INTERNATIONAL STANDARD

ISO/IEC/IEEE 14764

Third edition
2022-01

Software engineering — Software life cycle processes — Maintenance

Ingénierie du logiciel — Processus du cycle de vie du logiciel — Maintenance

Reference number
ISO/IEC/IEEE 14764:2022(E)

Contents

Page

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO/IEC documents should be noted. This document was drafted in accordance with the rules given in the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document are in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

ISO/IEC/IEEE 14764 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Systems and software engineering*, in cooperation with the Systems and Software Engineering Standards Committee of the IEEE Computer Society, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

This third edition cancels and replaces the second edition (ISO/IEC 14764:2006), which has been technically revised.

The main changes compared to the previous edition are as follows:

— alignment of the standard with ISO/IEC/IEEE 12207:2017 and updates to other ISO/IEC JTC1/SC7 standards;

— introduction of modern approaches to “maintenance”.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document provides guidance on the software maintenance process. Maintenance is a technical process in the life cycle of a software product, as described in ISO/IEC/IEEE 12207. The maintenance process contains the activities and tasks of the maintenance organization. This document is the result of the harmonization of ISO/IEC 14764 and IEEE Std 1219, and the update for ISO/IEC/IEEE 12207:2017.

Because maintenance consumes a major share of a software life cycle financial resources, it should be an important project consideration.

During operation of the software, problems may be detected that were not detected during verification, validation and acceptance. Therefore, a maintenance effort is needed to cope with these problems. This maintenance effort also covers software improvements needed to meet new or modified user requirements. Software maintenance is commonly needed when upgrading system components, such as operating systems and databases, as well as when changes are made to external software and systems' interfaces. Software maintenance is typically a significant portion of life cycle costs, even when a part of the system under maintenance includes COTS software.

Software maintenance organizations uses a number of specific tools, methods, and techniques. This document does not specify how to implement or perform the activities and tasks in the software maintenance process since these are dependent upon the formal agreement and organizational requirements. Maintenance is required on all types of software, whatever the technology, technique, or tool used to create it.

Software engineering — Software life cycle processes — Maintenance

1 Scope

1.1 Overview

This document provides guidance for the maintenance of software, based on the maintenance process and its activities and tasks defined in ISO/IEC/IEEE 12207:2017, 6.4.13. Moreover, this document describes the maintenance process in greater detail and establishes definitions for the various types of maintenance. This includes maintenance for multiple software products with the same maintenance resources. “Maintenance” in this document means software maintenance unless otherwise stated.

The document does not address the operation of software and the operational functions, e.g. backup, recovery, system administration, which are normally performed by those who operate the software. However, it does include the related disposal process defined in ISO/IEC/IEEE 12207:2017, 6.4.14.

This document is written primarily for managers, maintenance organizations, quality managers, users and acquirers of systems containing software.

Many of the activities and tasks discussed in this document apply equally to maintenance services, as well as to maintained software products. For example, in a COTS intensive system, maintenance services are performed to sustain the product in operations.

While the scope of this document is software maintenance, hardware and hardware costs are important considerations for maintenance.

1.2 Purpose

This document provides guidance on the maintenance process. It identifies how the maintenance process can be invoked during acquisition and operation. This document also emphasizes the following in the maintenance process: the maintainability of software products; the need for maintenance service models; and the need for a maintenance strategy.

1.3 Field of application

This document is intended to provide guidance for the planning for and maintenance of software products or services, whether performed internally or externally to an organization. It is not intended to apply to the operation of the software.

This document is intended to provide guidance for two-party situations and can be equally applied where the two parties are from the same organization. This document is intended to also be used by a single party as self-imposed tasks (ISO/IEC/IEEE 12207).

This document is not intended for software products that are “throw-away” or a “short-term” solution.

This document is intended for self-imposition by organizations that develop off-the-shelf software products to maintain such products. Maintenance is applied to computer programs, code, data, documents, and records. It is intended to apply to software products created during the development of the software product. This can include, for example, the test software, test databases, the software test environment (STE), or the software engineering environment (SEE).

This document is intended for use in all maintenance efforts, regardless of the life cycle model (e.g. incremental, waterfall, evolutionary, spiral, agile, continuous iterative development). This document is not restricted by size, complexity, criticality, reliability, or application of the software product.

1.4 Limitations

This document describes the framework of the maintenance process but does not specify the details of how to implement or perform the activities and tasks included in the process.

In this document, there are a number of lists. None of these is presumed to be exhaustive. They are intended as examples.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC/IEEE 12207, *Systems and Software engineering — Software life cycle processes*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions in ISO/IEC/IEEE 12207 and the following apply.

ISO, IEC and IEEE maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

— IEEE Standards Dictionary Online: available at https://dictionary.ieee.org

3.1.1
adaptive maintenance
modification of a software product, performed after delivery, to keep a software product usable in a changed or changing environment

Note 1 to entry: Adaptive maintenance provides *enhancements* (3.1.7) necessary to accommodate changes in the environment in which a software product operates. These changes help keep pace with the changing environment, e.g. an upgrade to the operating system results in changes in the applications.

3.1.2
additive maintenance
modification of a software product performed after delivery to add functionality or features to enhance the usage of the product

Note 1 to entry: Additive maintenance may be excluded from the definition of maintenance in the context of dependability that addresses recovery of a system to previous operational, functional and performance level, e.g. definition, monitor or measurement of availability, recoverability, or MTBF (mean time between failure).

Note 2 to entry: “Additive maintenance” type is distinguished from *“perfective maintenance”* (3.1.9) type and recognized as a different maintenance type to be able to:

— provide additional new functions or features to improve software usability, performance, *maintainability* (3.1.6), or other software attributes for the future;

— add functionality or features with relatively large additions or changes on software for improving software attributes after delivery with identified opportunities to negotiate any of additions or changes on maintenance strategy, methods, resources, agreements, or service levels between suppliers and acquirers.

Note 3 to entry: Additions or *enhancements* (3.1.7) can be handled through the maintenance process, while larger changes can involve a new development effort.

3.1.3
correction
<software> change that addresses and implements problem resolutions to recover gaps and to make software operational enough to meet defined operational requirements

Note 1 to entry: In this document, the term “correction” is mainly used as a maintenance type and to classify *modification requests (MR)* (3.1.8).

3.1.4
corrective maintenance
modification of a software product performed after delivery to correct discovered problems

Note 1 to entry: The modification repairs the software product to satisfy defined system requirements.

3.1.5
emergency maintenance
unscheduled modification performed to temporarily keep a system operational, pending *corrective maintenance* (3.1.4)

Note 1 to entry: Emergency maintenance may be performed to make a software system partially operational. It may include various modifications to the software or its parameters in order to limit operations, functionalities, inputs, outputs, interfaces, usability, etc.

Note 2 to entry: Emergency maintenance can be regarded as a corrective maintenance type.

3.1.6
maintainability
degree of effectiveness and efficiency with which a product or system can be modified

Note 1 to entry: Modifications can include *corrections* (3.1.3), improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.

Note 2 to entry: Maintainability includes installation of updates and upgrades.

Note 3 to entry: Maintainability can be interpreted as either an inherent capability of the product or system to facilitate maintenance activities, or the quality in use experienced by the maintainers for the goal of maintaining the product or system.

[SOURCE: ISO/IEC 25010:2011, 4.2.7, modified —"by the intended maintainers" at the end of the definition has been removed.]

3.1.7
enhancement
software change that addresses and implements a new requirement

Note 1 to entry: There are three types of software enhancements: adaptive, perfective and additive. An enhancement is not a software *correction* (3.1.3).

Note 2 to entry: In this document, the term “enhancements” is mainly used as a maintenance type and to classify *modification requests (MR)* (3.1.8).

3.1.8
modification request
MR
information item that identifies and describes proposed changes(s) to a product or service

Note 1 to entry: The MR may later be classified as a *correction* (3.1.3) or *enhancement* (3.1.7) and identified as *corrective maintenance* (3.1.4), *preventive maintenance* (3.1.10), *adaptive maintenance* (3.1.1), *additive maintenance* (3.1.2) or *perfective maintenance* (3.1.9). MRs are also referred to as change requests. See Figure 1.

