INTERNATIONAL STANDARD

ISO/IEC/ IEEE 12207

First edition
2017-11

Systems and software engineering — Software life cycle processes

Ingénierie des systèmes et du logiciel — Processus du cycle de vie du logiciel

Reference number
ISO/IEC/IEEE 12207:2017(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Institute of Electrical and Electronics Engineers, Inc
3 Park Avenue, New York
NY 10016-5997, USA

stds.ipr@ieee.org
www.ieee.org

Contents

Page

List of Illustrations

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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.

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 document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO, IEC, and IEEE 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 will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

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

For an explanation on 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 the following URL www.iso.org/iso/foreword.html.

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

This first edition of ISO/IEC/IEEE 12207 cancels and replaces ISO/IEC 12207:2008 (second edition), which has been technically revised.

Changes in this revision of ISO/IEC/IEEE 12207 were developed in conjunction with a corresponding revision of ISO/IEC/IEEE 15288:2015, *Systems and software engineering – System life cycle processes*. The purpose of these revisions is to accomplish the harmonization of the structures and contents of the two documents, while supporting the requirements of the engineering and assessment communities.

This document was developed with the following goals:

- provide a common terminology between the revision of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207;
- where applicable, provide common process names and process structure between the revision of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207; and
- enable the user community to evolve towards fully harmonized standards, while allowing backward compatibility.

This revision is intended to achieve a fully harmonized view of the system and software life cycle processes.

Introduction

The complexity of software systems has increased to an unprecedented level. This has led to new opportunities, but also to increased challenges for the organizations that create and utilize systems. These challenges exist throughout the life cycle of a system and at all levels of architectural detail. This document provides a common process framework for describing the life cycle of systems created by humans, adopting a Software Engineering approach. Software Engineering is an interdisciplinary approach and means to enable the realization of successful software systems. It focuses on defining stakeholder needs and required functionality early in the development cycle, documenting requirements, and performing design synthesis and system validation while considering the complete problem. It integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation and maintenance. It considers both the business and the technical needs of all stakeholders with the goal of providing a quality product that meets the needs of users and other applicable stakeholders. This life cycle spans the conception of ideas through to the retirement of a system. It provides the processes for acquiring and supplying systems. It helps to improve communication and cooperation among the parties that create, utilize and manage modern software systems in order that they can work in an integrated, coherent fashion. In addition, this framework provides for the assessment and improvement of the life cycle processes.

The processes in this document form a comprehensive set from which an organization can construct software life cycle models appropriate to its products and services. An organization, depending on its purpose, can select and apply an appropriate subset to fulfill that purpose.

This document can be used in one or more of the following modes:

a) By an organization — to help establish an environment of desired processes. These processes can be supported by an infrastructure of methods, procedures, techniques, tools and trained personnel. The organization may then employ this environment to perform and manage its projects and progress software systems through their life cycle stages. In this mode, this document is used to assess conformance of a declared, established environment to its provisions.

b) By a project — to help select, structure and employ the elements of an established environment to provide products and services. In this mode, this document is used in the assessment of conformance of the project to the declared and established environment.

c) By an acquirer and a supplier — to help develop an agreement concerning processes and activities. Via the agreement, the processes and activities in this document are selected, negotiated, agreed to and performed. In this mode, this document is used for guidance in developing the agreement.

d) By process assessors — to serve as a process reference model for use in the performance of process assessments that may be used to support organizational process improvement.

Systems and software engineering — Software life cycle processes

1 Scope

1.1 Overview

This document establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry. It contains processes, activities, and tasks that are applicable during the acquisition, supply, development, operation, maintenance or disposal of software systems, products, and services. These life cycle processes are accomplished through the involvement of stakeholders, with the ultimate goal of achieving customer satisfaction.

This document applies to the acquisition, supply, development, operation, maintenance, and disposal (whether performed internally or externally to an organization) of software systems, products and services, and the software portion of any system, Software includes the software portion of firmware. Those aspects of system definition needed to provide the context for software products and services are included.

This document also provides processes that can be employed for defining, controlling, and improving software life cycle processes within an organization or a project.

The processes, activities, and tasks of this document can also be applied during the acquisition of a system that contains software, either alone or in conjunction with ISO/IEC/IEEE 15288:2015, *Systems and software engineering—System life cycle processes*.

In the context of this document and ISO/IEC/IEEE 15288, there is a continuum of human-made systems from those that use little or no software to those in which software is the primary interest. It is rare to encounter a complex system without software, and all software systems require physical system components (hardware) to operate, either as part of the software system-of-interest or as an enabling system or infrastructure. Thus, the choice of whether to apply this document for the software life cycle processes, or ISO/IEC/IEEE 15288:2015, *Systems and software engineering—System life cycle processes*, depends on the system-of-interest. Processes in both documents have the same process purpose and process outcomes, but differ in activities and tasks to perform software engineering or systems engineering, respectively.

1.2 Purpose

The purpose of this document is to provide a defined set of processes to facilitate communication among acquirers, suppliers and other stakeholders in the life cycle of a software system.

This document is written for acquirers, suppliers, developers, integrators, operators, maintainers, managers, quality assurance managers, and users of software systems, products, and services. It can be used by a single organization in a self-imposed mode or in a multi-party situation. Parties can be from the same organization or from different organizations and the situation can range from an informal agreement to a formal contract.

The processes in this document can be used as a basis for establishing business environments, e.g., methods, procedures, techniques, tools and trained personnel. Annex A provides normative direction regarding the tailoring of these software life cycle processes.

1.3 Field of application

This document applies to the full life cycle of software systems, products, and services, including conception, development, production, utilization, support and retirement, and to their acquisition and supply, whether performed internally or externally to an organization. The life cycle processes of this document can be applied concurrently, iteratively and recursively to a software system and incrementally to its elements.

There is a wide variety of software systems in terms of their purpose, domain of application, complexity, size, novelty, adaptability, quantities, locations, life spans and evolution. This document describes the processes that comprise the life cycle of man-made software systems. It therefore applies to one-of-a-kind software systems, software systems for wide commercial or public distribution, and customized, adaptable software systems. It also applies to a complete stand-alone software system and to software systems that are embedded and integrated into larger, more complex and complete systems.

This document provides a process reference model characterized in terms of the process purpose and the process outcomes that result from the successful execution of the activity tasks. Annex B lists examples of artifacts and information items that may be associated with various processes. This document can therefore be used as a reference model to support process assessment as specified in ISO/IEC 33002:2015. Annex C provides information regarding the use of the software life cycle processes as a process reference model. Annex D describes the process constructs for use in the process reference model. Annex I provides the correspondence between this document and ISO/IEC/IEEE 12207:2008 at the level of process name and process outcome.

1.4 Limitations

This document does not prescribe a specific software life cycle model, development methodology, method, modelling approach, or technique. The users of this document are responsible for selecting a life cycle model for the project and mapping the processes, activities, and tasks in this document into that model. The parties are also responsible for selecting and applying appropriate methodologies, methods, models and techniques suitable for the project.

This document does not establish a management system or require the use of any management system standard. However, it is intended to be compatible with the quality management system specified by ISO 9001, the service management system specified by ISO/IEC 20000-1 (IEEE Std 20000-1), and the information security management system specified by ISO/IEC 27000.

This document does not detail information items in terms of name, format, explicit content and recording media. ISO/IEC/IEEE 15289 addresses the content for life cycle process information items (documentation).

2 Normative references

There are no normative references in this document.

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org
- ISO Online browsing platform: available at http://www.iso.org/obp
- IEEE Standards Dictionary Online: available at http://ieeexplore.ieee.org/xpls/dictionary.jsp

Definitions for other terms typically can be found in ISO/IEC/IEEE 24765, *System and software engineering — Vocabulary*, available at <www.computer.org/sevocab>.

3.1.1
acquirer
stakeholder that acquires or procures a product or service from a supplier

Note 1 to entry: Other terms commonly used for an acquirer are buyer, customer, owner, purchaser or internal/organizational sponsor.

3.1.2
acquisition
process of obtaining a system, product or service

3.1.3
activity
set of cohesive tasks of a process

3.1.4
agile development
software development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continual stakeholder feedback

[SOURCE: ISO/IEC/IEEE 26515: 2011]

3.1.5
agreement
mutual acknowledgement of terms and conditions under which a working relationship is conducted

EXAMPLE Contract, memorandum of agreement.

3.1.6
architecture
<system> fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

[SOURCE: ISO/IEC/IEEE 42010:2011]

3.1.7
architecture framework
conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders

EXAMPLE 1 Generalised Enterprise Reference Architecture and Methodologies (GERAM) [ISO 15704] is an architecture framework.

EXAMPLE 2 Reference Model of Open Distributed Processing (RM-*ODP) [ISO/IEC* 10746] is an architecture framework.

[SOURCE: ISO/IEC/IEEE 42010:2011]

3.1.8
architecture view
work product expressing the architecture of a system from the perspective of specific system concerns

[SOURCE: ISO/IEC/IEEE 42010:2011]

3.1.9
architecture viewpoint
work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns

[SOURCE: ISO/IEC/IEEE 42010:2011]

3.1.10
audit
independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria

3.1.11
baseline
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

[SOURCE: IEEE Std 828-2012]

3.1.12
business process
partially ordered set of enterprise activities that can be executed to achieve some desired end-result in pursuit of a given objective of an organization

3.1.13
concept of operations
verbal and/or graphic statement, in broad outline, of an organization's assumptions or intent in regard to an operation or series of operations

Note 1 to entry: The concept of operations frequently is embodied in long-range strategic plans and annual operational plans. In the latter case, the concept of operations in the plan covers a series of connected operations to be carried out simultaneously or in succession. The concept is designed to give an overall picture of the organization operations. See also operational concept (3.1.28).

Note 2 to entry: It provides the basis for bounding the operating space, system capabilities, interfaces and operating environment.

[SOURCE: ANSI/AIAA G-043A-2012e]

3.1.14
concern
<system> interest in a system relevant to one or more of its stakeholders

Note 1 to entry: A concern pertains to any influence on a system in its environment, including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.

[SOURCE: ISO/IEC/IEEE 42010:2011]

3.1.15
configuration item
item or aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process

EXAMPLE Software, firmware, data, hardware, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions and user manuals), facilities, services, materials, and naturally occurring entities

3.1.16
customer
organization or person that receives a product or service

EXAMPLE Consumer, client, user, acquirer, buyer, or purchaser.

Note 1 to entry: A customer can be internal or external to the organization.

3.1.17
design, verb
<process> to define the architecture, system elements, interfaces, and other characteristics of a system or system element

[SOURCE: ISO/IEC/IEEE 24765:2010, modified, changed 'components' to 'system element']

3.1.18
design, noun
result of the process in 3.1.17

Note 1 to entry: Information, including specification of system elements and their relationships, that is sufficiently complete to support a compliant implementation of the architecture

Note 2 to entry: Design provides the detailed implementation-level physical structure, behavior, temporal relationships, and other attributes of system elements.

3.1.19
design characteristic
design attributes or distinguishing features that pertain to a measurable description of a product or service

3.1.20
enabling system
system that supports a system-of-interest during its life cycle stages but does not necessarily contribute directly to its function during operation

EXAMPLE A configuration management system used to control software elements during software development.

Note 1 to entry: Each enabling system has a life cycle of its own. This document is applicable to each enabling system when, in its own right, it is treated as a system-of-interest.

3.1.21
environment
<system> context determining the setting and circumstances of all influences upon a system

[SOURCE: ISO/IEC/IEEE 42010:2011]

3.1.22
facility
physical means or equipment for facilitating the performance of an action, e.g., buildings, instruments, tools

3.1.23
incident
anomalous or unexpected event, set of events, condition, or situation at any time during the life cycle of a project, product, service, or system

3.1.24
information item
separately identifiable body of information that is produced, stored, and delivered for human use

[SOURCE: ISO/IEC/IEEE 15289:2015]

3.1.25
infrastructure
hardware and software environment to support computer system and software design, development, and modification

3.1.26
life cycle
evolution of a system, product, service, project or other human-made entity from conception through retirement

3.1.27
life cycle model
framework of processes and activities concerned with the life cycle, which can be organized into stages, acting as a common reference for communication and understanding

3.1.28
operational concept
verbal and graphic statement of an organization's assumptions or intent in regard to an operation or series of operations of a system or a related set of systems

Note 1 to entry: The operational concept is designed to give an overall picture of the operations using one or more specific systems, or set of related systems, in the organization's operational environment from the users' and operators' perspective. See also concept of operations (3.1.13).

[SOURCE: ANSI/AIAA G-043A-2012e]

3.1.29
operator
individual or organization that performs the operations of a system

Note 1 to entry: The role of operator and the role of user can be vested, simultaneously or sequentially, in the same individual or organization.

Note 2 to entry: An individual operator combined with knowledge, skills and procedures can be considered as an element of the system.

Note 3 to entry: An operator can perform operations on a system that is operated, or within a system that is operated, depending on whether or not operating instructions are placed within the system boundary.

3.1.30
organization
group of people and facilities with an arrangement of responsibilities, authorities and relationships

EXAMPLE company, corporation, firm, enterprise, institution, charity, sole trader, association, or parts or combination thereof.

Note 1 to entry: An identified part of an organization (even as small as a single individual) or an identified group of organizations can be regarded as an organization if it has responsibilities, authorities and relationships. A body of persons organized for some specific purpose, such as a club, union, corporation, or society, is an organization.

3.1.31
party
organization entering into an agreement

Note 1 to entry: In this document, the agreeing parties are called the acquirer and the supplier.

3.1.32
problem
difficulty, uncertainty, or otherwise realized and undesirable event, set of events, condition, or situation that requires investigation and corrective action

3.1.33
process
set of interrelated or interacting activities that transforms inputs into outputs

3.1.34
process outcome
observable result of the successful achievement of the process purpose

3.1.35
process purpose
high-level objective of performing the process and the likely outcomes of effective implementation of the process

Note 1 to entry: The purpose of implementing the process is to provide benefits to the stakeholders.

3.1.36
product
result of a process

Note 1 to entry: There are four agreed generic product categories: hardware (e.g., engine mechanical part); software (e.g., computer program procedures, and possibly associated documentation and data); services (e.g., transport); and processed materials (e.g., lubricant). Hardware and processed materials are generally tangible products, while software or services are generally intangible.

3.1.37
project
endeavour with defined start and finish criteria undertaken to create a product or service in accordance with specified resources and requirements

Note 1 to entry: A project is sometimes viewed as a unique process comprising coordinated and controlled activities and composed of activities from the Technical Management processes and Technical processes defined in this document.

3.1.38
<project> **portfolio**
collection of projects that addresses the strategic objectives of the organization

3.1.39
qualification
process of demonstrating whether an entity is capable of fulfilling specified requirements

3.1.40
quality assurance
part of quality management focused on providing confidence that quality requirements will be fulfilled

[SOURCE: ISO 9000:2015]

3.1.41
quality characteristic
inherent characteristic of a product, process or system related to a requirement

Note 1 to entry: Critical quality characteristics commonly include those related to health, safety, security assurance, reliability, availability and supportability.

3.1.42
quality management
coordinated activities to direct and control an organization with regard to quality

3.1.43
release
particular version of a configuration item that is made available for a specific purpose

EXAMPLE Test release.

3.1.44
requirement
statement that translates or expresses a need and its associated constraints and conditions

[SOURCE: ISO/IEC/IEEE 29148:2011, modified, NOTE has been removed.]

3.1.45
resource
asset that is utilized or consumed during the execution of a process

Note 1 to entry: Resources include those that are reusable, renewable or consumable.

EXAMPLE diverse entities such as funding, personnel, facilities, capital equipment, tools, and utilities such as power, water, fuel and communication infrastructures.

3.1.46
retirement
withdrawal of active support by the operation and maintenance organization, partial or total replacement by a new system, or installation of an upgraded system

3.1.47
risk
effect of uncertainty on objectives

Note 1 to entry: An effect is a deviation from the expected — positive or negative. A positive effect is also known as an opportunity.

Note 2 to entry: Objectives can have different aspects (such as financial, health and safety, and environmental goals) and can apply at different levels (such as strategic, organization-wide, project, product and process).

Note 3 to entry: Risk is often characterized by reference to potential events and consequences, or a combination of these.

Note 4 to entry: Risk is often expressed in terms of a combination of the consequences of an event (including changes in circumstances) and the associated likelihood of occurrence.

Note 5 to entry: Uncertainty is the state, even partial, of deficiency of information related to understanding or knowledge of an event, its consequence, or likelihood.

[SOURCE: ISO Guide 73:2009, definition 1.1]

3.1.48
safety
expectation that a system does not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered

3.1.49
security
protection against intentional subversion or forced failure; a composite of four attributes – confidentiality, integrity, availability, and accountability – plus aspects of a fifth, usability, all of which have the related issue of their assurance

[SOURCE: NATO AEP-67]

3.1.50
service
performance of activities, work, or duties

Note 1 to entry: A service is self-contained, coherent, discrete, and can be composed of other services.

Note 2 to entry: A service is generally an intangible product.

3.1.51
software element
system element that is software

3.1.52
software engineering
application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software

3.1.53
software item
source code, object code, control code, control data, or a collection of these items

Note 1 to entry: A software item can be viewed as a system element of this document and of ISO/IEC 15288:2015. Software items are typically configuration items.

3.1.54
software product
set of computer programs, procedures, and possibly associated documentation and data

Note 1 to entry: A software product is a software system viewed as the output (product) resulting from a process.

3.1.55
software system
system for which software is of primary importance to the stakeholders

Note 1 to entry: In the most general case, a software system is comprised of hardware, software, people, and manual procedures.

Note 2 to entry: In a software system, software is the leading driver in meeting system requirements.

3.1.56
software system element
member of a set of elements that constitute a software system

Note 1 to entry: A software system element can include one or more software units, software elements, hardware units, hardware elements, services, and other system elements and systems.

Note 2 to entry: A software system element can be viewed as a system element.

3.1.57
software unit
atomic-level software component of the software architecture that can be subjected to standalone testing

Note 1 to entry: Some software units are separately compilable pieces of code.

[SOURCE: ISO 26262-1:2011, modified, Note 1 to entry added.]

3.1.58
stage
period within the life cycle of an entity that relates to the state of its description or realization

Note 1 to entry: As used in this document, stages relate to major progress and achievement milestones of the entity through its life cycle.

Note 2 to entry: Stages often overlap.

3.1.59
stakeholder
individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations

EXAMPLE End users, end user organizations, supporters, developers, producers, trainers, maintainers, disposers, acquirers, supplier organizations and regulatory bodies.

Note 1 to entry: Some stakeholders can have interests that oppose each other or oppose the system.

3.1.60
supplier
organization or an individual that enters into an agreement with the acquirer for the supply of a product or service

Note 1 to entry: Other terms commonly used for supplier are contractor, producer, seller, or vendor.

Note 2 to entry: The acquirer and the supplier sometimes are part of the same organization.

3.1.61
system
combination of interacting elements organized to achieve one or more stated purposes

Note 1 to entry: A system is sometimes considered as a product or as the services it provides.

Note 2 to entry: In practice, the interpretation of its meaning is frequently clarified by the use of an associative noun, e.g., aircraft system or database management system. Alternatively, the word “system” is substituted simply by a context-dependent synonym, e.g., aircraft or database, though this potentially obscures a system principles perspective.

Note 3 to entry: A system can include the associated equipment, facilities, material, software, firmware, technical documentation, services and personnel required for operations and support to the degree necessary for use in its intended environment.

Note 4 to entry: See for comparison: enabling system, system-of-interest, system of systems.

3.1.62
system element
member of a set of elements that constitute a system

EXAMPLE Hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities or any combination.

Note 1 to entry: A system element is a discrete part of a system that can be implemented to fulfill specified requirements.

3.1.63
system-of-interest
SOI
system whose life cycle is under consideration

3.1.64
system of systems
SoS
set of systems that integrate or interoperate to provide a unique capability that none of the constituent systems can accomplish on its own

Note 1 to entry: Each constituent system is a useful system by itself, having its own management, goals, and resources, but coordinates within the SoS to provide the unique capability of the SoS.

3.1.65
systems engineering
interdisciplinary approach governing the total technical and managerial effort required to transform a set of stakeholder needs, expectations, and constraints into a solution and to support that solution throughout its life.

3.1.66
task
required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process

3.1.67
technical management
application of technical and administrative resources to plan, organize and control engineering functions

3.1.68
trade-off
decision-making actions that select from various requirements and alternative solutions on the basis of net benefit to the stakeholders

3.1.69
traceability
degree to which a relationship can be established among two or more logical entities, especially entities having a predecessor-successor or master-subordinate relationship to one another, such as requirements, system elements, verifications, or tasks

EXAMPLE Software features and test cases are typically traced to software requirements.

3.1.70
user
individual or group that interacts with a system or benefits from a system during its utilization

Note 1 to entry: The role of user and the role of operator are sometimes vested, simultaneously or sequentially, in the same individual or organization.

[SOURCE: ISO/IEC 25010:2011, modified, Note 1 to entry added.]

3.1.71
validation
confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled

Note 1 to entry: A system is able to accomplish its intended use, goals and objectives (i.e., meet stakeholder requirements) in the intended operational environment. The right system was built.

Note 2 to entry: In a life cycle context, validation involves the set of activities for gaining confidence that a system is able to accomplish its intended use, goals and objectives in an environment like the operational environment.

3.1.72
verification
confirmation, through the provision of objective evidence, that specified requirements have been fulfilled

Note 1 to entry: Verification is a set of activities that compares a system or system element against the required characteristics. This includes, but is not limited to specified requirements, design, descriptions, and the system itself. The system was built right.

[SOURCE: ISO 9000:2015, modified, Note 1 to entry added.]

3.2 Abbreviated terms

CCB	Configuration Control Board
CM	Configuration Management
COTS	Commercial-Off-The-Shelf
FCA	Functional Configuration Audit
FOSS	Free and Open Source Software
GUI	Graphical User Interface
NDI	Non-Developmental Items
QA	Quality Assurance
PCA	Physical Configuration Audit
PESTEL	Political, Economic, Social, Technological, Environmental, and Legal
PMI	Project Management Institute
PMP	Project Management Plan
PRM	Process Reference Model
SCM	Software Configuration Management
SDP	Software Development Plan
SEMP	Systems Engineering Management Plan
SOI	System-of-Interest
SoS	System of Systems
SWOT	Strengths, Weaknesses, Opportunities, Threats
WBS	Work Breakdown Structure

4 Conformance

4.1 Intended usage

The requirements in this document are contained in Clause 6 and Annex A. This document provides requirements for a number of processes suitable for usage during the life cycle of a software system or product. It is recognized that particular projects or organizations may not need to use all of the processes provided by this document. Therefore, implementation of this document typically involves selecting and declaring a set of processes suitable to the organization or project. There are two ways that an implementation can be claimed to conform to the provisions of this document — full conformance and tailored conformance.

There are two criteria for claiming full conformance. Achieving either criterion suffices for conformance, although the chosen criterion (or criteria) shall be stated in the claim. Claiming “full conformance to tasks” asserts that all of the requirements of the activities and tasks of the declared set of processes are achieved. Alternatively, claiming “full conformance to outcomes” asserts that all of the required outcomes of the declared set of processes are

achieved. Full conformance to outcomes permits greater freedom in the implementation of conforming processes and can be useful for implementing processes to be used in the context of an innovative life cycle model.

NOTE 1 Options for conformance are provided for needed flexibility in the application of this document. Each process has a set of objectives (phrased as "outcomes") and a set of activities and tasks that represent one way to achieve the objectives.

NOTE 2 Users who implement the activities and tasks of the declared set of processes can assert full conformance to tasks of the selected processes. Some users, however, might have innovative process variants that achieve the objectives (i.e., the outcomes) of the declared set of processes without implementing all of the activities and tasks. These users can assert full conformance to the outcomes of the declared set of processes. The two criteria — conformance to task and conformance to outcome — are necessarily not equivalent since specific performance of activities and tasks can require, in some cases, a higher level of capability than just the achievement of outcomes.

NOTE 3 When this document is used to help develop an agreement between an acquirer and a supplier, clauses of this document can be selected for incorporation in the agreement with or without modification. In this case, it is more appropriate for the acquirer and supplier to claim compliance with the agreement than conformance with this document.

NOTE 4 An organization (for example, national, industrial association, company) imposing this document, as a condition of trade, can specify and make public the minimum set of required processes, outcomes, activities, and tasks, which constitute suppliers' compliance with the conditions of trade.

NOTE 5 Requirements of this document are marked by the use of the verb "shall". Recommendations are marked by the use of the verb "should". Permissions are marked by the use of the verb "may". However, despite the verb that is used, the requirements for conformance are selected as described previously.

4.2 Full conformance

4.2.1 Full conformance to outcomes

A claim of full conformance declares the set of processes for which conformance is claimed. Full conformance to outcomes is achieved by demonstrating that all of the outcomes of the declared set of processes have been achieved. In this situation, the provisions for activities and tasks of the declared set of processes are guidance rather than requirements, regardless of the verb form that is used in the provision.

One intended use of this document is to facilitate process assessment and improvement. For this purpose, the objectives of each process are written in the form of 'outcomes' compatible with the provisions of ISO/IEC 33002. That standard provides for the assessment of the processes of this document, providing a basis for improvement. Users intending process assessment and improvement may use the process outcomes written in this document as the "process reference model" required by ISO/IEC 33002.

4.2.2 Full conformance to tasks

A claim of full conformance declares the set of processes for which conformance is claimed. Full conformance to tasks is achieved by demonstrating that all of the requirements of the activities and tasks of the declared set of processes have been achieved. In this situation, the provisions for the outcomes of the declared set of processes are guidance rather than requirements, regardless of the verb form that is used in the provision.

NOTE A claim of full conformance to tasks can be appropriate in contractual situations where an acquirer or a regulator requires detailed understanding of the suppliers' processes.

4.3 Tailored conformance

When this document is used as a basis for establishing a set of processes that do not qualify for full conformance, the clauses of this document are selected or modified in accordance with the tailoring process prescribed in Annex A. The tailored text, for which tailored conformance is claimed, is declared. Tailored conformance is achieved by demonstrating that the outcomes, activities, and tasks, as tailored, have been achieved.

5 Key concepts and application

5.1 Introduction

This clause is included to highlight and to help explain essential concepts on which this document is based.

NOTE Further elaboration of these concepts can be found in ISO/IEC TS 24748-1, ISO/IEC TR 24748-2, and ISO/IEC TR24748-3 on the application of life cycle management.

5.2 Software system concepts

5.2.1 Software systems

The software systems considered in this document are human-made, created and utilized to provide products or services in defined environments for the benefit of users and other stakeholders. These software systems can include the following system elements: hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, services, materials and naturally occurring entities. As viewed by the user, they are thought of as products or services.

This document applies to systems for which software is of primary importance to the stakeholders. It is based upon the general principles of systems engineering and software engineering. It is a fundamental premise of this document that software always exists in the context of a system. Since software does not operate without hardware, the processor upon which the software is executed can be considered as part of the system. Alternatively, hardware or services hosting the software system and handling communications with other systems can also be viewed as enabling systems or external systems in the operating environment.

The perception and definition of a particular software system, its architecture, and its elements depend on a stakeholder's interests and responsibilities. One stakeholder's system-of-interest can be viewed as a system element in another stakeholder's system-of-interest. Furthermore, a system-of-interest can be viewed as being part of the environment for another stakeholder's system-of-interest.

The following are key points regarding the characteristics of systems-of-interest:

a) defined boundaries encapsulate meaningful needs and practical solutions;

b) there is a hierarchical or other relationship between system elements;

c) an entity at any level in the system-of-interest can be viewed as a system;

d) a system comprises an integrated, defined set of subordinate system elements;

e) humans can be viewed as both users external to a system and as system elements (i.e., operators) within a system; and

f) a system can be viewed in isolation as an entity, i.e., a product; or as a collection of functions capable of interacting with its surrounding environment, i.e., a set of services.

Whatever the boundaries chosen to define the system, the concepts in this document are generic and permit a practitioner to correlate or adapt individual instances of life cycles to its system principles.

5.2.2 Software system structure

The life cycle processes in this document are described in relation to a software system that is composed of a set of interacting system elements (including software elements), each of which can be implemented to fulfill its respective specified requirements (Figure 1). Responsibility for the implementation of any system element may therefore be delegated to another party through an agreement.

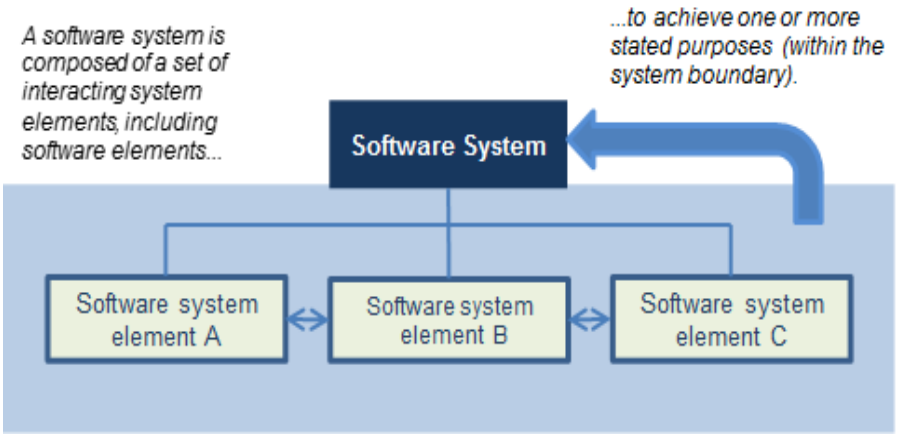

Figure 1 — Software system and software system element relationship

The relationship between the software system and its complete set of system elements can typically be represented showing relationships between the elements – often depicted as a hierarchy for the simplest of systems-of-interest. Decomposition is one approach to some software activities. Other approaches include the object-oriented approach, where the system elements are laid out in a flat (non-hierarchical) description such as in a network diagram. For more complex software systems-of-interest, a prospective system element may need to be considered as a system (that in turn is comprised of system elements) before a complete set of system elements can be defined with confidence (Figure 2). In this manner, the appropriate system life cycle processes are applied recursively to a system-of-interest to resolve its structure to the point where understandable and manageable software system elements can be implemented (created, adapted, acquired, or reused).

