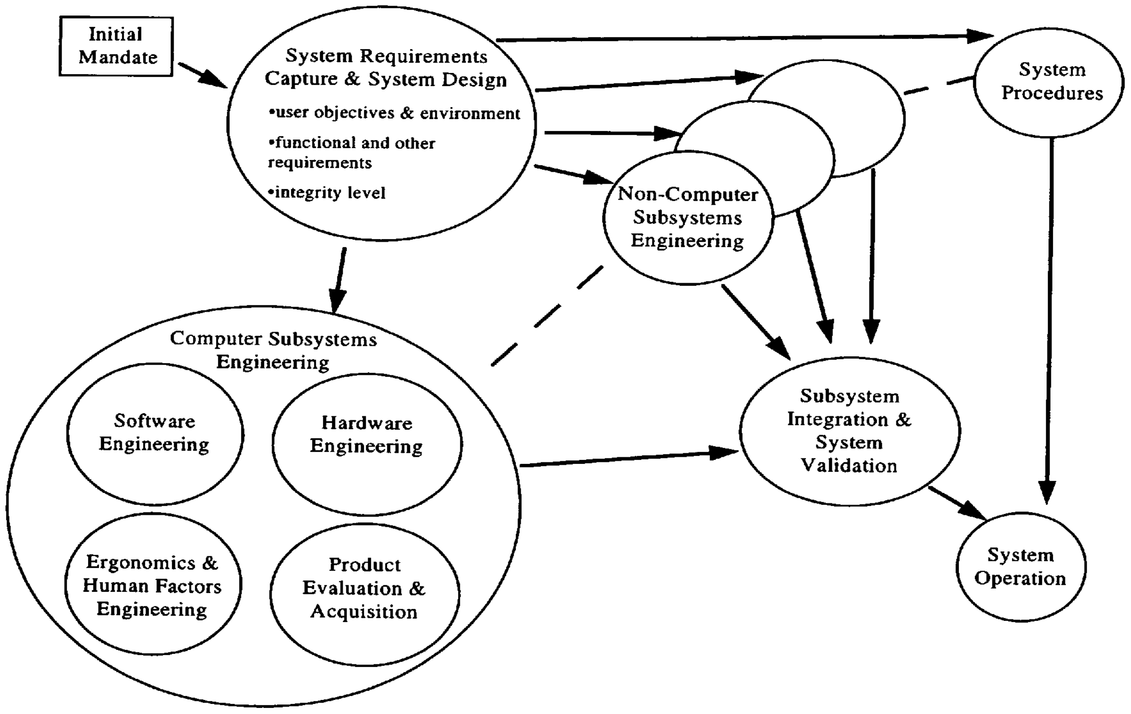
Initial Mandate
System Requirements Capture & System Design
•user objectives & environment
•functional and other requirements
•integrity level
System Procedures
Non-Computer Subsystems Engineering
Computer Subsystems Engineering
Software Engineering
Hardware Engineering
Ergonomics & Human Factors Engineering
Product Evaluation & Acquisition
Subsystem Integration & System Validation
System Operation

Figure 1 — Systems engineering context for software product evaluation and acquisition

5.2.2 Integrity level requirements

If the software is critical in containing non-trivial safety, security, financial, environmental and societal risk of a system within acceptable limits, its requred integrity level must have been established and properly documented prior to procurement and evaluation. Guidance to the integrity level determination process is given in ISO/IEC 15026. The resulting integrity level determines how the software is dealt with in the evaluation process.

5.2.3 Software requirements specification

Software requirements shall be defined using an appropriate well defined quality model. For this purpose the quality model and definitions in ISO/IEC 9126-1 should be used, unless there is a particular reason to use another model. This model defines six broad categories of characteristics of software in use: functionality, reliability, usability, efficiency, maintainability and portability. These can be further broken down into subcharacteristics which have measurable or assessable attributes.

Requirements should be defined in terms of external metrics (external metrics are defined in ISO/IEC 9126-2) which directly relate to user needs and should be documented in a requirements specification. Documenting the user needs can vary from producing an informal list of required functional and performance requirements to preparing a complete requirements specification for the product (or system if the product is embedded). The requirements specification may then form the basis for acquisition requirements used during the tendering step in the acquisition process and the basis against which subsequent product evaluation is performed.

5.2.4 Evaluations performed by others

The scope of the present evaluation process can be reduced through access to the results of evaluation activities performed by second or third parties as long as the results are trustworthy. Such evaluation activities can comprise preexisting certifications, product evaluations and/or process assessments. For example:

- software engineering processes for product development may be standardized to meet the requirements of ISO/IEC 12207, ISO 9000-3, or other sector-specific standards;
- the supplier's quality system under which the software is developed may be certified to ISO 9001 requirements by a third party;
- the software product may be evaluated by second or third parties to ISO/IEC 14598-5 or ISO/IEC 12119 requirements;
- the supplier's software process capability for acceptable product development may be assessed by third parties to ISO/IEC 15504-8 (in preparation);
- the software may be functionally evaluated as part of a larger system development phase;
- the software product may have been previously evaluated for another application with different integrity requirements;
- evaluations on the product may have been performed by other parties within the organization through informal or formal evaluation activities.

The additional costs and time required to obtain and interpret external evaluation results for the target application may affect the feasibility of this method. It may still be necessary to consult the with the evaluator or supplier in order to attain adequate confidence in the results of others.

NOTE Evaluation results of the supplier software engineering process, supplier quality system, or supplier capability alone are not sufficient criteria to demonstrate that a software product contains the required quality characteristics. Other product evaluation methods [e.g., such as those that specifically measure factors and attributes of quality that are appropriate to the requirements of the end-user] need to be executed.

5.3 Tailoring

The evaluation process can be applied to a wide range of acquisition requirements, integrity requirements, and evaluator objectives. For example:

- acquirers of software packages may wish to evaluate a software package using only ISO/IEC 12119;
- acquirers of software products may use ISO/IEC 14598-5 for independent evaluation;
- a small or sole acquirer may require a less formal evaluation process with minimum documentation of the evaluation;
- for consumer software the evaluation process objective may simply be to select, test and acquire one product from among a number of similar products. The formal acquisition process is then reduced to outright purchase, and does not include contract preparation.

The evaluation process should have the flexibility to accommodate the uniqueness of each application, to avoid unnecessary work or work that adds no value, and to provide a practical means of establishing the requisite confidence in the software. The required integrity level of the software largely determines the rigor and formality of the evaluation process.

The acquisition process can also be tailored as for the evaluation process using the tailoring guidance given in ISO/IEC 12207 and the required integrity level for the specific software product to be acquired. Acquisition of complete software systems with high integrity requirements will usually invoke the full set of acquisition activities and tasks, along with corresponding supply process activities and tasks, that are specified in ISO/IEC 12207. Generally, as the integrity level increases, so should the amount of rigor, and the number of activities and tasks associated with the acquisition process.

Table B.1 in Annex B shows an example of tailoring of integrated acquisition and evaluation process activities according to the target software integrity level requirements.

6 Evaluation during acquisition of “off-the-shelf” software products

The general software product evaluation process defined in ISO/IEC 14598-1 consists of four major steps, which are specifically implemented and refined to focus on the evaluation of end product quality during acquisition of “off-the-shelf” products in this part of ISO/IEC 14598. However, this does not preclude evaluation of intermediate products for specific quality characteristics. Hence, the details in the implementation of the steps differ, but are not inconsistent, with the details described in ISO/IEC 14598-1. The evaluation process is summarized in Table 1 below in terms of its steps and key tasks, as well as inputs and outputs.