Note 2 to entry: When formulating an MR, incidents, events and complaints should be reviewed as well as failure occurrence *problem reports* (3.1.11) and stakeholder’s maintenance requests.

Note 3 to entry: When classifying MRs, methods for prioritizing problems and root case analyses can be applied; and various approaches for preventing failure reoccurrences can be considered. For example, reoccurrence of similar failures can be prevented by preventive maintenance requests to fix potential latent faults, while similar failures in future can be avoided by perfective maintenance to enhance user interface preventing human error.

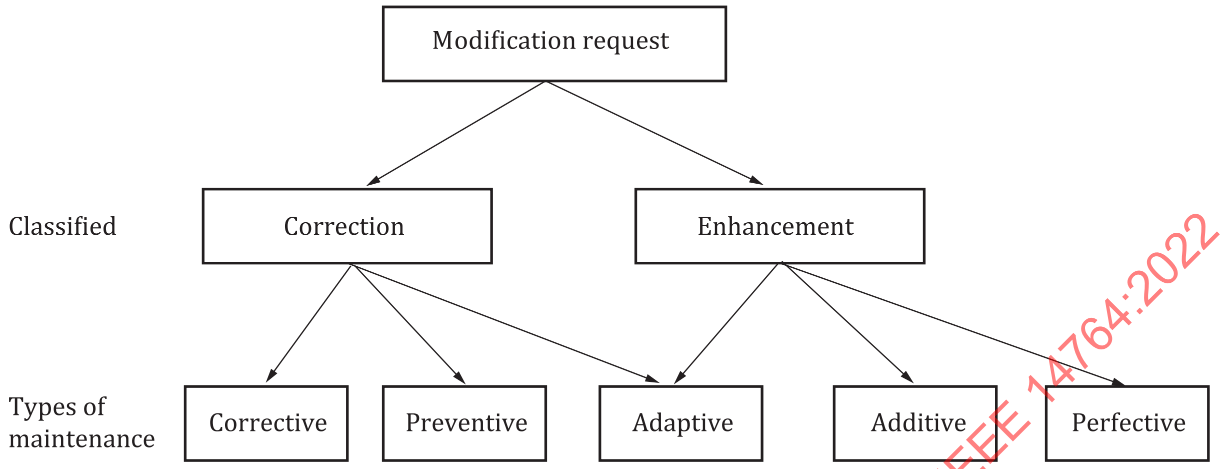

Figure 1 — Modification request

Note 4 to entry: In some organizations, adaptive maintenance is not considered to be an enhancement.

Note 5 to entry: Adaptive maintenance can be requested from MRs classified as enhancement, when the software is to be adapted to its evolving and changing environment or different environments on which the software is not initially intended to operate. Additionally, adaptive maintenance can be requested from MRs classified as correction, when the software is to be adapted to its environment that has already changed.

Note 6 to entry: Additive is an enhancement type that some organizations use and different from perfective in that some changes to existing software or systems is made.

Note 7 to entry: Some organizations sub-divide each type into “scheduled”, “unscheduled” and “emergency” types.

Note 8 to entry: Another way to classify MR’s is by “reactive” or “proactive” maintenance as used in SWEBOK. Reactive includes corrective and adaptive types; and proactive includes preventive, perfective and additive types.

3.1.9
perfective maintenance
modification of a software product to provide *enhancements* (3.1.7) for users, improvements of information for users, and recording to improve software performance, *maintainability* (3.1.6) or other software attributes

3.1.10
preventive maintenance
modification of a software product after delivery to correct latent faults in the software product before they occur in the live system

3.1.11
problem report
PR
document used to identify and describe problems detected in a software product

Note 1 to entry: PRs are either submitted directly to denote faults or established after impact analysis is performed on *modification requests* (3.1.8) and faults are found.

3.1.12
software maintenance
totality of activities required to provide support to a software system

Note 1 to entry: Activities are performed during the pre-delivery stage as well as the post-delivery stage.

Note 2 to entry: Pre-delivery activities include, for example, planning for post-delivery operations, supportability, and logistics determination. Post-delivery activities include, for example, software modification, training, and operating a help desk.

Note 3 to entry: In the context of dependability, maintenance is a combination of all technical and management actions intended to retain an item in, or restore it to, a state in which it can perform as required.

3.1.13
software sustainment
activities to support, maintain and operate a software system to ensure that the software system or service remains operational

Note 1 to entry: Software sustainment includes processes, procedures, people, material and information required to support, maintain and operate the software aspects of a system.

Note 2 to entry: Sustainment is defined in ISO/IEC/IEEE 24765 as activities performed to ensure that a product or service remains operational. In this document, software sustainment is defined to emphasize that activities for supporting, maintaining and operating are performed more concurrently and iteratively to sustain operational software systems.

3.1.14
transition
activities involved in moving a new or changed service, system, or component to or from an environment

3.2 Abbreviated terms

CM	configuration management
COTS	commercial-off-the-shelf software
MBSE	model-based systems engineering
SEE	software engineering environment
STE	software test environment

4 Conformance

This document provides guidance for the implementation of the maintenance process of ISO/IEC/IEEE 12207. The guidance in this document is completely consistent with ISO/IEC/IEEE 12207. Conformance cannot be claimed to this document but can be claimed to the ISO/IEC/IEEE 12207 maintenance process and related tailoring. The only mandatory clauses in this document come from ISO/IEC/IEEE 12207. The mandatory clauses contain requirements; and each requirement from ISO/IEC/IEEE 12207 that is duplicated in this document is boxed. The related ISO/IEC/IEEE 12207 clause number is listed after the boxed ISO/IEC/IEEE 12207 requirements.

5 Application of this document

5.1 General

This clause presents the maintenance process that is required to maintain software products.

5.2 Maintenance process

Maintenance is one of the technical life cycle processes that may be performed during the life cycle of software (ISO/IEC/IEEE 12207). The acquisition and supply agreement life cycle processes of ISO/IEC/IEEE 12207 may initiate the process implementation activity of the maintenance (life cycle) process through an agreement or contract. The operation process may call for maintenance through a modification request or problem report. The maintenance (life cycle) process invokes the technical life cycle processes of ISO/IEC/IEEE 12207 to develop any required enhancement, as a part of the maintenance strategy. The technical management processes, such as project planning, configuration management and quality assurance of ISO/IEC/IEEE 12207 are used by the maintenance (life cycle) process.

NOTE Significant maintenance changes can be treated as a software development project using the design definition, implementation, integration, and transition processes, among others.

Figure 2 shows the connection of the maintenance process in ISO/IEC/IEEE 12207 to other maintenance-related processes and the elaboration of the maintenance process into its defined activities. These maintenance activities are discussed in detail in Clause 6.

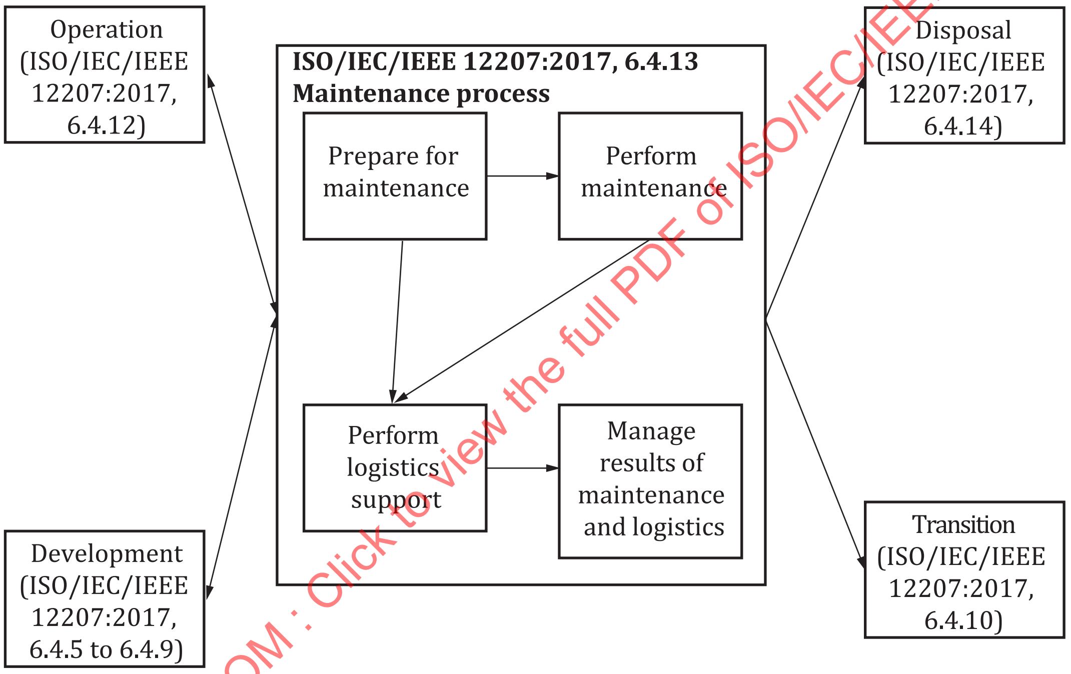
Operation (ISO/IEC/IEEE 12207:2017, 6.4.12)
ISO/IEC/IEEE 12207:2017, 6.4.13
Maintenance process
Prepare for maintenance
Perform maintenance
Perform logistics support
Manage results of maintenance and logistics
Disposal (ISO/IEC/IEEE 12207:2017, 6.4.14)
Development (ISO/IEC/IEEE 12207:2017, 6.4.5 to 6.4.9)
Transition (ISO/IEC/IEEE 12207:2017, 6.4.10)