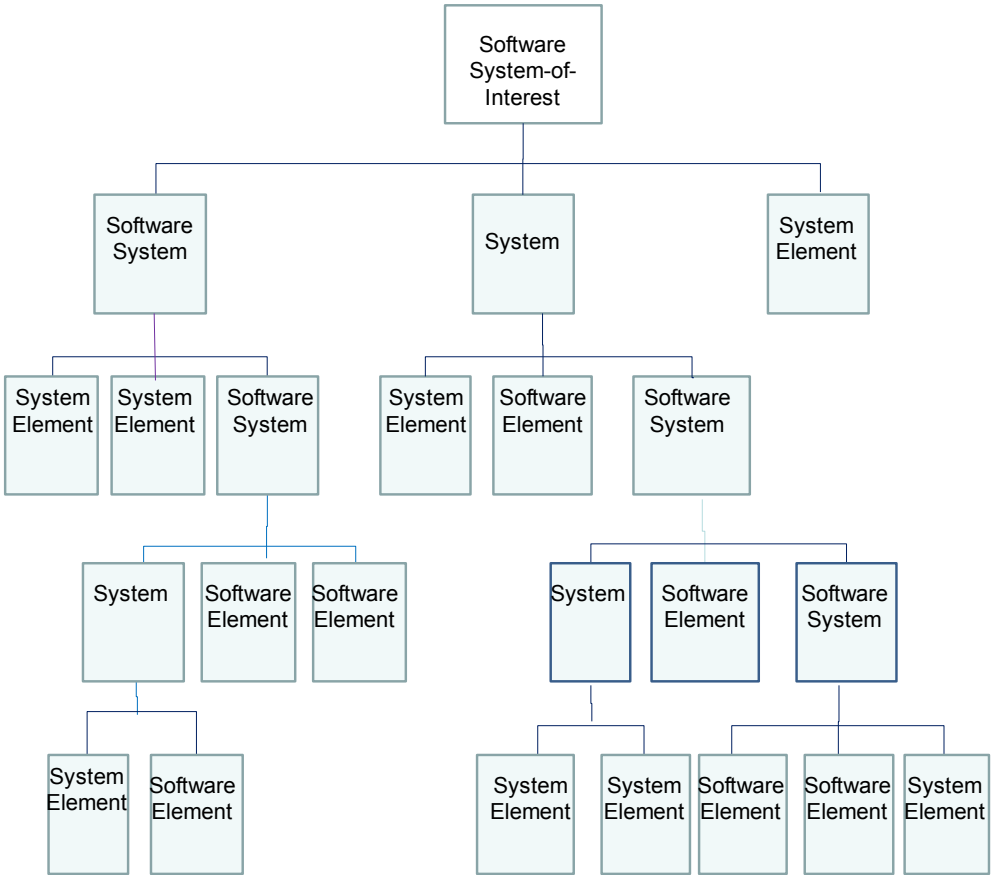

Figure 2 —Example of software system-of-interest structure

While Figures 1 and 2 imply a hierarchical relationship, in reality there are an increasing number of systems that, from one or more aspects, are not hierarchical, such as networks and other distributed systems. Annex G discusses the concept of a system of systems (SoS).

NOTE Decomposition is an activity fundamental to many software activities. Not all decompositions imply the designation of new software system elements and the corresponding recursive application of the activity. Designation of a decomposed construct as an element is necessary only when it is appropriate to apply distinct requirements, design, or implementation activities to its development. One example of an appropriate situation is when the element is to be developed by a distinct organization. Another example is when management determines that it is appropriate to distinctly monitor the status of the development or customization of the element.

5.2.3 Enabling systems

Throughout the life cycle of a system-of-interest, essential services are required from systems that are not directly a part of the operational environment of the system-of-interest, e.g., modelling system, training system, maintenance system. Each of these systems enables a part, e.g., a stage of the life cycle of the system-of-interest to be conducted. Termed "enabling systems", they facilitate progression of the system-of-interest through its life cycle.

The relationship between the services delivered to the operational environment by the system-of-interest and the services delivered by the enabling systems to the system-of-interest is shown in Figure 3. Enabling systems can be seen to contribute indirectly to the services provided by the system-of-interest. The interrelationships between the system-of-interest and the enabling systems can be bidirectional or a one-way relationship. In addition to interacting with enabling systems, the system-of-interest can also interact with other systems in the operating environment, shown as Systems A, B, and C. Requirements for interfaces with enabling systems and other systems in the operational environment are included in the requirements for the system-of-interest.

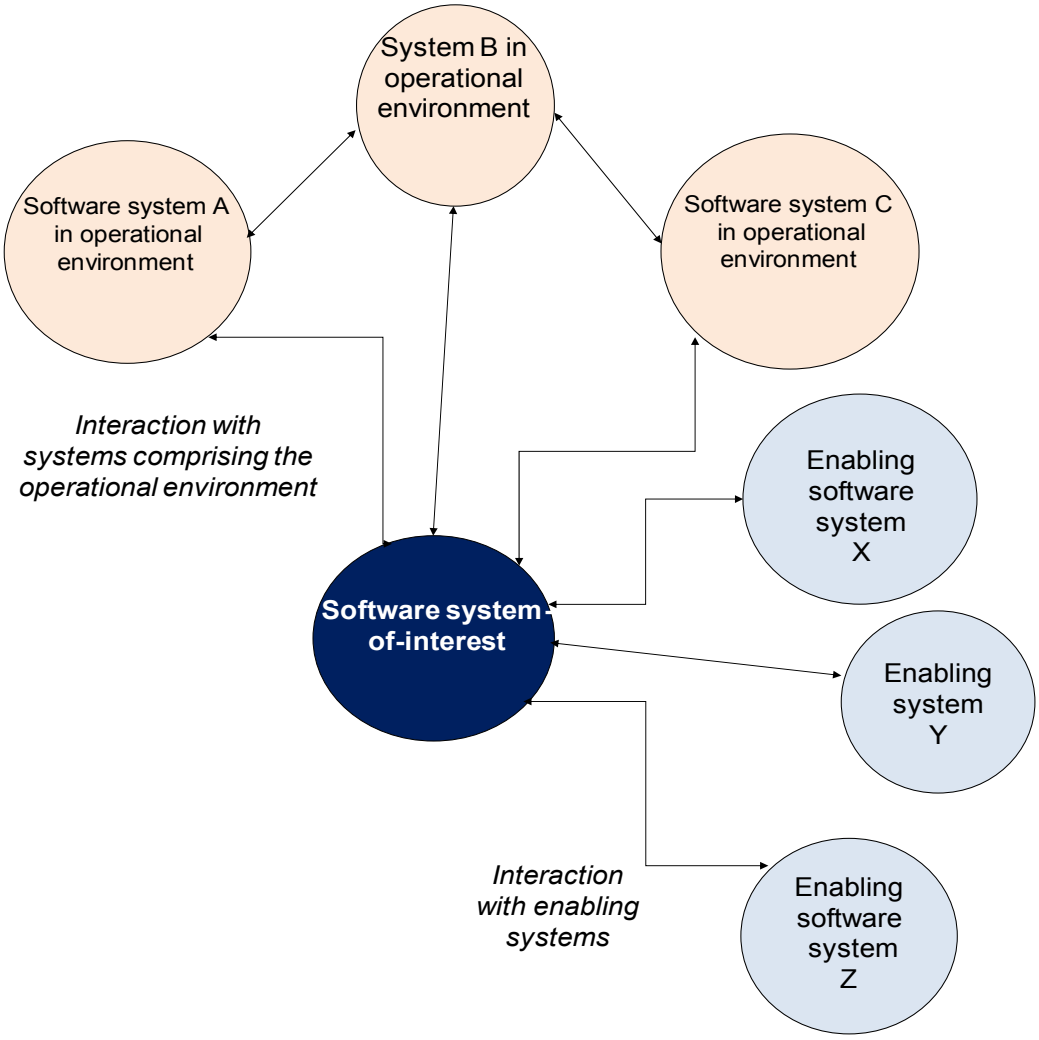

Figure 3 —Software system-of-interest, its operational environment and enabling systems

During a stage in the software life cycle, the relevant enabling systems and the system-of-interest are considered together. Since they are interdependent, they can also be viewed as a system. When a suitable enabling system does not already exist, the project that is responsible for the system-of-interest can be directly responsible for

creating and using the enabling system. Creating the enabling system can be viewed as a separate project and subsequently as another system-of-interest.

Further elaboration of these concepts can be found in ISO/IEC/IEEE 24748 (all parts) on the application of life cycle processes.

NOTE Enabling systems in software development include software development and test environments for target platforms.

5.2.4 Life cycle processes for the software system

In the software system, the requirements, architecture, and design processes at the system level result in an allocation of the system requirements to various elements. The software system-of-interest is implemented primarily by analyzing the software system requirements, architecture, and design and determining which functions will be implemented in software or by other elements, implementing the software and other elements, and integrating the elements as a software system. Therefore, a software product or service can be treated as an element of a software system.

In some cases, the architectural definition of a software system can indicate that it is appropriate to consider it as comprising a set of distinct subordinate elements. In turn, each of the software elements can be treated as a distinct software system as described previously. In these cases, this document may be applied recursively to procure or develop the subordinate elements.

This document has a strong relationship with ISO/IEC/IEEE 15288:2015, *Systems and Software Engineering--System Life Cycle Processes,* and is more applicable to software systems. To account for situations in which both ISO/IEC/IEEE 15288:2015 and ISO/IEC/IEEE 12207:2017 are applied (e.g., a development of a system containing software, or the development of a software system containing hardware), their process structures are harmonized to be identical. The processes of this document directly correspond to processes of ISO/IEC/IEEE 15288 with specialization for software products and services.

In the case where the system non-software elements have primary importance, an organization may decide to apply ISO/IEC/IEEE 15288 to perform the appropriate life cycle processes, activities and tasks. For each software element of the system, the organization may apply this document to create, adapt, acquire, or reuse the software elements.

5.3 Organization and project concepts

5.3.1 Organizations

When an organization, as a whole or a part, enters into an agreement, it is sometimes called a “party” to the agreement. Parties can be from the same organization or from separate organizations. An organization can be as small as a single individual, if the individual is assigned responsibilities and authorities.

In informal terms, the organization that is responsible for executing a process is sometimes referred to by the name of that process. For example, the organization executing the Acquisition process is sometimes called the “acquirer”. Other examples include supplier, implementer, maintainer, and operator.

A few other terms are applied to organizations in this document: “user” can be the organization or individuals that directly engage with or benefit from the utilization of the product or service; “customer” refers to the user and acquirer collectively; and “stakeholder” refers to an individual or organization with an interest in the system.

The processes and organizations are only related functionally. This document does not dictate or imply a structure for an organization, nor does it specify that particular processes are to be executed by particular parts of the organization. It is the responsibility of the organization that implements this document to define a suitable structure for the organization and assign appropriate roles for the execution of processes.

The processes in this document form a comprehensive set to serve various organizations. An organization, small or large, depending on its business purpose or its acquisition strategy, can select an appropriate set of the processes (and associated activities and tasks) to fulfill that purpose. An organization can perform one process or more than one process.

This document is intended to be applied by an organization internally or externally by two or more organizations. When applied internally, the two agreeing parties typically act under the terms of an agreement that may vary in formality under different circumstances. When applied externally, the two agreeing parties typically act under the terms of a contract. This document uses the term "agreement" to apply to either situation.

For the purpose of this document, any project is assumed to be conducted within the context of an organization. This is important, because a project is dependent upon various outcomes produced by the business processes of the organization, e.g., employees to staff the project and facilities to house the project. For this purpose, this document provides a set of "Organizational Project-Enabling" processes. These processes are not assumed to be adequate to operate a business; instead the processes, considered as a collection, are intended to state the minimum set of dependencies that the project places upon the organization.

5.3.2 Organization and project-level adoption

Modern businesses strive to develop a robust set of life cycle processes that are applied repeatedly to the projects and services of the business. Therefore, this document is intended to be useful for adoption at either the organization level or at the project level. An organization can adopt the document and supplement it with appropriate procedures, practices, tools and policies. In turn, a project of the organization typically conforms to the organization's processes rather than conforming directly to this document.

In some cases, projects may be executed by an organization that does not have an appropriate set of processes adopted at the organizational level. Such a project may apply the provisions of this document directly to the project.

5.4 Life cycle concepts

5.4.1 Software life cycle stages

Life cycles vary according to the nature, purpose, use and prevailing circumstances of the software system. Using stages concurrently and in different orders can lead to life cycle forms with distinctly different characteristics. Each stage has a distinct purpose and contribution to planning and executing the whole life cycle of the software system. Per ISO/IEC TS 24748-1, the typical system life cycle stages include concept, development, production, utilization, support, and retirement. Use of these terms to define stages is not normative. A common set of stages for a software system is concept exploration, development, sustainment, and retirement, with transitions between stages for the system as a whole and for its elements.

The stages represent the major life cycle periods associated with a software system and they relate to the state of the software system description or the software system itself. The stages describe the major progress and achievement milestones of the software system through its life cycle. They give rise to the primary decision gates of the life cycle. These decision gates are used by organizations to understand and manage the inherent uncertainties and risks associated with costs, schedule and functionality when creating or utilizing a software system. Using stages thus provides organizations with a framework within which organization management has high-level visibility and control of project and technical processes. Organizations define and employ stages differently to satisfy contrasting business and risk mitigation strategies.

The life cycle processes defined in this document are not aligned to any specific stage in a software life cycle. All of the life cycle processes involve planning, performance, and evaluation activities that should be considered for use at every stage.

Further elaboration of these concepts can be found in ISO/IEC/IEEE 24748 (all parts), on the application of life cycle management.

5.4.2 Life cycle model for the software system

Every software system has a life cycle. A life cycle can be described using an abstract functional model that represents the conceptualization of a need for the system, its realization, utilization, evolution and disposal.

A software system progresses through its life cycle as the result of actions, performed and managed by people in organizations, using processes for execution of these actions. The detail in the life cycle model is expressed in terms of these processes, their outcomes, relationships and sequence.

This document does not prescribe any particular life cycle model. Instead it defines a set of processes, termed life cycle processes, which can be used in the definition of the system's life cycle. Also, this document does not prescribe any particular sequence of processes within the life cycle model. The sequence of the processes is determined by project objectives and by selection of the life cycle model. Often, the development stage is subdivided more finely and in different ways.

One oft-cited set of software development stages are elicitation, requirements, design, construction, and testing- the predictive or "waterfall" model. If the stages are considered as sequential, then each stage is required to produce correct results before proceeding to the next stage. In practice, this is extremely difficult to achieve unless the requirements are known well and the initial cost estimates are accurate. In performing a waterfall, one risks performing extensive rework that does not properly fall within any of the planned stages, hence probably does not fall within any budget.

NOTE 1 Winston Royce, commonly recognized as an early analyst of life cycle process models, described the need for rework stages rather than the "waterfall" (a term that he did not use). Unfortunately, the rework stages were dropped from the "waterfall" model as it was popularly understood.

To deal with the issues of incompletely known requirements and inaccurate estimates, a number of other types of models have been proposed: incremental, spiral, iterative, and evolutionary (adaptive). These life cycle models can incorporate agile techniques and methods. These models can typically involve repeated performance of the life cycle processes and stages during the life cycle, e.g., for different increments of the software product, for more precise handling of exceptions to common functions, or for requirements that were not fully defined at the outset. These models can be applied across stages, such as development and utilization or deployment. Use of these models can affect software release strategies and acquisition strategies for software services.

EXAMPLE Software elements can be developed incrementally, and then held for block operational release at a convenient time in the organization's business cycle.

The "incremental development" model includes initial planning, initial requirements analysis, initial architectural definition, and initial validation, but allocates design, implementation, verification (and sometimes delivery) activities to a series of stages, each of which provides a portion of the intended functionality. The approach provides for some flexibility to respond to inaccurate cost or schedule estimates by moving functionality to later increments.

The "spiral" variation on incremental developmental proposes ordering the development of functionality based on risk, with the riskiest problems considered in the early increments. This provides some protection against cost surprises occurring late in the development cycle.

The "iterative development" model performs initial planning and then consists of a cyclic process of prototyping, testing, analyzing and refining the requirements and the solution. "Iterative" models repeatedly perform the life cycle processes to deliver prioritized system functions sooner, with refined or more complex elements of the system coming in later iterations.

The "evolutionary model" is intended to deal with incomplete knowledge of requirements. It provides for initial planning and initial architecture definition, but allocates requirements analysis, design, construction, verification, validation and delivery to a series of stages. Delivered capabilities that do not meet user needs can be reworked in subsequent stages of the evolution.

"Agile" methods actually can be applied within a variety of models. While Agile methods are common in executing an evolutionary lifecycle model, they can be used in other lifecycle models at various stages. What the methods have in common is an emphasis on continuous inspection and collaboration in the rapid production of working software in an environment where changes, including changes to requirements, are expected. Annex H provides information on the application of this document in an agile context.

NOTE 2 Selecting the name of a type of model does not satisfy the requirement to define a model comprised of stages, with defined purpose and outcomes accomplished via the processes of this document.

NOTE 3 ISO/IEC TS 24748-1, ISO/IEC TR 24748-2, ISO/IEC TR 24748-3, and ISO/IEC/IEEE 24748-4 provide additional detail regarding life cycle models and stages. The models described in this clause apply not only to software systems but also to other systems as described in ISO/IEC/IEEE 15288:2015.

5.5 Process concepts

5.5.1 Criteria for processes

The determination of the life cycle processes in this document is based upon three basic principles:

1) Each life cycle process has strong relationships among its outcomes, activities and tasks.

2) The dependencies among the processes are reduced to the greatest feasible extent.

3) A process is capable of execution by a single organization in the life cycle.

5.5.2 Description of processes

Each process of this document is described in terms of the following attributes:

a) The title conveys the scope of the process as a whole.

b) The purpose describes the goals of performing the process.

c) The outcomes express the observable results expected from the successful performance of the process.

d) The activities are sets of cohesive tasks of a process.

e) The tasks are requirements, recommendations, or permissible actions intended to support the achievement of the outcomes.

The processes and process groups in this document are identical in their purpose and outcomes with those in ISO/IEC/IEEE 15288:2015, *System and software engineering – System life cycle processes*, with one exception: the System/Software Requirements Definition process of this document is renamed from the System Requirements Definition process of ISO/IEC/IEEE 15288:2015. To emphasize this harmonization of systems and software system processes, the process purposes and outcomes are presented in boxes in Clause 6.

Software-specific activities, tasks, and work products are applied to achieve the outcomes of the processes in this document. Annex E provides additional process views.

Additional detail regarding this form of process description can be found in ISO/IEC TR 24774.

5.5.3 General characteristics of processes

In addition to the basic attributes described in the previous subclause, processes may be characterized by other attributes common to all processes. ISO/IEC 33020:2015 identifies common process attributes that characterize six levels of achievement within a measurement framework for process capability. Annex C includes the list of process attributes that contribute to the achievement of higher levels of process capability as defined in ISO/IEC 33020:2015.

5.5.4 Tailoring

Annex A, which is normative, defines the basic activities needed to perform tailoring. Note that tailoring may diminish the perceived value of a claim of conformance to this document. This is because it is difficult for other organizations to understand the extent to which tailoring may have deleted desirable provisions. An organization asserting a single-party claim of conformance to this document may find it advantageous to claim full conformance to a smaller list of processes rather than tailored conformance to a larger list of processes.

5.6 Process groups

5.6.1 Introduction

This document groups the activities that can be performed during the life cycle of a software system into four process groups. Each of the life cycle processes within those groups is described in terms of its purpose and

desired outcomes with a set of related activities and tasks that can be performed to achieve those outcomes. The four process groups and the processes included in each group are depicted in Figure 4 as follows:

a) Agreement processes;

b) Organizational Project-Enabling Processes;

c) Technical Management Processes; and

d) Technical Processes.

The processes described in this document are not intended to preclude or discourage the use of additional processes that organizations find useful. The order of the subclauses in which the processes are defined in this document does not determine the order in which the processes are performed during the system life cycle or any of its stages. A description of each process group is provided in the four subclauses that follow.

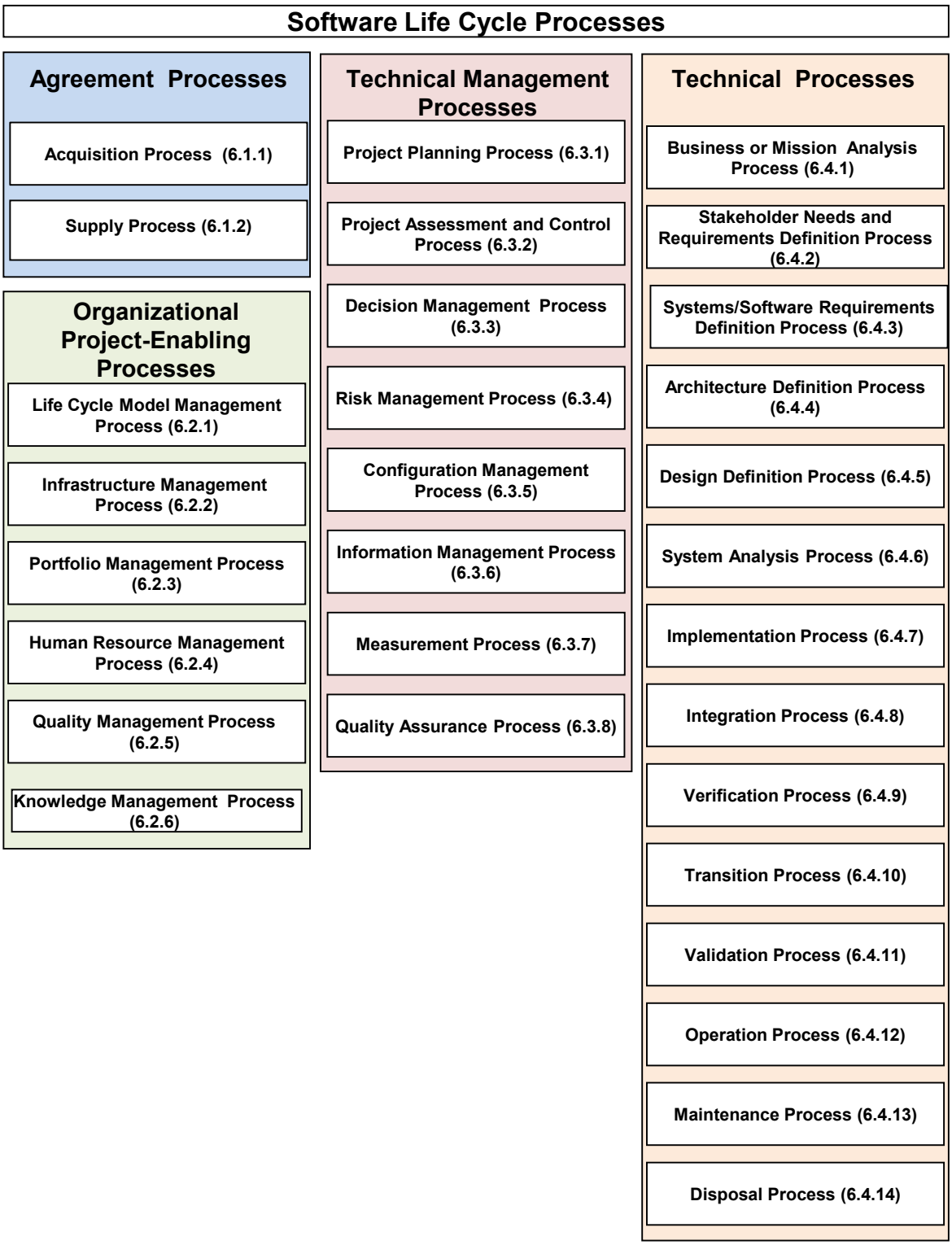

Figure 4 —Software life cycle processes

5.6.2 Agreement processes

Organizations are producers and users of software systems. One organization (acting as an acquirer) can task another (acting as a supplier) for products or services. This is achieved using agreements. Agreements allow both acquirers and suppliers to realize value and support business strategies for their organizations.

The Agreement processes are organizational processes that apply outside of the span of a project's life, as well as for a project's lifespan. Generally, organizations act simultaneously or successively as both acquirers and suppliers of software systems. The Agreement processes can be used with less formality when the acquirer and the supplier are in the same organization. Similarly, they can be used within the organization to agree on the respective

responsibilities of organization, project and technical functions. Figure 4 lists the processes contained in this process group.

5.6.3 Organizational project-enabling processes

The Organizational Project-Enabling processes are concerned with providing the resources to enable the project to meet the needs and expectations of the organization's stakeholders. The Organizational Project-Enabling processes are typically concerned at a strategic level with the management and improvement of the organization's business or undertaking, with the provision and deployment of resources and assets, and with its management of risks in competitive or uncertain situations. The Organizational Project-Enabling processes apply outside the span of a project's life, as well as during a project's lifespan.

The Organizational Project-Enabling processes establish the environment in which projects are conducted. The organization establishes the processes and life cycle models to be used by projects; establishes, redirects, or cancels projects; provides resources required, including human and financial; and sets and monitors the quality measures for software systems and other deliverables that are developed by projects for internal and external customers.

The Organizational Project-Enabling processes create a strong business image for many organizations and imply commercial and profit-making motives. Nevertheless, the Organizational Project-Enabling processes are equally relevant to non-profit organizations, since they are also accountable to stakeholders, are responsible for resources, and encounter risk in their undertakings. This document can be applied to non-profit organizations as well as to profit-making organizations. Figure 4 lists the processes contained in this process group.

5.6.4 Technical Management processes

The Technical Management processes are concerned with managing the resources and assets allocated by organization management and with applying them to fulfill the agreements into which the organization or organizations enter. The Technical Management processes relate to the technical effort of projects, in particular to planning in terms of cost, timescales and achievements, to the checking of actions to help ensure that they comply with plans and performance criteria and to the identification and selection of corrective actions that recover shortfalls in progress and achievement. These processes are used to establish and perform technical plans for the project, manage information across the technical team, assess technical progress against the plans for the software system, products, or services, control technical tasks through to completion, and aid in decision-making.

NOTE 1 Technical management is 'the application of technical and administrative resources to plan, organize and control engineering functions'. (ISO/IEC/IEEE 24765:2010)

Typically, several projects will co-exist in any one organization. The Technical Management processes can be employed at a corporate level to meet internal needs. Figure 4 lists the processes contained in this process group.

NOTE 2 Technical Management processes are applied during the performance of each Technical process.

5.6.5 Technical processes

The Technical processes are concerned with technical actions throughout the life cycle. Technical processes transform the needs of stakeholders into a product or service. By applying that product or operating that service, technical processes, provide sustainable performance, when and where needed in order to meet the stakeholder requirements and achieve customer satisfaction. The Technical processes are applied in order to create and use a software system, whether it is in the form of a model or is an operational product. The Technical processes apply at any level in a hierarchy of software system structure and at any stage in the life cycle. Figure 4 lists the processes contained in this process group.

5.7 Process application

The life cycle processes defined in this document can be used by any organization when acquiring, using, creating, or supplying a software system. They can be applied at any level in a system's hierarchy and at any stage in the life cycle.

The functions these processes perform are defined in terms of specific purposes, outcomes and the set of activities and tasks that constitute the process.

Each life cycle process in Figure 4 can be invoked, as required, at any time throughout the life cycle. The order that the processes are presented in this document does not imply any prescriptive order in their use. However, sequential relationships are introduced by the definition of a life cycle model. The detailed purpose and timing of use of these processes throughout the life cycle are influenced by multiple factors, including social, trading, organizational and technical considerations, each of which can vary during the life of a software system. An individual software life cycle is thus created through a selection and application of processes that will normally possess concurrent, iterative, recursive and time-dependent characteristics.

Concurrent use of processes can exist within a project (e.g., when design actions and preparatory actions for building a software system are performed at the same time), and between projects (e.g., when system elements are designed at the same time under different project responsibilities).

When the application of the same process or set of processes is repeated on the same system, the application is referred to as iterative. The iterative use of processes is important for the progressive refinement of process outputs, e.g., the interaction between successive verification actions and integration actions can incrementally build confidence in the conformance of the product. Iteration is not only appropriate but also expected. New information is created by the application of a process or set of processes. Typically, this information takes the form of questions with respect to requirements, analyzed risks or opportunities. Such questions should be resolved before completing the activities of a process or set of processes.

The recursive use of processes, i.e., the repeated application of the same process or set of processes applied to successive levels of system elements in a system's structure, is a key aspect of the application of this document. The outputs of processes at any level, whether information, artifacts or services, are inputs to the processes used at the level below (e.g., during top down design) or at the level above (e.g., during software system realization). The outcomes from one application are used as inputs to the next lower (or higher) system in the system structure to arrive at a more detailed or mature set of outcomes. Such an approach adds value to successive systems in the system structure.

The changing nature of the influences on the software system (e.g., operational environment changes, new opportunities for system element implementation, modified structure and responsibilities in organizations) requires continual review of the selection and timing of process use. Process use in the life cycle can be dynamic, responding to the many external influences on the software system. The life cycle approach also allows for incorporating the changes in the next stage. The life cycle stages assist the planning, execution and management of life cycle processes in the face of this complexity in life cycles by providing comprehensible and recognizable high-level purpose and structure. The set of processes within a life cycle stage are applied with the common goal of satisfying the exit criteria for that stage or the entry criteria of the formal progress reviews within that stage.

The discussion in this section on iterative and recursive use of software life cycle processes is not meant to imply any specific hierarchical, vertical, or horizontal structure for the system-of-interest, enabling system, organization, or project.

Where justified by product quality risks, detailed descriptions of process instances in the context of the specific product may also be created. Instantiation of processes involves identifying specific success criteria for a process instance, derived from the product requirements, and identifying the specific activities and tasks needed to achieve the success criteria, derived from the activities and tasks identified in this document. Creating detailed descriptions of process instances enables better management of product quality risks by establishing the link between the process and the specific product requirements.

Further elaboration of these concepts can be found in ISO/IEC/IEEE 24748 (all parts) on the application of life cycle processes.

5.8 Process reference model

Annex C defines a process reference model (PRM) at a level of abstraction higher than that of the detailed requirements contained in Clause 6. The PRM is applicable to an organization that is assessing its processes in order to determine the capability of these processes. The purpose and outcomes are a statement of the goals of the performance of each process. This statement of goals permits assessment of the effectiveness of the processes in ways other than simple conformity assessment.

NOTE In this document, the term "process reference model" is used with the same meaning as ISO/IEC 33001:2015: "model comprising definitions of processes in a domain of application described in terms of process purpose and outcomes, together with an architecture describing the relationships between the processes".

6 Software life cycle processes

6.1 Agreement processes

This subclause specifies the requirements for the establishment of agreements with organizational entities external and internal to the organization.

The Agreement processes consist of the following:

a) Acquisition process – used by organizations for acquiring products or services; and

b) Supply process – used by organizations for supplying products or services.

These processes define the activities necessary to establish an agreement between two organizations. If the Acquisition process is invoked, it provides the means for conducting business with a supplier. This may include products that are supplied for use as an operational software system, services in support of operational activities, software elements of a system, or elements of a software system being provided by a supplier. If the Supply process is invoked, it provides the means for an agreement in which the result is a product or service that is provided to the acquirer.

NOTE Security is an increasing concern in systems and software engineering. See ISO/IEC 27036, *Security techniques - Information security for supplier relationships*, for requirements and guidance for suppliers and acquirers on how to secure information in supplier relationships. Specific aspects of information security supplier relationships are addressed in ISO/IEC 27036-3:2013 and ISO/IEC 27036-4 (in development).

6.1.1 Acquisition process

6.1.1.1 Purpose

The purpose of the Acquisition process is to obtain a product or service in accordance with the acquirer's requirements.

NOTE As part of this process, the agreement is modified when a change request affecting the terms of the agreement is agreed to by both the acquirer and supplier.

6.1.1.2 Outcomes

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

a) A request for supply is prepared.

b) One or more suppliers are selected.

c) An agreement is established between the acquirer and supplier.

d) A product or service complying with the agreement is accepted.

e) Acquirer obligations defined in the agreement are satisfied.

6.1.1.3 Activities and tasks

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

NOTE 1 The activities and resulting agreement from this process often apply to suppliers in the supply chain, including subcontracted suppliers.