Table 1 — Evaluation Process During Acquisition of "Off-the-Shelf" Products

Inputs	Evaluation Step	Key Tasks	Outputs
System/ Software requirements	Establish evaluation requirements (subclause 6.1)	Specify objectives, purpose, and scope. Specify rigor of evaluation. Identify inputs to evaluation. Identify evaluation performed, or to be performed by others. Identify the acquisition process to be followed and how evaluation input requirements are to be communicated to the supplier.	Evaluation Requirements Specification
Evaluation Requirements	Specify the evaluation (subclause 6.2)	Select metrics that correlate to the characteristics of the software product. Establish rating levels. Select the most effective set of evaluation methods. Establish procedures for summarizing the results of the evaluation of different quality characteristics and other aspects that contribute to the assessment of quality of a software product in a particular environment.	Evaluation Specification
Evaluation Specification	Design the evaluation (subclause 6.3)	Prepare an evaluation plan describing the evaluation methods, and the schedule for evaluation. Identify the tie points between evaluation activities and acquisition activities.	Evaluation Plan
Evaluation Plan	Execute the evaluation (subclause 6.4)	Conduct the selected evaluation activities, and analyze and record the results to determine the suitability of the software product(s). Analyze the impact of identified deficiencies and options to regulate the use of the product. Draw conclusions with respect to the acceptability of the product and the ultimate decision to buy or not to buy.	Evaluation Records and Results

6.1 Step 1 - Establish evaluation requirements

6.1.1 Establish the purpose and scope of the evaluation

The evaluation process ***shall*** establish:

a) a set of software quality requirements using the quality model and definitions in ISO/IEC 9126-1, against which the software product can be objectively evaluated for fitness for use;

b) the appropriate priority which should be placed on the software quality characteristics;

c) a systematic basis for the evaluation appropriate for the integrity level of the application, which involves establishing requirements as to the level of rigor or detail required in the evaluation activities, as well as the inputs to, and the outputs from the evaluation process;

NOTE Figure 2 provides an overview of the software product evaluation process. It shows the different views of the inputs to the evaluation process and resulting outputs from the evaluation process, from an acquirer's perspective.

d) the acquisition process to be followed and how evaluation input requirements are to be communicated to the supplier;

NOTE See example of a combined evaluation and acquisition process in Figure C.1 in Annex C.

e) the scope, purpose and objectives of the evaluation by considering:

   1) whether the software product will be used for a specific application, for a collection of specific applications, or for a generic range of applications;

   2) whether any evaluation has been done by second or third parties, or whether any evaluation activities are planned to be performed later.

6.1.2 Specify evaluation requirements

The evaluation requirements specification ***should*** identify:

a) users and their goals, tasks, and characteristics, and the user environment for the product;

b) the integrity level of the software application (risk incurred in case of software error) and hence the level of rigor required for the evaluation process;

c) the regulatory requirements (what documentation is required to provide the regulator (if any) with assurance of the product's quality) (see *compliance* in ISO 9126-1);

d) the product boundaries and interfaces, including interfacing requirements (i.e. the type of data passed through the interface, the data format, the interface access mechanisms, failure/error handling, timing issues, interface behavior issues, and interface state dependencies and transitions) to the software product (see *interoperability* in ISO 9126-1);

e) integration requirements if the product is part of a larger system requiring integration with other components or products;

f) software quality requirements which include:

   1) distinction between mandatory and optional requirements;

   2) all assumptions, exceptions, limitations, exclusions, or unresolved issues needed to interpret or understand the requirements;

   3) user requirements for all important quality characteristics and their priority (i.e. if maintainability is considered important, then identify the specific maintainability requirements);

   4) all design and environmental constraints; i.e. functional or performance limitations imposed by the use of the software product, the level and complexity of integration of the software product with other existing software, custom software, and the hardware in the user application;

   5) all project management constraints; i.e. availability of resources and expertise to perform the evaluation activities, schedule and budget allowance, possible dependencies or risks, key assumptions, or assumptions about the evaluation effort itself;

   6) rationale for the use of a quality model other than the one defined in ISO/IEC 9126-1.

g) supplier services to be assessed; i.e., support capability, application development capability, and training capability;

h) special requirements to be assessed; i.e. specific technical feasibility issues or design implementation questions;

i) evaluation requirements that are consistent with each other (i.e. no conflicting requirements) and consistent with the integrity level of the application;

j) whether the product will be reused in future applications and the documentation necessary to support any future evaluations of the product;

k) the acquisition process, and information required from suppliers during tendering;

l) evaluations performed by other second or third parties that could be used to reduce the evaluation effort for the product.

NOTE The level of detail and completeness of the evaluation requirements specification directly affects the level of completeness of the evaluation, i.e., an evaluation based on only preliminary requirements cannot be considered a full evaluation, although the results may be used to direct further work in other phases, anticipate problems, or eliminate certain software products or suppliers. These would normally be done prior to the main evaluation activities as a part of the appropriate design or planning activity. Some evaluation work may also be required before the requirements are finalized.

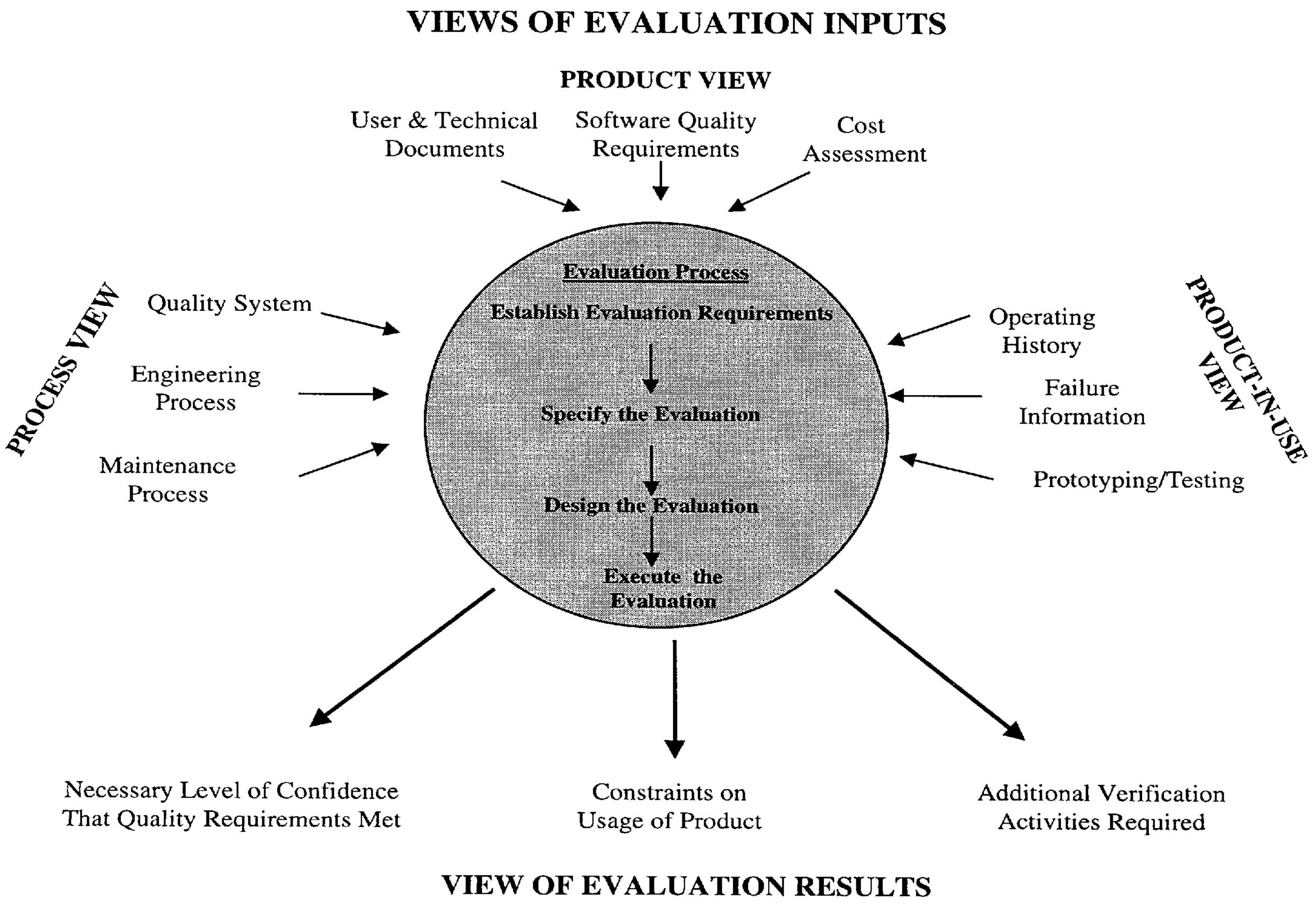
VIEWS OF EVALUATION INPUTS
PRODUCT VIEW
User & Technical Documents
Software Quality Requirements
Cost Assessment
PROCESS VIEW
Quality System
Engineering Process
Maintenance Process
Evaluation Process
Establish Evaluation Requirements
Specify the Evaluation
Design the Evaluation
Execute the Evaluation
Operating History
Failure Information
Prototyping/Testing
PRODUCT-IN-USE VIEW
Necessary Level of Confidence That Quality Requirements Met
Constraints on Usage of Product
Additional Verification Activities Required
VIEW OF EVALUATION RESULTS

Figure 2 — Software product evaluation process overview from acquirer's perspective

6.2 Step 2 - Specify the evaluation

The evaluation specification ***should*** be documented so that the evaluation process can be repeated by appropriately qualified personnel with repeatable results.