Figure 2 — Relationship between maintenance process and other processes of ISO/IEC/IEEE 12207 by clause number

5.3 Organization of this document

Clauses 6 to 9 provide an elaboration of the maintenance process from ISO/IEC/IEEE 12207. The boxed text preceding elaboration text contains the numbered clauses from ISO/IEC/IEEE 12207.

Clause 6 provides the details of the maintenance process including activities and tasks needed to implement the maintenance process. Clause 7 includes considerations, and issues encountered when planning maintenance, Clause 8 describes implementation issues. Clause 9 provides details for a maintenance plan.

6 Maintenance process

6.1 Maintenance activities and tasks

6.1.1 General

6.4.13 Maintenance process

6.4.13.1 Purpose

The purpose of the Maintenance process is to sustain the capability of the system to provide a service.

This process monitors the system's capability to deliver services, records incidents for analysis, takes corrective, adaptive, perfective and preventive actions and confirms restored capability.

[ISO/IEC/IEEE 12207:2017]

Software maintenance may be considered part of software sustainment, where support, maintenance and operation processes are managed and performed together.

6.4.13.1 Purpose (cont'd)

For software systems, the Maintenance process makes corrections, changes, and improvements to deployed software systems and elements. The software systems maintenance approach differs for systems that are freely available, in wide commercial distribution, or operating in a small number of controlled environments.

The need for software system maintenance can arise from multiple causes other than latent system defects, such as changes to interfaced systems or infrastructure, evolving security threats, and technical obsolescence of system elements and enabling systems over the system life cycle. Often the extension of capability, mid-life upgrade, or evolution of legacy systems becomes a new software system development project that will apply the set of processes within an appropriate life cycle. If so, the Portfolio Management process is the starting point to initiate the work. In other cases, software system maintenance is performed as a continuing series of prioritized work items, possibly on a level of effort basis. Maintenance of software system elements can include hardware, software, and services, such as communication or web services. Maintenance is closely connected with the Configuration Management process and software asset management and is performed concurrently with the other Technical processes.

NOTE ISO/IEC/IEEE 14764:2006 *Software Engineering — Software Life Cycle Processes — Maintenance* and ISO/IEC 16350, *Information technology — Systems and software engineering — Application management*, provide additional detail. The SWEBOK, Guide to the Software Engineering Body of Knowledge, Software Maintenance knowledge area discusses software maintenance fundamentals, key issues, measurement, techniques, maintenance process and support activities, and tools. The guide also discusses models, techniques and measures that support software reliability.

[ISO/IEC/IEEE 12207:2017]

The boxed text and notes from ISO/IEC IEEE 12207 are considered sufficient guidance for this subclause.

6.4.13.2 Outcomes

As a result of the successful implementation of the Maintenance process:

a) Maintenance constraints that influence system requirements, architecture, or design are identified.

b) Any enabling systems or services needed for maintenance are available.

c) Replacement, repaired, or revised system elements are made available.

d) The need for changes to address corrective, perfective, or adaptive maintenance is reported.

e) Failure and lifetime data, including associated costs, is determined.

[ISO/IEC/IEEE 12207:2017]

No specific guidance for this subclause.

6.4.13.3 Activities and tasks

The project shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the Maintenance process.

a) **Prepare for maintenance**. This activity consists of the following tasks:

1) Define a maintenance strategy, including consideration of the following:

i) Establishing priorities, typical schedules, and procedures for performing, verifying, distributing, and installing software maintenance changes in conformance with operational availability requirements;

ii) Establishing techniques and methods for becoming aware of the need for corrective, adaptive, and perfective maintenance;

iii) Periodic assessment of the design characteristics in case of evolution of the software system and of its architecture;

iv) Forecasting potential obsolescence of components and technologies using information on technical changes in related systems;

v) Establishing priorities and resources to obtain access to the correct versions of the product and product information needed for performing maintenance (e.g., scheduled or phased installation, maintenance patches or software upgrades);

vi) Measures for maintenance that will provide insight into performance levels, effectiveness, and efficiency, including access to historical fault and failure;

vii) Agreed rights to data and the impact on data in the system during problem resolution and maintenance activity;

viii) Approach to assure that counterfeit or unauthorized system elements are not introduced into the system;

ix) Impact of the maintenance change on other software systems elements versus the risk of leaving a reported software anomaly in place;

x) The skill and personnel levels required to effect system or software repairs or replacements, fixes, patches, updates, and upgrades, considering legal and regulatory requirements regarding health and safety, security, and the environment;

2) For non-software elements, define a logistics strategy throughout the life cycle, including acquisition and operational considerations: the number and type of replacement elements to be stored, their storage locations and conditions, their anticipated replacement rate, and their storage life and renewal frequency.

NOTE Supportability implications are considered early during concept exploration or development stages. Logistics helps to ensure that the necessary material and resources, in the right quantity and quality, are available at the right place and time throughout deployment and sustainment stages.

3) Identify constraints from maintenance to be incorporated in the system/software requirements, architecture, or design.

NOTE These often result from the need to 1) re-use existing maintenance and verification enabling systems; 2) re-use existing holdings of replaceable system element and accommodate re-supply limitations; 3) conduct maintenance in specific locations or environments. For example, software architectures and designs that emphasize encapsulation, modularity, and scalability can be simpler to maintain. Requirements to document the system design and construction can reduce the effort needed to reverse engineer systems and elements when maintenance is needed. The system architecture and design reflect the need to roll back, back up, and recover data during problem resolution. Functions to make the system available for remote diagnostics and maintenance can be incorporated in the architecture and design.

4) Identify trades such that the system and associated maintenance and logistics actions result in a solution that is affordable, operable, supportable, and sustainable.

NOTE The System Analysis and Decision Management processes are used to perform the assessments and trade decisions.

5) Identify and plan for the necessary enabling systems or services needed to support maintenance.

NOTE This includes identification of requirements and interfaces for the enabling systems. The selection of enabling systems for maintenance often reflects the need to re-use existing or equivalent design, development, and configuration management infrastructure as during the initial system implementation.

6) Obtain or acquire access to the enabling systems or services to be used.

NOTE The Validation process is used to objectively confirm that the maintenance enabling system achieves its intended use for its enabling functions.

[ISO/IEC/IEEE 12207:2017]

6.1.2 Maintenance strategy

To effectively implement the maintenance process, a strategy should be developed for performing the maintenance. In the maintenance strategy, various maintenance methods or techniques can be designated to perform the software maintenance process, consistent with the software system development and operation processes, such as performing development/maintenance concurrently with operational service provision, e.g. DevOps, or performing iterative development/maintenance with Agile practices. There may be multiple organizations providing maintenance, such as a distributed control system supplier, operating system supplier and application software suppliers.

The maintenance strategy should cover:

a) why maintenance is needed;

b) who can do what work;

c) what the roles and responsibilities of everyone involved are;

d) how the work is to be performed;

e) what resources are available for maintenance;

f) where maintenance is to be performed;

g) when maintenance can commence;

h) system description;

i) agreement protocols;

j) training;

k) controls;

l) an estimate of maintenance costs;

m) records and reports.