NOTE 2 IEEE Std 1062-2015, *IEEE Recommended Practice for Software Acquisition*, contains detailed activities for software acquisition alternatives, including custom-developed, off-the-shelf, and software as a service. IEEE Std 1062-2015 also provides software acquisition guidance for technical data rights and intellectual property considerations, for consideration

when safety assurance or information security requirements are of concern, and checklists of relevant issues for organizational consideration when establishing software acquisition processes.

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

1) Define a strategy for how the acquisition will be conducted.

NOTE 1 This strategy describes or references the life cycle model, risks and issues mitigation, a schedule of milestones, and selection criteria if the supplier is external to the acquiring organization. It also includes key drivers and characteristics of the acquisition, such as responsibilities and liabilities; specific models, methods, or processes; level of criticality; formality; and priority of relevant trade factors.

NOTE 2 The Decision Management process and System Analysis process are often used to support trade-offs for the acquisition strategy. Examples include: determining a make or buy decision, as well as the suitability of specific COTS or modified OTS solutions and supplier evaluation.

NOTE 3 If the strategy calls for acquisition of specific commercial-off-the-shelf software or open source software, acquisition can be limited to identifying the supplier, accepting or negotiating the conditions in a pre-defined license or lease or maintenance agreement, determining rights to intellectual property and data rights in the software system, and agreeing on the price.

NOTE 4 A factor to consider in agreements between suppliers is data rights and lateral access to constituent data and intellectual property. As an example, suppliers for one system component may need to collaborate with suppliers for another component and share source code. Agreements can enable this collaboration.

2) Prepare a request for the supply of a product or service that includes the requirements.

NOTE 1 If a supplier is external to the organization, then the request includes the business practices with which a supplier is expected to comply and the criteria for selecting a supplier.

NOTE 2 A definition of requirements is provided to one or more suppliers. The requirements are the stakeholder or the system/software requirements, depending on the type of acquisition approach, and using the associated requirements definition process.

NOTE 3 The acquirer develops the requirements by itself or retains a supplier to develop them. If the acquirer retains a supplier to develop requirements, the acquirer retains approval authority for the requirements developed by the supplier.

b) **Advertise the acquisition and select the supplier.** This activity consists of the following tasks:

1) Communicate the request for the supply of a product or service to potential suppliers; and

2) Select one or more suppliers.

NOTE To obtain competitive solicitations, proposals to supply are evaluated and compared against the selection criteria and ranked. The justification for rating each proposal is declared and suppliers commonly are informed why they were or were not selected.

c) **Establish and maintain an agreement.** This activity consists of the following tasks:

NOTE Project cost, schedule, and performance are monitored through the Project Assessment and Control process. Any identified issues that require agreement modifications are referred to this activity. Any proposals for changes to system elements or information are controlled through the Change Management activity of the Configuration Management process.

1) Develop an agreement with the supplier that includes acceptance criteria.

NOTE 1 This agreement ranges in formality from a written contract to a verbal agreement. Appropriate to the level of formality, the agreement establishes requirements, development and delivery milestones, verification, validation and acceptance conditions, exception-handling procedures, agreement change management procedures, and payment schedules, so that both parties of the agreement understand the basis for executing the agreement. Other provisions for agreements include rights and restrictions associated with technical data and intellectual property, acceptance test preparation and test environment details; and the extent of supplier involvement. The agreement identifies process requirements that need to be imposed on participating subcontractors, such as configuration management requirements, reporting of risks, and reporting of measures and measurement analysis.

NOTE 2 Acceptance criteria, such as acceptance testing, relate to how the product or service will satisfy its intended use in its operational environment. Acceptance testing can be performed using the Validation process. Exceptions that arise during the conduct of the agreement or with the delivered product or service are resolved according to the procedures established in the agreement.

2) Identify necessary changes to the agreement.

NOTE In requesting a change to the agreement, the acquirer or the supplier details its specifications, rationale, and background.

3) Evaluate impact of changes on the agreement.

NOTE Any change is investigated for impacts to project plans, schedule, cost, technical capability, and quality. A change can be handled within the existing agreement, can require a modification to the agreement, or can require a new agreement.

4) Negotiate the agreement with the supplier.

NOTE Agreement terms are negotiated between the acquirer and supplier. Negotiation occurs for the initial agreement, and as required for any changes. Changed agreements are based on the required change and identified impacts. Details are discussed and changed during negotiation, after which the acquirer and supplier accept the terms of an agreement and the agreement commences. For a written contract, this occurs when the contract is signed or as specified in the agreement.

5) Update the agreement with the supplier, as necessary.

NOTE 1 The result of the agreement modification is incorporated into the project plans and communicated to all affected parties.

NOTE 2 Agreements can specify the conditions under which the agreement will be terminated by either party, e.g., unexpected changes in strategy or available funding, or lack of satisfactory progress.

d) **Monitor the agreement.** This activity consists of the following tasks:

1) Assess the execution of the agreement.

NOTE 1 This includes confirmation that all parties are meeting their responsibilities according to the agreement. The Project Assessment and Control process is used to evaluate projected cost, schedule, performance, and the impact of undesirable outcomes on the organization. This information is combined with other assessments of the execution of the terms of the agreement. If execution of the agreement does not result in an acceptable product or service, the acquirer or supplier can terminate the agreement as allowed in its terms.

NOTE 2 Acceptance testing can be performed using the Validation process. Exceptions that arise during the conduct of the agreement or with the delivered product or service are resolved according to the procedures established in the agreement.

2) Provide data needed by the supplier and resolve issues in a timely manner.

e) **Accept the product or service.** This activity consists of the following tasks:

1) Confirm that the delivered product or service complies with the agreement.

NOTE If the agreed requirements are satisfied and the acceptance criteria are met, the supplier has fulfilled its obligation. Unaddressed exceptions, e.g., disputes over conduct of acceptance testing or product suitability for intended use, are a matter for agreement provisions relating to disputes, arbitration or applicable law and regulation.

2) Provide payment or other agreed consideration.

3) Accept the product or service from the supplier, or other party, as directed by the agreement.

4) Close the agreement.

NOTE The project is closed by the Portfolio Management process.

6.1.2 Supply process

6.1.2.1 Purpose

The purpose of the Supply process is to provide an acquirer with a product or service that meets agreed requirements.

NOTE As part of this process, the agreement is modified when a change request affecting the terms of the agreement is agreed to by both the acquirer and supplier.

6.1.2.2 Outcomes

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

a) An acquirer for a product or service is identified.

b) A response to the acquirer's request is produced.

c) An agreement is established between the acquirer and supplier.

d) A product or service is provided.

e) Supplier obligations defined in the agreement are satisfied.

f) Responsibility for the acquired product or service, as directed by the agreement, is transferred.

6.1.2.3 Activities and tasks

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

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

1) Determine the existence and identity of an acquirer who has a need for a product or service.

NOTE This is often generated through the Business or Mission Analysis process. For a product or service developed for consumers, an agent, e.g., a marketing function within the supplier organization, often represents the acquirer.

2) Define a supply strategy.

NOTE This strategy describes or references the life cycle model, risks and issues mitigation, and a schedule of milestones. It also includes key drivers and characteristics of the acquisition, such as responsibilities and liabilities; specific models, methods, or processes; level of criticality; formality; and priority of relevant trade factors.

b) **Respond to a request for supply of products or services**. This activity consists of the following tasks:

1) Evaluate a request for the supply of a product or service) to determine feasibility and how to respond.

2) Prepare a response that satisfies the solicitation.

c) **Establish and maintain an agreement.** This activity consists of the following tasks:

1) Negotiate an agreement with the acquirer that includes acceptance criteria.

NOTE This agreement ranges in formality from a written contract to a verbal agreement. The Supplier confirms that the requirements, delivery milestones, and acceptance conditions are achievable, that exception handling and agreement change management procedures and payment schedules are acceptable, and that they establish a basis for executing the agreement without unnecessary risks. Issues are discussed and resolved during negotiation, after which the acquirer and supplier accept the terms of an agreement and the agreement commences. For a contract, this occurs when the contract is signed.

2) Identify necessary changes to the agreement.

NOTE In requesting a change to the agreement, the acquirer or the supplier details its specifications, rationale, and background.

3) Evaluate impact of changes on the agreement.

NOTE Any change is investigated for impacts to project plans, schedule, cost, technical capability, or quality. A change can be handled within the existing agreement, can require a modification to the agreement, or can require a new agreement.

4) Negotiate the agreement with the acquirer, as necessary.

NOTE Changes to agreement terms are negotiated between the supplier and acquirer. This includes changes due to changing market context. Negotiation occurs for the initial agreement, and as required for any changes. Changed agreements are based on the required change and identified impacts.

5) Update the agreement with the acquirer, as necessary.

NOTE The result of the agreement modification is incorporated into the project plans and communicated to all affected parties.

d) **Execute the agreement.** This activity consists of the following tasks:

1) Execute the agreement according to the established project plans.

NOTE A supplier sometimes adopts, or agrees to use, acquirer processes.

2) Assess the execution of the agreement.

NOTE This includes confirmation that all parties are meeting their responsibilities according to the agreement. The Project Assessment and Control process is used to evaluate projected cost, schedule, performance, and the impact of undesirable outcomes on the organization. The change management activity of the Configuration Management process is used to control changes to the system elements. This information is combined with other assessments of the execution of the terms of the agreement. If execution of the agreement does not result in an acceptable product or service, the acquirer or supplier can terminate the agreement as allowed in its terms.

e) **Deliver and support the product or service.** This activity consists of the following tasks:

1) Deliver the product or service in accordance with the agreement criteria.

NOTE As stated in the agreement, acceptance criteria, such as acceptance testing, relate to how the product or service will satisfy its intended use in its operational environment. Unaddressed exceptions, e.g., disputes over conduct of acceptance testing or product suitability for intended use, are a matter for agreement provisions relating to disputes, arbitration or applicable law and regulation.

2) Provide assistance to the acquirer in support of the delivered product or service, per the agreement.

3) Accept and acknowledge payment or other agreed consideration.

4) Transfer the product or service to the acquirer, or other party, as directed by the agreement.

5) Close the agreement.

NOTE 1 The project is closed by the Portfolio Management process.

NOTE 2 Agreements can specify the conditions under which the agreement will be terminated by either party, e.g., unexpected changes in strategy or available funding, or lack of satisfactory progress.

6.2 Organizational Project-Enabling processes

The Organizational Project-Enabling processes help ensure the organization's capability to acquire and supply products or services through the initiation, support and control of projects. These processes provide resources and infrastructure necessary to support projects and help ensure the satisfaction of organizational objectives and

established agreements. They are not intended to be a comprehensive set of business processes that enable strategic management of the organization's business.

The Organizational Project-Enabling Processes consist of the following:

a) Life Cycle Model Management process;

b) Infrastructure Management process;

c) Portfolio Management process;

d) Human Resource Management process;

e) Quality Management process; and

f) Knowledge Management process.

6.2.1 Life cycle model management process

6.2.1.1 Purpose

The purpose of the Life Cycle Model Management process is to define, maintain, and assure availability of policies, life cycle processes, life cycle models, and procedures for use by the organization with respect to the scope of this document.

This process provides life cycle policies, processes, models, and procedures that are consistent with the organization's objectives, that are defined, adapted, improved and maintained to support individual project needs within the context of the organization, and that are capable of being applied using effective, proven methods and tools.

6.2.1.2 Outcomes

As a result of the successful implementation of the Life Cycle Model Management process:

a) Organizational policies and procedures for the management and deployment of life cycle models and processes are established.

b) Responsibility, accountability, and authority within life cycle policies, processes, models, and procedures are defined.

c) Life cycle models and processes for use by the organization are assessed.

d) Prioritized process, model, and procedure improvements are implemented.

6.2.1.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the Life Cycle Model Management process.

a) **Establish the process.** This activity consists of the following tasks:

NOTE The detail of the life cycle implementation within a project is dependent upon the complexity of the work, the methods used, and the skills and training of personnel involved in performing the work. A project tailors policies, processes, models, and procedures according to its requirements and needs, while maintaining alignment with regulations and organizational policies. Annex A contains information on tailoring.

1) Establish policies and procedures for process management and deployment that are consistent with organizational strategies.

2) Establish the processes that implement the requirements of this document and that are consistent with organizational strategies.

3) Define the roles, responsibilities, accountabilities, and authorities to facilitate implementation of processes and the strategic management of life cycles.

4) Define business criteria that control progression through the life cycle.

NOTE The decision-making criteria regarding entering and exiting each life cycle stage and key milestones are established. These are sometimes expressed in terms of business achievement.

5) Establish standard life cycle models for the organization that are comprised of stages and define the purpose and outcomes for each stage.

NOTE The life cycle model comprises one or more stage models, as needed. It is assembled as a sequence of stages that can overlap or iterate, as appropriate for the system-of-interest's scope, magnitude, complexity, changing needs and opportunities. Stages are illustrated in ISO/IEC TS 24748-1 using a commonly encountered example of life cycle stages. Specific examples for systems and software are provided in ISO/IEC TR 24748-2 and ISO/IEC TR 24748-3. The life cycle processes and activities are selected, tailored as appropriate and employed in a stage to fulfill the purpose and outcomes of that stage.

b) **Assess the process.** This activity consists of the following tasks:

NOTE ISO/IEC 33002:2015 provides a more detailed set of process assessment activities and tasks that can be aligned with the tasks shown below.

1) Monitor process execution across the organization.

NOTE This includes monitoring performance, analyzing process measures, and reviewing trends with respect to compliance with regulations, organizational policies, business criteria, and feedback from the projects regarding the effectiveness and efficiency of the processes.

2) Conduct periodic reviews of the life cycle models used by the projects.

NOTE This includes confirming the continuing suitability, adequacy and effectiveness of the life cycle models used by the projects and making improvements as appropriate. This includes the stages, processes and achievement criteria that control progression through the life cycle.

3) Identify improvement opportunities from assessment results.

NOTE Improvements can affect the stages, processes, and achievement criteria that control progression through the life cycle.

c) **Improve the process.** This activity consists of the following tasks:

1) Prioritize and plan improvement opportunities.

2) Implement improvement opportunities and inform relevant stakeholders.

NOTE Process Improvement includes improvements to any of the processes in the organization. Lessons learned are captured and available.

6.2.2 Infrastructure Management process

6.2.2.1 Purpose

The purpose of the Infrastructure Management process is to provide the infrastructure and services to projects to support organization and project objectives throughout the life cycle.

This process defines, provides and maintains the facilities, tools, and communications and information technology assets needed for the organization's business with respect to the scope of this document.

6.2.2.2 Outcomes

As a result of the successful implementation of the Infrastructure Management process:

a) The requirements for infrastructure are defined.

b) The infrastructure elements are identified and specified.

c) Infrastructure elements are developed or acquired.

d) The infrastructure is available.

6.2.2.3 Activities and tasks

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

a) **Establish the infrastructure.** This activity consists of the following tasks:

1) Define project infrastructure requirements.

NOTE 1 Infrastructure element examples include facilities, tools, hardware, software, services, and standards. In addition to the general infrastructure resources common to an organization to support its business processes, an organization can also provide projects with unique or shared enabling systems to support the project's technical processes.

NOTE 2 The infrastructure resource needs for the project are considered in context with other projects and resources within the organization, as well as within the policies and strategic plans of the organization. Business constraints and timelines that influence and control provision of infrastructure resources and services for the project are also evaluated. Project plans and future business needs contribute to the understanding of the resource infrastructure that is required. Physical factors (e.g., facilities), logistics needs, and human factors (including health and safety aspects) are also considered.

NOTE 3 ISO/IEC 27036, *Information security for supplier relationships*, provides guidance for addressing security of outsourced infrastructure.

2) Identify, obtain and provide infrastructure resources and services that are needed to implement and support projects.

NOTE An inventory asset registry is often established to track infrastructure elements and support reuse of infrastructure assets.

b) **Maintain the infrastructure.** This activity consists of the following tasks:

1) Evaluate the degree to which delivered infrastructure resources satisfy project needs.

2) Identify and provide improvements or changes to the infrastructure resources as the project requirements change.

6.2.3 Portfolio Management process

6.2.3.1 Purpose

The purpose of the Portfolio Management process is to initiate and sustain necessary, sufficient and suitable projects in order to meet the strategic objectives of the organization.

This process commits the investment of adequate organization funding and resources, and sanctions the authorities needed to establish selected projects. It performs continued assessment of projects to confirm they justify, or can be redirected to justify, continued investment.

For software systems, portfolio management also commonly refers to the management of a product line (portfolio of assets, products, and enabling systems, or service catalogue) to meet organizational or customer needs and objectives and support changes in technology. Management of assets is achieved through management of projects.

6.2.3.2 Outcomes

As a result of the successful implementation of the Portfolio Management process:

a) Business venture opportunities, investments or necessities are qualified and prioritized.

b) Projects are identified.

c) Resources and budgets for each project are allocated.

d) Project management responsibilities, accountability, and authorities are defined.

e) Projects meeting agreement and stakeholder requirements are sustained.

f) Projects not meeting agreement or satisfying stakeholder requirements are redirected or terminated.

g) Projects that have completed agreements and satisfied stakeholder requirements are closed.

6.2.3.3 Activities and tasks

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

a) **Define and authorize projects.** This activity consists of the following tasks:

1) Identify potential new or modified capabilities or missions.

NOTE The organization business strategy, concept of operations, or gap analysis or opportunity analysis is reviewed for current gaps, problems, or opportunities. A new capability or enterprise need is usually determined in the Business or Mission Analysis process, further defined in the Stakeholder Needs and Requirements Definition process, and managed through this process.

2) Prioritize, select and establish new business opportunities, ventures or undertakings.

NOTE These are usually consistent with the business strategy and action plans of the organization. The potential projects are prioritized, and thresholds established, to determine which projects will be executed. The characteristics of identified projects are often determined, including stakeholder value, risks and barriers to success, dependencies and inter-relationships, constraints, resource needs and mutual contention for resources. Each potential project is then assessed with respect to likelihood of success and cost-benefit. The Decision Management and System Analysis processes provide details on performing an analysis of alternatives.

3) Define projects, accountabilities and authorities.

4) Identify the expected goals, objectives, and outcomes of each project.

5) Identify and allocate resources for the achievement of project goals and objectives.

6) Identify multi-project interfaces and dependencies to be managed or supported by each project.

NOTE 1 This includes the use or reuse of enabling systems used by more than one project and the use or reuse of common system elements, including software elements, by more than one project.

NOTE 2 Understanding each project in the context of the enterprise architecture helps to ensure interfaces and constraints are identified.

7) Specify the project reporting requirements and review milestones that govern the execution of each project.

8) Authorize each project to commence execution of project plans.

NOTE Refer to the Project Planning process for additional information on developing project plans. Project plans are most useful when developed and approved early in the project life cycle.

b) **Evaluate the portfolio of projects.** This activity consists of the following tasks:

1) Evaluate projects to confirm ongoing viability.

NOTE Viability includes the following criteria:

i) The project is making progress towards achieving established goals and objectives.

ii) The project is complying with project directives.

iii) The project is being conducted according to approved project life cycle policies, processes, and procedures.

iv) The project remains viable, as indicated by, for example, continuing need for the service, practicable product implementation, and acceptable investment benefits.

2) Act to continue or redirect projects that are satisfactorily progressing or can be expected to progress satisfactorily by appropriate redirection.

c) **Terminate projects.** This activity consists of the following tasks:

1) Where agreements permit, act to cancel or suspend projects whose disadvantages or risks to the organization outweigh the benefits of continued investments.

NOTE Capture of lessons learned from canceled or failing projects can be especially useful for organizational improvement or use on other projects.

2) After completion of the agreement for products and services, act to close the projects.

NOTE Closure is accomplished in accordance with organizational policies and procedures, and the agreement.

6.2.4 Human Resource Management process

6.2.4.1 Purpose

The purpose of the Human Resource Management process is to provide the organization with necessary human resources and to maintain their competencies, consistent with business needs.

This process provides a supply of skilled and experienced personnel qualified to perform life cycle processes to achieve organization, project, and stakeholder objectives.

6.2.4.2 Outcomes

As a result of the successful implementation of the Human Resource Management process:

a) Skills required by projects are identified.

b) Necessary human resources are provided to projects.

c) Skills of personnel are developed, maintained or enhanced.

d) Conflicts in multi-project resource demands are resolved.

6.2.4.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the Human Resource Management process:

a) **Identify skills.** This activity consists of the following tasks:

1) Identify skill needs based on current and expected projects.

2) Identify and record skills of personnel.

b) **Develop skills.** This activity consists of the following tasks:

1) Establish skills development strategy.

NOTE This plan includes types and levels of training, categories of personnel, schedules, personnel resource requirements, and training needs.

2) Obtain or develop training, education or mentoring resources.

NOTE These resources include training materials that are developed by the organization or external parties, training courses that are available from external suppliers, computer based instruction.

3) Provide planned skill development.

4) Maintain records of skill development.

c) **Acquire and provide skills.** This activity consists of the following tasks:

NOTE This includes the recruitment and retention of personnel with experience levels and skills necessary to properly staff projects; staff assessment and review, e.g., their proficiency, motivation, ability to work in a team environment, as well as the need to be retrained, reassigned or reallocated.

1) Obtain qualified personnel when skill deficits are identified.

NOTE This includes using outsourced resources.

2) Maintain and manage the pool of skilled personnel necessary to staff ongoing projects.

3) Make project assignments based on project and staff-development needs.

4) Motivate personnel, e.g., through career development and reward mechanisms.

5) Control multi-project management interfaces to resolve personnel conflicts.

NOTE This includes conflicts of capacity in organizational infrastructure and supporting services and personnel resources among ongoing projects; or from project personnel being over-committed.

6.2.5 Quality Management process

6.2.5.1 Purpose

The purpose of the Quality Management process is to assure that products, services and implementations of the quality management process meet organizational and project quality objectives and achieve customer satisfaction.

6.2.5.2 Outcomes

As a result of the successful implementation of the Quality Management process:

a) Organizational quality management policies, objectives, and procedures are defined and implemented.

b) Quality evaluation criteria and methods are established.

c) Resources and information are provided to projects to support the operation and monitoring of project quality assurance activities.

d) Quality assurance evaluation results are gathered and analyzed.

e) Quality management policies and procedures are improved based upon project and organizational results.

6.2.5.3 Activities and tasks

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

NOTE Refer to ISO 9001:2015 for information and requirements to establish a Quality Management System.

a) **Plan quality management.** This activity consists of the following tasks:

1) Establish quality management policies, objectives, and procedures.

NOTE 1 ISO 9004:2009 contains guidelines for performance improvements.

NOTE 2 The policies, objectives, and procedures are based on the business strategy for customer satisfaction and risk management considerations.

2) Define responsibilities and authority for implementation of quality management.

NOTE Resources for quality management are often assigned from distinct organizations for independence from project management.

3) Define quality evaluation criteria and methods.

4) Provide resources and information for quality management.

b) **Evaluate quality management.** This activity consists of the following tasks:

1) Gather and analyze quality assurance evaluation results, in accordance with the defined criteria.

2) Assess customer satisfaction.

NOTE ISO 10004:2012 contains guidelines for monitoring and measuring customer satisfaction. The quality of the software system is also demonstrated by user satisfaction.

3) Conduct periodic reviews of project Quality Assurance activities for compliance with the Quality Management policies, objectives, and procedures.

NOTE Quality evaluation criteria and methods are established. Quality assessments address compliance with the project procedures and product conformance with quality characteristics.

4) Monitor the status of quality improvements on processes, products, and services.

c) **Perform corrective and preventive action.** This activity consists of the following tasks:

1) Plan corrective actions when quality management objectives are not achieved.

2) Plan preventive actions when there is a sufficient risk that quality management objectives will not be achieved.

3) Monitor corrective and preventive actions to completion and inform relevant stakeholders.

NOTE 1 Implementation of corrective and preventive action is performed in other relevant processes, such as Life Cycle Model Management or Project Assessment and Control.

NOTE 2 ISO 9001:2015, 0.3.3 and Annex A.4 describe preventive action to eliminate potential nonconformities as part of risk-based thinking.

6.2.6 Knowledge Management process

6.2.6.1 Purpose

The purpose of the Knowledge Management process is to create the capability and assets that enable the organization to exploit opportunities to re-apply existing knowledge.

This encompasses knowledge, skills, and knowledge assets, including system elements.

NOTE The re-application of existing knowledge is known as knowledge reuse and includes the reuse of knowledge about or from software elements.

6.2.6.2 Outcomes

As a result of the successful implementation of the Knowledge Management process:

a) A taxonomy for the application of knowledge assets is identified.

b) The organizational knowledge, skills, and knowledge assets are developed or acquired.

c) The organizational knowledge, skills, and knowledge assets are available.

d) Knowledge management usage data is gathered and analyzed.

6.2.6.3 Activities and tasks

The organization shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the Knowledge Management process:

a) **Plan knowledge management.** This activity consists of the following tasks:

1) Define the knowledge management strategy.

NOTE 1 This knowledge management strategy generally includes:

i) Identifying domains and their potential for the reapplication of knowledge.

ii) Plans for obtaining and maintaining knowledge, skills, and knowledge assets for their useful life.

iii) Characterization of the types of knowledge, skills, and knowledge assets to be collected and maintained.

iv) Criteria for accepting, qualifying, and retiring knowledge, skills, and knowledge assets.

v) Procedures for controlling changes to the knowledge, skills, and knowledge assets.

vi) Plans, mechanisms, and procedures for protection, control, and access to classified or sensitive data and information.

vii) Mechanisms for storage and retrieval.

NOTE 2 Knowledge management includes knowledge shared both internally within the organization and knowledge that is shared outside the organization with designated stakeholders, acquirers, and business partners, subject to intellectual property and non-disclosure agreements.

2) Identify the knowledge, skills, and knowledge assets to be managed.

3) Identify projects that can benefit from the application of the knowledge, skills, and knowledge assets.

b) **Share knowledge and skills throughout the organization.** This activity consists of the following tasks:

1) Establish and maintain a classification for capturing and sharing knowledge and skills across the organization.

NOTE This classification can include expert, common, and domain knowledge and skills, as well as lessons learned from other tasks.

2) Capture or acquire knowledge and skills.

3) Share knowledge and skills across the organization.

c) **Share knowledge assets throughout the organization.** This activity consists of the following tasks:

1) Establish a taxonomy to organize knowledge assets.

NOTE 1 The taxonomy includes the following:

i) Definition of the boundaries of domains and their relationships to others.

ii) Domain models capturing essential common and different features, capabilities, concepts, and functions.

iii) An architecture for a family of systems within the domain, including their common and different features.

NOTE 2 See ISO/IEC 26550 for more information on product line models. Refer to ISO/IEC/IEEE 42010:2011 for requirements on architecture frameworks, viewpoints, model kinds, views and models.

2) Develop or acquire knowledge assets.

NOTE Knowledge assets include system elements or their representations (e.g., reusable code libraries, reference architectures) architecture or design elements (e.g., architecture or design patterns), processes, criteria, or other technical information (e.g., training materials) related to domain knowledge, and lessons learned.

3) Share knowledge assets across the organization.

NOTE Automated search capabilities improve access to knowledge assets.

d) **Manage knowledge, skills, and knowledge assets.** This activity consists of the following tasks:

1) Maintain knowledge, skills, and knowledge assets.

2) Monitor and record the reuse of knowledge, skills, and knowledge assets.

3) Periodically reassess the currency of technology and market needs for the knowledge assets.

NOTE Assess the business benefits which the organization gained through the use of knowledge management practices.

6.3 Technical Management processes

The Technical Management processes are used to establish and evolve plans, to execute the plans, to assess actual achievement and progress against the plans, and to control execution through to fulfillment. Individual Technical Management processes may be invoked at any time in the life cycle and at any level in a hierarchy of projects, as required by plans or unforeseen events. The Technical Management processes are applied with a level of rigor and formality that depends on the risk and complexity of the project.

The scope of a technical management process is the technical management of a project or its products, to include the software product or system-of-interest.

NOTE This set of Technical Management processes is performed so that software system-specific technical processes can be conducted effectively. They do not comprise a management system or a comprehensive set of processes for project management, as that is not the scope of this document.

The Technical Management processes consist of the following:

a) Project Planning process;

b) Project Assessment and Control process;

c) Decision Management process;

d) Risk Management process;

e) Configuration Management process;

f) Information Management process;

g) Measurement process; and

h) Quality Assurance process.

Project Planning and Project Assessment and Control processes are key to all management practices. These processes establish the general approach for managing a project or a process. The other processes in this group provide a specific focused set of tasks for achieving a specialized management objective. They are all evident in the management of any undertaking, ranging from a complete organization down to a single life cycle process and its tasks. In this document, the project has been chosen as the context for describing processes. The same processes can also be applied in the performance of services.

6.3.1 Project Planning process

6.3.1.1 Purpose

The purpose of the Project Planning process is to produce and coordinate effective and workable plans.

This process determines the scope of the project management and technical activities, identifies process outputs, tasks and deliverables, establishes schedules for task conduct, including achievement criteria, and required resources to accomplish tasks. This is an ongoing process that continues throughout a project, with regular revisions to plans.

NOTE The strategies defined in each of the other processes provide inputs and are integrated in the Project Planning process. The Project Assessment and Control process is used to assess whether the plans are integrated, aligned, and feasible.

6.3.1.2 Outcomes

As a result of the successful implementation of the Project Planning process:

a) Objectives and plans are defined.

b) Roles, responsibilities, accountabilities, and authorities are defined.

c) Resources and services necessary to achieve the objectives are formally requested and committed.

d) Plans for the execution of the project are activated.

6.3.1.3 Activities and tasks

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

a) **Define the project.** This activity consists of the following tasks:

1) Identify the project objectives and constraints.

NOTE 1 Objectives and constraints include performance and other quality aspects, cost, time and customer and user satisfaction. Each objective is identified with a level of detail that permits selection, tailoring and implementation of the appropriate processes and activities.

NOTE 2 ISO/IEC 15026 *Systems and software assurance,* ISO/IEC 27001 *Information Security Management System* and ISO/IEC 27036, *Information Security for Supplier Relationships,* provide additional guidance on objectives and constraints related to assurance and security.

2) Define the project scope as established in the agreement.

NOTE This includes the relevant activities required to satisfy business decision criteria and complete the project successfully. A project can have responsibility for one or more stages in the complete software system life cycle. Project Planning includes defining appropriate actions for maintaining project plans, performing assessments and controlling the project.

3) Define and maintain a life cycle model that is comprised of stages using the defined life cycle models of the organization.

NOTE ISO/IEC TS 24748-1 provides detailed information regarding life cycle stages and the definition of an appropriate life cycle model. It defines a general set of exemplar system life cycle stages, including Concept, Development, Production, Utilization, Support and Retirement. It also identifies a generic exemplar set of software life cycle stages, including Needs determination, Concept exploration and definition, Demonstration and evaluation, Engineering/development, Production/manufacturing, Deployment/sales, Operations, Maintenance and support, and Retirement.

4) Establish a work breakdown structure (WBS) based on the deliverable products or the evolving architecture of the software system.

NOTE 1 Each element of the software system architecture, and appropriate processes and activities, are described with a level of detail that is consistent with identified risks. Related tasks in the work breakdown structure are grouped for performance. Project tasks identify work items being developed or produced. The Practice Management Standard for Work Breakdown Structures of the Project Management Institute (PMI) contains additional details on WBSs.