6.2.1 Select metrics

The evaluation specification ***should*** identify:

a) the characteristics of the product to be evaluated;

b) the external metrics that correlate to measurable aspects of quality when the software is used (Table B.2 in Annex B shows an example of external metrics and possible acceptance criteria. Notice it is up to the user to specify actual threshold acceptance numbers based on experience as there are no hard and fast rules for some of these numbers);

c) the "quality in use" metrics correlating to the user's view of the quality of the system containing the software (see example of quality in use metrics in Table B.3 in Annex B);

d) sufficient criteria for metrics to describe their acceptable range (e.g. how much operating history is necessary to provide a reasonable degree of assurance for a given quality characteristic and a given integrity level (see Annex D.4 and D.5 for specific details on operating history));

e) any packaged evaluation modules;

f) the level of coverage relative to the evaluation requirements that is necessary after review of any previous evaluations conducted by others;

g) checklists to be answered by the evaluation;

h) a list of examples which would help to answer the questions;

i) test cases to be used;

j) data to be collected and analyzed, and their format;

k) the specific evaluation methods to be used, which include review or assessment of one or more of:

   1) Software product user and technical documentation (including on-line documentation) (see Annex D.1);

   2) Software product evaluation based on supplier courses and training (see Annex D.2);

   3) Software engineering process, including intermediate software products (see Annex D.3);

   4) Product operating history with supplier (see Annex D.4);

   5) Product operating history with customers (see Annex D.5);

   6) Supplier capability, support, and quality system (see Annex D.6);

   7) Prototyping or other evaluation methods (see Annex D.7);

   8) Product deficiency lists and related information (usually found on web sites).

l) the methods for assessing the evaluation results;

m) suitable methods of ranking the assessments to allow selection, when selecting a product from among similar products;

n) any rating schemes useful for comparing more than one software product. The rating scheme may be weighted in accordance with the priority of the quality characteristics.

6.2.2 Select the evaluation methods

A combination of evaluation methods ***should*** be specified to allow selection of the product or to establish the product's fitness for use. Areas to be evaluated include:

a) whether some of the considerations may be mutually conflicting (e.g. "Are costs of the selected [assessment] methods within budget?" may not be compatible with "Do the methods collectively address all of the evaluation requirements?"). In this case it is up to the evaluator to make the necessary tradeoffs based on the prioritization of evaluation requirements;

NOTE Table B.4 in Annex B shows an example of ranking evaluation methods for software quality characteristics by cost and effectiveness.

b) whether the evaluation provides adequate coverage or scope in the combination of methods chosen considering:

   1) how to demonstrate that software meets its specifications;

   2) overlapping of coverage of methods to provide additional confidence;

   3) whether the set of activities, as a whole, provides an acceptable level of assurance that there is complete coverage for those software quality characteristics of concern;

   4) the degree to which the methods complement each other;

   5) the effectiveness and objectivity of each method in addressing a variety of characteristics;

   6) the variety of distinct approaches among the evaluation methods (e.g. some review, some analysis, and some testing based methods);

   7) taking credit for any evaluation activities on the application that will ultimately be conducted as part of the overall system development life-cycle;

   8) crediting evaluations performed by others;

c) use of "informal" preliminary evaluation activities like reviews or surveys or peer/user anecdotal experience, trade journal product reviews, accessible product user documentation, or database repositories of product reviews, in order to narrow the selection of products considered functionally suitable for further evaluation.

6.2.3 Evaluations performed by others

Before crediting evaluations performed by others, the following ***should*** be considered:

a) whether the evaluation addressed the evaluation requirements with a degree of rigor consistent with the integrity level of the application;

b) whether the evaluation report identified the version of the software product being assessed, the extent of the evaluation, the decision criteria used, and the conclusions reached;

c) whether the evaluation report identified any deficiencies in the software product or the software engineering process, recommended any corrective action to address those deficiencies, and whether the corrective action was carried out;

d) whether the evaluator had appropriate expertise, including:

   1) experience in performing evaluation and analysis;

   2) experience in software quality relative to internationally accepted standards;

   3) expertise in software engineering issues;

   4) total independence from the supplier.

6.3 Step 3 - Design the evaluation

The product evaluation plan ***should*** identify:

a) whether the supplier or third party is willing and able to provide access to the required documentation, equipment, tools, software, courses and/or training, and the costs associated with this;

b) any conditions associated with the provision of access to any confidential or proprietary information;

c) whether the supplier or third party is willing and able to provide access to individuals with the correct expertise to answer questions, and what are the costs associated with this, including travel costs;

d) the expertise that is required by the evaluator to conduct the evaluation based on the evaluation requirements, and any costs associated with obtaining this particular expertise;

e) any pre-tests required to establish that a product is fit for full scale testing;

f) any costs associated with providing the test environment (e.g. test hardware, support equipment and tools, specialized personnel) for performing the evaluation;

g) responsibility for the evaluation and the required schedule;

h) any limitation or deficiency in the evaluation method in providing assurance of quality, and whether that limitation or deficiency is covered elsewhere in the plan; e.g. the evaluation method is unable to cover all subcharacteristics for a specific quality characteristic;

i) any interdependencies between the various evaluation methods used; i.e., order dependencies (information obtained in one test may be useful in another), establishing an optimal sequence of evaluation methods;

j) the resources required, the cost for the total evaluation, and the cost of each evaluation method;

k) the tie points between evaluation activities and acquisition activities (See example of a combined evaluation and acquisition process in Figure C.1 in Annex C);

l) the decision points in the evaluation process which determine when and why the evaluation should be considered complete (i.e. acceptance or rejection criteria) and should be stopped;

m) the following for each evaluation activity planned:

   1) the procedures and techniques which should be followed;

   2) the information inputs and outputs and documentation required;

   3) any format and content requirements on any documentation produced;

n) the rationale, justification, and assumptions behind any unusual or exceptional decisions made in defining the evaluation plan are documented;

o) evaluation tools;

p) the procedures for developing and validating metrics, and for standardizing the evaluation process, metrics and measures.

Note 1 ISO/IEC 14598-6 defines a concept of an evaluation module that systematically collects all information necessary to carry out an evaluation of a specific aspect of a quality characteristic applying a specific evaluation technique or method.

Note 2 In scheduling the evaluation methods, it is important to recognize that a high degree of interdependence exists between the various evaluation methods, i.e. information obtained by one method may influence the focus of another method. As the nature of evaluation is iterative, issues may be revisited as information is gained. Therefore the evaluation plan will likely be altered as the evaluation is conducted. For example, it may be common for more detailed levels of evaluation to be considered either as unnecessary, or as an additional requirement, once the evaluation progresses.

Note 3 The evaluation of software products may be carried out in stages at different points in a development life cycle, or all at once, at one point in the life cycle. Different individuals or groups may be responsible for performing different parts of the evaluation. When the evaluation is done in stages, the steps of the evaluation activity are repeated in each stage until no further work is required (See example of a staged evaluation process in Annex E).

6.4 Step 4 - Execute the evaluation

6.4.1 Execute the evaluation methods

a) The evaluation ***should*** be performed, documented and analyzed to:

1) establish an appropriate degree of confidence that the software product is able to meet the evaluation requirements;

2) identify any specific deficiencies with regard to the evaluation requirements and any additional evaluations needed to determine the scope of those deficiencies;

3) identify any special limitations or conditions placed on the use of the software product;

4) identify any weaknesses or omissions in the evaluation itself and any additional evaluation that is needed;

5) identify any options for the use of the software product uncovered by the evaluation;

b) The records for the execution of the evaluation should identify:

1) the execution of the evaluation according to the procedures specified in the evaluation plan;

2) the stepwise execution of evaluation procedures (including data used, set-up process, and any state information), the evaluation results (answers to all questions and references to the sources of the answers), and the software product version numbers;

3) any limitations, constraints, deficiencies, or exclusions in an evaluation activity, including their impact on the use, configuration, modification, or general maintenance of the software product over time;

4) the evaluators and their qualifications;

5) any differences between the product versions assessed and the corresponding evaluation inputs; i.e., documentation or courses;

6) resolutions or "work-arounds" in the event of a deficiency.