6.1.3 Maintenance planning

To accomplish maintenance planning the following tasks should be performed, although not all tasks may be necessary for all maintenance activities, especially when activities are focused on MRs or PRs:

a) develop maintenance plans and procedures;

b) establish MR/PR procedures;

c) implement configuration management.

The maintenance strategy includes the strategy to be used to maintain the system, while the maintenance plans and procedures should provide a more detailed approach on how to accomplish the maintenance.

Note The contents of a maintenance plan are detailed in Clause 9.

6.1.4 Maintenance plans and procedures

The following tasks support definition of maintenance plans and procedures:

a) determine the scope of maintenance;

b) analyse maintenance organization alternatives;

c) conduct resource analyses;

d) estimate maintenance costs;

e) perform a maintainability assessment of the system;

f) determine transition requirements;

g) determine transition milestones;

h) identify the maintenance process to be used;

i) develop maintenance procedures for selected maintenance tasks.

NOTE 1 Transition to a new system is often performed concurrently with disposal of an existing system, entailing data migration from the old system to its replacement (from ISO/IEC/IEEE 12207:2017, 6.4.10.1). Migration of software and data are often performed as a part of maintenance. This is the major reason why g) and h) are included in the maintenance plans and procedures.

NOTE 2 Consistent with ISO/IEC/IEEE 12207, ISO/IEC/IEEE 15289 specifies content for maintenance plans (ISO/IEC/IEEE 15289:2019, 10.31), maintenance procedures (ISO/IEC/IEEE 15289:2019, 10.32), problem management procedure (ISO/IEC/IEEE 15289:2019, 10.36), and problem reports (ISO/IEC/IEEE 15289:2019, 10.37).

6.4.13.3 Activities and tasks (cont'd)

b) **Perform maintenance**. This activity consists of the following tasks:

1) Review stakeholder requirements, complaints, events, incident and problem reports to identify corrective, adaptive, perfective and preventive maintenance needs.

NOTE For software systems with iterative life cycles, changing requirements can be considered as the source for adaptive and perfective maintenance activities. For software maintenance, this process makes corrections, changes, and improvements to deployed software, as well as patching and updates to maintain system security.

2) Analyse the impact of maintenance changes on data structures, data, and related software functions, user documentation, and interfaces.

NOTE Reviews and analyses often include factors such as the category of maintenance action; size of modification; cost involved; time to modify; and impacts on performance, safety or security.

3) Upon encountering unexpected faults that cause a software system failure, restore the system to operational status.

NOTE Restoration to full or degraded operational status can often be accomplished through a rollback, workaround or by identifying and correcting the cause of the fault. If full restoration is delayed or not possible, the system is restored to a degraded mode, consistent with the contingency planning. If possible, the fault is replicated using a distinct environment similar to the operational environment and the root causes of the fault are identified. The Configuration Management process, especially release management activities, is invoked to control scheduled and emergency changes to the system.

4) Implement the procedures for correction of flaws (defects) and errors, or for replacement or upgrade of system elements.

NOTE 1 Correction of flaws and errors uses problem resolution and can be handled through the Quality Assurance and Project Assessment and Control processes.

NOTE 2 Typically, regression testing is performed to verify that the maintenance change has not introduced other issues, i.e., complete and correct implementation of the new and modified requirements without effect on the performance of the original, unmodified requirements. The Transition process can be applied for deployment of major maintenance changes; minor fixes are typically handled as part of the Maintenance process. Actions are recorded in order to facilitate future maintenance and problem resolution, and for logistics analyses of degradable system elements.

NOTE 3 System and data recovery procedures and maintenance information are often made available on media that is usable at the point of performing maintenance.

5) Perform preventive maintenance by replacing, patching, augmenting, or upgrading software system elements, to improve the performance of a software system that is projected to reach unacceptable service levels, e.g., lack of capacity due to increases in demand or stored data, or to avoid unacceptable operating conditions, e.g., running with outdated security software.

6) Identify when adaptive or perfective maintenance is required.

NOTE Adaptive and perfective maintenance actions usually involve change to the system/software requirements, architecture and design. A new project can be started to modify the existing software system.

[ISO/IEC/IEEE 12207:2017]

6.1.5 Maintenance requirements review

In ISO/IEC/IEEE 12207:2017, 6.4.13.3 b) 1) for connected systems, log files can be used to track operational system performance and unreported problems/defects.

When formulating MRs, incidents, events and complaints should be reviewed in conjunction with failure occurrence problems and stakeholder's maintenance requests.

When incidents and failure occurrence problems are reviewed, potential risks can be identified and possibly mitigated by maintenance.

6.1.6 Maintenance changes impact analysis

ISO/IEC/IEEE 12207:2017, 6.4.13.3 b) 2) includes “quality”.

The following tasks should be performed:

a) develop an identification scheme for MRs/PRs;

b) develop a scheme for categorizing and prioritizing MRs/PRs;

c) determine the procedures for an operator to submit a MR/PR;

d) identify the information needs and issues that need to be tracked and reported to the users, and identify measures that provide feedback on those information needs and issues;

e) determine how temporary workarounds will be provided to the operators;

f) track the work-around(s) through to removal;

g) determine what follow-up feedback will be provided to the users.

Additionally, the following may be performed in mature organizations:

h) set targets for MR/PR closure rates;

i) determine the type of data that will need to be tracked over time, e.g. MR/PR aging data;

j) determine how data are entered into the status accounting database if used.

NOTE For analysis of the impact of maintenance changes, both the existing software system and the one under incremental maintenance/migration can be operated in parallel and monitored for quality evaluation, verification or validation with the comparisons between their outputs, results or performance.

6.4.13.3 Activities and tasks (Cont'd)

c) **Perform logistics support.** This activity consists of the following tasks:

NOTE The logistics actions enable the software system to sustain operational readiness. The actions include provisions for staffing, supply support, support equipment, technical data needs (user documentation) and agreed data rights, training support, communications, equipment/computing resource support, and facilities.

1) Obtain resources to support the software system through its life cycle or the project's life (acquisition logistics).

NOTE Acquisition Logistics considerations are included in the agreement resulting from the Agreement processes. This includes performing analysis to identify cost-effective changes to the initial design of the system for supportability and ease of maintenance, as well as arrangements for distributing software fixes and upgrades during utilization/deployment. These decisions are often constrained by availability requirements and impact the supply chain management.

2) Monitor the quality and availability of replacement elements and enabling systems, their delivery mechanisms and their continued integrity during storage.

NOTE Operational logistics involves the concurrent adjustment of both the system-of-interest and enabling systems throughout the operational life to help ensure effective and efficient delivery of software functions. It also includes availability of skilled resources. For example, reliable enabling systems are available with the capacity to read software stored on previous media formats, or to migrate backup files to current media and currently maintained enabling systems.

3) Implement mechanisms for software system or element distribution, including packaging, handling, storage and communications or transportation needed for items during the life cycle.

NOTE 1 Software distribution and installation is typically automated. Software packages commonly include software license terms, including data rights, and elements for software asset management. Logistics planning for other systems elements is often required to support the objectives of the Integration and Transition processes.

NOTE 2 Consider the need to store spare elements or backup copies of software onsite or in additional locations, to maintain software system capabilities, as required (perhaps at a reduced level for contingency operations).

4) Confirm that logistics actions to fulfil software system or element supportability requirements or achieve operational readiness are planned and implemented.

NOTE These logistic actions can include staffing, supply support, support equipment, technical data needs (user documentation, instructions, lists), training support, communications, equipment/computing resource support, and facilities.

[ISO/IEC/IEEE 12207:2017]

6.1.7 Maintenance measurements

ISO/IEC/IEEE 12207:2017, 6.4.13.3 a) 1) vi) includes consideration of system and software product quality including reliability, performance, efficiency, maintainability and so on (see also ISO/IEC 25010).

It is helpful to periodically apply measures for maintainability during maintenance such as measures of modularity quantifying complexity of structures of system and software, modifiability describing frequency of modifications done correctly without bringing defects or degradation into system or software.