NOTE 2 For projects with agile or iterative methods, a WBS element can correspond to the primary features, from a user perspective, to be produced during iterations.

5) Define and maintain the processes that will be applied on the project.

NOTE 1 These processes are based on the defined processes of the organization (see Life Cycle Model Management process). Annex A contains information on tailoring that can be used to address project-specific needs. The definition of the processes includes the entry and exit criteria, inputs, process sequence constraints (predecessor/successor relationships), process concurrency requirements (what processes and tasks are to be worked concurrently with other process area tasks or activities), Measures of Effectiveness/Measures of Performance attributes, and scope and cost parameters (for critically important cost estimation).

NOTE 2 Identifying interfaces with other projects or organizational units is addressed through the Portfolio Management process.

b) **Plan project and technical management.** This activity consists of the following tasks:

1) Define and maintain a project schedule based on management and technical objectives and work estimates.

NOTE This includes definition of the duration, relationship, dependencies and sequence of activities, achievement milestones, resources employed and the reviews and schedule reserves for risk management necessary to achieve timely completion of the project.

2) Define achievement criteria for the life cycle stage decision gates, delivery dates and major dependencies on external inputs or outputs.

NOTE The time intervals between internal reviews are defined in accordance with organizational policy on issues such as business and system criticality, schedule and technical risks.

3) Define the costs and plan a budget.

NOTE Budgeted costs are based on the schedule, software size and complexity estimates, labor estimates, infrastructure costs, procurement items, acquired service and enabling system estimates, and budget reserves for risk management.

4) Define roles, responsibilities, accountabilities, and authorities.

NOTE This includes defining the project organization, staff acquisitions, and the development of staff skills. Authorities include, as appropriate, the legally responsible roles and individuals, e.g., design authorization, safety authorization, and those responsible for applicable certifications or accreditations.

5) Define the infrastructure and services required.

NOTE This includes defining the capacity needed, its availability and its allocation to project tasks. Infrastructure includes facilities, services, tools, communications, and information technology assets. The requirements for enabling systems and services for each life cycle stage are also specified.

6) Plan the acquisition of materials and enabling systems and services supplied from outside the project.

NOTE 1 This includes, as necessary, plans for solicitation, supplier selection, acceptance, contract administration and contract closure. The agreement processes are used for the planned acquisitions.

NOTE 2 ISO/IEC 27036, *Information security for supplier relationships*, provides guidance for acquisition of infrastructure and services.

7) Generate and communicate a plan for project and technical management and execution, including reviews.

NOTE 1 Technical planning for the software system is often captured in a Systems Engineering Management Plan (SEMP) or a Software Engineering Management Plan or a Software Development Plan (SDP). ISO/IEC/IEEE 24748-5 provides more detail on software engineering technical management planning and includes an annotated outline for an SDP. Planning for the project is often captured in a Project Management Plan. ISO/IEC/IEEE 16326 provides more detail on project planning.

NOTE 2 The strategy activities and tasks from each of the other processes provide inputs and are integrated in the Project Planning process. The Project Assessment and Control process is used to help ensure that the plans are integrated, aligned, and feasible.

c) **Activate the project.** This activity consists of the following tasks:

1) Obtain approval to start the project.

NOTE Approval to start (authorization to proceed) is provided through the Portfolio Management process.

2) Submit requests and obtain commitments for necessary resources to perform the project.

NOTE This includes access to enabling systems or services.

3) Implement project plans.

6.3.2 Project assessment and control process

6.3.2.1 Purpose

The purpose of the Project Assessment and Control process is to assess if the plans are aligned and feasible; determine the status of the project, technical and process performance; and direct execution to help ensure that the performance is according to plans and schedules, within projected budgets, to satisfy technical objectives.

This process evaluates, periodically and at major events, the progress and achievements against requirements, plans and overall business objectives. Information is provided for management action when significant variances are detected. This process also includes redirecting the project activities and tasks, as appropriate, to correct identified deviations and variations from other technical management or technical processes. Redirection may include re-planning as appropriate.

6.3.2.2 Outcomes

As a result of the successful implementation of the Project Assessment and Control process:

a) Performance measures or assessment results are available.

b) Adequacy of roles, responsibilities, accountabilities, and authorities is assessed.

c) Adequacy of resources is assessed.

d) Technical progress reviews are performed.

e) Deviations in project performance from plans are investigated and analyzed.

f) Affected stakeholders are informed of project status.

g) Corrective action is defined and directed, when project achievement is not meeting targets.

h) Project replanning is initiated, as necessary.

i) Project action to progress (or not) from one scheduled milestone or event to the next is authorized.

j) Project objectives are achieved.

6.3.2.3 Activities and tasks

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

a) **Plan for project assessment and control**. This activity consists of the following task:

1) Define the project assessment and control strategy.

NOTE The strategy identifies the expected Project Assessment and Control activities, including planned assessment methods and timeframes, and necessary management and technical reviews.

b) **Assess the project.** This activity consists of the following tasks:

1) Assess alignment of project objectives and plans with the project context.

2) Assess management and technical plans against objectives to determine adequacy and feasibility.

3) Assess project and technical status against appropriate plans to determine actual and projected cost, schedule, and performance variances.

4) Assess the adequacy of roles, responsibilities, accountabilities, and authorities.

NOTE This includes assessment of the adequacy of personnel competencies to perform project roles and accomplish project tasks. Objective measures are used wherever possible, e.g., efficiency of resource use, project achievement.

5) Assess the adequacy and availability of resources.

NOTE Resources include infrastructure, personnel, funding, time, or other pertinent items. This task includes evaluating the reuse of existing processes and infrastructure resources, and confirming that intra-organizational commitments are satisfied.

6) Assess progress using measured achievement and milestone completion.

NOTE This includes collecting and evaluating data for labor, material, service costs, and technical performance, as well as other technical data about objectives, such as affordability. These are compared against measures of achievement. This includes conducting effectiveness assessments to determine the adequacy of the evolving software system against requirements. It also includes the readiness of enabling systems to deliver their services when needed.

7) Conduct required management and technical reviews, audits and inspections.

NOTE These are formal or informal, and are conducted to determine readiness to proceed to the next stage of the life cycle or project milestone, to help ensure that project and technical objectives are being met, or to obtain feedback from stakeholders.

8) Monitor critical processes and new technologies.

NOTE This includes identifying and evaluating technology maturity and feasibility of technology insertion. Technology maturity is the readiness of a technology for operational use, and is often measured on a scale from low (exists as a concept only) to high (proven in operational use).

9) Analyze measurement results and make recommendations.

NOTE Measurement results are analyzed to identify deviations, variations or undesirable trends from planned values that include potential concerns, and to make appropriate recommendations for corrections or preventive actions. This includes, where appropriate, statistical analysis of measures that indicates trends, e.g., fault density to indicate quality of outputs, distribution of measured parameters that indicate process repeatability.

10) Record and provide status and findings from assessment tasks.

NOTE These are generally designated in the agreement, policies and procedures.

11) Monitor process execution within the project.

NOTE This includes the analysis of process measures and review of trends with respect to project objectives. Any improvement actions identified can be handled through the Quality Assurance process, the Quality Management process, or the Life Cycle Model Management process.

c) **Control the project.** This activity consists of the following tasks:

1) Initiate necessary actions needed to address identified issues.

NOTE 1 This task occurs when project or technical achievement is not meeting planned targets. This includes preventive, corrective, and problem resolution actions. Actions generally require replanning or reassignment of personnel, tools and infrastructure assets when inadequacy or unavailability has been detected, or when project or technical achievement exceeds targets or plan. They often impact the cost, schedule, or technical scope or definition. Actions sometimes require changes to the implementation and execution of the life cycle processes.

NOTE 2 Actions are recorded and reviewed to confirm their adequacy and timeliness.

2) Initiate necessary project replanning.

NOTE 1 Project replanning is initiated when project objectives or constraints have changed, or when planning assumptions are shown to be invalid.

NOTE 2 Any change that requires a change to the agreement between acquirer and supplier invokes the Acquisition and Supply processes.

3) Initiate change actions when there is a contractual change to cost, time or quality due to the impact of an acquirer or supplier request.

NOTE This includes consideration of modified terms and conditions for supply or initiating new supplier selection, which invokes the Acquisition and Supply processes.

4) Authorize the project to proceed toward the next milestone or event, if justified.

NOTE The Project Assessment and Control process is used to reach agreement on milestone completion.

6.3.3 Decision Management process

6.3.3.1 Purpose

The purpose of the Decision Management process is to provide a structured, analytical framework for objectively identifying, characterizing and evaluating a set of alternatives for a decision at any point in the life cycle and select the most beneficial course of action.

This process is used to resolve technical or project issues and respond to requests for decisions encountered during the software life cycle, in order to identify the alternative(s) that provides the preferred outcomes for the situation. The methods most frequently used for Decision Management are the trade study and engineering analysis. Each of the alternatives is assessed against the decision criteria (e.g., cost impact, schedule impact, programmatic constraints, regulatory implications, technical performance characteristics, critical quality characteristics, and risk). Results of these comparisons are ranked, via a suitable selection model, and are then used to decide on an optimal solution. Key study data (e.g., assumptions and decision rationale) are typically maintained to inform decision-makers and support future decision-making.

NOTE When it is necessary to perform a detailed assessment of a parameter for one of the criteria, the System Analysis process can be employed to perform the assessment.

6.3.3.2 Outcomes

As a result of the successful implementation of the Decision Management process:

a) Decisions requiring alternative analysis are identified.

b) Alternative courses of action are identified and evaluated.

c) A preferred course of action is selected.

d) The resolution, decision rationale and assumptions are identified.

6.3.3.3 Activities and tasks

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

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

1) Define a decision management strategy.

NOTE A decision management strategy includes the identification of roles, responsibilities, accountabilities, and authorities. The strategy considers the need for obtaining information input and for returning a timely decision. It includes the identification of decision categories and a prioritization scheme. Decisions often arise as a result of an effectiveness assessment, a technical trade-off, a problem needing to be solved, an action needed as a response to risk exceeding the acceptable threshold, or a new opportunity or approval for project progression to the next life cycle stage. Organization or project guidelines determine the degree of rigor and formality to apply to the decision analysis.

2) Identify the circumstances and need for a decision.

NOTE Problems or opportunities and the alternative courses of action that will resolve their outcome are recorded, categorized and reported.

3) Involve relevant stakeholders in the decision-making in order to draw on experience and knowledge.

NOTE It is good practice to identify the subject matter expertise needed for the analysis and the decision.

b) **Analyze the decision information.** This activity consists of the following tasks:

1) Select and declare the decision management strategy for each decision.

NOTE The degree of rigor required to resolve these problems or opportunities is determined, as well as the data and system analysis needed for evaluating the alternatives. Define the timeframe to reach a decision.

2) Determine desired outcomes and measurable selection criteria.

NOTE The desired value for quantifiable criteria and the threshold value(s) beyond which the attribute will be unsatisfactory are determined, as well as weighting factors for the criteria.

3) Identify the trade space and alternatives.

NOTE If a large number of alternatives exist, they are qualitatively screened to reduce alternatives to a manageable number for further detailed systems analysis. This screening is often based on qualitative assessments of such factors as risk, cost, schedule, and regulatory impacts.

4) Evaluate each alternative against the criteria.

NOTE The System Analysis process is used, as necessary, to quantify specific criteria for each trade alternative to be evaluated. This includes new design parameters, different architecture characteristics, and range of values for critical quality characteristics. The System Analysis process assesses the range of parameter variations in order to obtain a sensitivity analysis for each of the trade alternatives evaluated. These results are used to establish the feasibility of the various trade alternatives.

c) **Make and manage decisions.** This activity consists of the following tasks:

1) Determine preferred alternative for each decision.

NOTE Alternatives are evaluated quantitatively, using the selection criteria. The selected alternative generally provides an optimization of, or improvement in, an identified decision.

2) Record the resolution, decision rationale, and assumptions.

3) Record, track, evaluate and report decisions.

NOTE 1 This includes records of problems and opportunities and their disposition, as stipulated in agreements or organizational procedures and in a manner that permits auditing and learning from experience.

NOTE 2 This allows the organization to confirm that problems have been effectively resolved, that adverse trends have been reversed, and that advantage has been taken of opportunities.

6.3.4 Risk Management process

6.3.4.1 Purpose

The purpose of the Risk Management process is to identify, analyze, treat and monitor the risks continually.

The Risk Management process is a continual process for systematically addressing risk throughout the life cycle of a system product or service. It can be applied to risks related to the acquisition, development, maintenance or operation of a system.

NOTE Risk is defined in ISO Guide 73:2009 as “The effect of uncertainty on objectives”. This has an attached Note 1, “An effect is a deviation from the expected — positive and/or negative”. A positive risk is commonly known as an opportunity, and can be addressed within the Risk Management process.

6.3.4.2 Outcomes

As a result of the successful implementation of the Risk Management process:

a) Risks are identified.

b) Risks are analyzed.

c) Risk treatment options are identified, prioritized, and selected.

d) Appropriate treatment is implemented.

e) Risks are evaluated to assess changes in status and progress in treatment.

6.3.4.3 Activities and tasks

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

NOTE ISO/IEC/IEEE 16085 provides a more detailed set of risk management activities and tasks. This risk management process is aligned with ISO 31000:2009 *Risk management — Principles and Guidelines*, and ISO Guide 73:2009, *Risk management — Vocabulary.* ISO 9001:2015 describes planning for risks and opportunities in 6.1.

a) **Plan risk management.** This activity consists of the following tasks:

1) Define the risk management strategy.

NOTE This includes the risk management process of supply chain suppliers and describes how risks from suppliers will be raised to the next level(s) for incorporation in the project risk process.

2) Define and record the context of the Risk Management process.

NOTE 1 This includes a description of stakeholders' perspectives, risk categories, and a description (perhaps by reference) of the technical and managerial objectives, assumptions and constraints. The risk categories include the relevant technical areas of the software system and facilitate identification of risks across the product life cycle. As noted in ISO 31000, the aim of this step is to generate a comprehensive list of risks based on those events that might create, enhance, prevent, degrade, accelerate, or delay the achievement of objectives.

NOTE 2 Opportunities, which are one type of risk, provide potential benefits for the software system or project. Each of the opportunities pursued has associated risks that detract from the expected benefit. This includes the risks associated with not pursuing an opportunity, as well as the risk of not achieving the effects of the opportunity.

b) **Manage the risk profile.** This activity consists of the following tasks:

1) Define and record the risk thresholds and conditions under which a level of risk may be accepted.

2) Establish and maintain a risk profile.

NOTE The risk profile records: the risk management context; a record of each risk's state including its likelihood of occurrence, consequences, and risk thresholds; the priority of each risk based on risk criteria supplied by the stakeholders; and the risk action requests along with the status of their treatment. The risk profile is updated when there are changes in an individual risk's state. The priority in the risk profile is used to determine the application of resources for treatment.

3) Periodically provide the relevant risk profile to stakeholders based upon their needs.

c) **Analyze risks.** This activity consists of the following tasks:

1) Identify risks in the categories described in the risk management context.

NOTE Risks are commonly identified through various analyzes, such as safety, reliability, security, and performance analyzes; technology, architecture, and readiness assessments; and trade studies. These risks are often identified early in the life cycle and continue into the utilization, support, and retirement of the software system. Additionally, risks are often identified through the analysis of measurements of the evolving software system.

2) Estimate the likelihood of occurrence and consequences of each identified risk.

NOTE Consequences of a risk typically involve technical, schedule, cost, or quality impacts.

3) Evaluate each risk against its risk thresholds.

4) For each risk that does not meet its risk threshold, define and record recommended treatment strategies and measures.

NOTE Risk treatment strategies include, but are not limited to, eliminating the risk, reducing its likelihood of occurrence or severity of consequence, or accepting the risk. Treatments also include taking or increasing risk in order to pursue an opportunity. Measures provide information about the effectiveness of the treatment alternatives.

d) **Treat risks.** This activity consists of the following tasks:

1) Identify recommended alternatives for risk treatment.

2) Implement risk treatment alternatives for which the stakeholders determine that actions should be taken to make a risk acceptable.

3) When the stakeholders accept a risk that does not meet its threshold, consider it a high priority and monitor it continually to determine if future risk treatment actions are necessary or if its priority has changed.

4) Once a risk treatment is selected, coordinate management action.

NOTE The Project Assessment and Control process can be applied.

e) **Monitor risks.** This activity consists of the following tasks:

1) Continually monitor risks and the risk management context for changes and evaluate the risks when their state has changed.

2) Implement and monitor measures to evaluate the effectiveness of risk treatments.

3) Continually monitor for the emergence of new risks and sources throughout the life cycle.

6.3.5 Configuration Management process

6.3.5.1 Purpose

The purpose of Configuration Management is to manage and control system elements and configurations over the life cycle. Configuration Management (CM) also manages consistency between a product and its associated configuration definition.

Software configuration management (SCM) applies to both the software system and its interfaces. The purpose of interface management is to agree with interface partners on the exchange of data through communications among software systems and services. Annex E (see E.5) provides an example of an Interface Management Process View.

Software configurations are changed through the controlled release of a new version. The purpose of a release is to authorize and effect the availability of a software feature, function, or system for a specific purpose, with or without restrictions to a subset of users.

6.3.5.2 Outcomes

As a result of the successful implementation of the Configuration Management process:

a) Items requiring configuration management are identified and managed.

b) Configuration baselines are established.

c) Changes to items under configuration management are controlled.

d) Configuration status information is available.

e) Required configuration audits are completed.

f) System releases and deliveries are controlled and approved.

6.3.5.3 Activities and tasks

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

NOTE ISO/IEC/IEEE 19770 provides procedures and requirements for an IT asset management system.

a) **Plan configuration management.** This activity consists of the following tasks:

1) Define a configuration management strategy, including approaches for the following:

i) Governance of CM, including roles, responsibilities, accountabilities, and authorities, and use of configuration control (change control) boards; and

ii) Consideration of the level of risk and impact in approval of configuration baselines and regular and emergency change requests.

NOTE Regularly scheduled changes to apply software patches using approved procedures or check-in and check-out of unit-tested software elements under development are typically performed automatically, or reviewed and approved daily as a matter of routine. In comparison, significant changes to the software system design with major impact on project cost and schedule can involve extensive analyzes, consultations with suppliers, stakeholder reviews, and approvals at the highest levels of the organization.

iii) Coordination of CM across the set of acquirer, supplier, and supply chain organizations for the life of the software system, or the extent of the agreement or project, as appropriate.

iv) Control of access and changes to and disposition of configuration items.

v) The necessary baselines to be established, including criteria or events for commencing configuration control and maintaining baselines of evolving configurations.

vi) Control of software licenses, data rights, and other intellectual property assets.

vii) Frequency, priorities, and content of software versions and releases.

viii) The audit strategy and the responsibilities for validating continuous integrity and security of the configuration definition information.

ix) Change management, including preparing stakeholders and especially users for changes in operational software systems and services.

NOTE 1 For complex software systems, trade-off studies are performed, e.g., to select an appropriate automated tool to support SCM needs and scope as identified in the strategy.

NOTE 2 Additional guidance regarding configuration management activities can be found in ISO 10007, IEEE Std 828, and SAE ANSI/EIA-649-B.

NOTE 3 The SWEBOK, Guide to the Software Engineering Body of Knowledge, provides detailed discussion on SCM. This knowledge area addresses SCM in the context of a system, SCM project and process planning, SCM plan and outline, tool selection, subcontractor control, surveillance and other audits, software configuration items and relationships, software libraries, and Configuration Management process activities.

NOTE 4 The SCM strategy is commonly documented in a plan, e.g., a configuration management plan, or sometimes in a project's SEMP, SDP, or Project Management Plan (PMP). The strategy planning for Configuration Management is coordinated through the Project Planning process. In establishing points to establish baselines and conduct audits, CM planning is aligned with the software life cycle. The frequency of recurring SCM activities aligns with the iteration of technical processes and stages. SCM planning typically includes deciding when to review Configuration Management planning, what conditions require updating the CM plan, and who is authorized to change the CM plans and items held in configuration control.

2) Define the storage, archive and retrieval procedures for configuration items, CM artifacts, and records.

NOTE The locations and conditions of storage for software system configuration items, such as source code and executable software, are established in accordance with designated levels of integrity, security and safety.

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

1) Select the software system elements to be uniquely identified as configuration items subject to configuration control.

EXAMPLE Configuration items subject to configuration control in software systems usually include system/software requirements specifications, interface specifications; product and system elements (e.g., software objects, hardware, and services) while under development, baseline configurations or software versions established for transition between stages and as released for operational use; master ("gold") copies of source code or executable software for different platforms or versions; site-specific configurations in operational use; and information items, such as agreements, architecture models, service descriptions and operational procedures; and items in enabling systems.

NOTE 1 Unique identification can be applied to software components, versions, or to individually licensed copies. The identifiers are in accordance with relevant standards and product sector conventions, such that the items under configuration control are unambiguously traceable to their supplier and to their specifications or equivalent recorded descriptions. Information items are often identified and managed separately from other configuration items.

NOTE 2 Software configuration identifiers facilitate traceability when more than one developer or maintenance programmer is working on the same software function, so that various branches of code can be successfully reassembled and tested.

NOTE 3 The ISO/IEC 19770 standard (multiple parts) provides an IT Asset management system for tracking software licenses.

2) Identify the attributes of configuration items.

NOTE 1 Attributes refers to item status, or physical or logical features useful for managing or maintaining the software system. Appropriate attributes can differ for hardware and software configuration items.

NOTE 2 Configuration attributes and identifiers can reflect a decomposition of the software system, so that configuration items are tracked at the level at which change needs to be controlled.

EXAMPLE Software from external suppliers can be tracked by its license and maintenance agreement, which can involve tracking to the location, number or size of systems where it is used or the number of concurrent users allowed. Software versions can be traced to the stakeholder requirements which they implement.

3) Define baselines through the life cycle.

NOTE 1 Baselines capture the evolving configuration states of software system elements at designated times or under defined circumstances. The content for the baselines is developed through the technical processes, but is formalized at a point in time through the Configuration Management process. Baselines form the basis for subsequent changes. Selected baselines typically become formalized between acquirer and supplier, depending on the practices of the industry and the contractual involvement of the acquirer in the configuration management process. There are generally three major types of baselines at the system level: functional baseline, allocated baseline, and product baseline. These vary by domain or local strategy.

NOTE 2 Software systems development often entails establishing multiple developmental baselines to address evolving software configuration needs at key points in the life cycle, e.g., to allow for simultaneous control of software versions during software design, prototype, integration, and test releases. This can involve distributed configuration management responsibilities and access limitations for archives, e.g., software development or test libraries and a master configuration support library.

4) Obtain acquirer and supplier agreement to establish a baseline.

NOTE The Project Assessment and Control process is used to reach agreement. When software is being developed for commercial or internal use, the acquirer or project sponsor can be the authority to approve a baseline.

c) **Perform configuration change management** This activity consists of the following tasks:

NOTE Configuration change management establishes procedures and methods for managing change to a baseline once it is established. This is sometimes referred to as configuration control. The term 'change management' is also used for managing change to organizational procedures and business workflow.

1) Identify and record Requests for Change and Requests for Variance.

NOTE A request for variance is often referred to as a deviation, waiver, or concession.

2) Coordinate, evaluate, and disposition Requests for Change and Requests for Variance.

NOTE 1 Evaluation commonly includes analysis of rationale and need versus impact on the software and interoperating systems, considering risks and opportunities, quality, users, schedule, and cost. A decision is made on whether to implement or deny the change request.

NOTE 2 Requests for Change and Requests for Variance are often under the formal control of a Configuration Control Board (CCB).

3) Track and manage approved changes to the baseline, Requests for Change and Requests for Variance.

NOTE 1 This task involves prioritization, tracking, scheduling, and closing changes. Changes are then made through the Technical Processes. These changes are verified or validated through the Verification and Validation processes, to help ensure that the approved changes have been correctly applied.

NOTE 2 Changes and rationales are typically recorded when approved and when completed.

d) **Perform release control.** This activity consists of the following tasks:

1) Identify and record release requests, identifying the software system elements in a release.

NOTE 1 The life cycle model helps determine the frequency of iterative or incremental software releases. The Integration process is used to select and configure a release package, software version, update or patch. These changes are verified or validated through the Verification or Validation processes. Changes are made through the Technical processes, particularly Transition.

EXAMPLE Releases for a software test, for software or system qualification or other formal tests, or for trial (beta) or operational use.

2) Approve software system releases and deliveries.

NOTE 1 Releases often involve prioritization, tracking, scheduling, and closing changes. Approval of a release for operational use can include acceptance of the verified and validated changes. Criteria for approval of a release often includes rollback plans or contingency plans in the event of an unsuccessful release.

NOTE 2 For software systems, automated version control tools can help ensure that only the correct source code versions are accessed, updated, tested and documented for approved changes by appropriate personnel, and released.

3) Track and manage distribution of software system releases to specified environments or software deliveries.

NOTE Master copies or copies of incremental changes to released software versions can be maintained for the life of the system or project in a controlled environment. Software suppliers often track delivered copies of licensed software to the acquirer in order to provide agreed software maintenance. The software system release is stored and distributed in accordance with agreement and with the policies of the organizations involved.

e) **Perform configuration status accounting.** This activity consists of the following tasks:

1) Develop and maintain the CM status information for software system elements, baselines, and releases.

NOTE 1 Configuration status accounting provides the data on the status of controlled products needed to make decisions regarding system elements throughout the product life cycle. For example, software status can include past, current, and planned progression through stages in the life cycle for software functions and completion of verification and validation activities for software elements. Configuration status information permits forward and backward traceability to other configuration states. Configuration status records are maintained through the software or project life cycle and then archived according to agreements, relevant legislation or organizational practice.

NOTE 2 The recording, retrieval and consolidation of the current configuration status and the status of previous configurations to confirm information correctness, timeliness, integrity and security is managed. Audits are performed to verify conformance of a baseline to an architecture view, interface control documents, software license agreements, and other agreement requirements.

2) Capture, store and report configuration management data.

f) **Perform configuration evaluation.** This activity consists of the following tasks:

1) Identify the need for CM audits, and schedule the events.

2) Verify the product configuration meets the configuration requirements by comparing requirements, constraints, and waivers (variances) with results of formal verification activities, which can involve sampling methods.

3) Monitor the incorporation of approved configuration changes.

4) Assess whether the software system meets functional and performance capabilities identified for the baseline.

NOTE This is sometimes called a functional configuration audit (FCA), which assures that the product configuration meets specified requirements.

5) Assess whether the operational software system elements conform to the approved configuration information.

NOTE This is sometimes called a physical configuration audit (PCA). For software items, criteria for the PCA can include whether specified configuration items are installed on designated systems according to software license or agreement.

6) Record the CM audit results and disposition action items.

6.3.6 Information Management process

6.3.6.1 Purpose

The purpose of the Information Management process is to generate, obtain, confirm, transform, retain, retrieve, disseminate and dispose of information, to designated stakeholders.

Information management plans, executes, and controls the provision of information to designated stakeholders that is unambiguous, complete, verifiable, consistent, modifiable, traceable, and presentable. Information includes technical, project, organizational, agreement, and user information. Information is often derived from data records of the organization, system, process, or project.

NOTE Managed information has these quality characteristics: unambiguous, complete, verifiable, consistent, modifiable, traceable, and presentable.

6.3.6.2 Outcomes

As a result of the successful implementation of the Information Management process:

a) Information to be managed is identified.

b) Information representations are defined.

c) Information is obtained, developed, transformed, stored, validated, presented, and disposed of.

d) The status of information is identified.

e) Information is available to designated stakeholders.

6.3.6.3 Activities and tasks

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

NOTE ISO/IEC/IEEE 15289 summarizes requirements for the content of life cycle process information items (documentation) and provides guidance on their development.

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

1) Define the strategy for information management.

NOTE Information about the same topic can be developed in different ways at different points in the life cycle and for different audiences.

2) Define the items of information that will be managed.

NOTE This includes the information that will be managed during the software life cycle and possibly maintained for a defined period beyond. This is done according to organizational policy, agreements, or legislation.

3) Designate authorities and responsibilities for information management.

NOTE Due regard is paid to information and data legislation, security and privacy, e.g., ownership, agreement restrictions, rights of access to data and ownership of data, intellectual property and patents. Where restrictions or constraints apply, information is identified accordingly. Staff members with knowledge of such items of information are informed of their obligations and responsibilities.

4) Define the content, formats and structure of information items.

NOTE The information originates and terminates in many forms (e.g., audiovisual, textual, graphical, numerical) and mediums (e.g., electronic, printed, magnetic, optical). Organization constraints, e.g., infrastructure, inter-organizational communications, and distributed project workings, are taken into account. Relevant information item standards and conventions are used according to policy, agreements and legislation constraints.

5) Define information maintenance actions.

NOTE Information maintenance includes status reviews of stored information for integrity, validity and availability. It also includes any needs for replication or transformation to an alternative medium, as necessary, either to retain infrastructure as technology changes so that archived media can be read or to migrate archived media to newer technology.

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

1) Obtain, develop, or transform the identified items of information.

NOTE This includes collecting the data, information, or information items from appropriate sources (e.g., resulting from any life cycle process), and writing, illustrating, or transforming it into usable information for stakeholders. It includes reviewing, validating, and editing information per information standards.

2) Maintain information items and their storage records, and record the status of information.

NOTE 1 Information items are maintained according to their integrity, security and privacy requirements. The status of information items is maintained, (e.g., version description, date of issue or validity date, record of distribution, security classification). Legible information is stored and retained in such a way that it is readily retrievable.

NOTE 2 The source data and tools used to transform information, along with the resulting documentation is placed under configuration control in accordance with the Configuration Management process. ISO/IEC/IEEE 26531 provides requirements for content management systems useful for life cycle information and documentation.

3) Publish, distribute or provide access to information and information items to designated stakeholders.

NOTE Information is provided to designated stakeholders in an appropriate form, as required by agreed schedules or defined circumstances. Information items include documentation used for certification, accreditation, license or assessment ratings, as required.

4) Archive designated information.

NOTE Archiving is done in accordance with the audit, knowledge retention, and project closure purposes. The media, location and protection of the information are selected in accordance with the specified storage and retrieval periods, and with organization policy, agreements and legislation. Arrangements are put in place to retain necessary information items after project closure.

5) Dispose of unwanted, invalid or unvalidated information.

NOTE This is done according to organization policy, and security and privacy requirements.

6.3.7 Measurement process

6.3.7.1 Purpose

The purpose of the Measurement process is to collect, analyze, and report objective data and information to support effective management and demonstrate the quality of the products, services, and processes.

NOTE Measures have these quality characteristics: verifiable, meaningful, actionable, timely, and cost-effective.

6.3.7.2 Outcomes

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

a) Information needs are identified.

b) An appropriate set of measures, based on the information needs, is identified or developed.

c) Required data is collected, verified, and stored.

d) The data is analyzed and the results interpreted.

e) Information items provide objective information that supports decisions.

6.3.7.3 Activities and tasks

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

NOTE 1 ISO/IEC 15939 provides a more detailed set of measurement activities and tasks that are aligned with the activities and tasks in this document.

NOTE 2 ISO 9001:2015 specifies Quality Management System requirements for measurement and monitoring.

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

1) Define the measurement strategy.

2) Describe the characteristics of the organization that are relevant to measurement, such as business and technical objectives.

3) Identify and prioritize the information needs.