6.4.2 Analyze the evaluation results

The records for analysis of the evaluation activities ***should*** identify:

a) each deficiency, any relevant analysis, and how each deficiency was resolved. Resolution of deficiencies may include the fact that:

1) one of the other evaluation methods has provided assurance that the deficiency is not major; e.g., extensive operating history may make up for a deficient software engineering process;

2) a satisfactory "work-around" can be found to alleviate the impact of the deficiency; e.g., modification to the product, disable or remove unneeded functionality, regenerate missing design requirements using reverse engineering;

3) the original requirement is not mandatory and the deficiency can be accepted;

4) the deficiency is acceptable provided that the use of the software product will be controlled by specific conditions or limitations;

5) additional evaluation work is required to resolve the deficiency or gaps in the evaluation;

b) any additional evaluations performed to resolve any identified deficiencies:

1) to determine the scope or impact of a deficiency;

2) to establish confidence that there is no deficiency;

3) to verify that a work-around is technically feasible and/or suitable and acceptable;

4) to verify the correct and acceptable performance of the software once a design change or changes have been made to correct deficiency;

c) in a case where it is necessary to limit or control the use of the software product, whether the limitation:

1) interferes with the software product meeting the mandatory requirements of the application;

2) impacts on the application's design, budget, and schedule;

3) requires additional evaluation work;

4) introduces any possibility of failure in the application;

d) any exclusions from scope of evaluation and/or restrictions on the results for each evaluation, such as:

1) 'This evaluation does not include a detailed review of the functionality of the product.' or

2) 'This software product is deemed to be qualified to the required integrity level provided a full evaluation of the required functionality for the product is completed successfully.'

e) the integrated results of all the evaluation activities to allow an overall conclusion for the evaluation of the software product to be made.

6.4.3 Draw conclusions

The conclusions ***should*** state whether the software product is adequate and appropriate for use in the intended application, taking into consideration the application integrity level and the actual evaluation requirements. If it is not possible to use the software product "as is", due to deficiencies discovered or due to lack of evaluation information, then it is necessary to make recommendations to perform further evaluation or to control or limit the use of the software product in its target application.

The conclusions may be formalized using a "statement of requirements compliance" which would describe for each of the specific requirements the feature(s), function(s), or service(s) of the software product used to meet that requirement, and the evaluation method (s) that provide adequate confidence that the requirement has been met. Potential design strategies such as implementation of design diversity, redundancy in configuration, interface integrity checks and recovery techniques may compensate for deficiencies or potential failures of the software product.

It is possible that the evaluation may result in a decision not to accept the software product for use, or a decision not to attempt to comply with the evaluation requirements, and in a recommendation to re-evaluate alternative approaches. The ultimate decision is to buy or not to buy.

The decision to buy can then result in a contract to purchase the software product with possible additional evaluation in the form of product acceptance testing.

The decision not to buy can result in possible alternatives which include modifying the software product, developing a custom software product, or changing the requirements.

7 Evaluation during acquisition of custom software and modifications to existing software

This clause focuses on the application of the evaluation process during acquisition of custom software or modifications to existing software. Figure C.2 in Annex C shows an example of the combined acquisition and evaluation process for custom software and modifications to existing software.

7.1 Step 1 - Establish evaluation requirements

The process for establishing the evaluation requirements defined in 6.1 also applies in this case. The evaluation requirements form the basis for acquisition requirements that can be sent out as part of a request for a proposal to pre-selected suppliers. For modifications to existing software, evaluation needs to focus primarily on the changed parts of the software product and their interfaces.

Preliminary evaluation should be performed to pre-select suppliers on the basis of their capability, quality program and software engineering process prior to a request for proposal.

7.2 Step 2 - Specify the evaluation

6.2 specifies the evaluation for 'off-the-shelf' software which also applies to the evaluation for custom software and modifications for existing software. However, additional measurement is required as part of the supplier development process to predict end product quality based on measuring the quality of intermediate products. ISO/IEC 14598-3 provides the requirements and guidance for measuring intermediate product quality during development.

7.3 Step 3 - Design the evaluation

6.3 applies to the acquisition of custom software and modifications to existing software with the following additional considerations.

Selecting a supplier during tendering may require that the supplier upgrade the software engineering or maintenance process to that required to achieve the target integrity requirements prior to software development or modification.

Required evaluation activities then become part of the supplier performance process, i.e., during the process of conducting supplier development, verification, joint review and audit, testing and validation activities. These requirements are specified in the quality or development plan that the supplier is required to follow. The acquirer monitors supplier performance to the plan and establishes the requirements for the plan in the contractual agreement between the supplier and the acquirer.

7.4 Step 4 - Execute the evaluation

The requirements for execution of the evaluation in 6.4 also apply here, except that actual evaluation is carried out through supplier performance and acquirer monitoring activities according to the quality plan. A successful software product acceptance test is a necessary criterion before product delivery to the acquirer. The acquirer may decide to make additional modifications to the software product to address deficiencies found during evaluation, before using the product.

Annex A
(informative)

Definitions from other standards

For the purposes of this part of ISO/IEC 14598, definitions are from ISO/IEC 14598-1: 1997 unless otherwise stated.

A.1 acquirer
an organization that acquires or procures a system, software product or software service from a supplier.

NOTE The acquirer could be one of the following: buyer, customer, owner, user, purchaser.

[ISO/IEC 12207: 1995]

A.2 acquisition
the process of obtaining a system, software product or software service. [ISO/IEC 12207: 1995]

A.3 attribute
a measurable physical or abstract property of an entity.

NOTE Attributes can be internal or external.

A.4 audit
conducted by an authorized person for the purpose of providing an independent assessment of software products and processes in order to assess compliance with requirements. [ISO/IEC 12207: 1995]

A.5 baseline
a formally approved version of a configuration item, regardless of media, formally designated and fixed at a specific time during the configuration item's life cycle. [ISO/IEC 12207: 1995]

A.6 CASE tool
a software product that can assist software engineers by providing automatic support for software life-cycle activities as defined in ISO/IEC 12207: 1995.

NOTE 1 A CASE tool may provide support in only selected functional areas or in a wide variety of functional areas.

NOTE 2 CASE tools may be used in several modes:

- As stand alone tools; in this case, only compatibility with environmental elements should be addressed;
- In small groups which communicate directly with one another; it may be supposed that integration is predefined, perhaps proprietarily;
- In the presence of a larger framework of the SEE; in this case the ability of the tool to use the relevant services of the framework should be addressed.

[ISO/IEC 14102: 1995]

A.7 configuration item
an entity within a configuration that satisfies an end use function and that can be uniquely identified at a given reference point. [ISO/IEC 12207: 1995]

A.8 contract
a binding agreement between two parties, especially enforceable by law, or a similar internal agreement wholly within an organization, for the supply of software service or for the supply, development, production, operation, or maintenance of a software product. [ISO/IEC 12207: 1995]

A.9 developer
an organization that performs development activities (including requirements analysis, design, testing through acceptance) during the software life cycle process. [ISO/IEC 12207: 1995]

A.10 direct measure
a measure of an attribute that does not depend upon a measure of any other attribute.

A.11 evaluation module
a package of evaluation technology for a specific software quality characteristic or subcharacteristic which includes:

- evaluation methods and techniques;
- inputs to be evaluated;
- data to be measured and collected;
- acceptance criteria;
- supporting procedures and tools.

[ISO/IEC 14598-6:new]

A.12 external measure
an indirect measure of a product derived from measures of the behaviour of the system of which it is a part.

NOTE 1 The system includes any associated hardware, software (either custom software or off-the-shelf software) and users.

NOTE 2 The number of faults found during testing is an external measure of the number of faults in the program because the number of faults are counted during the operation of a computer system.

NOTE 3 External measures can be used to evaluate quality attributes closer to the ultimate objectives of the design.

A.13 external quality
the extent to which a product satisfies stated and implied needs when used under specified conditions.

A.14 failure
the termination of the ability of an item to perform a required function or its inability to perform within previously specified limits. [ISO/IEC 15026: 1997]

A.15 fault
an incorrect step, process or data definition in a computer programme.

NOTE This definition is taken from IEEE 610.12-1990.

A.16 firmware
the combination of a hardware device and computer instructions or computer data that reside as read-only software on the hardware device. The software cannot be readily modified under program control. [ISO/IEC 12207: 1995]

A.17 implied needs
needs that may not have been stated but are actual needs when the entity is used in particular conditions.

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

A.18 indicator
a measure that can be used to estimate or predict another measure.

NOTE 1 The predicted measure may be of the same or a different software quality characteristic.

NOTE 2 Indicators may be used both to estimate software quality attributes and to estimate attributes of the production process. They are imprecise indirect measures of the attributes.

A.19 indirect measure
a measure of an attribute that is derived from measures of one or more other attributes.