6.1.8 Replacement elements monitoring of quality and availability

If available, a maintenance enabling system can be set up to automatically gather and monitor the types of problems that have occurred during maintained, migrated or replaced software. Software functions are used or operated by an individual user or operator of the software system, for analysing problems and for evaluating the quality of the updated software.

6.4.13.3 Activities and tasks (Cont'd)

d) **Manage results of maintenance and logistics.** This activity consists of the following tasks:

1) Record incidents and problems, including their resolutions, and significant maintenance and logistics results.

NOTE This includes anomalies due to the maintenance strategy, the maintenance enabling systems, execution of the maintenance and logistics, or incorrect system definition. The Project Assessment and Control and Quality Assurance processes are used to perform maintenance problem identification and resolution, e.g., analyze the data to identify the root cause, enable corrective or improvement actions, and record lessons learned. This activity can include changes to logistics or software distribution procedures. Changes to the software system requirements, architecture, or design are done within other Technical processes.

2) Identify and record trends of incidents, problems, and maintenance and logistics actions.

NOTE 1 Trend data and problem resolution reports are used to inform operations and maintenance personnel, customers, and other stakeholders and projects that are creating or utilizing similar system entities.

NOTE 2 Incident and problem reporting, including resulting action taken, is tracked through the incident and process management activity of the Quality Assurance process.

3) Maintain traceability of the system elements being maintained.

NOTE Bidirectional traceability is maintained between the recorded maintenance actions and the software system elements and life cycle artifacts. Changes in software asset management, such as assignment of software licenses to replacement systems, are recorded.

4) Provide key artifacts and information items that have been selected for baselines.

NOTE The Configuration Management process is used to establish and maintain configuration items and baselines and to track licenses and data rights. This process identifies candidates for the baseline, and the Information Management process controls the information items, such as maintenance procedures.

5) Monitor and measure customer satisfaction with system and maintenance support.

NOTE ISO 10004:2012 contains guidelines for monitoring and measuring customer satisfaction. When customer satisfaction data is collected, it is then used in the Quality Management process.

[ISO/IEC/IEEE 12207:2017]

6.1.9 Maintenance and logistics results management

This activity and tasks are often activated after the software transition or maintenance processes start and are called iteratively when the need for modification arises.

6.2 Problem and modification analysis

6.2.1 Problem and modification analysis task

During the problem and modification analysis task, perform the following:

a) analyse MRs/PRs;

b) replicate or verify the problem;

c) develop options for implementing the modification, including estimated timeline of implementation;

d) document the MR/PR, results, and modification options;

e) obtain approval for the selected modification option.

Input for the problem and modification analysis activity should be a validated modification request or problem report, system/project records and information items, and requirements records and information items.

Before modifying the system, analyse the MR/PR to determine its impact on the organization, the existing system, and the interfacing systems; develop and document recommended potential solutions; and obtain approval to implement the selected solution.

6.2.2 MR/PR feasibility review

To determine whether the requested MR/PR is feasible, the following tasks should be performed:

a) determine if there is adequate staff to implement the proposed change;

b) determine if there is adequate budget to implement the proposed change;

c) determine if sufficient resources are available and whether this modification affects ongoing or projected projects;

d) determine the operational issues to be considered; for example, what are the anticipated changes to system interface requirements, the expected useful life of the system, the operational priorities, if it is not implemented;

e) determine handling priority;

f) classify the type of maintenance;

g) determine the impact to current and future users;

h) determine safety and security implications;

i) estimate the size and magnitude of the modification;

j) identify where the MR/PR impact would affect other items;

k) evaluate any software or hardware constraints that may result from the changes;

l) determine short-term and long-term costs, including cost of management and technical staff for implementation;

m) determine the value of the benefit of making the modification;

n) determine the impact on existing schedules;

o) document any project or software risks resulting from the impact analysis;

p) determine the level of test and evaluation required;

q) place developed artefacts under CM.

6.2.3 MR/PR replication or verification

In order to help ensure that the requested problem reports are valid, replicate or verify problems by performing the following tasks (unless already completed as part of evaluating the cost/schedule impact):

a) develop a test strategy to verify the problem;

b) obtain affected software version from CM;

c) install affected version;

d) run test to verify problem, preferably with a copy of the affected data;

e) document test results.

If the problem cannot be replicated for some reason (e.g. confidentiality of the data), other items such as organization rules, policies, records and information items should be checked.

Based upon the analysis, options are developed for implementing the modification.

6.2.4 MR/PR implementation

The following tasks should be performed:

a) determine if a work-around exists for problems; if so, provide the work-around to the operator or user to be used in the interim until a more permanent correction can be implemented (this task at times can be unnecessary for adaptive or perfective maintenance);

b) define firm requirements for the modification;

c) develop different options to implement the modification;

d) determine the impacts the options have on the system hardware and users;

e) perform a risk analysis for each of the options identified;

f) record acceptance or rejection of the proposed option;

g) develop an agreed-upon plan to implement the change.

6.2.5 MR/PR reviewing and recording

The following reviewing and recording tasks should be performed:

a) verify that system information (e.g. comments in code, requirements, design specifications, and test scripts) and information for users has been created or updated to reflect maintenance changes;

b) review the proposed test strategy and schedule for accuracy;

c) review resource estimates for accuracy;

d) update the status accounting database;

e) include a disposition recommendation to indicate whether the MR/PR is to be approved or disapproved.

Approval should also be obtained when maintenance is performed when agreements are not used to initiate maintenance.

6.2.6 MR/PR approval

The following tasks may be performed to obtain approval:

a) provide analysis results for approval by appropriate CM groups;

b) participate at discussions regarding the modification;

c) upon approval, update the status of the modification request; and upon approval, update the requirements if the request is an enhancement (improvement).

During the modification implementation task, develop and test the modification of the software product (may be as part of the ISO/IEC/IEEE 12207 technical processes if identified in the maintenance strategy).

The maintenance organization performs analysis, then invokes the implementation process of ISO/IEC/IEEE 12207 to effect the modification (if identified as part of the maintenance strategy).

The maintenance organization conducts analysis and determines which records and information items, software units, and versions thereof need to be modified.

6.2.7 MR/PR additional analysis recording

The results of this additional analysis should be recorded. This effort includes the following tasks:

a) identify the elements to be modified in the existing system, and identify changes to testing process / scripts so that defects can be captured in future releases;

b) identify the interface elements affected by the modification;

c) identify the records to be updated;

d) update the records.

6.2.8 MR/PR implementation using technical processes

To implement the modifications through the technical processes, the following tasks should be performed:

a) define and record test and evaluation criteria for testing and evaluating the modified and the un-modified parts (software units, components, and configuration items) of the system;

b) implement the new and modified requirements completely and correctly avoiding impact on the unmodified software functions.

The system/software requirements definition process in ISO/IEC/IEEE 12207 (technical processes) is satisfied by the process implementation and problem and modification analysis activities of the maintenance process.

6.2.9 MR/PR review of modified system

Review(s) should be conducted with the organization authorizing the modification to determine the integrity of the modified system.

The following tasks should be performed:

a) establish traceability of the MR/PR as required by the users' process (e.g. tracing between requirements, design, code and test), from requirements through to design, and to code;

b) verify testability of the code;

c) verify conformance with coding standards;

d) verify that only necessary software components are modified;

e) verify that the new software components are integrated properly;

f) check records and information items for accuracy;

g) ensure software files are successfully compiled for testing;

h) perform testing for the changes (may be performed on a separate test environment);

i) perform system tests on a fully integrated system (may be performed on a separate test environment);

j) develop test report.

6.2.10 MR/PR approval and implementation

Approval should be obtained for the satisfactory completion of the modification as specified in the contract. If maintenance has been implemented without an agreement, approval should also be obtained.

The following tasks should be performed:

a) obtain approval through the QA life cycle supporting process (ISO/IEC/IEEE 12207);

b) verify that the process has been followed;

c) release the change to the affected systems or make it available for download;

d) conduct functional and physical configuration audits, as required;

e) notify the operators;

f) perform installation and training as required by agreement.

These tasks can be performed remotely as implied in c) above or as a part of "self-maintenance" by the organization.