NOTE The information needs are based on the organization's business objectives, the project objectives, identified risks, and other items related to project decisions. Measurements can relate to projects, processes, products, or decisions.

4) Select and specify measures that satisfy the information needs.

NOTE Measures are defined that are verifiable and cost-effective.

5) Define data collection, analysis, access, and reporting procedures.

6) Define criteria for evaluating the information items and the Measurement process.

7) Identify and plan for the necessary enabling systems or services to be used.

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

1) Integrate manual or automated procedures for data generation, collection, analysis and reporting into the relevant processes.

NOTE This task can involve change impacts to other life cycle processes to accomplish procedural integration.

2) Collect, store, and verify data.

3) Analyze data and develop information items.

4) Record results and inform the measurement users.

NOTE The measurement analysis results are reported to relevant stakeholders in a timely, usable fashion to support decision making and assist in corrective actions, risk management, and improvements. Results are reported to decision process participants, technical and management review participants, and product and process improvement process owners.

6.3.8 Quality Assurance process

6.3.8.1 Purpose

The purpose of the Quality Assurance process is to help ensure the effective application of the organization's Quality Management process to the project.

Quality Assurance focuses on providing confidence that quality requirements will be fulfilled. Proactive analysis of the project life cycle processes and outputs is performed to assure that the product being produced will be of the desired quality and that organization and project policies and procedures are followed.

6.3.8.2 Outcomes

As a result of the successful implementation of the Quality Assurance process:

a) Project quality assurance procedures are defined and implemented.

b) Criteria and methods for quality assurance evaluations are defined.

c) Evaluations of the project's products, services, and processes are performed, consistent with quality management policies, procedures, and requirements.

d) Results of evaluations are provided to relevant stakeholders.

e) Incidents are resolved.

f) Prioritized problems are treated

NOTE Outcomes a) through d) align with the outcomes of the Quality Management process activities and tasks.

6.3.8.3 Activities and tasks

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

NOTE IEEE Std 730-2014, *Software Quality Assurance Processes*, provides additional detail.

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

1) Define a Quality Assurance strategy. The strategy is consistent with the organizational Quality Management policies and objectives and includes:

i) Priorities for applying Quality Assurance resources to processes and tasks that have the most significant impact on the quality of the delivered products and services;

ii) Defined roles, responsibilities, accountabilities, and authorities;

iii) Evaluation criteria and methods for processes, products, and services, including criteria for product or service acceptance;

iv) Activities appropriate to each supplier (including subcontractors);

v) Required verification, validation, monitoring, measurement, review, inspection, audit, and test activities specific to the products or services; and

vi) Problem resolution and process and product improvement activities.

NOTE In software projects, activities and tasks that have significant impact on product quality include obtaining agreement on new and changed requirements, performance of peer reviews and unit testing, analysis of problem reports and feedback from users; validating completion of corrective actions assigned at project milestone reviews, and root cause analysis of defects.

2) Establish independence of quality assurance from other life cycle processes.

NOTE Resources for quality assurance are often assigned from distinct organizations for independence from project management.

b) **Perform product or service evaluations.** This activity consists of the following tasks:

1) Evaluate products and services for conformance to established criteria, contracts, standards, and regulations.

NOTE This task includes verifying if criteria for product or service acceptance are reflected in verification and validation activities. Derived system/software quality requirements are usually associated with quality characteristics during requirements definition processes. ISO/IEC 25010 and ISO/IEC 25030 provide additional information on system/software quality characteristics.

2) Monitor that verification and validation of the outputs of the life cycle processes are performed to determine conformance to specified requirements.

c) **Perform process evaluations.** This activity consists of the following tasks:

1) Evaluate project life cycle processes for conformance.

2) Evaluate tools and environments that support or automate the process for conformance.

3) Evaluate supplier processes for conformance to process requirements.

NOTE Consider items such as a collaborative software development environment, process measures that suppliers are required to provide, or a risk process that suppliers are required to use. This includes surveillance reviews of process implementation through the supply chain.

d) **Manage QA records and reports.** This activity consists of the following tasks:

1) Create records and reports related to quality assurance activities.

NOTE Create records and reports in accordance with organizational, regulatory, and project requirements, using the information management process.

2) Maintain, store, and distribute records and reports.

3) Identify incidents and problems associated with product, service, and process evaluations.

NOTE This includes the capture of lessons learned. Responsibilities for resolution are identified.

e) **Treat incidents and problems.** This activity consists of the following tasks:

NOTE 1 In the terminology of quality management, problems are often described as "non-conformities" when, if left untreated, they can cause the project to fail to meet its requirements.

NOTE 2 For additional information and examples of problem categories and priority classifications, see ISO/IEC TS 24748-1:2016, Annex C.

1) Record, analyze and classify incidents.

2) Identify selected incidents to associate with known errors or problems.

3) Record, analyze and classify problems.

NOTE Analysis results include potential treatment options.

4) Identify root causes and treatment of problems where feasible.

5) Prioritize treatment of problems (problem resolution) and track corrective actions.

NOTE Implementation is done in the Technical processes after initiation by the Project Assessment and Control process. Organizational procedures for problem escalation can help focus resources on lagging problem resolutions.

6) Analyze trends in incidents and problems.

7) Identify improvements in processes and products that may prevent future incidents and problems.

NOTE The Risk Management process is used to treat risks and opportunities. The Life Cycle Model Management process is used to improve the organization's processes.

8) Inform designated stakeholders of the status of incidents and problems.

9) Track incidents and problems to closure.

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; and

n) Disposal process.

NOTE 1 For software systems, these processes can be recursively applied at more inclusive or more detailed levels for software system definition and realization.

NOTE 2 For software systems, these processes are often performed concurrently, iterating between one another to establish a solution that has satisfactory trade-offs with respect to requirements, critical performance measures, and critical quality characteristics. At any level of abstraction, requirements and models are made consistent via iterations of applicable technical processes. When requirements and models are not directly capable of being implemented, the technical processes are applied recursively at a more detailed level or through different system views.

NOTE 3 The concept of life cycle stages and the application of these processes in any stage are described in detail in ISO/IEC TS 24748-1. It has a complete set of example stages and stage outcomes for the enactment of technical processes within a software life cycle.

NOTE 4 Interface management is a set of activities that cut across software engineering processes. These cross-cutting activities of the Technical and Technical Management processes apply and track as a specific view of the processes and software system. See Annex E (E.5) for an example of an Interface Management process view.

NOTE 5 ISO/IEC 27002 *Code of practice for information security controls* and ISO/IEC 27034 *Application security*, provide guidance for applying security concerns in the Technical processes for software systems. See Annex E (E.6) for a sample Software Assurance Process View.

6.4.1 Business or Mission Analysis process

6.4.1.1 Purpose

The purpose of the Business or Mission Analysis process is to define the business or mission problem or opportunity, characterize the solution space, and determine potential solution class(es) that could address a problem or take advantage of an opportunity.

NOTE 1 Business and Mission Analysis is related to the organization encompassing stakeholders concerned by the activities of the software life cycle. This process interacts with the organization's strategy, which is generally outside the scope of ISO/IEC/IEEE 12207. The results of the organization's strategic analysis include the organizational Concept of Operations, strategic goals and plans, new market or mission elements, and identified problems and opportunities. The organization's strategy establishes the context within which the business or mission analysis is performed. The organizational Concept of Operations relates to the leadership's intended way of operating the organization. It describes the organization's assumptions and how it intends to use, acquire, or supply the system to be developed, existing systems, and possible future systems in support of an overall operation or series of operations of the business. In the case that the organization is the system-of-interest, the organization's strategy is part of the system definition.

NOTE 2 This process has application through the life of the software system solution and can be revisited if there are changes in the environment, needs, or other drivers.

NOTE 3 In some domains, Business or Mission Analysis relates to the concept of identifying and analyzing capabilities that are needed or desired by the organization. This process focuses on the necessary capabilities and interacts with the Portfolio Management process for identifying the trade space that can address the capability. The identified problems or opportunities

are often translated into target capabilities. As applicable within a given domain, the problem or opportunity space includes the target capabilities.

6.4.1.2 Outcomes

As a result of the successful implementation of the Business or Mission Analysis process:

a) The problem or opportunity space is defined.

b) The solution space is characterized.

c) Preliminary operational concepts and other concepts in the life cycle stages are defined.

d) Candidate alternative solution classes are identified and analyzed.

e) The preferred candidate alternative solution class(es) are selected.

f) Any enabling systems or services needed for business or mission analysis are available.

g) Traceability of business or mission problems and opportunities and the preferred alternative solution classes is established.

6.4.1.3 Activities and tasks

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

a) **Prepare for Business or Mission Analysis.** This activity consists of the following tasks:

1) Review identified problems and opportunities in the organization strategy with respect to desired organization goals or objectives.

NOTE This includes problems or opportunities with respect to the organization business or mission, vision, Concept of Operations, and other organization strategic goals and objectives. This includes identified deficiencies or gaps in existing capabilities, systems, products, or services.

2) Define the business or mission analysis strategy.

NOTE This includes the approach to be used to identify and define the problem space, characterize the solution space and select a solution class.

3) Identify and plan for the necessary enabling systems or services needed to support business or mission analysis.

NOTE This includes identification of requirements and interfaces for enabling systems. Enabling systems for business or mission analysis include the business systems and repositories of the organization or other accessible entities.

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

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

b) **Define the problem or opportunity space.** This activity consists of the following tasks:

1) Analyze customer complaints, problems and opportunities in the context of relevant trade-space factors.

NOTE 1 This analysis is focused on understanding the scope, basis, or drivers of the problems or opportunities, as opposed to the synthesis that is the focus of system analysis and decision management needed for trade studies. The focus here includes changes in mission requirements, business opportunities, capabilities, performance improvement, or lack of existing systems, security and safety improvement, factors such as cost and effectiveness, regulation changes, user dissatisfaction, and PESTEL factors (Political, Economic, Social, Technological, Environmental, and Legal). Relevant factors can be identified through external, internal, or SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis.

NOTE 2 The outputs of the analysis are considered as part of the portfolio management decisions.

2) Define the mission, business, or operational problem or opportunity.

NOTE This definition includes the context and any key parameters, without regard to a specific solution, since the solution can be an operational change, a change to an existing product or service, or a new system.

c) **Characterize the solution space**. This activity consists of the following tasks:

1) Define preliminary operational concepts and other concepts in life cycle stages.

NOTE 1 This involves the identification of major stakeholder groups, such as customers, users, administrations, regulators, and system owners, that are defined in the Stakeholder Needs and Requirements Definition process.

NOTE 2 Preliminary life cycle concepts include preliminary acquisition concepts, preliminary deployment concepts, preliminary operational concepts, preliminary support concepts, and preliminary retirement concepts. Operational concepts include high level operational modes or states, operational scenarios, potential use cases, or usage within a proposed business strategy. These concepts enable feasibility analysis and evaluation of alternatives. These concepts are further refined within the Stakeholder Needs and Requirements Definition process.

NOTE 3 The operating environment can have known vulnerabilities associated with specific security threats and safety hazards. These vulnerabilities need to be understood in association with the product under development. The system and human interfaces are an element of the system assurance context and related vulnerabilities are examined in the context of mission-critical threats.

2) Identify candidate alternative solution classes that span the potential solution space.

NOTE These classes can range from simple operational changes to various software system developments or modifications. This solution space can include the identification of existing assets, systems, and software products suitable for reuse, and changes in services that can address the need for operational or functional modifications. This includes deducing what potential expected services will be needed. The solution space characterization often invokes the Architecture Definition process for a user architecture viewpoint, resulting in architecture views (e.g., capability views, program views and operational views) as proposed by ISO/IEC/IEEE 42010.

d) **Evaluate alternative solution classes.** This activity consists of the following tasks:

1) Assess each alternative solution class.

NOTE 1 Each alternative solution class is assessed against defined criteria that are established based on the organization's strategy. Feasibility of the solution class is one key decision criteria. The Portfolio Management process provides some criteria to be considered.

NOTE 2 The System Analysis process is used to assess the value of each criterion for each alternative solution class. Structured affordability trade-offs are recommended. Including cost as a criterion will aid affordability decisions. The assessment of alternatives can include modeling, simulation, analytical techniques, or expert judgment to understand the risks, feasibility and value of the alternative candidate solution classes.

2) Select the preferred alternative solution class(es).

NOTE The Decision Management process is used to evaluate alternatives and to guide selection. Selected alternatives are validated in the context of the organization's strategy. Feedback on risks, feasibility, market factors, and alternatives is provided for use in updating the organization's strategy.

e) **Manage the business or mission analysis**. This activity consists of the following tasks:

1) Maintain traceability of business or mission analysis.

NOTE Through the life cycle, bidirectional traceability is maintained between the business and mission problems and opportunities and the preferred alternative solution classes with the organizational strategy, stakeholder needs and requirements, and system analysis results supporting decisions.

2) 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. This process identifies candidates for the baseline, and the Information Management process controls the information items.

6.4.2 Stakeholder Needs and Requirements Definition process

6.4.2.1 Purpose

The purpose of the Stakeholder Needs and Requirements Definition process is to define the stakeholder requirements for a system that can provide the capabilities needed by users and other stakeholders in a defined environment.

It identifies stakeholders, or stakeholder classes, involved with the system throughout its life cycle, and their needs. It analyzes and transforms these needs into a common set of stakeholder requirements that express the intended interaction the system will have with its operational environment and that are the reference against which each resulting operational capability is validated. The stakeholder requirements are defined considering the context of the system-of-interest with the interoperating systems and enabling systems.

NOTE The *SWEBOK, Guide to the Software Engineering Body of Knowledge*, Software Requirements knowledge area discusses software requirements fundamentals (e.g., definition, types, properties, quality characteristics) and other topics, such as stakeholders, requirements elicitation, analysis, and management that provide additional guidance for software systems.

6.4.2.2 Outcomes

As a result of the successful implementation of the Stakeholder Needs and Requirements Definition process:

a) Stakeholders of the system are identified.

b) Required characteristics and context of use of capabilities and concepts in the life cycle stages, including operational concepts, are defined.

c) Constraints on a system are identified.

d) Stakeholder needs are defined.

e) Stakeholder needs are prioritized and transformed into clearly defined stakeholder requirements.

f) Critical performance measures are defined.

g) Stakeholder agreement that their needs and expectations are reflected adequately in the requirements is achieved.

h) Any enabling systems or services needed for stakeholder needs and requirements are available.

i) Traceability of stakeholder requirements to stakeholders and their needs is established.

6.4.2.3 Activities and tasks

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

a) **Prepare for Stakeholder Needs and Requirements Definition.** This activity consists of the following tasks:

1) Identify the stakeholders who have an interest in the software system throughout its life cycle.

NOTE This includes individuals and classes of stakeholders who are users, operators, supporters, developers, producers, trainers, maintainers, disposers, acquirer and supplier organizations, parties responsible for external interfacing entities, regulatory bodies, and others who have a legitimate interest in the system. Where direct communication is not practicable (e.g., for consumer products and services), representatives or designated proxy stakeholders are selected.

2) Define the stakeholder needs and requirements definition strategy.

NOTE Some stakeholders have interests that oppose the acquirer's interests (e.g., market competitors, hackers, terrorists) or oppose each other. When the stakeholder interests oppose each other, but do not oppose the software system, this process is intended to gain consensus among the stakeholder classes to establish a common set of acceptable requirements. The intent or desires of those that oppose the acquirers, or detractors of the system, are addressed through the Risk Management process, threat analyzes of the System Analysis process, or the system/software requirements for security, adaptability, or resilience. In this case, the stakeholder needs are not satisfied, but rather addressed in a manner to help ensure system assurance and integrity if actions from the detractors are encountered.

3) Identify and plan for the necessary enabling systems or services needed to support stakeholder needs and requirements definition.

NOTE This includes identification of requirements and interfaces for the enabling systems. Enabling systems for stakeholder needs and requirements definition include tools for facilitation and requirements management.

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

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

b) **Define stakeholder needs.** This activity consists of the following tasks:

1) Define context of use within the concept of operations and the preliminary life cycle concepts.

NOTE Context of use is often captured using a Context of Use Description [ISO/IEC 25063]. Preliminary life cycle concepts are developed by the Business or Mission Analysis process.

2) Identify stakeholder needs.

NOTE 1 Identification of stakeholder needs includes elicitation of needs directly from the stakeholders, identification of implicit stakeholder needs based on domain knowledge and context understanding, and documented gaps from previous activities. Needs often include measures of effectiveness. Functional analysis is often used to aid the elicitation of needs. Also quality characteristics of the quality model in ISO/IEC 25010 and quality model application to requirements analysis in ISO/IEC 25030 are useful to elicit and identify quality requirements of non-functional requirements, which are often implicit stakeholder needs.

NOTE 2 The *SWEBOK, Guide to the Software Engineering Body of Knowledge*, Software Requirements knowledge area discusses some additional techniques for eliciting and clarifying software requirements, such as, prototyping, observation, user stories to determine required functionality, data mining, and analyzing competitors' products.

NOTE 3 Stakeholder needs describe the needs, wants, desires, expectations and perceived constraints of identified stakeholders. Understanding stakeholder needs for the minimum security and privacy requirements necessary for the operational environment minimizes the potential for disruption in plans, schedules, and performance. If significant issues are likely to arise relating to users and other stakeholders and their involvement in or interaction with a software system, recommendations for identifying and treating human-system issues can be found in ISO TS 18152.

3) Prioritize and down-select needs.

NOTE The Decision Management process is typically used to support prioritization. The System Analysis process is used to analyze needs for feasibility or other factors.

4) Define the stakeholder needs and rationale.

NOTE Needs concentrate on system purpose and behavior, and are described in the context of the operational environment and conditions. It is useful to trace needs to their sources and rationale.

c) **Develop the operational concept and other life cycle concepts.** This activity consists of the following tasks:

NOTE Other life cycle concepts can include acquisition concepts, deployment concepts, support concepts, security concepts, and retirement concepts. In this activity, the preliminary life cycle concepts defined within the Business or Mission Analysis process are further developed in the context of specific stakeholder needs, as associated scenarios and interactions are defined. See ISO/IEC/IEEE 29148:2011 Clauses 5 and 6 for more information on operational concepts, and ISO/IEC/IEEE 29148:2011 Annex A for an annotated outline of a System Operational Concept.

1) Define a representative set of scenarios to identify the required capabilities that correspond to anticipated operational and other life cycle concepts.

NOTE 1 Scenarios are used to analyze the operation of the system in its intended environment in order to identify additional needs or requirements that perhaps have not been explicitly identified by any of the stakeholders, e.g., legal, regulatory and social obligations The context of use of the system is identified and analyzed, including the activities that users perform to achieve system objectives, the relevant characteristics of the users (e.g., expected training and knowledge, frequency of system use, responsibilities, accessibility concerns), the physical environment (e.g., available light, temperature) and any equipment to be used (e.g., protective or communication equipment). The social and organizational influences on users that affect system use or constrain its design are analyzed when applicable. Scenarios centered on attackers, their environments, tools, techniques, and capabilities are key considerations for operational concept development. Scenarios are prioritized in order to reflect the weighted importance of the various operational needs.

NOTE 2 These scenarios often motivate updates to the operational or other life cycle concepts. Abuse and failure scenarios highlight the need for additional functional requirements (or more specific derived requirements) to mitigate risks that are identified in the abuse or failure scenarios.

2) Identify the factors affecting interactions between users and the system.

i) Anticipated physical, mental, and learned capabilities of the users;

ii) Workplace, environment and facilities, including other equipment in the context of use;

iii) Normal, unusual, and emergency conditions; and

iv) Operator and user recruitment, training and culture.

NOTE 1 Usability requirements take into account human capabilities and skills limitations. Where possible, applicable standards, e.g., ISO 9241, and accepted professional practices are used.

NOTE 2 If usability is important, usability requirements are planned, specified, and implemented through the life cycle processes. Refer to ISO TS 18152 for information on human-system issues and ISO/IEC 25060:2010 for information on usability.

d) **Transform stakeholder needs into stakeholder requirements.** This activity consists of the following tasks:

1) Identify the constraints on a system solution.

NOTE These constraints can result from 1) instances or areas of stakeholder-defined solution; 2) implementation decisions made at higher levels of system hierarchical structure; 3) required use of defined enabling, legacy, or interfacing systems, system elements, resources, and staff; or 4) stakeholder-defined affordability objectives. Include those that are unavoidable consequences of existing agreements, management decisions and technical decisions.

2) Identify the stakeholder requirements and functions that relate to critical quality characteristics, such as assurance, safety, security, environment, or health.

NOTE 1 See ISO/IEC/IEEE 15026 for additional information on system and software assurance.

NOTE 2 Identifying safety risks facilitates the identification of safety requirements and functions. Safety risks include those associated with methods of operations and support, health and safety, threats to property and environmental influences. Use applicable standards, and accepted professional practices. For example, IEC 61508:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems*, provides detailed requirements.

NOTE 3 Identifying security risks facilitates the identification of additional security requirements and functions. If warranted, include applicable areas of system security, such as physical, procedural, communications, computers, programs, data and emissions. This includes access and damage to protected personnel, properties and information, compromise of sensitive information, and denial of approved access to property and information. This also includes the required security functions, such as mitigation and containment, referencing applicable standards and accepted professional practices where mandatory or relevant. See Annex E (E.6) for a life cycle software assurance view.

NOTE 4 See ISO/IEC 25030 for further information regarding quality characteristics from a quality in use perspective.

3) Define stakeholder requirements, consistent with life cycle concepts, scenarios, interactions, constraints, and critical quality characteristics.

NOTE 1 See ISO/IEC/IEEE 29148:2011 Clauses 5 and 6 for more information on stakeholder requirements, and Clauses 8 and 9 for a description of and an annotated outline for a Stakeholder Requirements Specification.

NOTE 2 The stakeholder requirements are reviewed at key decision times in the life cycle to help ensure that account is taken of changes in needs.

NOTE 3 The stakeholder requirements are recorded in a form suitable for requirements management through the life cycle. These records establish the stakeholder requirements baseline, and retain changes of need and their origin throughout the software life cycle. These records are the basis for traceability to decisions made by the Business or Mission Analysis process as well as stakeholder needs, system requirements, and subsequent software system elements.

NOTE 4 The stakeholder requirements are the basis of the validation criteria for the software system and its elements.

e) **Analyze stakeholder requirements.** This activity consists of the following tasks:

1) Analyze the complete set of stakeholder requirements.

NOTE 1 Stakeholder requirements are analyzed for characteristics of individual requirements, as well as characteristics of the set of requirements. Potential analysis characteristics include that the requirements are necessary, implementation-free, unambiguous, consistent, complete, singular, feasible, traceable, verifiable, affordable, and bounded. ISO/IEC/IEEE 29148 provides additional information on characteristics of requirements.

NOTE 2 The System Analysis process is used to assess feasibility and affordability. The Verification and Validation processes are used in the review of stakeholder requirements.

2) Define critical performance measures that enable the assessment of technical achievement.

NOTE This includes defining technical and quality measures and critical performance parameters associated with each effectiveness measure identified in the stakeholder requirements. The critical performance measures (e.g., measures of effectiveness and measures of suitability) are defined, analyzed and reviewed to help ensure stakeholder requirements are met and to help ensure identification of project cost, schedule or performance risk associated with any non-compliance. ISO/IEC 15939 provides a process to identify, define and use appropriate measures. INCOSE TP-2003-020-01, *Technical Measurement,* provides information on the selection, definition and implementation of critical performance measures. The ISO/IEC 25000 series of standards provides relevant quality measures.

3) Feed back the analyzed requirements to applicable stakeholders to validate that their needs and expectations have been adequately captured and expressed.

4) Resolve stakeholder requirements issues.

NOTE This includes requirements that violate the characteristics for individual requirements or the set of requirements as defined in ISO/IEC/IEEE 29148.

f) **Manage the stakeholder needs and requirements definition.** This activity consists of the following tasks:

1) Obtain explicit agreement with designated stakeholders on the stakeholder requirements.

NOTE This includes confirming that stakeholder requirements are expressed correctly, comprehensible to originators, and that the resolution of conflict in the requirements has not corrupted or compromised stakeholder intentions.

2) Maintain traceability of stakeholder needs and requirements.

NOTE Through the life cycle, bidirectional traceability is maintained between the stakeholder needs and requirements, organizational strategy, and business and mission problems and opportunities. Stakeholder requirements are traced to the system/software requirements during the System/Software Requirements Definition process. Traceability is often maintained using an appropriate data repository.

3) 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. This process identifies candidates for the baseline, and the Information Management process controls the information items.

For this process, the stakeholder needs, stakeholder requirements, and operational concept are typical information items that are baselined.

6.4.3 System/Software requirements definition process

6.4.3.1 Purpose

The purpose of the System/Software Requirements Definition process is to transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user.

This process creates a set of measurable system requirements that specify, from the supplier's perspective, what characteristics, attributes, and functional and performance requirements the system is to possess, in order to satisfy stakeholder requirements. As far as constraints permit, the requirements should not imply any specific implementation.

NOTE 1 From a high-level view of the software system, this process can be used to define the overall requirements of the system. As the software system is decomposed into elements, each element, in turn, is treated as a system, function, or set of functions and this process can be used to further specify requirements. Requirements analyzes and tools support traceability of requirements between the software system and its elements.

NOTE 2 The *SWEBOK, Guide to the Software Engineering Body of Knowledge*, Software Requirements knowledge area discusses software requirements definition, analysis, modelling, specification, validation, management and other topics that provide additional guidance for software systems.

NOTE 3 The wording of the outcomes of the System/Software Requirements Definition process differs slightly from the outcomes in the System Requirements Definition process of ISO/IEC/IEEE 15288:2015. Use of the wording "system/software requirements" emphasize the applicability of this document to software systems, having software requirements and systems requirements. This is intended to assist users who define systems requirements and software requirements hierarchically or in different stages.

6.4.3.2 Outcomes

As a result of the successful implementation of the System/Software Requirements Definition process:

a) The system or element description, including interfaces, functions and boundaries, for a system solution is defined.

b) System/software requirements (functional, performance, process, non-functional, and interface) and design constraints are defined.

c) Critical performance measures are defined.

d) The system/software requirements are analyzed.

e) Any enabling systems or services needed for system/software requirements definition are available.

f) Traceability of system/software requirements to stakeholder requirements is developed.

6.4.3.3 Activities and tasks

The project shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the System/Software Requirements Definition process.

a) **Prepare for System/Software Requirements Definition.** This activity consists of the following tasks:

1) Define the functional boundary of the software system or element in terms of the behavior and properties provided.

NOTE The functional boundary definition is partly based on the context of use and operational scenarios defined in the frame of the Stakeholder Needs and Requirements Definition process. This includes the software system's stimuli (input)

and its responses to users and external systems, and an analysis and description of the required interactions between the software system and its operational environment in terms of interface properties and constraints, such as procedural flows, calling orders, data formats and flows, throughput, and timing. This establishes the expected software system behavior, expressed in quantitative terms, at its boundary. For software, boundaries are commonly expressed in Application Program Interfaces (API) and a graphical user interface (GUI) or interface files or services, including data formats. Annex E (E.5) provides an interface management view of the life cycle processes.

2) Define the system/software requirements definition strategy.

NOTE This includes the approach to be used to identify and define, and manage the system/software requirements with the selected life cycle model, e.g., evolutionary, incremental or iterative. Many factors can influence the strategy, e.g., complexity of the software system and information and functions to be managed; need for ready access and common understanding by multiple team members; degree of collaborative involvement by the acquirer or user representatives throughout the development stage; whether the project involves a new development, a modification, re-use or integration of existing systems; and process documentation requirements including period of retention. The life cycle model will influence when and how often the system/software requirements definition will be done. Annex H describes the progressive development of requirements in projects using agile methods.

3) Identify and plan for the necessary enabling systems or services needed to support system/software requirements definition.

NOTE This includes identification of requirements and interfaces for the enabling systems. Enabling systems for requirements definition include tools for facilitation and requirements management. Tools for software requirements management which are integrated with software development, test, and CM can simplify tracking and expedite software construction. Plans for enabling systems and description of modeling techniques used in support of the System/Software Requirements Definition process can be incorporated into an SDP.

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

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

b) **Define system/software requirements.** This activity consists of the following tasks:

1) Define each function that the software system or element is required to perform.

NOTE 1 Software functions can be described in use cases, user stories, or scenarios, and involve the transformation of data and information to achieve user needs (stakeholder requirements). In some cases, functions are derived from analysis of critical quality characteristics, such as performance, security or availability (e.g., system diagnosing function or highly frequent data backup function for reliability).

NOTE 2 Enabling functions that are required to support the system-of-interest in achieving its functionality are also identified and defined concurrently with the function of the system-of-interest. This is necessary to help ensure that the enabling functions in the system environment are identified and accounted for.

2) Identify required states or modes of operation of the software system.

NOTE 1 States or modes of operation can be modeled and represented in multiple modeling techniques and perspectives to give a sufficiently complete description of the desired system or element requirements.

NOTE 2 Conditions for performance of functions often involve interoperability across functions or elements. For example, some software requirements (e.g., a performance timing limit) can be allocated across multiple software system elements, affecting treatment of the requirement in a test case or regression test.

3) Define necessary implementation constraints.

NOTE For software elements, this includes the implementation decisions that are allocated from architecture definition at higher levels in the software system, introduced by stakeholder requirements, or solution limitations. Implementation constraints include the conditions under which the system is to be capable of performing the function, the conditions under which the system is to commence performing that function (input) and the conditions under which the system is to cease performing that function (output).

4) Identify requirements that relate to risks, criticality of the software system, or critical quality characteristics.

NOTE 1 Non-functional requirements and critical quality characteristics in software systems commonly include those related to health, safety, security and software assurance, reliability, availability and supportability (maintainability), and time constraints for throughput and performance. The System Analysis process can be used to determine appropriate values for performance requirements, considering the anticipated cost of achieving them and their impact on operation and use of the system.

NOTE 2 Analysis and definition of safety considerations include those requirements relating to methods of operation and maintenance, environmental influences, and risk of personnel injury. It also includes expressing each safety-related function and its associated integrity, in terms of the necessary risk reduction and allocation to designated safety-related systems. Applicable standards are used concerning functional safety, e.g., IEC 61508, and environmental protection, e.g., ISO 14001. Analysis includes security considerations, e.g., those related to compromise and protection of sensitive information, data and material. The security-related risks are defined, including administrative, personnel, physical, computer, communication, network, emission and environment factors using, as appropriate, applicable security standards. Refer to ISO/IEC/IEEE 15026-4 for system and software assurance guidance. ISO/IEC 27036 provides guidance for information security requirements for the outsourcing of products and services. ISO 25030 provides guidance for external system quality factors and characteristics. Annex E (E.6) provides a software assurance view of the life cycle processes.

NOTE 3 For software systems intended for human interaction, human-factors engineering (ergonomics) specifications are considered. For systems that have usability requirements, recommendations for obtaining a desired level of usability can be found in ISO/FDIS 9241-220, *Ergonomics of human-system interaction — Part 220: Processes for enabling, executing and assessing human-centred design within organizations*.