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

A.20 integrity level
a denotation of a range of values of a property of an item necessary to maintain system risks within acceptable limits. For items that perform mitigating functions, the property is the reliability with which the item must perform the mitigating function. For items whose failure can lead to a threat, the property is the limit on the frequency of that failure. [ISO/IEC 15026: 1997]

A.21 intermediate software product
a product of the software development process that is used as input to another stage of the software development process.

NOTE In some cases an intermediate product may also be an end product.

A.22 internal measure
a measure of the product itself, either direct or indirect.

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

A.23 internal quality
the totality of attributes of a product that determine its ability to satisfy stated and implied needs when used under specified conditions.

NOTE 1 The term "internal quality", used in this standard to contrast with "external quality", has essentially the same meaning as "quality" in ISO 8402: 1994.

NOTE 2 The term "attribute" is used with the same meaning as the term "characteristic" used in 4.1.1, as the term "characteristic" is used in a more specific sense in ISO/IEC 9126.

A.24 measure (verb)
make a measurement.

A.25 measure (noun)
the number or category assigned to an attribute of an entity by making a measurement.

A.26 measurement
the use of a metric to assign a value from a scale (which may be a number or category) to an attribute of that entity.

NOTE Measurement can be qualitative when using categories. For example, some important attributes of software products, e.g. the language of a source program (ADA, C, COBOL, etc.) are qualitative categories.

A.27 metric
the defined measurement method and the measurement scale.

NOTE 1 Metrics can be internal or external, and direct or indirect.

NOTE 2 Metrics include methods for categorising qualitative data.

A.28 "off-the-shelf" product
product that is already developed and available; usable either "as is" or with modification. [ISO/IEC 12207:1995]

A.29 operator
an organization that operates the system. [ISO/IEC 12207:1995]

A.30 quality
the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs.

NOTE 1 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).

NOTE 2 In ISO/IEC 14598 the relevant entity is a software product.

[ISO 8402 :1994]

A.31 quality evaluation
systematic examination of the extent to which an entity is capable of fulfilling specified requirements.

NOTE The requirements may be formally specified, as when a product is developed for a specific user under a contract, or specified by the development organisation, as when a product is developed for unspecified users, such as consumer software, or the requirements may be more general, as when a user evaluates products for comparison and selection purposes.

[ISO 8402: 1994]

A.32 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 1 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 2 The definition of quality in use in 14598 does not currently include the new characteristic of "safety".the extent to which a product used by specified users meets their needs to achieve specified goals with effectiveness, productivity and satisfaction in specified contexts of use.

NOTE 3 Usability is defined in ISO 9241-11 in a similar way to the definition of quality in use in ISO/IEC 9126-1. Quality in use may be influenced by any of the quality characteristics, and is thus broader than usability, which is defined in ISO/IEC 9126-1 in terms of understandability, learnability, operability, attractiveness and compliance.

[ISO/IEC 9126-1.1)]

A.33 quality model
the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality.

A.34 rating
the action of mapping the measured value to the appropriate rating level. Used to determine the rating level associated with the software for a specific quality characteristic.

A.35 rating level
a scale point on an ordinal scale which is used to categorise a measurement scale.

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

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

A.36 request for proposal [tender]
a document used by the acquirer as the means to announce its intention to potential bidders to acquire a specified system, software product or software service. [ISO/IEC 12207: 1995]

A.37 scale
a set of values with defined properties.

NOTE Examples of scales are: a nominal scale which corresponds to a set of categories; an ordinal scale which corresponds to an ordered set of scale points; an interval scale which corresponds to an ordered scale with equidistant scale points; and a ratio scale which not only has equidistant scale point but also possess an absolute zero. Metrics using nominal or ordinal scales produce qualitative data, and metrics using interval and ratio scales produce quantitative data.

1) To be published. Until this part is published ISO/IEC 9126 :1991 should be used.

A.38 software
all or part of the programs, procedures, rules, and associated documentation of an information processing system.

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

[ISO/IEC 2382-1: 1993]

A.39 software product
the set of computer programs, procedures, and possibly associated documentation and data.

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

[ISO/IEC 12207: 1995]

A.40 supplier
an 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.

NOTE 1 The term "supplier" is synonymous with contractor, producer, seller, or vendor.

NOTE 2 The acquirer may designate a part of its organization as supplier.

[ISO/IEC 12207: 1995]

A.41 system
an integrated composite that consists of one or more of the processes, hardware, software, facilities and people, that provides a capability to satisfy a stated need or objective. [ISO/IEC 12207: 1995]

A.42 user
an individual that uses the software product to perform a specific function.

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

A.43 validation
confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled.

NOTE 1 In design and development, validation concerns the process of examining a product to determine conformity with user needs.

NOTE 2 Validation is normally performed on the final product under defined operating conditions. It may be necessary in earlier stages.

NOTE 3 "Validated" is used to designate the corresponding status.

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

[ISO 8402:1994]

A.44 verification
confirmation by examination and provision of objective evidence that specified requirements have been fulfilled.

NOTE 1 In design and development, verification concerns the process of examining the result of a given activity to determine conformity with the stated requirement for that activity.

NOTE 2 "Verified" is used to designate the corresponding status.

[ISO 8402: 1994]

Annex B
(informative)

Tables

Table B.1 — Example tailoring of evaluation / acquisition activities per target software integrity

Evaluation/ acquisition Activities	Target software integrity		
	low	medium	high
prepare software requirements specification	maybe informally	yes	yes
prepare acquisition requirements	no	yes	yes
preselect suppliers and prepare request for tender	maybe	maybe	maybe
assess supplier capability	no	no	yes
evaluate supplier software process	no	no	yes
evaluate executable product	yes	yes	yes
evaluate product operating history with supplier	maybe informally	yes	yes
evaluate product operating history with customers	maybe informally	yes	yes
prepare contract	no	yes	yes
perform formal product acceptance test	no	yes	yes
perform additional evaluation	no	maybe	yes

NOTE 1 An example of low integrity software is "shrink-wrapped" word processing or spread sheet software. However, if the spread sheet software is used to calculate critical safety parameters for a chemical processing plant, the integrity level of the software necessarily increases unless it is reduced by mitigating factors, i.e., alternate means of calculation.

NOTE 2 An example of medium integrity software is graphical users interface (GUI) software used in a plant process monitoring system with no control capability.

NOTE 3 An example of high integrity software is Unix operating system software used in an air traffic control system.

Table B.2 — Example specifying software quality characteristics, subcharacteristics, external metrics

Target software integrity	Prioritized quality characteristics	Selected Subcharacteristic	Selected external metric	Possible acceptance criteria
LOW	1. Functionality	Accuracy	number of accurate results against precalculated expected results.	95%
	2. Usability	Operability	the number of evaluated clear messages received against total number of messages reviewed.	80%
	3. Portability	Installability	number of modules to be recompiled against the total number of modules when transferring to a new platform.	< 6 modules
	4. Efficiency	Time Behaviour	elapsed time between the start of system action and receipt of system response.	<5 seconds
	5. Reliability	Fault Tolerance	number of failures to detect erroneous input against number of erroneous inputs entered.	25%
	6. Maintainability (not required)	-not required-	-not required-	-
HIGH	1. Reliability	Availability	mean time between failures for a specified operating period.	> 6 months
	2. Functionality	Suitability	the number of mandatory requirements met in software requirements specification against total number of mandatory requirements.	100%
	3. Maintainability	Changeability	the number of modules required to be changed for identified likely changes	1
	4. Efficiency	Resource Utilization	the percentage of CPU load over a specified operating period under worst case operation conditions.	80%
	5. Usability	Understandability	the length of time required for specified users to learn how to use the software to produce specified results.	< 10 minutes
	6. Portability (not required)	- not required-	-not required-	-

Table B.3 — Example specifying Software Quality in Use Metrics

User	External Quality Goal Characteristics	Quality in Use Attributes	Example Quality In Use Measures	Acceptance Criteria
Operator, end user	functionality, reliability, usability, efficiency	Effectiveness (relating goals of using system/product to accuracy and completeness with which these goals can be achieved)	-number of mistakes made in interpreting display results - number of deviations from a specified format in transcribing document	- the closer to zero the better - the closer to zero the better
		efficiency (level of effectiveness achieved related to the expenditure of mental or physical effort)	- task time and labour costs of user's time - cost of the resources and the equipment used - cost of any training required by the user	- the lower the better - the lower the better - the lower the better
		Satisfaction (comfort and acceptability of use)	- user interaction satisfaction questionnaires. - ratio: positive to negative comments during use. - cognitive workload questionnaires.	- the higher the better - the higher the better - population average per user type
Maintainer, installer	maintainability, portability	Effectiveness (relating goals of using system/product to accuracy and completeness with which these goals can be achieved)	- number of modules affected per change request. - number of successful installations over the number of installation attempts	- the lower the better - the closer to 1.0 the better
		efficiency (level of effectiveness achieved related to the expenditure of mental or physical effort)	- task time and labour costs of user's time. - cost of the resources and the equipment used. - cost of any training required by the user.	- the lower the better - the lower the better - the lower the better
		Satisfaction (comfort and acceptability of use)	- user interaction satisfaction questionnaires. - ratio: positive to negative comments during use - cognitive workload questionnaires.	- the higher the better - the higher the better - population average per user type