7 Software disposal

7.1 General

Once a software product has reached the end of its useful life, a process for disposing of the software should be established. An analysis should be performed to assist in making the decision to dispose a software product. The analysis is often economic-based and may be included in the disposal strategy. Analysis should determine if it is cost effective to:

a) retain capability because it is still a valid requirement;

b) retain outdated technology;

c) shift to new technology by developing a new software or updated software product;

d) develop, for example, either

   1) an update for the existing product adapted to the new, non-obsolete environment,

   2) a new software product to achieve modularity,

   3) a new software product to facilitate maintenance,

   4) a new software product to achieve standardization, or

   5) a new software product to facilitate supplier independence.

The software product may be replaced by a new software product; but in some cases, it may not be replaced. In order to dispose of a software product, the tasks needed to accomplish the disposal should be determined, developed and recorded. Consideration should be given to storing data, information, and other system artefacts, from the software product to be disposed, to allow access in the future.

All artefacts from the disposal process should be controlled by CM.

7.2 Disposal strategy

7.2.1 General

A disposal strategy to remove active support by the operation and maintenance organizations should be developed and recorded. Due account should be taken of what happens to the hardware on which the software is stored, including design artefacts. Along with cyber security concerns this also protects intellectual property or trade secrets. The strategy implementation tasks should involve users.

7.2.2 Disposal strategy items

The strategy should address the items listed below:

a) cessation of full or partial support after a certain period of time;

b) archiving or destruction of the software product and its associated records and information items;

c) assign responsibility for any future residual support issues;

d) transition to the new software product (e.g. COTS, new product development), if applicable;

e) accessibility, access controls, and retention requirements of archived data.

7.2.3 Disposal tasks

As part of disposal, perform the following tasks:

a) analyse the disposal requirements;

b) determine the impact of retiring the software product, including but not limited to, technical interfaces to third party users, interfaces to data warehouse(s), statutory reporting, software and hardware infrastructure;

c) establish a schedule for disposing of the software product;

d) identify the responsibility for any future residual support;

e) define and document the disposal resources.

Users should be given notification of the disposal schedules and activities.

7.2.4 Disposal notification

Notifications should include the following:

a) description of the replacement or upgrade with its date of availability;

b) statement of why the support product is no longer to be supported;

c) description of other support options available, once support has been removed.

7.2.5 Disposal notification tasks

Perform the following tasks:

a) identify all the locations which are affected;

b) identify site specific issues;

c) promulgate the strategy and schedule;

d) process site feedback.

When the scheduled disposal happens, notification should be sent to all concerned. All associated development document records, including logs, code, data, records and information items, should be placed in archives, when appropriate.

7.2.6 Post-disposal tasks

The following tasks should be performed:

a) promulgate changes to the disposal schedule;

b) document site specific issues and how they can be resolved;

c) archive the old software, records and data;

d) remove the old equipment.

Data used by or associated with the retired software product should be accessible in accordance with the contract requirements for data protection and audit applicable to the data.

Consideration should be given to archiving records where they can be accessed, if necessary in the future.

7.2.7 Archiving records

Perform the following task: store the disposed software, data and information items obtained during the disposal process.

8 Implementation considerations

8.1 General

The (software) maintenance life cycle process begins with development of a strategy and planning for maintenance and ends with the disposal of the software product. It includes modification of code, records and information items due to a problem or need for improvement. The objective of the maintenance process is to modify an existing software product while preserving its integrity. Some systems contain software. Systems engineering aspects are discussed in ISO/IEC/IEEE 15288. This clause provides implementation considerations.

The maintenance process is needed because the operational environment detects errors and because it introduces the need for new and/or modified capability. If the software product is developed using model-based systems engineering (MBSE) or infrastructure tools, maintenance is still needed. MBSE and infrastructure tools facilitate maintenance but do not eliminate the requirement for maintenance. If no application code is developed, i.e., the software product consists solely of off-the-shelf products, maintenance may still be required to replace obsolete products and updates for security issues. Maintenance of off-the-shelf software products by the acquirer or supplier can involve modification of the interfaces, both data and operational, to the product.

Consideration should be given to implicit requirements and constraints imposed on the original software licence owner. Circumstances may have changed and some of the original requirements may no longer be applicable.

During implementation of the technical processes of ISO/IEC/IEEE 12207, any problems detected are recorded and monitored. MRs or PRs are submitted. Often, these are referred to as change requests. The decision management process of ISO/IEC/IEEE 12207 analyses and resolves problems. It also determines if an MR/PR is a problem or an enhancement. The CM process of ISO/IEC/IEEE 12207 records and reports the status of MRs/PRs. The configuration control activity of the CM process then decides whether to approve the request. Approved MRs/PRs are then implemented by initiating the maintenance process.

Maintenance is needed to help the software product satisfy the user requirements. Maintenance is applicable to software developed using any development life cycle model (e.g. incremental, waterfall, evolutionary, agile, continuous iterative development).

Constraints imposed by the operational environment impact the maintenance process. Often there are 24/7 non-stop operations/maintenance service environments. Software maintenance needs to be performed on systems that cannot be stopped easily. Specific maintenance strategies should be put in place for this type, especially if the system is unable to be stopped for maintenance. Maintenance to such software should be carefully planned in order not to degrade the agreed upon service level. Recovery procedures should be prepared in case maintenance results in a general system failure.

Maintenance may consume a significant portion of life cycle costs. Analysis of the types of maintenance performed helps to provide an understanding of the costs.

8.2 Types of maintenance

Corrective maintenance refers to changes necessitated by actual errors in a software product. If the software product does not meet its requirements corrective maintenance is performed. Emergency maintenance is an unscheduled fix performed to temporarily keep a system operational pending corrective maintenance.

Preventive maintenance refers to the changes necessitated by detecting potential errors in a software product.

Adaptive, additive and perfective maintenance refers to changes to a software product. These changes are those that were not in the specifications or the released software. Adaptive changes are those changes necessary to accommodate a changing environment. Adaptive changes include changes to implement new system interface requirements, new system requirements, or new hardware requirements. Additive changes are those where little if any change is made to the existing software. Perfective changes improve the software product's performance or maintainability. A perfective change can entail providing new functionality improvements for users or reverse engineering to create documentable functionality improvements.

Software maintenance often requires change to an existing structure or system, i.e. software modifications are introduced into an existing architecture and should allow for constraints imposed by the design structure. Adaptive and perfective maintenance can be very costly and time consuming in the case the software was not designed to be scalable for new features. In such case they may consume a significant portion of maintenance costs.

8.3 Agreements for maintenance

The acquirer may enter into an agreement with the original supplier to perform maintenance or a separate third party may be the maintenance organization. Maintenance can also be provided by internal two-party agreements. If the software consists of COTS an internal or third-party agreement may be sufficient.

Maintenance service models should be agreed upon. The models should address the types of maintenance and include new development. When the types of maintenance to be used are identified a comprehensive contract can be made either with a fixed price, or if cost prohibitive, with a time and materials cost basis. Possible types include as extremes:

a) type 1 - blanket contract with fixed amounts for maintenance, which includes the types of maintenance and can include new development;

b) type 2 - split contract for maintenance, which typically includes corrective maintenance for an agreed period.

ISO/IEC/IEEE 12207 provides detailed tasks for the derivation of an agreement between the acquirer and supplier. This should be used to aid the derivation of a maintenance agreement whether acquirer or supplier are from the same or different organizations. These agreements are sometimes referred to as service level agreements. Specific maintenance issues are discussed later.

If the acquirer requires software maintenance from the supplier after delivery, or at the end of a warranty period, this should be stipulated in the agreement. Requirements for updates to information for users should be stipulated in the agreement. Training should also be stipulated. The supplier should then if required prepare procedures for the maintenance task, keep these procedures up to date and check that the activities comply with the agreement requirements and prepared procedures. The items to be maintained, the maintenance procedures, and the time for which they are to be maintained, should be specified in the maintenance plan.

The supplier (the maintenance organization) and the acquirer should first agree on a maintenance agreement and stipulate if required, procedures to incorporate modifications into the maintained software products. Similar procedures may be used by the software licence owner and third-party maintenance organizations.

These procedures should include:

a) basic rules used to determine when the software can be locally corrected or when a new baseline, using the technical processes of ISO/IEC/IEEE 12207 for installation and release, is required;

b) descriptions of types of releases depending on their frequency or their effects on software operation (e.g. emergency releases, periodic releases);

c) ways in which the acquirer is to be informed on the status of current or future changes;

d) methods to confirm that the changes cannot introduce other problems into the software;

e) classification of change according to criticality distinction (e.g. major/minor) which dictates how a change is authorized, processed, and approved.