5) Define system/software requirements and requirements attributes, including the following:

   i) Data elements, data structures and formats, and database or data retention requirements;

   ii) User interfaces and user documentation (information for users) and user training;

   iii) Interfaces with other systems and services;

   iv) Functions and non-functional characteristics, including critical quality characteristics and cost targets;

   v) Transition of operational processes and data from existing automated and manual systems, migration approach and schedule, software installation and acceptance of the product; and

   vi) Requirement attributes, such as rationale; priority; traceability to software system elements, test cases, and information items; methods of verification; inclusion in approved baselines; and evaluated risk.

NOTE 1 Requirements definition involves iterative and recursive steps in parallel with other life cycle processes. Depending on the life cycle model that is being employed, it is useful to compare the resources to be spent in assuring initial correctness of requirements versus the resource needed to evolve requirements based on verification and validation results.

NOTE 2 The system/software requirements and attributes are recorded with a level of detail and in a form suitable for requirements management through the life cycle. See ISO/IEC/IEEE 29148:2011 Clauses 5 and 6 for more information on requirements, and Clauses 8 and 9 for a description of and an annotated outline for a System Requirements Specification and a Software Requirements Specification.

c) **Analyze system/software requirements.** This activity consists of the following tasks:

1) Analyze the complete set of system/software requirements.

NOTE 1 Requirements are analyzed for characteristics of individual requirements, as well as characteristics of the set of requirements. Potential analysis characteristics include that the requirements are necessary, implementation-free, unambiguous, consistent, complete, singular, feasible, traceable, verifiable, affordable, and bounded. The Verification process is used to determine if requirements meet the attributes and characteristics of good requirements. In some cases, the technical and economic feasibility of validating and verifying alternative formulations of requirements is evaluated. ISO/IEC/IEEE 29148 provides additional information on characteristics of requirements.

NOTE 2 The System Analysis process can be used to assess feasibility, affordability, balance and other requirements characteristics. The System Analysis process is used to determine appropriate values for requirement parameters, considering the estimated cost, schedule, and technical performance of the software system.

NOTE 3 Anticipating that some requirements can be achieved incrementally or even deferred or waived, requirements can be prioritized.

2) Define critical performance measures that enable the assessment of technical achievement.

NOTE This includes defining technical and quality measures and critical performance parameters associated with each effectiveness measure identified in the software system element requirements. The critical performance measures (e.g., measures of performance and technical performance measures) are analyzed and reviewed to help ensure system/software requirements are met and to help ensure identification of project cost, schedule or performance risk associated with any non-compliance. ISO/IEC 15939 provides a process to identify, define and use appropriate measures. INCOSE TP-2003-020-01, *Technical Measurement*, provides information on the selection, definition and implementation of critical performance measures. The ISO/IEC 25000 series of standards provides relevant quality measures.

3) Feed back the analyzed requirements to applicable stakeholders for review.

NOTE Feedback helps validate that the specified requirements have been adequately captured and expressed. Confirmation is made that they are a necessary and sufficient response to stakeholder requirements and a necessary and sufficient input to other processes, in particular software architecture, design, and verification. The Validation process is used to determine if the system/software requirements address the users' needs.

4) Identify and resolve issues, deficiencies, conflicts, and weaknesses within the complete set of requirements.

NOTE This includes requirements that are not verifiable, ambiguous, violate the characteristics for individual requirements, or are inconsistent with others in the set of requirements. Resolution of issues with requirements can be iterative within certain life cycle models.

d) **Manage system/software requirements.** This activity consists of the following tasks:

NOTE Maintaining system/software requirements includes defining, recording, and controlling the baseline, typically under formal configuration management, along with managing changes resulting from the application of other life cycle processes such as architecture or design.

1) Obtain explicit agreement on the system/software requirements.

NOTE This includes confirming that system/software requirements are expressed correctly, comprehensible to originators and implementers, and that the resolution of conflict in the requirements is consistent with stakeholder decisions.

2) Maintain traceability of the system/software requirements.

NOTE Through the life cycle, bidirectional traceability is maintained between the system/software requirements and the stakeholder requirements, architectural entities, interface definitions, analysis results, verification methods or techniques, and allocated, decomposed, and derived requirements. Traceability allows verification that achievable stakeholder requirements are met by one or more system or element requirements, and such requirements meet or contribute to meeting at least one stakeholder requirement. Traceability is often facilitated by an appropriate data repository or integrated development and test infrastructure.

3) 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. This process identifies candidates for the baseline, and the Information Management process controls the information items, such as requirements specifications. For this process, the system/software requirements are typical artifacts that are baselined.

6.4.4 Architecture Definition process

6.4.4.1 Purpose

The purpose of the Architecture Definition process is to generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views.

Iteration of the Architecture Definition process with the Business or Mission Analysis process, System/Software Requirements Definition process, Design Definition process, and Stakeholder Needs and Requirements Definition process is often employed so that there is a negotiated understanding of the problem to be solved and a satisfactory solution is identified. The results of the Architecture Definition process are widely used across the life cycle processes. Architecture definition may be applied at many levels of abstraction, highlighting the relevant detail that is necessary for the decisions at that level.

NOTE 1 System architecture deals with fundamental principles, concepts, properties, and characteristics and their incorporation into the system-of-interest. Architecture definition has more uses than as merely a driver or part of design. Refer to ISO/IEC/IEEE 42010:2011 for more information about architecture description and the uses and nature of architecture.

NOTE 2 The Architecture Definition process supports identification of stakeholders and their concerns. As the process unfolds, insights are gained into the relation between the requirements specified for the software system and the emergent properties and behaviors of the system that arise from the interactions and relations between the system elements. An effective architecture is as design-agnostic as possible to allow for maximum flexibility in the design trade space. Even for a single-product software system, the design of the product will likely change over time while the architecture remains constant. An effective architecture also highlights and supports trade-offs for the Design Definition process and possibly other processes, such as Portfolio Management, Project Planning, System/Software Requirements Definition, and Verification.

NOTE 3 Architecture Definition can apply to a product line rather than a single software system. A product line architecture describes the structural properties for building a group of related systems with common components and interrelationships. In product line architectures, the architecture necessarily spans several designs. The architecture serves to make the product line cohesive and helps ensure compatibility and interoperability across the product line. ISO/IEC 26550:2013 describes establishing a domain architecture for a product line.

NOTE 4 The *SWEBOK (Guide to the Software Engineering Body of Knowledge*) Software Requirements, Software Design and Software Engineering Models and Methods knowledge areas discuss key aspects of software architecture in relationship to the system, as well as with respect to iteration with design.

6.4.4.2 Outcomes

As a result of the successful implementation of the Architecture Definition process:

a) Identified stakeholder concerns are addressed by the architecture.

b) Architecture viewpoints are developed.

c) Context, boundaries, and external interfaces of the system are defined.

d) Architecture views and models of the system are developed.

e) Concepts, properties, characteristics, behaviors, functions, or constraints that are significant to architecture decisions of the system are allocated to architectural entities.

f) System elements and their interfaces are identified.

g) Architecture candidates are assessed.

h) An architectural basis for processes throughout the life cycle is achieved.

i) Alignment of the architecture with requirements and design characteristics is achieved.

j) Any enabling systems or services needed for architecture definition are available.

k) Traceability of architecture elements to stakeholder and system/software requirements is developed.

6.4.4.3 Activities and tasks

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

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

1) Review pertinent information and identify key drivers of the architecture.

NOTE 1 Key drivers are identified by reviewing: (a) market studies, industry projections, competitor product plans, and scientific findings; (b) organizational strategies, organizational-level concept of operations, organizational policies and directives, regulatory and legal constraints, and stakeholder requirements; (c) mission or business concept of operations, system-of-interest and related system operational concept, operational environment, technology roadmaps, and system/software requirements, and (d) other factors that impact the suitability of the software system through its life cycle. This analysis of key drivers typically builds from the Business or Mission Analysis, Stakeholder Requirements Definition, and System/software Requirements Definition processes.

NOTE 2 Key drivers of the architecture can include architecture styles and patterns, elements, principles such as replaceable components, feasibility of implementation and integration; availability of COTS and open source components; data sources for data-intensive systems; and performance implications. The effect of choosing various design elements can be lessened if the software system is properly architected.

2) Identify stakeholder concerns.

NOTE 1 Stakeholders are initially identified in the Stakeholder Needs and Requirements process. Additional stakeholders are usually identified during the Architecture Definition process. Stakeholder concerns related to architecture include system integrity concerns that the software system will be compromised intentionally or unintentionally via a threat agent or cause accidents as a safety hazard. Stakeholder. expectations or constraints are often associated with the system's life cycle stages, such as utilization (e.g., availability, security, effectiveness, usability, interoperability with existing systems, availability or risks to data in the system), support (e.g., the supportability of the system over its projected life-span, obsolescence management), evolution of the software system and its environment (e.g., adaptability, scalability, survivability), production (e.g., distribution, testability), and retirement (e.g., sensitive data eradication or retention).

NOTE 2 Concerns affecting software system architecture include data sources and performance implications for data-intensive systems, and constraints on the use of outsourced, existing, newly developed, proprietary, commercially available, or open source software elements, including software licensing. While software architecture is ideally design-agnostic, the feasibility of implementing the architecture in an affordable software system is a significant constraint for most systems.

3) Define the Architecture Definition roadmap, approach, and strategy.

NOTE This includes the identification of opportunities to communicate with designated stakeholders, the definition of architecture review activities, evaluation approach and criteria, measurement approach, and measurement methods (refer to the Measurement process). The roadmap shows how the architecture will evolve to an envisioned end state and often has a longer timeframe than for the current system-of-interest. The approach is the manner in which the work will be accomplished, such as how to engage with stakeholders, how to vet the results, or where to do the work. The strategy deals with the systematic plan of action for implementing the approach consistent with the roadmap.

4) Define architecture evaluation criteria based on stakeholder concerns and key requirements.

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

NOTE This includes identification of requirements and interfaces for the enabling systems and services. Enabling systems for architecture definition can include tools for collaboration and architecture development, and architecture reuse repositories for artifacts such as architecture patterns, models, and reference architectures.

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 enabling system achieves its intended use for its enabling functions. The Infrastructure Management process supports reuse of enabling systems.

b) **Develop architecture viewpoints.** This activity consists of the following tasks:

1) Select, adapt, or develop viewpoints and model kinds based on stakeholder concerns.

2) Establish or identify potential architecture framework(s) to be used in developing models and views.

NOTE Some architecture frameworks identify stakeholders and their concerns, and relevant viewpoints that address those concerns, while other architecture frameworks are more general in their guidance. Viewpoints specify the kinds of models to be used and how the resulting models can be used to generate architecture views. Refer to ISO/IEC/IEEE 42010 for more information on architecture framework and architecture description practices.

3) Capture rationale for selection of framework(s), viewpoints and model kinds.

4) Select or develop supporting modelling techniques and tools.

NOTE Both the SWEBOK and ISO/IEC TR 24748-3 describe modeling techniques that support Architecture Definition and Design Definition of software elements.

c) **Develop models and views of candidate architectures.** This activity consists of the following tasks:

1) Define the software system context and boundaries in terms of interfaces and interactions with external entities.

NOTE This task is mainly based on the outcomes of the Business or Mission Analysis process, and is performed concurrently with the Stakeholder Needs and Requirements Definition process. It consists of identifying the entities external to the software system (i.e., existing and projected systems, products, and services that constitute the system context) and defining the boundaries of the software system (i.e., interactions with these external entities through the interfaces that cross the boundaries). The external entities include the necessary enabling systems. The Architecture Definition process defines interfaces to the extent needed to support essential architectural decisions and understanding. These interface definitions are then refined by the Design Definition process.

2) Identify architectural entities and relationships between entities that address key stakeholder concerns and critical software system requirements.

NOTE Architecture is not necessarily concerned with all requirements, but rather only with those system/software requirements that drive the architecture. On the other hand, the Design Definition process addresses and takes into account all the requirements. Sometimes, through the Architecture Definition process there will be requirements that are deemed to be inappropriate, unaffordable, or unsuitable. These are requirements issues that are resolved through iteration of the System/Software Requirements Definition process. It is also important that the architecture addresses key stakeholder concerns since not all of these will be captured in requirements.

3) Allocate concepts, properties, characteristics, behaviors, functions, or constraints that are significant to architecture decisions of the software system to architectural entities.

NOTE The items being allocated can be physical, logical, or conceptual.

4) Select, adapt, or develop models of the candidate architectures of the software system.

NOTE It is common to use models in architecture definition. The models used are those that best address key stakeholder concerns. Refer to ISO/IEC/IEEE 42010 for how this can be done. Historically, it has been common to use logical and physical models in architecture definition. Information on logical and other models is provided in Annex F.

5) Compose views from the models in accordance with identified viewpoints to express how the architecture addresses stakeholder concerns and meets stakeholder and system/software requirements.

6) Harmonize the architecture models and views with each other.

NOTE Correspondence rules from frameworks are one way to establish harmony between views. See ISO/IEC/IEEE 42010.

d) **Relate the architecture to design.** This activity consists of the following tasks:

1) Identify software system elements that relate to architectural entities and the nature of these relationships.

NOTE Sometimes the software system elements are initially notional until Design Definition has occurred since this depends on the actual design(s) to be done. Sometimes a “reference architecture” is created using these notional system elements as a means to convey architectural intent and to check for design feasibility.

2) Define the interfaces and interactions among the software system elements and external entities.

NOTE This is defined at level of detail necessary to convey the architectural intent and can be further refined in the Design Definition process.

3) Partition, align and allocate requirements to architectural entities and system elements.

4) Map software system elements and architectural entities to design characteristics.

5) Define principles for the software system design and evolution.

EXAMPLE Principles can include interoperability, use of selected design patterns, ease of replacing and upgrading system elements, or security levels.

e) **Assess architecture candidates.** This activity consists of the following tasks:

1) Assess each candidate architecture against constraints and requirements.

2) Assess each candidate architecture against stakeholder concerns using evaluation criteria.

NOTE The System Analysis process and the Risk Management process can be used to support this task.

3) Select the preferred architecture(s) and capture key decisions and rationale.

NOTE The Decision Management process can be used to support this task.

4) Establish the architecture baseline of the selected architecture.

NOTE The architecture baseline is composed of models, views and other relevant architecture descriptions.

f) **Manage the selected architecture.** This activity consists of the following tasks:

1) Formalize the architecture governance approach and specify governance-related roles and responsibilities, accountabilities, and authorities related to design, quality, security, and safety.

2) Obtain explicit acceptance of the architecture by stakeholders.

NOTE The Validation process is used to confirm that the architecture models and views reflect stakeholder requirements, that stakeholder concerns are addressed, and to help ensure that future iterations of software system architecture better address stakeholder concerns.

3) Maintain concordance and completeness of the architectural entities and their architectural characteristics.

NOTE The entities to be checked are not only technical. These are also, for example, legal, economical, organizational and operational entities that are normally part of stakeholder requirements and concerns.

4) Organize, assess and control evolution of the architecture models and views to help ensure that the architectural intent is met and the architectural vision and key concepts are correctly implemented.

5) Maintain the architecture definition and evaluation strategy.

NOTE This includes updating the architecture based upon technological (e.g., obsolescence), implementation, or operational experience. This includes the management of external and internal interfaces that are defined at this level of software system decomposition.

6) Maintain traceability of the architecture.

NOTE Throughout the life cycle, traceability is often maintained among the architectural entities or elements (models, views, and viewpoints), the requirements (including allocated, decomposed, and derived) and stakeholder concerns, software system design, interface definitions, analysis results, and verification methods or techniques.

7) 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. This process identifies candidates for the baseline. The Information Management process controls the information items, such as architecture descriptions (architecture models, architecture views, evaluations, and traceability).

6.4.5 Design Definition process

6.4.5.1 Purpose

The purpose of the Design Definition process is to provide sufficient detailed data and information about the system and its elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

For software systems, design activities typically iterate with activities in System/Software Requirements Definition and Architecture Definition. Design definition is typically applied iteratively and incrementally to develop a detailed design, including software elements, interfaces, databases, and user documentation. Software design is usually concurrent with software implementation, integration, verification, and validation. Annex H discusses software design using agile methods. During design and implementation, further process application refines allocation of evolving requirements among software elements.

NOTE 1 The Design Definition process is driven by requirements that have been vetted through the architecture and more detailed analyzes of feasibility. Architecture focuses on suitability, viability, and desirability, whereas design focuses on compatibility with technologies and other design elements and feasibility of implementation and integration. An effective architecture is as design-agnostic as possible to allow for maximum flexibility in the design trade space.

NOTE 2 This process provides feedback to the software system architecture to consolidate or confirm the allocation, partitioning and alignment of architectural entities.

6.4.5.2 Outcomes

As a result of the successful implementation of the Design Definition process:

a) Design characteristics of each system element are defined.

b) System/software requirements are allocated to system elements.

c) Design enablers necessary for design definition are selected or defined.

d) Interfaces between system elements composing the system are defined or refined.

e) Design alternatives for system elements are assessed.

f) Design artifacts are developed.

g) Any enabling systems or services needed for design definition are available.

h) Traceability of the design characteristics to the architectural entities of the system architecture is established.

NOTE Design definition considers applicable technologies and their contribution to the system solution. Design provides the 'implement-to' level of the definition, such as drawings, state diagrams, stories, and detailed design descriptions. For software elements, this process can result in a detailed design description that can be verified against requirements and the software architecture. Even if the software design is not fully specified in a formal description, it is sufficiently detailed to permit software implementation (construction) and test planning.

6.4.5.3 Activities and tasks

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

NOTE The SWEBOK, *Guide to the Software Engineering Body of Knowledge*, provides detailed discussion on software design. This knowledge area addresses fundamentals, key issues, design strategies and methods, and design notations.

a) **Prepare for software system design definition.** This activity consists of the following tasks:

1) Define the design definition strategy, consistent with the selected life cycle model and anticipated design artifacts.

NOTE The software design strategy can include initial or incremental decomposition into system elements; creation of various views of automated procedures, data structures and control systems; selection of design patterns, or progressively more detailed definition of objects and their relationships.

2) Select and prioritize design principles and design characteristics.

NOTE Design principles include controlling ideas such as abstraction, modularization and encapsulation, separation of interface and implementation, concurrency, and persistence of data. Security considerations include the principle of least privilege, layered defenses, restricted access to system services, and other considerations to minimize and defend the system attack surface. Design characteristics include, for example, availability, fault tolerance and resilience, scalability, usability, capacity and performance, testability, portability, and affordability.

3) Identify and plan for the necessary enabling systems or services needed to support design definition.

NOTE This includes identification of requirements and interfaces for the enabling systems. Enabling systems for design definition include selection of software and system platforms, programming languages, design notations and tools for collaboration and design development, design reuse repositories (for product lines, design patterns, and design artifacts), and design standards.

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

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

b) **Establish designs related to each software system element.** This activity consists of the following tasks:

1) Transform architectural and design characteristics into the design of software system elements.

NOTE Characteristics apply to physical and logical system elements, such as database structures, provisions for memory and storage, software processes and controls, external interfaces such as user interfaces, or services. ISO 9241-210 provides human centred design/ergonomic design guidelines.

2) Define and prepare or obtain the necessary design enablers.

NOTE Design enablers include models, equations, algorithms, calculations, formal expressions and values of parameters, patterns, and heuristics, which are associated with design characteristics using adequate representation such as drawings, logical diagrams, flowcharts, coding conventions, logic patterns, information models, business rules, user profiles, scenarios, use cases or user stories, and tables of metrics and their values, e.g., function points or user story points.

3) Examine design alternatives and feasibility of implementation.

NOTE 1 For the software system and software elements, typically reuse, adaptation, outsourced service, or new development are examined.

NOTE 2 Assess the feasibility of realizing design characteristics. If warranted by assessment results, examine other alternative design options or perform trade-offs in the architecture or requirements when design characteristics are impractical to implement.

4) Refine or define the interfaces among the software system elements and with external entities.

NOTE Interfaces are identified and defined in the Architecture Definition process (see 6.4.4) to the level or extent needed for the architecture intent and understanding. These are refined in the Design Definition process based on the design characteristics, interfaces, and interactions of software elements with other elements composing the software system and with external entities. Additional interfaces are sometimes identified and defined that were not addressed in the architecture definition.

5) Establish the design artifacts.

NOTE This task formalizes the design characteristics of the software system elements through dedicated artifacts, depending on the implementation technology. Examples of artifacts include prototypes, data models, pseudocode, entity-relationship diagrams, use cases, user role and privilege matrixes, interface specifications, service descriptions, and

procedures. Design artifacts are developed, obtained, or modified for selected alternatives. The data is associated with detailed acceptable margins for implementation (if relevant at this process or task iteration).

c) **Assess alternatives for obtaining software system elements.** This activity consists of the following tasks:

1) Determine technologies required for each element composing the software system.

NOTE Several technologies are sometimes used for a given software system element, e.g., internet presence, embedded systems, adaptation of open source software, human operator roles.

2) Identify candidate alternatives for the software system elements.

NOTE Alternatives include newly designed and constructed items; adaptations of existing product lines, components, objects, or services; or acquisition or reuse of Non-Developed Items (NDI). NDI include COTS (Commercial-Off-The-Shelf) or FOSS (Free and Open Source Software) packages or elements, reuse of a previous design, or existing assets, including acquirer provided items.

3) Assess each candidate alternative against criteria developed from expected design characteristics and element requirements to determine suitability for the intended application.

NOTE A make-or-buy decision and resulting implementation and integration approach typically involve trade-offs of the design criteria, including cost. Design choices commonly consider enabling systems required to test the candidate alternative (test-driven design and development) and sustainability over the system life, including maintenance costs. The Maintenance process can be used to determine the suitability of the design for long-term maintenance and sustainability.

4) Choose the preferred alternatives among candidate design solutions for the software system elements.

NOTE The System Analysis process can be used for analyzes and assessments to support the Decision Management process in performing the selection. Design reviews are conducted using the Validation process.

d) **Manage the design.** This activity consists of the following tasks:

1) Capture the design and rationale.

NOTE Commonly captured information includes the software system elements and affiliated requirements and design data, e.g., for software elements, internal and external interfaces, data structures, implementation and test requirements, unit aggregation data for integration, and test cases. Rationale typically includes information about major implementation options and enablers. The resultant design is controlled in accordance with the strategy.

2) Establish traceability between the detailed design elements, the system/software requirements, and the architectural entities of the software system architecture.

NOTE 1 This task facilitates providing feedback to the Architecture Definition process for potential modifications, for example, to modify the allocation of software system elements in order to obtain the expected architectural characteristics; or possibly to modify the expected architectural characteristic due to factors discovered during the design process, or to make stakeholders aware of the potential impacts.

NOTE 2 Through the life cycle, bidirectional traceability is maintained between the design and the verification methods or techniques, and software system element requirements. Allocations and design properties are assigned to software elements, software units and affiliated artifacts, at a detailed enough level to permit software testing and implementation, including construction.

3) Determine the status of the software system and element design.

NOTE 1 The Measurement process is used to establish measures for the completeness and quality of the design as it progresses. The Verification and Validation processes are invoked to verify and validate the detailed design and implementation

NOTE 2 This includes periodic assessment of the design characteristics in case of evolution of the software system and of its architecture, as well as forecasting potential obsolescence of components and technologies, their replacement by others over time in the life cycle of the software system, and the consequences for the design definition. The Risk Management process is typically applied to evaluate risks in the design strategy, initial design, and the evolving design.

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 for artifacts such as design models. This process identifies candidates for the baseline, and the Information Management process controls the information items, such as design descriptions and specifications.

6.4.6 System Analysis process

6.4.6.1 Purpose

The purpose of the System Analysis process is to provide a rigorous basis of data and information for technical understanding to aid decision-making across the life cycle.

The System Analysis process applies to the development of inputs needed for any technical assessment. It can provide confidence in the utility and integrity of system requirements, architecture, and design. System analysis covers a wide range of differing analytic functions, levels of complexity, and levels of rigor. It includes mathematical analysis, modelling, simulation, experimentation, and other techniques to analyze technical performance, system behavior, feasibility, affordability, critical quality characteristics, technical risks, life cycle costs, and to perform sensitivity analysis of the potential range of values for parameters across all life cycle stages. It is used for a wide range of analytical needs concerning operational concepts, determination of requirement values, resolution of requirements conflicts, assessment of alternative architectures or system elements, and evaluation of engineering strategies (integration, verification, validation, and maintenance). Formality and rigor of the analysis will depend on the criticality of the information need or work product supported, the amount of information/data available, the size of the project, and the schedule for the results.

NOTE The System Analysis process can be employed for the entire software system or any element. This process is often used in conjunction with the Decision Management process.

6.4.6.2 Outcomes

As a result of the successful implementation of the System Analysis process:

a) System analyzes needed are identified.

b) System analysis assumptions and results are validated.

c) System analysis results are provided for decisions.

d) Any enabling systems or services needed for system analysis are available.

e) Traceability of the system analysis results is established.

6.4.6.3 Activities and tasks

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

a) **Define the system analysis strategy and prepare for system analysis.** This activity consists of the following tasks:

1) Identify the problem or question that requires analysis.

NOTE This includes technical, functional, and non-functional objectives of the analysis. Non-functional objectives include critical quality characteristics, various properties, technology maturity, and technical risks. The problem statement or question to be answered by the analysis is essential to establish the objectives of the analysis and the expectations and utility of the results.

2) Identify the stakeholders of the analysis.

3) Define the scope, objectives, and level of fidelity of the analysis.

NOTE The necessary level of fidelity (accuracy or precision) is a factor in determining the appropriate level of rigor.

4) Select the methods to support the analysis.

NOTE The methods are chosen based on time, cost, fidelity, technical drivers, and criticality of analysis. Analysis methods have a wide range of levels of rigor and include expert judgment, worksheet computations, parametric estimates and calculations, historical data and trend analysis, engineering models, simulation, visualization, and prototyping. Due to cost and schedule constraints, most projects typically perform system analysis only for critical characteristics.

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

NOTE This task includes identification of requirements and interfaces for the enabling systems. The system analysis enabling systems include the tools, relevant models, and potential data repositories needed to support the analysis. The methods chosen will be a major factor in determining what tools are appropriate to support the analysis. This also includes determining the availability of reusable or other relevant models and data, or resources.

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

NOTE The Infrastructure Management process enables the provision of systems analysis services. The Validation process is used to objectively confirm that the enabling system achieves its intended use for its enabling functions.

7) Collect the data and inputs needed for the analysis.

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

1) Identify and validate contexts and assumptions.

2) Apply the selected analysis methods to perform the required analysis.

3) Review the analysis results for quality and validity.

NOTE The results are coordinated with associated analyzes that have been previously completed.

4) Establish conclusions and recommendations.

NOTE The appropriate subject matter experts and stakeholders are identified and engaged in this task.

5) Record the results of the system analysis,

c) **Manage the system analysis**. This activity consists of the following tasks:

1) Maintain traceability of the analysis results.

NOTE Through the life cycle, bidirectional traceability is maintained between the analysis results and any software system item for which the analysis is supporting a decision or providing rationale (e.g., system/software requirement values, architecture alternatives). This is often facilitated by an appropriate data repository.

2) 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. This process identifies candidates for the baseline, and the Information Management process controls the information items. For this process, the analysis results or reports are typical information items that are managed.

6.4.7 Implementation process

6.4.7.1 Purpose

The purpose of the Implementation process is to realize a specified system element.

This process transforms requirements, architecture, and design, including interfaces, into actions that create a system element according to the practices of the selected implementation technology, using appropriate technical specialties or disciplines. This process results in a system element that satisfies specified system requirements (including allocated and derived requirements), architecture, and design.

For software systems, the purpose of the Implementation process is to realize a software system element.

Software system elements can include hardware, software, and services. For software implementation, this process transforms specified designs, behavior, interfaces and implementation constraints into actions that create a software system element implemented as a software product or service, also known as a "software item". Software implementation results in a software element that satisfies specified requirements through verification and stakeholder requirements through validation. Software implementation includes various combinations of construction (coding of newly built software elements), acquisition of new software packages (e.g., from open source or a commercial or organizational source) or re-use of existing elements (with or without modification).

Software implementation commonly involves use of the Agreement processes to obtain non-developmental items (NDI), such as hardware and operating systems (the platform) or enabling systems and services. Software implementation is usually performed concurrently with software integration. Implementation is typically performed along with all of the Technical Management processes and many of the Technical processes, especially:

a) The Verification process, which provides objective evidence that the software implementation fulfills its specified requirements and identifies anomalies (errors, defects, faults) in implementation-related information items, (e.g., system/software requirements, architecture, design, or other descriptions), processes, software elements, items, units;

b) The Validation process, which confirms that the implementation fulfils requirements for a specific intended use of a software work product.

6.4.7.2 Outcomes

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

a) Implementation constraints that influence the requirements, architecture, or design are identified.

b) A system element is realized.

c) A system element is packaged or stored.

d) Any enabling systems or services needed for implementation are available.

e) Traceability is established.

6.4.7.3 Activities and tasks

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

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

1) Define an implementation strategy, with consideration of the following:

i) development policies and standards, including standards that govern applicable safety, security, privacy and environmental practices; programming or coding standards; unit test policies; and language-specific standards for implementing security features;

ii) For reused or adapted software, methods to determine the level, source, and suitability of the reused system elements and security of the supply chain;

iii) procedures and methods for software development (construction) and development of unit tests; and the use of peer reviews, unit tests, and walkthroughs during implementation;

iv) use of CM control during software construction;

v) change management considerations for manual processes;

vi) implementation priorities to support data and software migration and transition, along with retirement of legacy systems;

vii) creation of manual or automated test procedures to verify that a software unit meets its requirements before creation of the software unit (test-driven development); and

viii) comprehensive or specialized life cycle development and support environments for realizing and managing requirements, models and prototypes, deliverable system or software elements, and test specifications and test cases.

NOTE The implementation strategy is commonly recorded in a project's SDP or SEMP, or sometimes in a PMP.

2) Identify constraints from the implementation strategy and implementation technology on the system/software requirements, architecture characteristics, design characteristics, or implementation techniques.

NOTE 1 Constraints include current or anticipated limitations of the chosen implementation technology (e.g., for software, the operating system, database management system, web services), acquirer furnished materials or system elements for adaptation, and limitations resulting from the use of required implementation-enabling systems.

NOTE 2 The implementation strategy for software typically identifies and allocates 'implement-to' criteria, e.g., software architecture and design characteristics, system/software requirements including software assurance, usability considerations, configuration management, traceability, or other conditions to be satisfied. These criteria can clarify appropriate unit aggregation levels, specifications, and constraints.

3) Identify and plan for the necessary and distinct software environments, including enabling systems or services needed to support development and testing.

NOTE Implementation of software commonly uses distinct environments that are separated under configuration control from the operational (production) environment. Common Implementation process, enabling systems, and services include comprehensive or specialized life cycle development and support environments for realizing and managing requirements, models and prototypes, deliverable elements, and test environments, specifications and test cases; simulators for external systems, training systems; and content management systems for user documentation.

4) Obtain or acquire access to the software environments and other enabling systems or services.

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

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

NOTE Throughout the Implementation process the Verification process is used to objectively confirm the system elements conform to requirements. The Validation process is used to objectively confirm the element is suitable to be used in its intended operational environment according to stakeholder requirements.

1) Realize or adapt software elements, according to the strategy, constraints, and defined implementation procedures.