Table B.4 — Example of Cost-Effectiveness Ranking of Evaluation Methods

Evaluation	Cost	Effectiveness Ranking of Evaluation Methods per Software Quality Characteristic					
Method	Ranking	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability
End Product Documents, Courses and Training	Low	High	Low	High	Medium	Low	High
Intermediate Products	High	Low	Medium	Low	Medium	High	High
Operating History - Supplier	Medium	Medium	High	Low	Low	Medium	Medium
Operating History – Customer	Medium	High	Medium	High	Medium	High	High

NOTE 1 This table presents a *qualitative* estimated ranking of the relative effectiveness and the gross relative cost of evaluation methods relative to specific product quality characteristics. This effectiveness rating assumes that the evaluation is conducted successfully and to the appropriate degree of rigor. This table can be used to select the target inputs that need to be evaluated to fully assess the adequacy of product quality characteristics against those specified in the software requirements specification. The evaluations required can be used to estimate total cost of product evaluation.

NOTE 2 Cost and effectiveness rankings shown are relative and dependent on the extent of the evaluation conducted.

Annex C
(informative)

Figures

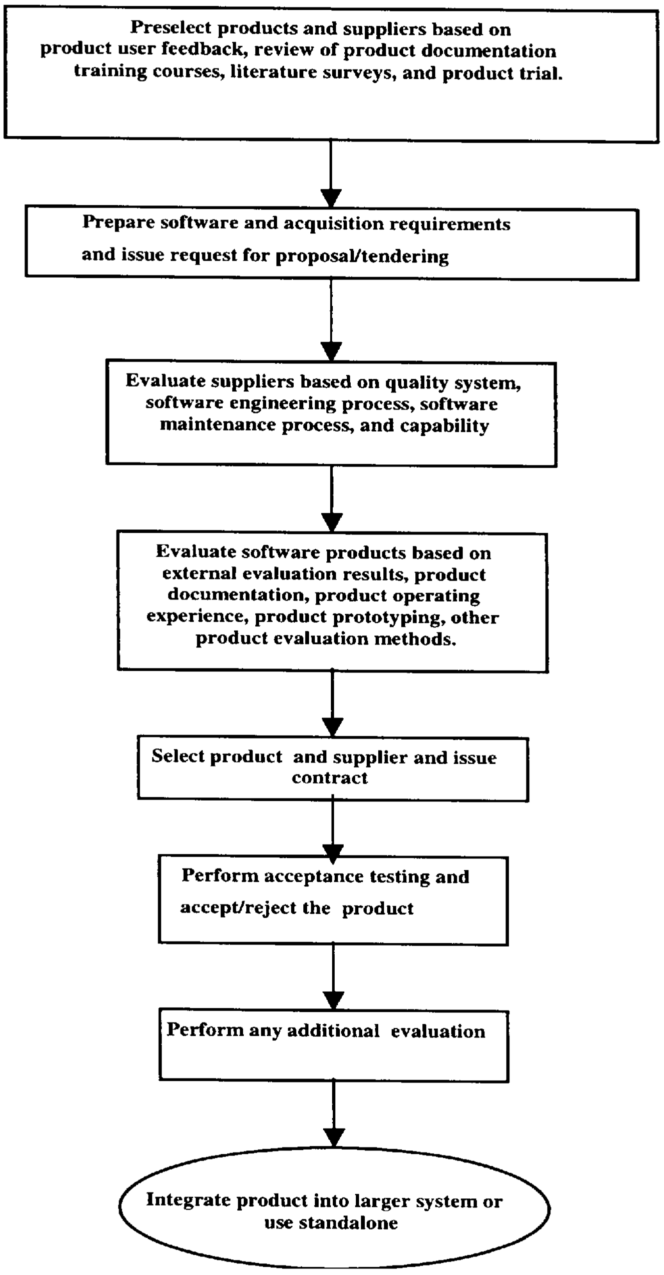
Preselect products and suppliers based on product user feedback, review of product documentation training courses, literature surveys, and product trial.
Prepare software and acquisition requirements and issue request for proposal/tendering
Evaluate suppliers based on quality system, software engineering process, software maintenance process, and capability
Evaluate software products based on external evaluation results, product documentation, product operating experience, product prototyping, other product evaluation methods.
Select product and supplier and issue contract
Perform acceptance testing and accept/reject the product
Perform any additional evaluation
Integrate product into larger system or use standalone

Figure C.1 — Example evaluation/acquisition process for "off-the-shelf" products

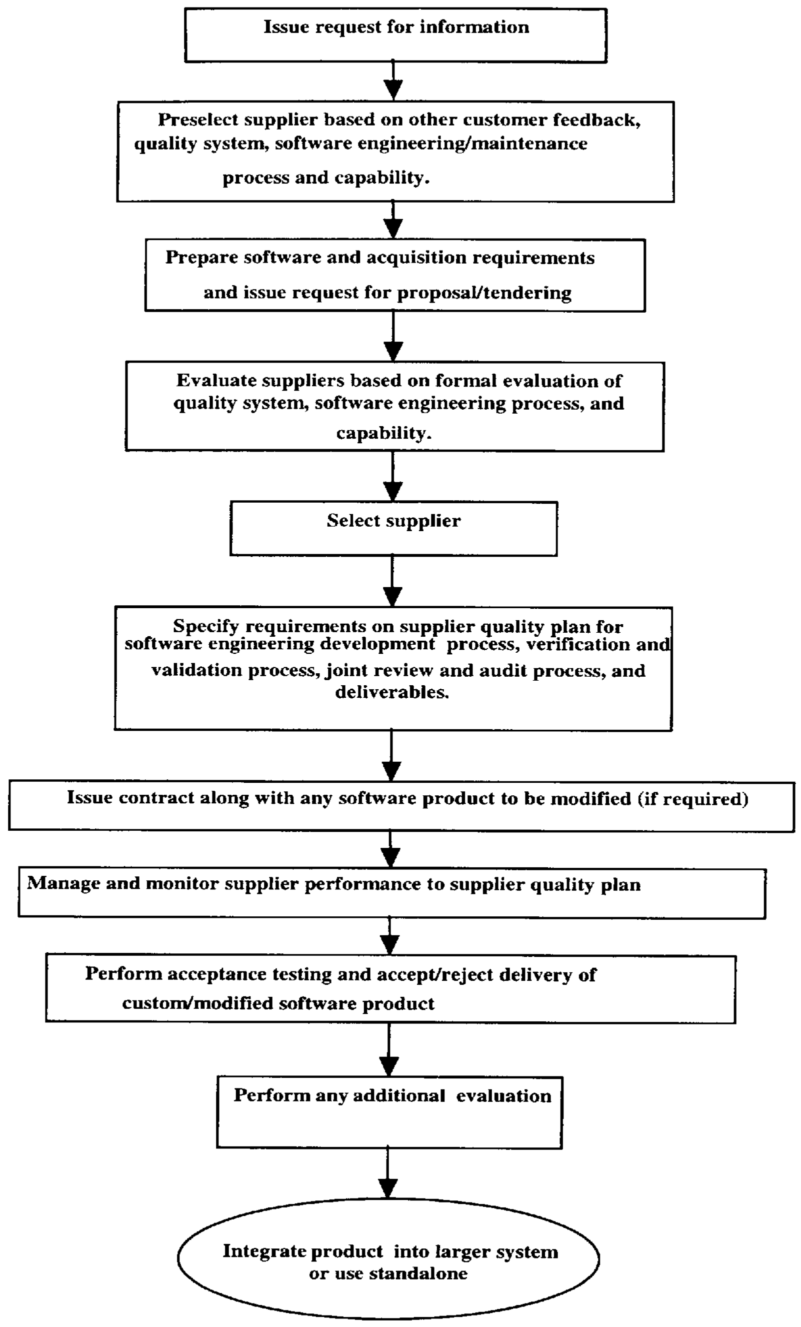
Issue request for information
Preselect supplier based on other customer feedback, quality system, software engineering/maintenance process and capability.
Prepare software and acquisition requirements and issue request for proposal/tendering
Evaluate suppliers based on formal evaluation of quality system, software engineering process, and capability.
Select supplier
Specify requirements on supplier quality plan for software engineering development process, verification and validation process, joint review and audit process, and deliverables.
Issue contract along with any software product to be modified (if required)
Manage and monitor supplier performance to supplier quality plan
Perform acceptance testing and accept/reject delivery of custom/modified software product
Perform any additional evaluation
Integrate product into larger system or use standalone

Figure C.2 — Example evaluation/acquisition process for custom software or modifications to existing software

Annex D
(informative)

Evaluation methods

D.1 Review of user and technical product documentation (including on-line documentation)

Product documentation may provide all the information necessary to make an evaluation of the functionality and usability requirements as well as other issues such as portability and maintainability. It may be possible to gain access to pertinent software product documentation without actually purchasing it, either by borrowing the documents or purchasing a documentation set. Although reviewing the software product documentation may not be as efficient as taking a course or training, it may prove to be the most economical, especially if the evaluator has the pertinent expertise.