NOTE ISO/IEC/IEEE 41062 includes detailed procedures for software acquisition, including acquisition of maintenance services.

8.4 Tools for maintenance

A potential means of containing software maintenance costs is to use MBSE and/or infrastructure tools. These environments and tools aid software maintenance activities. The vision for MBSE/infrastructure is an interrelated set of environment and tools supporting all aspects of software development and maintenance (ISO/IEC TR 14471). This interrelated collection of tools should be brought together in the form of a software engineering environment (SEE) to support the methods, policies, guidelines, and standards that support software maintenance activities. A software test environment (STE) should also be provided for the maintenance organization so that the modified software product can be tested in a non-operational environment. The SEE provides the tools to initially develop and modify the software products. The STE provides the test environment. The STE may be used to test the modified software products in a non-operational environment. Any use of tools and special environments should be part of the maintenance strategy.

8.5 Software maintenance measurement

The measurement process from ISO/IEC/IEEE 12207 should be implemented to identify, define, select, apply validate, and improve software measurement for software maintenance. ISO/IEC/IEEE 15939 provides additional measurement information. ISO/IEC 25023 also provides quality measures for maintainability of system or software.

As part of software measurement, it is useful to determine the effort (in terms of resources expended) for corrective, preventive, adaptive, and perfective maintenance. Data may be collected, analysed, and interpreted in order to facilitate maintenance process improvement and to obtain a better understanding of where maintenance costs are being expended. Empirical measurement data may be collected in order to assist life cycle estimating and where appropriate to explain the quality issues of a software product and to explain what can be done to improve the developers' processes within the current project or programme.

8.6 Definition of process

The software maintenance process (Clause 6) should be clearly defined so that all maintenance personnel follow the same process. The measures should support the process and related software process improvement efforts.

8.7 Early involvement in development

8.7.1 General

Involvement of those expected to be the maintenance organization for the software during software development (technical processes in ISO/IEC/IEEE 12207) can benefit and ease future maintenance.

Data suggests that the cost of software maintenance and the maintenance organization's ability to conduct software maintenance is greatly influenced by what occurs or does not occur during software development. In some cases, the maintenance organization cannot be involved due to contractual or other reasons. Specifically, when maintenance is outsourced to a third party, there is often no opportunity for involvement. When the maintenance organization can be involved during development, this should be as early as possible in the process.

8.7.2 Maintenance organization functions

These should include:

a) planning for the logistics of supporting the software product and defining the SEE;

b) planning for knowledge transfer;

c) maintaining the software product;

d) planning for the transition of software products from development to maintenance.

Planning is discussed in detail in Clause 6. Early maintenance organization involvement in projects can help in stating, establishing and clarifying the maintainability requirements of the software. ISO/IEC 25000 should be used to explicitly define maintainability and other software quality characteristics. Maintainability can be improved by maintenance organization participation in the quality assurance, verification, and validation supporting life cycle processes of ISO/IEC/IEEE 12207.

8.7.3 Maintenance organization involvement

It can be useful for those to be involved in maintenance to:

a) participate in review;

b) perform code analysis;

c) trace requirements;

d) perform verification and validation.

8.8 Applying life cycle processes to strengthen maintainability

8.8.1 General

Software maintainability and maintenance are important aspects of dependability. Maintainability is an important feature of software for the acquirer, supplier, and user. Maintainability requirements should be included in the strategy activity of the acquisition process of ISO/IEC/IEEE 12207 and should be evaluated throughout the technical processes of ISO/IEC/IEEE 12207. Variations in design should be monitored throughout development for impact to maintainability. Various measures should be used to define and assess the quality of software. Both qualitative and quantitative evaluation is important. There are five maintainability sub-characteristics that address modularity, reusability, analysability, modifiability and testability of software. These five dimensions affect the effort, time spent in maintenance and ease of software changes (ISO/IEC 25010).

8.8.2 Maintainability and the technical processes

Maintainability requirements, which include non-functional requirements, should be developed and agreed upon early in a software project. When software is acquired from a third party, agreement on the level of maintainability required should be developed between the acquirer and supplier as part of the implementation activities of ISO/IEC/IEEE 12207.

The capability to monitor and evaluate maintainability criteria (or objectives) identified for each requirement can be developed during software development. The capability describes qualitative and quantitative software maintainability requirements specified by the customer. It defines the criteria and the ways of checking them. Qualitative requirements (e.g. usability, maintainability) are used to define the techniques employed to facilitate maintenance costing and resources. Quantitative requirements are used to define maintainability ratings, rating levels, or quality criteria and the measures used to determine values or indicators throughout the various software life cycle phases.

The maintainability sub-characteristics that have been specified should be reviewed and controlled during software development. The maintenance strategy should emphasize the need to sustain and improve the maintainability of the software after its initial development and acceptance. Suppliers should implement requirements for maintainability and maintenance organizations should monitor implementation. The effort should be part of the software maintenance strategy.

One of the key factors in applying ISO/IEC/IEEE 12207 is the development of a software maintenance strategy. Accordingly, a maintenance strategy should be developed; and maintenance should be planned (Clause 6).

A software maintenance strategy can also be established prior to design. Early maintenance organization involvement in a software project has the potential to reduce maintenance costs. There are many actions, including software maintenance planning, to be performed during the technical processes.

Maintainability is affected by the architecture, design, the coding and its programming language and the testing activities. ISO/IEC TR 19759 provides additional information regarding good architecture and design approaches that help maintainability.

8.8.3 Maintainability and specific activities in the technical processes

8.8.3.1 General

6.4 Technical processes

The Technical processes are used to define the requirements for a software system, to transform the requirements into an effective product, to permit consistent reproduction of the product where necessary, to use the product to provide the required services, to sustain the provision of those services, and to dispose of the product when it is retired from service.

The Technical processes define the activities that enable organization and project functions to optimize the benefits and reduce the risks that arise from technical decisions and actions. These activities enable software systems and services to possess the timeliness and availability, cost effectiveness, functionality, reliability, maintainability, producibility, usability, and other qualities required by acquiring and supplying organizations. They also enable products and services to conform to the expectations or legislated requirements of society, including health, safety, security, and environmental factors.

The Technical processes consist of the following:

a) Business or Mission Analysis process;

b) Stakeholder Needs and Requirements Definition process;

c) System/Software Requirements Definition process;

d) Architecture Definition process;

e) Design Definition process;

f) System Analysis process;

g) Implementation process;

h) Integration process;

i) Verification process;

j) Transition process;

k) Validation process;

l) Operation process;

m) Maintenance process;

n) Disposal process.

[Notes removed.]

[ISO/IEC/IEEE 12207:2017]

8.8.3.2 Stakeholder needs and requirements definition process

Maintenance organizations' needs, as one of the stakeholders, should be defined and transformed to maintenance requirements, as well as users, operators, customers and acquirers. The following aspects affect maintainability and should be considered:

a) the identification of variation in maintenance organizations, because maintenance organizations can be end users of the software, trained maintenance personnel, field support personnel, operators of system using the software and so on;

b) maintenance organizations needs and requirements relating to effectiveness and efficiency of maintenance, for examples, performing maintenance without stopping the system/software operation, functional or performance degradation, and exposures of critical risks to health, finance or environment;

c) operational concept for maintenance addressing and resolving maintenance concerns, e.g. remote software maintenance for real time systems of space satellites deployed far distant from maintenance organizations.

8.8.3.3 System/Software requirements definition process

The software specifications should exhaustively and unambiguously describe the maintainability requirements of the software. The following aspects affect maintainability and should be considered:

a) the identification and definition of functions;

b) the accuracy and logical organization of data;

c) interfaces (machine and users), particularly future interfaces;

d) the performance requirements, including the effects of any corrections and additions;

e) requirements imposed by the planned environment including scalability requirements and projected system growth;

f) the granularity of requirements as it impacts the ease or difficulty of traceability;

g) security design.

Also, in system/software requirements definition process in ISO/IEC/IEEE 12207:2017, 6.4.3.3.b) 5) "Define system/software requirements and requirements attributes", the following is included: "v) Transition of operational processes and data from existing automated and manual systems, migration approach and schedule, software installation and acceptance of the product (ISO/IEC/IEEE 12207:2017, 6.4.3)."