NOTE 1 Software elements are acquired, identified for reuse from organizational assets, or developed (constructed). Software elements that are acquired can range from a simple product purchase in accordance with organizational or project purchasing rules to a complex acquisition of a software system that involves the Acquisition and Supply processes. Adaptation includes configuration of software elements that are reused or modified. Construction can involve software coding, adaptive reuse and integration of existing units, refactoring, database development, and construction of manual or automated test procedures for each unit.

NOTE 2 For software elements that are developed, at the lowest level of implementation executable software units are constructed (often with associated data structures, application programming interfaces, service descriptions, user documentation, test cases, or other elements), controlled, made available to authorized roles, and stored according to the CM procedures for development artifacts.

NOTE 3 The *SWEBOK, Guide to the Software Engineering Body of Knowledge* provides detailed discussion on Software Construction. This knowledge area addresses fundamentals, management, measurement, practical considerations (e.g., construction design, languages, testing, reuse and integration), construction technologies (e.g., object oriented, error and exception handling, executable models, distributed software), and tools and environments.

2) Realize or adapt hardware elements of software systems.

NOTE Hardware elements are acquired or fabricated using applicable techniques relevant to the physical implementation technology and materials selected. As appropriate, hardware elements are verified for conformance to specified system requirements and critical quality characteristics. In the case of repeated system element implementation (e.g., mass production, replacement system elements) the implementation procedures and fabrication processes are defined and can be automated to achieve consistent and repeatable producibility. Some common hardware elements in software systems include integrations of acquired COTS systems, special modifications, e.g., for test or operational environments, and hardware controls with embedded software.

3) Realize or adapt service elements of software systems.

NOTE Service elements include a set of services to be provided. ISO/IEC 20000 (IEEE Std 20000) applies to management of system elements realized in services, including strategy, design, and transition. As appropriate, service elements are verified for conformance to the system requirements and service criteria. For example, operational resource elements are verified for conformance to the system requirements and operational concept. Service elements can include network communications, training, software packaging and distribution services, software customization services for customer-specific needs, operational and security monitoring, and user assistance.

4) Evaluate software unit and affiliated data or other information according to the implementation strategy and criteria.

NOTE 1 Criteria for evaluation commonly include satisfaction of unit requirements and test criteria, unit test coverage, traceability requirements, consistency with software element requirements or design, internal unit requirement consistency, and feasibility for further process activity, e.g., integration, verification, validation, operations and maintenance.

NOTE 2 Use the *Manage results of implementation* activity to record construction and address anomalies.

5) Package and store the software system element.

NOTE Contain the software system element in order to achieve continuance of its characteristics. Conveyance and storage, and their durations, can influence the specified containment. For software, a master copy of the implemented software (electronic or on physical media) is stored in a controlled location and made available to authorized roles (e.g., for use in the Integration and Transition processes). Configuration and product information is captured by the Configuration Management and Information Management processes when the element is stored.

6) Record objective evidence that the software system element meets requirements.

NOTE Evidence is provided in accordance with supply agreements, legislation and organization policy. Evidence includes element modifications made due to processing changes or non-conformances found during the Verification and Validation processes. The objective evidence is part of the element's as-implemented configuration baseline established through the Configuration Management process and includes the results of unit testing, analysis, inspections, walk-through events, demonstrations, product or technical reviews, or other verification exercises.

c) **Manage results of implementation.** This activity consists of the following tasks:

1) Record implementation results and anomalies encountered.

NOTE This includes anomalies due to the implementation strategy, the implementation enabling systems, or incorrect software system definition. The Project Assessment and Control and Quality Assurance processes are used to analyze the data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

2) Maintain traceability of the implemented software system elements.

NOTE 1 To support traceability throughout the life cycle during operations and maintenance, sources of software licenses and other system assets in the supply chain are recorded. The information management and configuration management processes are used to maintain license and maintenance support terms for a software application and its required infrastructure (host system). The ISO/IEC 19770 standards provide requirements for an IT asset management system.

NOTE 2 Bidirectional traceability is maintained between the implemented elements and the software system architecture; design, and related requirements, including interface requirements and definitions that are necessary for implementation; and validation and verification plans, procedures, and results.

3) 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. This process identifies candidates for the baseline, and the Information Management process controls the information items. For this process, the software system elements (e.g., source code), software packages, and unit test results are typical artifacts that are baselined.

6.4.8 Integration process

6.4.8.1 Purpose

The purpose of the Integration process is to synthesize a set of system elements into a realized system (product or service) that satisfies system/software requirements, architecture, and design.

This process assembles the implemented system elements. Interfaces are identified and activated to enable interoperation of the system elements as intended. This process integrates the enabling systems with the system-of-interest to facilitate interoperation.

Software system integration iteratively combines implemented software system elements to form complete or partial system configurations in order to build a product or service. Software integration is typically performed daily or continuously during development and maintenance stages, using automated tools. Continuous integration involves frequent inclusion or replacement and archiving of items in software libraries under CM control.

NOTE Interfaces are defined by the Architecture Definition and Design Definition processes. The Integration process coordinates with these other processes to check that the interface definitions, as implemented and integrated, are adequate and that they take into account the integration needs.

6.4.8.2 Outcomes

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

a) Integration constraints that influence system requirements, architecture, or design, including interfaces, are identified.

b) Approach and checkpoints for the correct operation of the assembled interfaces and system functions are defined.

c) Any enabling systems or services needed for integration are available.

d) A system composed of implemented system elements is integrated.

e) The interfaces between the implemented system elements that compose the system are checked.

f) The interfaces between the system and the external environment are checked.

g) Integration results and anomalies are identified.

h) Traceability of the integrated system elements is established.

6.4.8.3 Activities and tasks

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

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

1) Define the integration strategy.

NOTE 1 Integration builds sequences of progressively more complete software system element or software item configurations. It is dependent on applicable software system element availability and is consistent with a fault isolation and diagnosis strategy. Successive applications of the Integration process and the Verification process, and when appropriate the Validation process, are repeated for elements in the system structure until the system-of-interest has been realized. Simulators or prototypes are typically utilized for system elements that are not yet implemented, e.g.,

receiving data from interfacing systems. Integrating the implemented software system elements is based on the priorities of the related requirements and architecture definition, typically focusing on the interfaces, while minimizing integration time, cost, and risks. Software system integration commonly maintains version control through the Configuration Management process for selection of configuration items to be integrated.

NOTE 2 For software integration, the integration strategy typically is consistent with a regression strategy which is applied for re-verifying software elements when related software units (and potentially associated requirements, design and user documentation) are changed.

NOTE 3 Defining a strategy for software unit and element integration commonly accompanies defining the strategy for other processes that occur concurrently, such as:

i) The Implementation process to help ensure timely coordination of Implementation and Integration process tasks and enabling systems, e.g., combined software development and test environments to support automated or continuous implementation and integration of software units and elements.

ii) The Verification process to provide objective evidence that the integrated software fulfils its specified requirements and to identify anomalies (errors, defects, faults) in integration-related information items, (e.g., system/software requirements, architecture, design, test, or other descriptions), processes, software elements, items, units.

iii) The Validation process to confirm that a work product fulfils requirements for a specific intended use of an integrated software function.

iv) The Quality Assurance process to support integration process and work product audits and inspections and to address problem, non-conformance, or incident reporting and handling.

NOTE 4 The integration strategy is commonly recorded in a plan, e.g., an integration plan, or a project's SDP or SEMP.

2) Identify and define criteria for integration and points at which the correct operation and integrity of the interfaces and the selected software system functions will be verified.

NOTE 1 Detailed verification of the interfaces is performed using the Verification process. Software integration typically involves combining software elements, resulting in a set of integrated software elements, that is consistent with the software design, and that satisfies the functional and non-functional system/software requirements on an equivalent of the operational environment.

NOTE 2 For projects involving multiple suppliers or development teams, the availability of software system elements for integration is typically part of the project schedule with milestones under the Project Assessment and Control process. Integration proceeds as the software is verified in its functionality, performance, and suitability for site-specific or platform-specific environments. At major integration points, e.g., completion of a stage, element, or version, check points for reviews and validation with stakeholders are typically held. The frequency of these reviews is related to the selected life cycle model and development method.

3) Identify and plan for the necessary enabling systems or services needed to support integration.

NOTE This includes identification of requirements and interfaces for the enabling systems. Enabling systems for integration commonly include integration facilities, specialized equipment, training systems, discrepancy reporting systems, simulators, measurement devices, and environmental security. For software, this can involve regression test suites and CM systems for the integrated testing of software systems, incident and problem reporting systems, simulators representing external systems or undeveloped elements, and software library management systems for development operations. Changes or specializations needed for the enabling systems to support the integration tasks need to be identified and defined. Typically, the enabling systems or services used for integration during development stages can also help support system element integration as the software system and enabling environments evolve to operational status. This "DevOps" approach supports iterative software system implementation, integration, verification, transition, validation, operation and maintenance processes.

4) Obtain or acquire access to the enabling systems or services to be used to support integration.

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

5) Identify constraints for integration to be incorporated in the system/software requirements, architecture or design.

NOTE This includes requirements such as accessibility, supply chain security, safety for integrators, required interconnections for sets of implemented software system elements and for enablers, and interface constraints.

b) **Perform integration**. Successively integrate software system element configurations until the complete system is synthesized. This activity consists of the following tasks:

1) Obtain implemented software system elements in accordance with agreed schedules.

NOTE The implemented software system elements are provided from the developers or received from suppliers, the acquirer, or other resources and typically placed under CM control. The elements are handled in accordance with relevant health, safety, security and privacy considerations.

2) Integrate the implemented elements.

NOTE 1 This task is performed to achieve software system element configuration (complete or partial) connecting the implemented elements as prescribed in the integration strategy, using the defined procedures, interface control descriptions, and the related integration enabling systems.

NOTE 2 In terms of software, integrating the implemented elements can involve linking together pieces of object code or simply bringing together the implemented elements that are part of the software configuration in a methodical piece by piece approach. Software elements are typically compiled into a "build" so that branched units are properly linked or merged in the assembled element. Firmware elements are fabricated, often as prototypes, and installed in hardware elements. If software functions are not yet available for integration, emulated functionality (stubs or scaffolding) can be used to temporarily support integration of software elements or represent input from external interfaces. Successful aggregations result in an integrated software element, that is stored and available for further processing, i.e., additional software system element integration, verification, or validation.

NOTE 3 Anti-counterfeit, anti-tamper, system and software assurance and interoperability concerns can arise when performing integration and identifying and defining checkpoints. Integration and Verification processes often use fictitious data for security or privacy considerations. ISO/IEC/IEEE 15026 and the ISO/IEC 27000 series include information on assurance, integrity, and security considerations affecting integration.

3) Check that the integrated software interfaces or functions run from initiation to an expected termination within an expected range of data values.

NOTE As part of the acceptance of the implemented software system elements, selected elements are checked to help ensure they meet acceptance criteria as specified in the integration strategy and applicable agreements. Checking can include conformance to the agreed configuration, compatibility of interfaces, and the presence of mandatory information items. The Project Assessment and Control process can be used in accordance with the integration strategy to plan and conduct technical reviews of the integrated software system elements, e.g., a test readiness review to help ensure the integrated element or system with its affiliated data and information items is ready for qualification testing.

c) **Manage results of integration.** This activity consists of the following tasks:

1) Record integration results and anomalies encountered.

NOTE This includes anomalies due to the integration strategy, the integration enabling systems, execution of the integration or incorrect system or element definition. Where inconsistencies exist at the interface between the system, its specified operational environment and systems that enable the utilization stage, the deviations lead to corrective actions. Anomaly resolution typically involves the Technical Processes, often repetitive application of the Implementation process. The Quality Assurance and Project Assessment and Control process are used to analyze the data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

2) Maintain traceability of the integrated software system elements.

NOTE Bidirectional traceability is maintained between the integrated system elements and the software system architecture, design, and system or element requirements, such as use cases, and including interface requirements and definitions that are necessary for integration. Integrated software elements and their components are identified by version. Versions of integrated software elements are commonly traceable to implemented units, test procedures, and test cases.

3) 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. The Integration process identifies candidates for the baseline, and the Information Management process controls the

information items. For this process, the test cases, regression tests, and automated test scripts are typical artifacts that are baselined. The integration strategy is a typical information item that is baselined.

6.4.9 Verification process

6.4.9.1 Purpose

The purpose of the Verification process is to provide objective evidence that a system or system element fulfils its specified requirements and characteristics.

The Verification process identifies the anomalies (errors, defects, or faults) in any information item (e.g., system/software requirements or architecture description), implemented system elements, or life cycle processes using appropriate methods, techniques, standards or rules. This process provides the necessary information to determine resolution of identified anomalies.

Verification can be performed across all technical processes. The Verification process is typically used at key points in a software system's life cycle to demonstrate that the requirements (including functional and non-functional requirements) have been met, or that process outcomes have been achieved or process activities have been performed. Different domains and engineering or development communities can identify the milestones, verification strategies and criteria differently.

For software systems, the Verification process is typically instantiated for the following purposes:

a) To confirm that a software work product or service properly reflects the specified requirements (often called software verification);

b) To confirm that the integrated software product meets its defined requirements (often called software qualification testing); and

c) To confirm that the implementation of each system/software requirement is tested for compliance and that the software system is ready for delivery (often called system qualification testing).

NOTE 1 The Verification process determines that the "product is built right". The Validation process determines that the "right product is built".

NOTE 2 ISO/IEC/IEEE 29119 *Systems and software engineering — Software testing* (in multiple parts) provides detailed processes and techniques for verification performed through testing. IEEE Std 1012-2012, *IEEE Standard for System and Software Verification and Validation*, provides additional details about these processes for systems, software, hardware, and interfaces being developed, maintained, or reused.

NOTE 3 The *SWEBOK, Guide to the Software Engineering Body of Knowledge*, provides detailed discussion on Software Testing. This knowledge area addresses fundamentals, terminology, issues, techniques, application, process planning, measures, tools, practical considerations, and references. The guide also discusses Software Verification and Validation in terms of Software Quality Management processes, and identifies methods and techniques that support both Verification and Validation. The SWEBOK also addresses topics such as software construction for verification and software engineering models and methods support.

6.4.9.2 Outcomes

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

a) Constraints of verification that influence the requirements, architecture, or design are identified.

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

c) The system or system element is verified.

d) Data providing information for corrective actions is reported.

e) Objective evidence that the realized system fulfills the requirements, architecture and design is provided.

f) Verification results and anomalies are identified.

g) Traceability of the verified system elements is established.

6.4.9.3 Activities and tasks

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

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

1) Define the verification strategy, which includes the following:

NOTE 1 A verification strategy generally focuses on minimizing cost, schedule, or risk, providing a balanced approach for confirming that the software system or element has been "built right".

NOTE 2 The verification strategy and schedule account for dynamic changes when anomalous results (events, incidents, or problems) occur. According to the progress of the project, planned verification actions are redefined or rescheduled when unexpected events or system evolutions occur.

NOTE 3 The verification strategy can be documented in a plan, e.g., a verification plan, or a project's SDP or SEMP.

i) Identify the verification scope, including the software system, element, or artifact, the properties to be verified, and the expected results.

NOTE Overall verification scope includes the software system-of-interest or system elements, including interfaces. For each verification action, the scope identifies the software system, element, or artifact to be verified (e.g., the actual system, or a model, a mock-up, a prototype, code, a procedure, a plan or other document) and the expected results, such as conformance, or performance, fault tolerance, and recovery after service interruption. The properties to be verified can include requirements, architecture and design characteristics, integration, and accuracy of documentation. Design characteristics can include security implications of the design in the context of the planned operational environment and the achievement of critical quality characteristics as stated in the requirements.

ii) Identify the constraints that potentially limit the feasibility of verification actions.

NOTE Constraints include technical feasibility, cost, time, availability of verification enablers or qualified personnel, contractual constraints, and characteristics such as criticality of the mission. Such constraints often factor into verification strategy determination, e.g., whether an organizationally independent verification effort is necessary or justified.

iii) Identify verification priorities.

NOTE In software systems, verification of every possible scenario (100% code coverage) is typically infeasible. The verification strategy typically includes trading off what will be verified (scope) against the constraints or limits, and deducing what verification actions to perform and how many iterations of verification actions and rework are needed to reduce risk. A model-based testing approach can enable the generation and management of multiple scenarios. Potential verification actions that are candidates for deletion are evaluated for the risks their withdrawal imposes.

2) Identify constraints from the verification strategy to be incorporated in the system/software requirements, architecture, or design.

NOTE This includes practical limitations of accuracy, uncertainty, repeatability that are imposed by the verification enablers, the associated measurement methods, the need for software system integration, and the availability, accessibility and interconnection with enablers.

3) Define the purpose, conditions and conformance criteria for each verification action.

4) Select appropriate verification methods or techniques and associated criteria for verification actions, such as inspection, analysis, demonstration, or testing.

NOTE 1 The selection of verification methods or techniques is made according to the type of system, the purpose of the verification, the objectives of the project, and the acceptable risks. Verification methods or techniques include inspection

(including code walkthroughs and peer review), analysis (including modelling and simulation, and analogy/similarity), demonstration, and dynamic and static testing.

NOTE 2 The selected verification approach, methods, and techniques can be coordinated with relevant stakeholders to help ensure the verification approach is acceptable.

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

NOTE Verification enabling systems include verification facilities, qualified personnel, equipment, simulators, test automation tools, and incident and problem management systems. Software system verification is typically performed in distinct controlled environments that do not interfere with operational software or ongoing development. If the enabling systems for verification differ in capability from the planned operational environment, the Measurement process can be used to calibrate the performance of the verification enabling systems and suitability for the verification action.

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

NOTE The acquisition of the enabling systems can be done through various ways such as rental, procurement, development, reuse of organizational assets, or subcontracting. Usually the acquisition of the complete set of enablers is a mix of these ways. The Validation process is used to objectively confirm that the verification enabling system achieves its intended use for its enabling functions.

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

1) Define the verification procedures, each supporting one or a set of verification actions.

NOTE Verification procedures, which can be performed by automated scripts, include the requirements to be verified, the type of software system element or artifact to be verified (e.g., the actual system, or a model, a mock-up, a prototype, code, a procedure, a plan, or other information item), and the expected results (success criteria), such as conformance, or performance of a function or capacity in terms of response time or throughput. The procedures identify the purpose of the verification with success criteria (expected results), the verification technique to be applied, the necessary enabling systems (facilities, equipment), and the environmental conditions to perform each verification procedure (resources, qualified personnel, specialized procedural set-up or work instructions). Verification procedures include how the verification procedure results will be recorded, analyzed, stored, and reported.

2) Perform the verification procedures.

NOTE Verification occurs, in accordance with the verification strategy, at the appropriate time in the schedule, in the defined environment, with defined enabling systems and resources. The performance of a verification action consists of capturing a result from the execution of the verification procedure; comparing the obtained and recorded result with the expected result; and deducing a degree of correctness (or success/failure) of the submitted element.

c) **Manage results of verification.** This activity consists of the following tasks:

1) Review verification results and anomalies encountered and identify follow-up actions.

NOTE 1 This includes anomalies due to the verification strategy, the verification enabling systems, execution of the verification, or incorrect system definition. The Project Assessment and Control and Quality Assurance processes are used to analyze the data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

NOTE 2 The evaluation of verification results and follow-up corrective action can vary greatly depending on the purpose of the verification. For elements of software, examples include a modification or waiver of requirements, a simple defect fix for a failed software element, followed by re-verification, or major project re-direction based on a failure to attain a key milestone, e.g., failed software system qualification testing. Often simple or recommended solutions to anomalies discovered during verification, are recorded with the verification result to facilitate analysis and potential corrective action.

2) Record incidents and problems during verification and track their resolution.

NOTE 1 Performing problem resolution is handled through the Quality Assurance and Project Assessment and Control processes. During software verification, the conditions under which the problem occurred are documented so that if possible, the problem can be duplicated and the root cause of the software defect identified. Changes to the requirements, architecture, design, or system elements are done using other Technical processes.

3) Obtain stakeholder agreement that the software system or element meets the specified requirements.

4) Maintain traceability of the verified software system elements.

NOTE Bidirectional traceability is maintained between the verified system elements and the record of verification activity, system architecture, design, or system/software requirements.

5) 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. This process identifies candidates for the baseline, and the Information Management process controls the information items. For this process, the verification strategy and verification procedures are typical information items.

6.4.10 Transition process

6.4.10.1 Purpose

The purpose of the Transition process is to establish a capability for a system to provide services specified by stakeholder requirements in the operational environment.

This process moves the system in an orderly, planned manner into the operational status, such that the system is functional, operable and compatible with other operational systems. It installs a verified system, together with relevant enabling systems, e.g., planning system, support system, operator training system, user training system, as defined in agreements. This process is used at each level in the system structure and in each stage to complete the criteria established for exiting the stage. It includes preparing applicable storage, handling, and shipping enabling systems.

For software systems, the purpose of the Transition process is to establish a capability for a system to provide services in a different environment.

The Transition process is often used for recurring deployments of software to different environments, e.g., from a development environment to a test or maintenance environment, or between various test environments, or from one operational environment to another (e.g., rehosting or use of cloud services). Transitions to backup or contingent sites are typically planned and rehearsed for business continuity and disaster recovery. Transition for software systems can involve the physical relocation of hardware, the installation and activation or deactivation of physical or virtual infrastructure or enabling systems in different locations, or no change to the physical infrastructure. Transition can involve changes to the data sources, data structure, or updates or upgrades of functional software. Transition includes recurring scheduled or emergency patches and fixes for security and other concerns. Transition can involve transfer between organizations and also encompasses the addition of a large group of new users to an existing software system or service. Transition to a new system often is performed concurrently with retirement and disposal of an existing system, entailing data migration from the old system to its replacement.

NOTE Transition can involve knowledge transfer using the Knowledge Management process.

6.4.10.2 Outcomes

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

a) Transition constraints that influence system/software requirements, architecture, or design are identified.

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

c) The site is prepared.

d) The system, as installed in its operational location, is capable of delivering its specified functions.

e) Operators, users and other stakeholders necessary to the system utilization and support are trained.

f) Transition results and anomalies are identified.

g) The installed system is activated and ready for operation.

h) Traceability of the transitioned elements is established.

6.4.10.3 Activities and tasks

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

a) **Prepare for the software system transition.** This activity consists of the following tasks:

1) Define a strategy for managing software releases and other software system transitions, including the following considerations:

i) establishing the type of transition and transition success criteria;

ii) determining the frequency of recurring transitions, such as updates and upgrades to development, test, and operational software systems;

iii) minimizing security risks, disruption, and downtime during transition;

iv) archiving, destroying, or converting and validating data from previous systems to the new system; including data received through external interfaces;

v) contingency planning for problem resolution, backup and return to the last working system version;

vi) scheduling transitions consistent with ongoing business processing, with phased or synchronized transition of systems

vii) change management for stakeholders, including interface partners, human operators, system administrators, and software system or service users;

NOTE Change management activities are often conducted to design changes in business processes associated with the new system, plan the transition in business processes, and gain user commitment to productive use of the new system.

viii) associated strategies for validation of the transitioning system or element;

ix) initiating user support and maintenance activities with the transfer and update of system design documentation, user documentation, and test procedures; and

x) concurrent execution of the Transition, Operations, and Disposal processes, when a new system is commissioned and an old system is decommissioned.

NOTE The strategy includes roles and responsibilities, approval authority, use of readiness reviews and training.

2) Identify and define facility, site, communications network, or target environment changes needed for software system installation or transition.

NOTE For each transition, identify and define any needed changes in infrastructure or enabling systems. A site survey can be performed to identify needed changes in the physical environment to install or use the software system, such as changes to maintain the physical and information security of the system.

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. User roles are established and user accounts and access controls are implemented.

4) Prepare detailed transition information, such as plans, schedules, and procedures.

NOTE 1 The transition strategy is commonly recorded in a plan, e.g., a transition plan, or a project's SDP or SEMP. Transition schedules help validate that sufficient resources and infrastructure are available to support the transition, so that activities can be executed within a reasonable timeframe to minimize disruption. Schedules can include rehearsals

for complex transitions, in which procedures, such as database and system backup and restore and software installation, are tested to verify durations and correct results.

NOTE 2 During a specified period of changeover or concurrent operation, the transfer of services is managed so that continuing conformance to persistent stakeholder needs or an agreed level of service is achieved. If a period of parallel operations for both the old and new systems is needed, special procedures are identified and developed for receiving and utilizing data from interface partners.

5) Identify system constraints from transition to be incorporated in the software system requirements, architecture or design.

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.

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

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

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

1) Prepare the site of operation or virtual environment in accordance with installation requirements.

NOTE Site preparation is conducted in accordance with applicable health, safety, security and environmental regulations. Virtual environments and new communication resources are initialized and verified. Shipping and receiving of physical system elements and enabling systems is arranged.

2) Deliver the software system or element for installation at the correct location and time.

NOTE 1 Typically software is delivered electronically. For physical media, hardware, and embedded software systems, it is sometimes necessary to account for temporary storage prior to delivery or installation.

NOTE 2 Deliver agreed information items in electronic or physical form, such as training material, logistics support packages, or user documentation.

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. Licenses and maintenance agreements for system elements, and other intellectual property, are transferred according to agreements.

4) Provide user documentation and training for the operators, users, and other stakeholders necessary for product utilization and support.

5) Perform activation and check-out, including the following as agreed:

NOTE 1 This task takes the steps needed to activate the product to an operational state, including start-up, assessment of environmental conditions, and other readiness evaluations, in accordance with operational procedures, organizational policies, and regulations. Where the exact location or environment of operation is not available or when software will be accessed from multiple or mobile locations, a representative example is selected.

NOTE 2 Acceptance tests are sometimes defined in the agreement to demonstrate satisfactory installation. This task interacts with the Validation process to objectively confirm that the system fulfills stakeholder requirements in the operational environment. Acceptance tests, as specified in agreements, can define the criteria that demonstrate that the software system entity possesses the capability to deliver the required functions and services when installed and sustained in its operational environment. Specific attention is given to the key functions and logical interfaces.

NOTE 3 As part of the Configuration Management process, a physical configuration audit (PCA) and update of as-built documentation is often performed at the time of system activation. Anti-counterfeit provisions can be confirmed.

i) Demonstrate proper installation of the software system.

NOTE This task can include integrity checks of data and operations, e.g., that the software code and data representations properly initialize, execute, and terminate as specified.

ii) Demonstrate the installed or transitioned product is capable of delivering its required functions.

NOTE This is an operational readiness task that examines readiness of functional capability for an operational state. Specific attention is given to the data interfaces and security concerns: information assurance and interoperability functions are exercised.

iii) Demonstrate the functions provided by the system are sustainable by the enabling systems.

NOTE This is an operational readiness task that examines readiness of enabling systems for an operational state. For example, activation of monitoring, problem reporting, access control, backup and recovery, and user assistance (customer support) are demonstrated.

iv) Review the software system for operational readiness.

NOTE This includes the results of functional demonstrations, validation activities, and sustainment demonstrations. A readiness review can be conducted. Deficiencies, risks, and problems that impact the success of the transition are resolved, accepted for waiver, or closed.

v) Commission the software system for operations.

NOTE This includes providing support to the users, administrators, and operators during the operations commencement (commissioning) of the system.

c) **Manage results of transition.** This activity consists of the following tasks:

1) Record transition results and anomalies encountered.

NOTE This includes anomalies due to the transition strategy, the transition enabling systems, execution of the transition or incorrect software system or database system definition. Where inconsistencies exist between the system, its operational environment, and enabling systems, the deviations are resolved through corrective actions, including requirement changes. The Project Assessment and Control and Quality Assurance processes are used to analyze the data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

2) Record transition incidents and problems and track their resolution.

NOTE Performing problem resolution is handled through the Quality Assurance and Project Assessment and Control processes. During transition, the conditions under which the problem occurred are documented so that if possible, the problem can be duplicated and the root cause of the defect identified. Changes to the requirements, architecture, design, or software system elements are done using other Technical processes.

3) Maintain traceability of the transitioned software system elements.

NOTE Bidirectional traceability is maintained between the transitioned and deployed system and elements and the approved and controlled versions of the software system and enabling systems.

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, including transitioned software system elements. This process identifies candidates for the baseline, and the Information Management process controls the information items. For this process, the transition strategy, training material, and installation, transition and data migration procedures, and user documentation are typical information items that are baselined.

6.4.11 Validation process

6.4.11.1 Purpose

The purpose of the Validation process is to provide objective evidence that the system, when in use, fulfils its business or mission objectives and stakeholder requirements, achieving its intended use in its intended operational environment.

The objective of validating a system or system element is to acquire confidence in its ability to achieve its intended mission, or use, under specific operational conditions. Validation is ratified by stakeholders. This process provides the necessary information so that identified anomalies can be resolved by the appropriate technical process where the anomaly was created.

The Validation process is typically used at key points in a product's life cycle to demonstrate that the product's requirements for stakeholder intended operational use have been met. Validation is also applicable to the software engineering artifacts (viewed as software system elements). Different domains and engineering or development communities can identify the milestones, validation strategies and criteria differently.

For software systems, highly iterative life cycle models often feature frequent involvement by the acquirer, user representative, or other stakeholders to validate, e.g., the priority of requirements for inclusion in an iteration, the usability of the software interface through prototypes, and the suitability of the software for performing business tasks and fulfilling the operational concept.

For software systems, the following are purposes of the Validation process:

a) To confirm that the requirements for a specific intended use of the software work product are fulfilled (often called software validation); and

b) To achieve confidence (especially with an acquirer or customer) that the delivered product meets stakeholder requirements and is fit for use (often called software acceptance testing).

NOTE 1 The validation process determines that the "right product is built". The verification process determines that the "product is built right".

NOTE 2 Acceptance criteria, as used for acceptance testing, include criteria to determine whether the delivered product is fit to use or not. Acceptance criteria for acceptance can be specified and agreed between two parties, i.e., an acquirer and a supplier, and included in the stakeholder requirements.

NOTE 3 IEEE Std 1012-2012, *IEEE Standard for System and Software Verification and Validation*, provides detailed requirements. The *SWEBOK, Guide to the Software Engineering Body of Knowledge*, discusses software verification and validation in terms of software Quality Management processes, and contains methods and techniques that support both verification and validation. The SWEBOK also addresses topics such as requirements and model validation.

6.4.11.2 Outcomes

As a result of the successful implementation of the Validation process

a) Validation criteria for stakeholder requirements are defined.

b) The availability of services required by stakeholders is confirmed.

c) Constraints of validation that influence the requirements, architecture, or design are identified.

d) The system or system element is validated.

e) Any enabling systems or services needed for validation are available.

f) Validation results and anomalies are identified.

g) Objective evidence that the realized system or system element satisfies stakeholder needs is provided.

h) Traceability of the validated system elements is established.

6.4.11.3 Activities and tasks

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

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

1) Define the validation strategy, which includes the following:

NOTE 1 A validation strategy generally focuses on minimizing cost, schedule, or risk by progressively building confidence in the quality and suitability of the software system for the stakeholders.

NOTE 2 The validation strategy reflects the life cycle model, and often involves repeated validation for iterative, incremental, or evolutionary life cycles.

NOTE 3 The validation strategy can be documented in a plan, e.g., an acceptance plan, or a project's SDP or SEMP.

i) Identify the validation scope, including the characteristics of the software system, element, or artifact to be validated, and the expected results of validation.