D.2 Evaluation based on supplier courses and training

Product courses are offered for many software products, either through the supplier, or a third party. In the case of software products where no courses exist, it may be possible to arrange for special training from experienced users or developers of the software product. The advantage that product courses or training offer is allowing the evaluator to focus on specific areas and gain specific information on the functionality and usability of the software product in a short period of time. It may be possible to gain the same information through review of the software product documentation but that may prove more time consuming. The additional cost of a course or training needs to be weighed against efficiency in gaining information and the generality of the course material.

D.3 Assessment of software engineering process

The software engineering process assessment is a means of determining the quality of the software product by examining the interim products of the process; i.e. product quality plan, requirements specifications, architecture descriptions, detailed design descriptions, code listings, verification and validation records, code inspection and testing records, etc. To achieve this, it is necessary to define what constitutes an acceptable documentation baseline for software engineering processes which will provide adequate assurance in the quality of the resulting software product.

An acceptable baseline can be defined by tailoring ISO/IEC 12207 requirements to the target integrity level in order to specify the required development and related support activities. This consists of determining:

a) required processes;

b) required process output documentation (Note that ISO/IEC 14598-5 also lists in an informative annex a product document set that can be used for evaluating the software engineering process);

c) the requirements on process and process output documentation.

This assessment may be coupled with a supplier process capability level determination as defined in ISO/IEC 15504-8. For example, the software process documentation baseline defined in Table D.1 may be required for products of medium to high integrity requirements:

Table D.1 —Software Process Documentation Baseline

[illegible] 12207:1995 [illegible]	Output(s)	ISO/IEC 12207:1995 Requirements (subclause)
planning	software quality / development plan and supporting procedures	7.1.2
system requirements analysis	system requirements specification	5.3.2
system requirements verification	verification records	6.4.2.3
system architectural design	system design description	5.3.3
system design verification	verification records	6.4.2.4
software requirements analysis	software requirements specification	5.3.4
software requirements verification	verification records	6.4.2.3
software architectural design	software architecture description	5.3.5
software detailed design	software design description	5.3.6
software design verification	verification records	6.4.2.4
software coding	code	5.3.7
code verification	verification records	6.4.2.5
software unit testing	test records	5.3.7
software integration	test records	5.3.8, 6.4.2.6
software validation	test records	5.3.9, 6.5
system integration	test records	5.3.10
system validation	test records	5.3.11, 6.5
configuration management	plan, status reports, releases, change requests	6.2
training	training records	7.4

For systems with high integrity requirements additional process and product requirements may be required by sector standards, e.g., IEC 880 , Draft IEC 1508 , DOA-167A , MOD-55, etc.

Then the supplier quality/development plan and associated methodology procedures can be used to assess supplier compliance to this target baseline. The level of compliance may then be determined by identifying the major deficiencies and assessing their potential impact on the quality of the software product. Additional evaluation or workarounds can address the impact of the deficiencies.

It should be recognized that there are a variety of software engineering processes which are effective for particular organizations and different types of software products. The evaluation process should have the flexibility to accommodate a variety of reasonable software engineering processes and methods.

It is recommended that the software engineering review be done in a staged manner as shown in the example in Annex E. When it is deemed that the integrity level of software does not require a full evaluation of the software engineering process, the review may be halted after stage I or II.

D.4 Review of operating history with the supplier

A review of the operating history with the supplier can provide a very effective means of indicating the quality of the software product. This is achieved by reviewing the sales figures of the software product and details of the industries and the applications in which it is used. This review also addresses the history of revisions to the software, the way in which revisions are maintained, the way in which deficiency reports from customers are dealt with, and details of known deficiencies. The most convenient way to conduct the review is by interviewing the supplier's engineering, sales, and customer support staff and by examining any supporting records.

D.4.1 Operating history requirements

a) the sales figures should be at least six months old; i.e., the number of sales used in the evaluation will only include those sold over six months before the evaluation takes place. This criteria is based on the fact that it may take up to six months for the software product to be delivered, installed, commissioned, and placed in service;

b) the software product should have gone through at least one major revision and there should be viable operating history data available on that revision. This is based on the assumption that the quality of the software product will depend on the amount of refinement it has gone through;

c) there is a means for users of the software product to feed back deficiency reports to the supplier, and that there is evidence of this happening, and of resulting dispositions being implemented.

D.4.2 Operating history review

Review of product operating history should determine:

a) whether the software was produced by modifying another product and whether that product's operating history can be used. This would be contingent on the number of changes and the extent of the changes made to the software product;

b) the number of unit-years of operation for the software product. This is calculated by the following steps:

    1) calculate the Sales Year = (initial sales [total sales in first year] + cumulative final total up to six months from present [assuming it typically takes 6 months delay before a unit is operational]) * (number of years the software product is in the market) / 2.

    2) calculate Unit-Years of operation = (Sales Year * duty cycle [percentage of the time the software is operational] * percentage of units actually in operation [this is pertinent, for example, to firmware in which a number of units of the host hardware may be kept as spares]).;

c) if the operating experience provides evidence pertinent to the functionality required by the application. For example, is it used in similar applications by other customers?;

d) if the operating experience provides evidence that the breadth of the software product's functionality has been exercised. For example, has it been used in a wide variety of applications and industries?;

e) the number of unit-years of operation for each revision of the software and for each special option of the software product;

f) the differences between the revisions, the extent of the changes, and whether the changes are isolated;

g) whether documented evidence exists to support the operating history data;

h) how revisions of the software product and revisions of any related hardware are controlled and tracked;

i) whether it is possible to order a specific revision of the software product and the implications of ordering a revision which is not current;

j) the supplier process for accepting and disposing of deficiency reports received from customers;

k) whether customers are kept abreast of reported deficiencies;

l) any outstanding deficiencies in the software and their impact.

D.5 Review of operating history with customers

A review of the operating history with actual customers who use or have used the software product in one of their applications allows the evaluator to get relatively unbiased answers to specific questions based on realistic operating conditions. Depending on how similar the customer applications are to the proposed application, the assurance gained in overall quality or specific functionality may be as useful as that obtained from prototyping or even extensive trial usage. The most convenient way to conduct the review is by interviewing the customers at their site of operations and possibly even viewing demonstrations or supporting records.

The evaluator conducting an operating history review with customers should:

a) establish the degree of similarity of the application(s) to the proposed application;

b) attempt to view the software product in operation or obtain other supporting evidence;

c) question the customer(s) on the form and quality of support provided by the supplier;

d) determine the amount of error free operation.

D.6 Review of supplier capability, support, and quality system

Evaluation of the supplier's level of support capability and ability to maintain the software should address:

a) financial stability, experience, and capabilities;

b) product development environmental support, including assembly tools, operating system used, and maintenance and use of other component/library objects;

c) product interfaces to other products or product groupings, including interface standards;

d) contingencies for third party involvement;

e) sufficient resource commitment to product support;

f) sufficient customer base to justify continued support;

g) sufficient maintenance service for bug fixes, environmental support;

h) adequate references from installed base;

i) formalized and sufficient release and revision control procedures and evidence of practice;

j) formalized regression testing and formal evaluation of design changes;

k) documented and practiced problem reporting and resolution procedures;

l) quality system in place;

m) standards used for hardware and software;

n) plans for future development; i.e., strategy in place relative to current market position;

o) product warranty.