8.8.3.4 Architecture definition process

This process transforms the requirements for the software item into an architecture that describes its top-level structure and identifies its software components. An example of this architecture is when a user application layer and infrastructural system operating layer are appropriately independent for an internet type system, such architecture strongly helps to separately maintain application from infrastructural software.

Architectural and software design considerations can greatly simplify maintenance, such as:

a) separation of concerns;

b) encapsulation or containerization of functions;

c) loosely coupled functions;

d) avoidance of duplication/repetition;

e) designing for scalability and additional functions;

f) separation of data from functions.

8.8.3.5 Design definition process

Software which has a clearly discernible design is easier to maintain, especially when the design relies on compartmentalization of functions and traceability of software elements to requirements. Clear indications in the design of software features intended to handle errors or security issues improve testability as well as maintainability.

8.8.3.6 Implementation process

Features of the implementation process of ISO/IEC/IEEE 12207 which affect maintainability are the choice of the program structure, the breakdown into entities and the flow of data through them. As in other activities, it is important to use the data processing knowledge of the programming teams since this can, in particular, reveal possibilities of using parts of existing programs or libraries which have already proven their dependability.

The following aspects, all of which affect maintainability, should be taken into account in choosing the programming language:

a) expertise availability;

b) language portability;

c) language legibility;

d) language stability;

e) self-records and information items;

f) tolerance of programming "tricks" which reduce program clarity;

g) program structuring possibilities;

h) the ease with which new releases can be produced;

i) data structuring possibilities;

j) availability of a compiler and other such tools;

k) stability of a compiler and other such tools;

l) test possibilities during compiling, program implementation and testing;

m) the availability of software engineering and software test environments to assist in production, debugging, configuration management and the satisfaction of reliability and quality requirements;

n) viability period of the various development tools;

o) evaluation of the length of time the supplier proposes to maintain the hardware, software, system, and test environments along with an evaluation of the software tools.

This process also develops, documents, and tests software items and databases.

Software maintainability can be improved by accurate implementation records.

Suggestions for improving maintainability include:

— ensuring legibility;

— pursuing structured code;

— reducing code complexity;

— providing accurate code comments;

— using indentation and white space;

— eliminating classic traps such as language weaknesses and compiler dependent constructs;

— using techniques to facilitate error-tracing;

— ensuring traceability of source code to design;

— using coding standards;

— reducing complexity of decisions and control flow;

— conducting inspections of code and test cases;

— maintaining records during the development cycle.

It is also suggested for improving maintainability to implement additional program code, a function, or a tool to transfer data or database from the existing formats to maintained and updated ones for support of data and software migration.

The implementation process supports migration per ISO/IEC/IEEE 12207:2017, 6.4.7.3 a) 1), "Define an implementation strategy", which includes the following: "vi) implementation priorities to support data and software migration and transition, along with disposal of legacy systems."

8.8.3.7 Verification process

Verification helps confirm that the modified software meets its requirements, without introducing new defects into other parts of the system. ISO/IEC/IEEE 29119-2 provides additional detail on software testing strategy and techniques. The test cases used during software development should be kept for regression testing after modification. In addition, the development history of a program should be available for maintenance in order to better understand the evolution of the software during development. New test cases may be needed for any added functionality.

8.8.3.8 Transition process

Software transition is a controlled and coordinated sequence of actions wherein software is moved from one environment to another and from the organization developing changes to the organization performing software maintenance. This can be a transition of processes within one organization. If maintenance responsibility transfers from one organization to another, a transition strategy should be developed. The strategy should address:

a) the transfer of hardware, software, data, support services, and experience from the developer or acquirer to the maintenance organization;

b) the tasks needed for the maintenance organization to implement the software maintenance strategy (e.g. staffing, training, installation, replicating maintenance problems);

c) assessing knowledge transfer and records or information items;

d) outstanding problems and changes by priority;

e) assessing the test environment readiness;

f) transfer of as-built configuration information for the source code and object code, including open or deferred problem reports or change orders, number and location of media masters which may need to be updated during maintenance.

g) data privacy requirements.

Additionally, when the software system-of-interest is maintained to have different features from the existing system, software system requirements on migration for maintenance organizations including users (e.g. required announcement, communication, tutorial, training) and on migration enabling systems (e.g. required storage to back-up the existing database, of the required system for both updated and existing software system running concurrently) should be addressed as one or more software system requirements related to maintenance. See the tasks of the transition process in ISO/IEC/IEEE 12207:2017, 6.4.10.

6.4.10.3 a) Prepare for the software system transition.

3) Identify information needs and arrange for user documentation and training of operators, users, and other stakeholders necessary for system utilization and support.

NOTE Transition includes migration or activation of user access to the software system.

6) Identify and plan for the necessary enabling systems or services needed to support transition.

NOTE 1 This includes identification of requirements and interfaces for the enabling systems. Transition often involves the use of highly automated infrastructure to deliver, install, and activate or inactivate software. For electronic software distribution, temporary or continuing changes in connectivity are often needed for software and data migration and continuing sustainment. Enabling systems can include backup or alternate systems for use during a transitional period.

6.4.10.3 b) Perform the transition.

3) Install the product in its physical or virtual operational location and interface to its environment.

NOTE The product installation includes configuring it with required operational data, changes to the environment, or business process changes. Databases are instantiated and data migration is performed as applicable.

[ISO/IEC/IEEE 12207:2017]

Guidance for migration is included in ISO/IEC/IEEE 12207:2017.

8.8.3.9 Validation process

This activity helps ensure the requirements of the software system have been met, after constituent software items have been developed and verified. Validation helps confirm that the software system can accomplish its intended use, goals and objectives.

8.9 Records and information items

Maintenance organizations are often faced with providing maintenance for a software product for which little or no records or information items exist. When faced with this situation, the necessary records or information items should be created before the final transition. If appropriate records or information items are not created before the transition to maintenance, records or information items should be created before maintenance is required, or, as maintenance is being performed. The needed records or information items should be created while performing the following tasks in order to prepare for maintenance:

a) understand the problem domain (the type of application); read any records or information items (if available), discuss the software product with developers (if available), and operate the software product;

b) learn the structure and organization (e.g. control flow, data flow, data structures, call graph) of the software product; inventory the software product, place the software product under CM, rebuild the software product from the CM libraries, and analyse the structure of the software product;

c) determine what the software product is doing; review specifications (if available), review overall structure, analyse call trees, read the code, provide oral presentations to other maintenance organizations, and where important add comments to code;

d) gradually build confidence by addressing low risk changes and progressively more risky and complex changes to maintain the stability of the software.

Maintenance organizations should record information about the software product as the guidance listed above is performed. Records or information items such as specifications, programmers' maintenance guides, user's manuals, and installation procedures, should be updated or created as necessary.

There are various factors that influence records or information items creation/updating in the maintenance environment. Some factors include access to source code, availability of tools to analyse the code, ability to operate the software product to determine capabilities, and availability of a software test environment (STE).

9 Software maintenance plan

9.1 General

9.1.1 Overview

This subclause discusses development of the software maintenance plan. The plan prepares for the human and materiel resources required to provide software maintenance for software products. Maintenance organizations should monitor the development effort for maintainability. Results from maintainability analyses should be used as aids in planning for maintenance. This analysis should be provided as input into development of the maintenance plan. The software maintenance plan is described in detail in the 9.1.2 to 9.1.13.

9.1.2 Identification and control of the plan

The following topics should be considered to set the context for identification and control of the plan:

a) the date of issue;

b) the status of the plan;

c) the issuing organization;

d) the approval authority;

e) description of the change procedure for the plan;

f) change history section.

9.1.3 Scope of maintenance

Scope relates to how responsive maintenance will be. It should define how much support will be provided. Budgetary constraints often dictate the scope of maintenance. The scope of maintenance should address:

a) types of maintenance to be performed;

b) cost, resource and budgetary constraints;

c) regulatory requirements;

d) level of records to be maintained;

e) responsiveness;

f) level of training to be provided;

g) delivery support;

h) help desk support;

i) open-source software management (if part of the solution);

j) software license management. (it should be clearly defined who manages what licenses);