D.7 Prototyping and other evaluation methods

D.7.1 Prototyping

Prototyping is a method of evaluation that can be used to resolve or fine tune requirements, to determine the technical feasibility of using the software product, or to eliminate unknowns or technical risks associated with specific functionality or usability requirements and their implementation. Prototyping may or may not utilize the full suite of functionality of the software product or address the entire set of requirements of the application.

It should be noted that prototyping often requires the supplier to provide access to specialized equipment, personnel and documentation. Costs and schedule implications for these and other factors such as special environmental conditions or services should be considered in determining the feasibility of prototyping the software product.

In addition to the general requirements for an evaluation, prototyping should:

a) use examples which adequately address the requirements being assessed and provide realistic re-creation and simulation of key operating parameters;

b) be adequately documented so that it may be repeated by a third party;

c) should make use of historical data pertinent to the proposed application where available.

D.7.2 Other evaluation methods

ISO/IEC 14598-5 lists in an informative annex, a list of evaluation techniques attached to evaluation levels and software quality characteristics. The evaluation levels can correspond to integrity levels for the evaluation. Additional evaluation methods include:

a) analysis of software architecture design (maintainability);

b) fault tree analysis of software (safety, reliability);

c) statistical random usage based testing of software product (reliability);

d) dynamic analysis of code to check syntax and semantics for correctness (reliability);

e) hazards analysis of software design (safety, reliability);

f) review of software requirements specification (functionality);

g) code inspection (functionality);

h) black-box testing of software (functionality);

i) benchmark testing (efficiency);

j) analysis of requirements traceability (maintainability);

k) simulated faults at the interfaces between components (robustness).

For complex software of high integrity, fault tree analysis or hazards analysis of software design can be used to isolate "critical" software modules for evaluation. This may eliminate the need to perform rigorous evaluation on software that has no impact on the integrity of the application.

Annex E
(informative)

Example of staged evaluation process

An example of a staged evaluation showing the relative effort required for three target areas at each stage is outlined below.

E.1 Stage 1: Planning - Requirements Stage

E.1.1 Product documentation, courses and training

Some review in this area is almost always done initially to confirm or determine the basic functionality of the software product. This initial review may be done by others. Depending on the integrity level, consideration should be given to documenting this informal or preliminary evaluation.

E.1.2 Software engineering process

It is highly advantageous to begin the evaluation of the software engineering process for the software product at this point, usually by reviewing the software product quality/development plan. This would provide some indication of whether the software engineering process is likely to produce documentation and procedures that meet the required general characteristics of the documentation set for the relevant integrity level. Where appropriate, the review of other process evaluations may be considered to supplement this effort.

General characteristics and not specific requirements should be considered at this point. The following points should also be noted:

a) an exact match of the supplier documentation to the defined set is not likely or necessary;

b) in assessing software engineering processes, note the context of the process in the overall process, the importance of previous steps leading up to the process, and the quality of the inputs to the process.

E.1.3 Operating history

An initial review of the operating history with both the supplier and appropriate customers can provide an early indication of further evaluation work required. For example, a short operating history and/or few appropriate customers may lead to the need to consider prototyping, external evaluation review or an increase in the degree of rigor of the software engineering process evaluation.

E.2 Stage 2: Design - Acquisition Stage

E.2.1 Product documentation, courses and training

A more comprehensive review for the required functionality of the software product is needed in this phase. If product courses or documentation are not available to substantiate the required functionality, then a more rigorous review of the operating history with appropriate customers should be undertaken.

E.2.2 Software engineering process

At this stage the actual implementation of the quality plan is assessed. This would involve a review of the actual intermediate products produced to ensure their conformance to the general characteristics of those identified. If this stage coincides with the tendering process, the time element may restrict the review to a check of key documents only. Tenders should always contain a phrase that allows for examination of any documentation referenced in the software product quality plan before the contract award.

Because of its relatively higher cost, a software engineering process review would not normally be considered for software products of lower integrity levels unless it was felt that the operating history was not extensive enough to provide assurance of the product's reliability.

E.2.3 Operating history

A more extensive operating history review that concentrates on the particular version and required functions of the existing software product should be conducted in this stage. If the review of product courses, documentation, and the operating history evaluation with appropriate customers does not provide adequate confidence of the product's reliability and functionality, prototyping should be considered.

E.3 Stage 3: Full Evaluation Stage

E.3.1 Product documentation, courses and training

The evaluation for product functionality by a review of the product documentation or attendance at courses should be completed if not done in previous stages.

E.3.2 Software engineering process

This phase would entail the full evaluation of the software engineering process where the specific requirements of each process/product are compared to those of the supplier. Additional points to consider include:

a) each requirement should be addressed to determine if the supplier's software engineering process does or does not comply with the intention of the requirement. This can often be most readily achieved by interviewing the supplier's software engineering staff; therefore it is not feasible or effective to determine it during the tendering process;

b) supplier personnel may also be asked to show evidence that their product is of adequate quality. This may provide a perspective which highlights areas not specifically covered by the baseline software process. It may also help to make the connection between specific areas in the baseline software process and parts of their process which are not immediately evident to the evaluator;

c) some requirements have an element of subjectivity and will require interpretation by the evaluator. There may be a wide variety of methods which meet the intent of the requirements;

d) the review should establish the revision history of the software product and should also determine the extent of change between revisions. This could include examination of reported deficiencies. Consideration should be given to:

   1) the status of all known deficiencies;

   2) the degree of review, evaluation, and testing carried out for all changes to fix deficiencies;

   3) the stability of the current design revision;

   4) time elapsed since current release date;

   5) number of changes and size or magnitude of those changes.

Where deficiencies exist, or detailed review is not feasible, consideration should be given to additional evaluation methods like prototyping or black box testing.

E.3.3 Operating history

The evaluation of the operating history should be completed if not done in previous phases.

Bibliography

Standards

[1] ISO/IEC Guide 2:1991, *General terms and their definitions concerning standardization and related activities.*

[2] ISO/IEC 2382-1:1993, *Information technology - Vocabulary - Part 1: Fundamental terms.*

[3] ISO/IEC 2382-20:1990, *Information technology - Vocabulary - Part 20: System development.*

[4] ISO 8402:1994, *Quality management and quality assurance - Vocabulary.*

[5] ISO 9001:1994, *Quality systems - Models for quality assurance in design, development, production, installation and servicing.*

[6] ISO/IEC 12119:1994, *Information technology - Software packages - Quality requirements and testing.*

[7] ISO/IEC 14102:1995, *Information technology - Guideline for the evaluation and selection of CASE tools.*

[8] Radio Technical Commission for Aeronautics (RTCA) DO-178B/ED-12B, *Software Considerations in Airborne Systems and Equipment Certification.*

[9] Ministry of Defence (UK) (MOD), Interim Defence Standard, 00-55 (Parts 1,2)/Issue 1, *The Procurement of Safety Critical Software in Defence Equipment.*

[10] IEC 880 -1986, *Software for computers in the safety systems of nuclear power stations*

[11] IEEE Std 1062-1993, *Recommended Practice for Software Acquisition.*

Other References

[12] Tremaine, DR. and de Grosbois, JFP. Guideline for the Qualification of Predeveloped Software. 907-C-H-69002-0201, Ontario Hydro/AECL, April 22, 1993.

[13] Ferguson, JR. and DeRiso, ME. Software Acquisition: A Comparison of DoD and Commercial Practices. CSU/SEI-94-SR-9, Software Engineering Institute, October, 1994.

[14] Baker, ER., Cooper, L., Corson, BA. and Stevens, AE. Software Acquisition Management Maturity Model (SAM^3). *Program Manager.* July-August, 1994, pp. 43-49.

[15] Scott, JA., Preckshot, GG. and Gallagher, JM. Using Commercial-Off-the-Shelf (COTS) Software in High Consequence Safety Systems (Draft Preprint). Lawrence Livermore National Laboratory, November, 1995.

[16] Cochrane, Gail. Use of COTS/NDI In Safety-Critical Systems. (based on original Report prepared for the Federal Aviation Authority), TRW, 1996.

[17] Brown, AW. and Wallnau, KC. Engineering of Component-Based Systems. *Proceedings Second IEEE International Conference on Engineering of Complex Computer Systems.* October, 1996.

[18] Bevan, N. and Azuma, M. Quality in Use: Incorporating human factors into the software engineering lifecycle. *Proceedings of the Third International Symposium on Software Engineering Standards.* May, 1997.

[19] Voas, J. and Miller, K. Interface Robustness for COTS-Based Systems. *IEE Symposium on COTS and Safety-Critical Systems.* Savoy Place, London, January, 1997.

ICS 35.080
Price based on 34 pages