NOTE Software system validation is typically performed both in distinct controlled environments that do not interfere with operational software or ongoing development, as well as in operational environments, typically before full operational use (e.g., beta testing or acceptance testing for a specified duration with agreed criteria). Scope includes stakeholder requirements, including related views of the system (e.g., scenarios or concept of operation) to be evaluated. The scope depends on what is appropriate for the systems life cycle stage: the system-of-interest or a system element or engineering artifact, such as a concept description or document, an operational scenario, a model, a mock-up, or prototype. The scope also includes evaluating that the software product or service is usable in its intended environment for the principal or critical functions. Additional characteristics to be validated can include usability of the documentation; fault tolerance, resilience, and recovery features of the software.

ii) Identify the constraints that potentially limit the feasibility of validation actions.

NOTE Constraints include practical limitations of accuracy, uncertainty, repeatability that are imposed by the validation enablers, the associated measurement methods, and the availability, accessibility and interconnection with enablers. The validation strategy is constrained by the progress of the project; in particular, planned validation actions are redefined or rescheduled when unexpected events or system evolutions occur. Validation can be extended to include ongoing measurements of user satisfaction and customer complaints.

iii) Identify validation priorities.

NOTE 1 To make effective use of stakeholders' time and expertise, validation typically focuses on stakeholder priorities, while verification is used for non-functional requirements. Potential validation actions that are candidates for deletion are evaluated for the risks their withdrawal imposes

NOTE 2 The supplier, the acquirer, or an agent of the acquirer participates in or performs validation. The responsibility is often designated in the agreement.

2) Identify system constraints from the validation strategy to be incorporated in the stakeholder requirements.

3) Define the purpose, conditions and conformance criteria for each validation action.

4) Select appropriate validation methods or techniques and associated criteria for each validation action.

NOTE 1 Software system validation methods or techniques include inspection, analysis, analogy/similarity, demonstration, simulation, peer review, and testing. Software validation techniques typically include demonstrations, inspection, reviews and code walkthroughs, usability tests, and trial use of the software (e.g., beta testing, operational testing, user testing, or an acceptance test with agreed criteria). The selection of validation methods or techniques is made according to the type of system, the purpose of the validation, the objectives of the project, legal and regulatory requirements, and the acceptable risks of a validation action. For software systems with human interaction, usability testing is commonly used to validate that representative users can achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. Further details for usability testing are found in ISO/IEC TR 25060:2010

Systems and software engineering — Systems and software product Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for usability: General framework for usability-related information.

NOTE 2 Where appropriate, validation steps or states are defined (e.g., in-house validation, on-site validation, operational validation) that progressively build confidence in conformance of the evolving software system, and assist diagnosis of any encountered discrepancies.

NOTE 3 Criteria for stakeholder acceptance of service performance is commonly stated as a service level and recorded in a service level agreement (SLA). Service levels typically measure capacity, availability, reliability, and timely response for services, and result in performance requirements for the supporting systems.

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

NOTE Enabling systems can include simulators, usability laboratory or test facility, qualified personnel, stakeholder and user representatives, according to selected validation methods or techniques. This includes identification of requirements and interfaces for enabling systems.

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

NOTE The infrastructure management process or acquisition process can be invoked to obtain access to enabling systems, such as through rental, procurement, development, reuse, or subcontracting. Usually access to the complete set of enablers is a mix of these ways. The Validation process is also used to objectively confirm that the validation enabling system achieves its intended use for its enabling functions.

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

1) Define the validation procedures, each supporting one or a set of validation actions.

NOTE Validation procedures identify stakeholder requirements to be validated, the associated software system artifact (e.g., the actual system, or a model, a mock-up, a prototype, code, a set of instructions or other information item), and the expected results (success criteria), such as completed and timely performance of a function. The procedures identify the purpose of the validation with success criteria (expected results), the validation technique to be applied, the necessary enabling systems (facilities, equipment), and the environmental conditions to perform each validation procedure (resources, qualified personnel, participating stakeholders, and specialized procedural set-up or work instructions). Validation strategy includes how the validation procedure results will be recorded, analyzed, stored, and reported.

2) Perform the validation procedures in the defined environment.

NOTE Validation occurs, in accordance with the validation strategy, at the appropriate time in the schedule, in a defined environment (such as the operational environment, a similar test environment, or other representative environment), with defined enablers and resources. The performance of a validation action typically consists of capturing execution results, comparing the obtained result with the success criteria, and deducing a degree of compliance or stakeholder satisfaction with the software system, element, service, or engineering artifact.

c) **Manage results of validation.** This activity consists of the following tasks:

1) Review validation results and anomalies encountered and identify follow-up actions.

NOTE 1 Confirm that the services of the system that are required by stakeholders are available. Anomalies can result from the validation strategy, the validation enabling systems, execution of the validation, incorrect system definition, or inefficient or ineffective system design, implementation, and integration.

NOTE 2 The evaluation of validation results and follow-up actions can include acceptance of the anomaly as a low risk occurrence. Corrective action can vary greatly depending on the impact of the validation result. For elements of software, examples include a simple defect fix for a failed software element, additional training for users, corrections and clarifications in documentation, or major project re-direction based on a failure to attain a key milestone, e.g., failed software system acceptance testing.

2) Record incidents and problems during validation and track their resolution.

NOTE The Project Assessment and Control process and Quality Assurance process are used to analyze the data to identify the root cause of problems, enable corrective or improvement actions, and to record lessons learned. During software validation, the gap between stakeholder expectations and system performance is documented so that if possible, the root cause of the discrepancy can be identified. Problem resolution typically involves determining the severity and impact of the problem and whether or when a software discrepancy is to be corrected or accepted for a time as a known error. Often simple or recommended solutions to anomalies discovered during validation are recorded with the validation

result to facilitate analysis and potential corrective action. Actual changes to the stakeholder and system/software requirements, architecture, design, or system elements are done within other Technical processes.

3) Obtain stakeholder agreement that the software system or element meets the stakeholder needs.

4) Maintain traceability of the validated system elements.

NOTE Bidirectional traceability is maintained between the validated system elements and the stakeholder requirements and record of validation results.

5) 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. This process identifies candidates for the baseline (such as the validated software system or element), and the Information Management process controls the information items. For this process, the validation strategy and validation results are typical information items that are baselined.

6.4.12 Operation process

6.4.12.1 Purpose

The purpose of the Operation process is to use the system to deliver its services.

This process establishes requirements for and assigns personnel to operate the system, and monitors the services and operator-system performance. In order to sustain services, it identifies and analyzes operational anomalies in relation to agreements, stakeholder requirements and organizational constraints.

The Operation process typically aims to control or reduce the cost of operations while sustaining an acceptable or improved level of service.

Software systems can have dedicated infrastructure, but are typically operated in distributed environments where other software systems and services (e.g., the internet) are active. The security, availability, and operational performance of the software system-of-interest are thus a matter of concern within a larger system of systems. It can include coordination with pre-existing, concurrent or continuing services delivered by other systems that provide identical or similar services.

NOTE ISO/IEC 20000-1:2011 (IEEE Std 20000-1) is a service management system standard that specifies requirements for the design, transition, delivery and improvement of managed operational services, and supports the Operation process to achieve its purpose.

6.4.12.2 Outcomes

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

a) Operation constraints that influence system/software requirements, architecture, or design are identified.

b) Any enabling systems, services, and material needed for operation are available.

c) Trained, qualified operators are available.

d) System product services that meet stakeholder requirements are delivered.

e) System product performance during operation is monitored.

f) Support to the customer is provided.

6.4.12.3 Activities and tasks

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

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

1) Define an operation strategy, including the following considerations:

i) The expected or agreed capacity, availability, response time, and security of services as they are introduced, routinely operated and withdrawn from service;

ii) The human resources strategy, depending on the need to define training and qualification requirements, train or obtain personnel to control and monitor software system operations, administer system access, and support customer service requests and user assistance;

iii) The release criteria and schedules of the software system to permit modifications that sustain existing or enhanced services;

iv) The approach to implement the operational modes in the Operational Concept, including normal operations and preparations for, and testing of, envisioned types of contingency operations;

v) Measures for operation that will provide insight into performance levels;

vi) The operational and occupational safety strategy for operators and others using or in contact with the software system during operation, accounting for safety regulations; and

vii) The environmental protection and sustainability strategy for operating the software system.

NOTE ISO/IEC 16350 *Information technology – Systems and software engineering – Application management*, provides guidance for operational aspects.

2) Identify system constraints from operation to be incorporated in changes to the system/software requirements, architecture, design, implementation, or transition.

3) Identify and plan for the necessary enabling systems or services needed to support operation.

NOTE This includes identification of operational requirements and interfaces for the enabling systems. Special modes of the operational software system, e.g., a training mode, can sometimes be active alongside or instead of a full operational mode. Enabling systems include monitoring for changes in threats to the software system.

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

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

5) Identify or define training and qualification requirements for personnel needed for software system operation.

NOTE The training and qualification includes awareness of the software system in its operational environment and a defined program of familiarization, with appropriate failure detection and isolation instruction. Operator knowledge, skill and experience requirements guide the personnel selection criteria, and where relevant, their authorization to operate is confirmed. The scope of qualification depends on the system-of-interest and its environment. For example, in some environments regulatory requirements include certification of operators, whereas in others there is no certification requirement.

6) Depending on the need for human intervention and control of operations, assign trained, qualified personnel to be operators.

NOTE With due regard for separation of duties, such as for administrative control of system access and investigation of security issues, many modern software products minimize the need for operators as distinct from end users, Operators commonly support enabling systems, such as cloud services, database and system software, security monitors, data storage, and help desk.

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

1) Use the software system in its intended operational environment.

NOTE Where agreed, continuous service capacity and quality is maintained when the software system replaces an existing system or element that is being retired.

2) Apply materials and other resources, as required, to operate the software system and sustain its services.

NOTE This includes energy sources for hardware, connectivity for software, and human or automated operators.

3) Monitor software system operation, including consideration of the following:

i) Managing adherence to the operation strategy (e.g., operational procedures);

ii) Recording and reporting significant events, such as possible breaches of software and data confidentiality and integrity;

iii) Operating the software system in a safe manner and compliant with legislated guidelines e.g., those concerning occupational safety and environmental protection; and

iv) Recording when software system or service performance is not within acceptable parameters.

NOTE This includes anomalies due to the operation strategy, the operation enabling systems, execution of the operation, or incorrect software system definition. The system sometimes exhibits unacceptable performance when system elements implemented in hardware have degraded or exceeded their useful life or the system's operational environment affects the software operation, e.g., workload above capacity thresholds, utilization by contending applications, security hacks, or software defects.

4) Consistent with the operational strategy, develop and, where feasible, automate operational procedures to minimize the risk of operational anomalies.

NOTE This includes procedures for handling routine (pre-approved) change requests and service requests, trouble-shooting and incident reporting, especially for security incidents.

5) Consistent with the operational strategy, analyze measurements to confirm that:

i) Service performance is within acceptable parameters or agreed service levels for the agreed workload;

ii) System and service availability and response times are acceptable;

iii) Cost of operation is consistent with objectives and constraints; and

iv) Potential improvements are identified and prioritized.

NOTE Operator feedback and suggestions are often useful input for improving software system operational performance. The Quality Assurance and Measurement processes can be applied.

6) Perform contingency operations, if necessary.

NOTE This includes operating the software system in a degraded mode, performing back-out and restore operation, system shutdown, implementation of work-around procedures to restore operation, or other modes for special conditions. If needed, the operator performs steps necessary to enter into contingency operations and possibly power down the system. Contingency operations are performed in accordance with pre-established procedures for such an event. Often these procedures are accompanied by a continuity plan.

c) **Manage results of operation.** This activity consists of the following tasks:

1) Record results of operation and anomalies encountered.

NOTE The Project Assessment and Control and Quality Assurance processes are used to analyze the incident and problem data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

2) Record operational incidents and problems and track their resolution.

NOTE 1 Performing incident and problem resolution is handled through the Quality Assurance and Project Assessment and Control processes. Changes to the requirements, architecture, design, or software system elements are done using other Technical processes.

NOTE 2 If an incident is experienced during operation, the operator records the incident (or is alerted to an automated notification) and performs actions prescribed in validated operating procedures to restore normal operations. Some procedures allow for provision of a temporary workaround solution until root cause analysis can be performed.

NOTE 3 During software operation, the conditions under which the problem occurred are typically documented, consistent with maintaining or restoring operational availability, so that if possible, the problem can be duplicated in a test environment and the root cause identified. Problem resolution usually involves determining the severity and impact of the problem and whether or when the problem is to be corrected or accepted for a time as a known error.

3) Maintain traceability of the operational services and configuration items.

NOTE Bidirectional traceability is maintained between the operational services and the business or mission needs, operational concept, concept of operations, and stakeholder requirements. The operational configuration items are traceable to the released versions and validated through PCA or FCA.

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

NOTE This process identifies candidates for the baseline, and the Information Management process controls the information items, such as reports on operational service performance. Key artifacts (information items) for Operations are listed in Annex B.

d) **Support the customer.** This activity consists of the following tasks:

1) Provide assistance and consultation to the customers and users to resolve complaints, incidents, problems, and service requests.

NOTE 1 Assistance and consultation includes providing or recommending sources for training, documentation, vulnerability resolution, anti-counterfeit activities, and other services supporting effective use of the software system.

NOTE 2 Customer support can include communication with customers of services, users, and other stakeholders to receive service requests and change requests, resolve complaints, and provide information on the resolution of incidents and problems.

2) Record and monitor requests and subsequent actions for support.

3) Determine the degree to which the delivered software system or services satisfy the needs of the customers and users.

NOTE The results are analyzed and the required actions to restore or amend software system or services to provide continued customer satisfaction and software system usability are identified. Wherever possible the benefit of such action is agreed with stakeholders or their representatives. The customer satisfaction data also serves as an input to the Quality Management process.

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.

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.

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.

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; and

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.

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

NOTE Reviews and analyzes 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 analyzes 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.

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 fulfill 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.

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.

6.4.14 Disposal process

6.4.14.1 Purpose

The purpose of the Disposal process is to end the existence of a system element or system for a specified intended use, appropriately handle replaced or retired elements, and to properly attend to identified critical disposal needs (e.g., per an agreement, per organizational policy, or for environmental, legal, safety, security aspects).

This process deactivates, disassembles and removes the system or any of its elements from the specific use. It addresses any waste products, consigning them to a final condition and returning the environment to its original or an acceptable condition. The waste products can be in-process resulting during any life cycle stage, e.g., waste materials during fabrication. This process destroys, stores, or reclaims system elements and waste products in an environmentally sound manner, in accordance with legislation, agreements, organizational constraints and stakeholder requirements. Disposal includes preventing expired, non-reusable, or inadequate elements from getting back into the supply chain. Where required, it maintains records in order that the health of operators and users, and the safety of the environment, can be monitored. When part of the system will continue to be in use in a modified form, the Disposal process helps ensure the proper handling of the portion being retired.

Disposal of software systems encompasses the termination of services and disposal of software elements, stored data, media and firmware, information items, and associated hardware elements that will not be reused or transitioned to another system. The Disposal process is intended to be applicable in any stage of a software systems life cycle. For software, the Disposal process applies throughout the life cycle to source code or executable

copies of the software, personally identifiable or controlled data used in the software system, and associated information items, retained under centralized configuration control or distributed for use, e.g., disposing of prototypes in early life cycle stages, and decommissioning elements replaced from modifications during utilization/deployment and support stages. When the system-of-interest is being modified for technology or capability upgrades, only the impacted elements are deactivated and removed.

NOTE The Business or Mission Analysis process and Decision Management process are typically applied to address the impact on stakeholders of system disposal and potential new system capabilities.

6.4.14.2 Outcomes

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

a) Disposal constraints are provided as inputs to requirements, architecture, design, and implementation.

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

c) The system elements or waste products are destroyed, stored, reclaimed or recycled in accordance with requirements, e.g., safety and security requirements.

d) The environment is returned to its original or an agreed state.

e) Records of disposal actions and analysis are available.

6.4.14.3 Activities and tasks

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

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

1) Define a disposal strategy for the software system, to include each system element and to identify and address critical disposal needs, including the following considerations:

i) Permanent termination of the system's functions and delivery of services, e.g., physical destruction of data storage devices, or transition of the software system elements for future reuse in modified or adapted form;

ii) Identification of ownership and responsibility for retention or destruction of data and intellectual property in the software system;

iii) Transformation of the product into, or retention in a socially and physically acceptable state, thereby avoiding subsequent adverse effects on stakeholders, society and the environment;

iv) The health, safety, security and privacy concerns applicable to disposal actions and to the long-term condition of resulting physical material and information;

v) Notification to relevant stakeholders of significant disposal activities, e.g., retirement or replacement of a system, software products or services, retirement schedule, or replacement options; and

vi) Identification of schedules, actions, responsibilities, and resources for disposal activities.

2) Identify constraints on disposal for the system/software requirements, architecture and design characteristics, or implementation techniques.

NOTE This includes access to and availability of archives or long-term storage locations and available skilled resources for system deactivation and communication with stakeholders and interface partners.

3) Identify and plan for the necessary enabling systems or services needed to support disposal.

NOTE This includes identification of requirements and interfaces for the enabling systems.

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

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

5) Specify containment facilities, storage locations, inspection criteria and storage periods, if the software system or data is to be stored, consistent with security and environmental considerations.

6) Define preventive methods to preclude disposed elements and materials that should not be repurposed, reclaimed or reused from re-entering the supply chain.

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

1) Deactivate the software system or element to prepare it for removal.

NOTE Interfaces to other systems are considered, in accordance with special procedures or instructions, and relevant health, safety, security and privacy constraints.

2) Remove the software system, its elements, its data, and non-reusable material from use or production for appropriate disposition and action.

NOTE The disposition includes reuse, recycling, reconditioning, overhaul, or destruction. The disposition and subsequent actions are conducted in accordance with relevant safety, security, privacy and environmental standards, directives and laws. Elements of the software system that have useful life remaining, either in their current condition or following modification, are transferred to other systems-of-interest or organizations. Where appropriate, consider refurbishing system elements to extend their useful life.

3) Withdraw impacted operating staff from the software system or system element and record relevant operating knowledge.

NOTE Reallocate, redeploy or retire operators. This is conducted in accordance with relevant safety, security, privacy and environmental standards, directives and laws. Act to safeguard and secure operator's knowledge and skills, using the Knowledge Management process.

4) Reuse, recycle, recondition, overhaul, archive, or destroy designated software system elements.

NOTE Handle system elements and their parts that are not intended for reuse in a manner that will assure they do not get back into the supply chain.

5) Conduct destruction of the system elements, as necessary, to reduce the amount of waste treatment or to make the waste easier to handle.

NOTE When the element is non-maintainable or non-recyclable, it is necessary to prevent the elements from getting back into the supply chain, e.g., complete erasure of all software from all system storage media and removal of license keys, data, and interfaces. This activity includes obtaining the destruction services to melt, crush, incinerate, demolish or eradicate the system or its elements as necessary.

c) **Finalize the disposal.** This activity consists of the following tasks:

1) Confirm that detrimental health, safety, security, and environmental conditions following disposal have been identified and treated.

2) Return the environment to its original state or to a state that is specified by agreement.

3) Archive information gathered through the lifetime of the product to permit audits and reviews in the event of long-term hazards to health, safety, security and the environment, and to permit future software system creators and users to build a knowledge base from experience.

Annex A
(normative)

Tailoring process

A.1 Introduction

This Annex provides requirements for the tailoring of this document.

NOTE 1 Tailoring is not a requirement for conformance to this document. In fact, tailoring is not permitted if a claim of "full conformance" is to be made. If a claim of "tailored conformance" is made, then this process is applied to perform the tailoring.

NOTE 2 Additional guidance for tailoring can be found in ISO/IEC/IEEE 24748 (all parts) on the application of life cycle processes.

A.2 Tailoring process

A.2.1 Purpose

The purpose of the Tailoring process is to adapt the processes of this document to satisfy particular circumstances or factors that:

a) surround an organization that is employing this document in an agreement;

b) influence a project that is required to meet an agreement in which this document is referenced;

c) reflect the needs of an organization in order to supply products or services.

A.2.2 Outcomes

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

a) Modified or new life cycle processes are defined to achieve the purposes and outcomes of a life cycle model.

A.2.3 Activities and tasks

If this document is tailored, then the organization or project shall implement the following tasks in accordance with applicable policies and procedures with respect to the Tailoring process, as required.

a) Identify and record the circumstances that influence tailoring. These influences include, but are not limited to:

1) stability of, and variety in, operational environments;

2) risks, commercial or performance, to the concern of interested parties;

3) novelty, size and complexity;

4) starting date and duration of utilization;

5) integrity issues such as safety, security, privacy, usability, availability;

6) emerging technology opportunities;

7) profile of budget and organizational resources available;

8) availability of the services of enabling systems;

9) roles, responsibilities, accountabilities and authorities in the overall life cycle of the system; and

10) the need to conform to other standards.

b) In the case of properties critical to the system, take due account of the life cycle structures recommended or mandated by standards relevant to the dimension of the criticality.

c) Obtain input from parties affected by the tailoring decisions. This includes, but may not be limited to:

1) the system stakeholders;

2) the interested parties to an agreement made by the organization; and

3) the contributing organizational functions.

d) Make tailoring decisions in accordance with the Decision Management process to achieve the purposes and outcomes of the selected life cycle model.

NOTE 1 Organizations establish standard life cycle models as a part of the Life Cycle Model Management process. It is sometimes appropriate for an organization to tailor processes of this document in order to achieve the purposes and outcomes of the stages of a life cycle model to be established.

NOTE 2 Projects select an organizationally-established life cycle model for the project as a part of the Project Planning process. It is sometimes appropriate to tailor organizationally adopted processes to achieve the purposes and outcomes of the stages of the selected life cycle model.

NOTE 3 In cases where projects are directly applying this document, it is sometimes appropriate to tailor processes of this document in order to achieve the purposes and outcomes of the stages of a suitable life cycle model.

e) Select the life cycle processes that require tailoring and delete selected outcomes, activities, or tasks.

NOTE 1 Irrespective of tailoring, organizations and projects are always permitted to implement processes that achieve additional outcomes or implement additional activities and tasks beyond those required for conformance to this document.

NOTE 2 An organization or project sometimes encounter a situation where there is the desire to modify a provision of this document. Modification is to be avoided because of unanticipated consequences on other processes, outcomes, activities or tasks. If necessary, modification is performed by deleting the provision (making the appropriate claim of tailored conformance) and, with careful consideration of consequences, implementing a process that achieves additional outcomes or performs additional activities and tasks beyond those of the tailored standard.

Annex B
(informative)

Examples of process information items

Table B.1 provides a possible set of work products, including artifacts, records, information items, and data stores associated with each process. This list is not all-inclusive: for each process, an organization may decide to develop a policy, plan, procedures, reports, and records, to demonstrate the outcomes or perform the activities and tasks. Where less intensive documentation is considered sufficient, information items can be combined. Also the organizational policies and procedures can be applied or tailored for each process and project. Typical item titles are shown, including common examples of alternate titles in parentheses.

NOTE See ISO/IEC/IEEE 15289 for guidance on content and management of Information Items.

Artifacts, records, record stores, and information items are usually initiated in one process and revised, enhanced, or completed in other processes. For convenience, they are listed once in this table, in a process where they are commonly initiated. When the software system artifacts and information are transformed or elaborated by another process, traceability is maintained and a traceability mapping can be produced. For example, traceability can be maintained between organizational and project processes, and between requirements, architecture and design elements, and verification cases.

Table B.1 — Sample information items by process

Process Group	Process	Typical Item Title	Item Type
Agreement processes			
	Acquisition process		
		Acquisition Plan	info
		Request for Supply (e.g., Request for Proposal, Request for Tender)	info
		Agreement Change Request	info
		Agreement (e.g., Contract)	info
		Agreement Change Management Procedure	info
		Delivery Acceptance Report	info
	Supply process		
		Supply Response (e.g., Proposal, Tender)	info
		Agreement Change Request	info
		Agreement Change Management Procedure	info
		Supply Delivery Records (for system, software, product or service)	record
Organizational Project-Enabling processes			
	Life Cycle Model Management process		
		Organizational Policies	info
		Life Cycle processes	artifact
		Life Cycle Process Description	info
		Life Cycle Model	artifact
		Organizational Procedures (Process Management Procedures)	info
		Process Assessment Report	info
		Process Improvement Plan	info
		Process Improvement Report	info
	Infrastructure Management process		
		Infrastructure Requirements	artifact
		Infrastructure Element	artifact
		Infrastructure Description	info
		Infrastructure Change Request	info
	Portfolio Management process		
		Project Portfolio	store
		Project Budget	artifact
		Project Authorization	info
	Human Resource Management process		
		Skill Needs Report	info
		Skill Development Assets (Training Materials)	artifact
		Skill Development Records (Skill Inventory, Training Record)	record

Process Group	Process	Typical Item Title	Item Type
		Qualified Personnel	artifact
		Staff Assignment Records	record
	Quality Management process		
		Quality Management Plan (Policies, Objectives)	info
		Quality Management Procedures	info
		Quality Assurance Assessment Results	record
		Corrective and Preventive Action Report (Problem Management Report)	info
	Knowledge Management process		
		Knowledge Management Plan	info
		Knowledge Management Procedures	info
		Knowledge Asset Records	record
		Knowledge Assets	artifact
Technical Management processes			
	Project Planning process		
		Project Plan (e.g., Project Technical Management Plan, Systems or Software Engineering Management Plan, Software Development Plan, Transition Plan)	info
		Work Breakdown Structure	artifact
		Resource Request	info
		Project Schedule	artifact
		Project Infrastructure and Services Requirements	artifact
	Project Assessment and Control process		
		Measurement Analysis Results and Recommendations	info
		Project Assessment Report	info
		Review Minutes	info
		Authorization to Proceed to Next Milestone	info
	Decision Management process		
		Decision Request	info
		Decision Records	record
	Risk Management process		
		Risk Management Plan	info
		Risk Profile	record
		Risk Action Request	info
	Configuration Management process		
		Configuration Management Plan	info
		Configuration Management Procedures	info
		Configuration Management Records	record
		Configuration Baseline	artifact
		CM Change/Variance Request	info
		Configuration Status Report	info
		Configuration Evaluation Report	info
		System/Software Release Report	info
	Information Management process		
		Information Item Archive	store
		Information Management Procedures	info
		Information Management Report	info
	Measurement process		
		Measurement Records	record
		Measurement Procedures	info
		Measurement Information Needs Report	info
		Measurement Report	info
	Quality Assurance process		
		Quality Assurance Procedures	info
		Quality Assurance Evaluation Report	info
		Quality Assurance Records	record
		Incident Records	record
		Problem Records	record
Technical processes			
	Business or Mission Analysis process		
		Preliminary Life Cycle Concept	artifact
		Solution Alternative Classes Assessment Report	info
	Stakeholder Needs and Requirements Definition process		
		Operational Concept	artifact
		Stakeholder Needs Assessment	info
		Stakeholder Requirements	artifact
		Stakeholder Requirements Specification	info
		Stakeholder Requirements Report	info
		Critical Performance Measures	artifact

Process Group	Process	Typical Item Title	Item Type
	System/Software Requirements Definition process		
		System or Element Description	info
		System/Software Requirements	artifact
		System/software requirements Specification	info
		Requirements Change Request	info
	Architecture Definition process		
		Architecture Viewpoints	artifact
		Architecture Views and Models (Architecture Description)	artifact
		Interface Definition (initial)	artifact
	Design Definition process		
		Design Artifact	artifact
		Design Artifacts Report (Design description)	info
		Interface Specification	info
	System Analysis process		
		System Analysis Report	info
	Implementation process		
		Software System Element	artifact
		Implementation Procedures	info
		Implementation Records (unit test results)	record
	Integration process		
		Interface Control Description	info
		Integration and Test Procedures	info
		Integrated Software System Elements (software library)	artifact
		Integration Records	record
	Verification process		
		Verified System	artifact
		Verification Procedures	info
		Verification Records	record
		Verification Report	info
	Transition process		
		Prepared Site for Operations	artifact
		Transitioned System/Software	artifact
		Transition Records	record
	Validation process		
		Validated System	artifact
		Validation Procedures	info
		Validation Records	record
		Validation Report	info
	Operation process		
		Continuity plan	info
		Operational procedures (User documentation)	info
		Operation Records	record
		Problem Report	info
		Customer Support Request	info
		Customer Support Records	record
		Operation Report	info
	Maintenance process		
		Replacement System Element	artifact
		Maintenance Procedures (Logistics Procedures)	info
		Maintenance (Logistics) Records	record
		Maintenance Requests	record
		Maintenance (Logistics) Report	info
	Disposal process		
		Disposal Records	record
		Archive Report	info

Annex C
(informative)

Process Reference Model for assessment purposes

C.1 Introduction

Some users of this document may desire to assess the implemented processes in accordance with the ISO/IEC 33000 series standards for process assessment. This annex provides a Process Reference Model (PRM) suitable for use in conjunction with those standards.

The PRM is composed of the processes in the body of this document, including the name, statement of purpose, and statement of outcomes for each process. Subclause C.3 identifies the processes in the process reference model and the clauses in which they are defined.

C.2 Conformance with ISO/IEC 33004

C.2.1 General

ISO/IEC 33004 subclause 5.3 places requirements on process reference models suitable for assessment by that standard. The following sections quote the ISO/IEC 33004 requirements for process reference models and describe how these are met by this document. In each of the following subclauses the *italicized* text quotes the requirement from the text of ISO/IEC 33004 and the non-italicized (upright) text describes the manner in which the requirement is satisfied in this document.

C.2.2 Requirements for process reference models

A Process Reference Model shall contain: [ISO/IEC 33020, 5.3.1]

a) A declaration of the domain of the process reference model. This is provided in Clause 1.

b) A description of the relationship between the process reference model and its intended context of use. This is provided by Clause 5.

c) Descriptions, meeting the requirements of [33004] 5.4 [below], of the processes within the scope of the process reference model; This is provided in Clause 6 in the description of each process.

d) A description of the relationship between the processes defined within the process reference model. This is provided in 5.6

The process reference model shall document the community of interest of the model and the actions taken to achieve consensus within that community of interest: [ISO/IEC 33020, 5.3.2]

a) the relevant community of interest shall be characterized or specified; The relevant community of interest is the users of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207.

b) the extent of achievement of consensus shall be documented; Both ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207 are international standards satisfying the consensus requirements of ISO/IEC and IEEE. (Not applicable).

c) if no actions are taken to achieve consensus, a statement to this effect shall be documented. (Not applicable).

The processes defined within a process reference model shall have unique process descriptions and identification. [ISO/IEC 33020, 5.3.3] The process descriptions are unique. The identification is provided by unique names and by the clause numbering of this annex.

C.2.3 Process descriptions

The fundamental elements of a process reference model are the descriptions of the processes within the scope of the model.

The process descriptions in the process reference model incorporate a statement of the purpose of the process, which describes at a high level the overall objectives of performing the process, together with the set of outcomes that demonstrate successful achievement of the process purpose.

A process description shall meet the following requirements:

a) a process shall be described in terms of its purpose and outcomes;

b) the set of process outcomes shall be necessary and sufficient to achieve the purpose of the process; and

c) process descriptions shall not contain or imply aspects of the process quality characteristic beyond the basic level of any relevant process measurement framework conformant with ISO/IEC 33003.

A process outcome describes one of the following: [ISO/IEC 30004:2015, 5.4]

a) production of an artifact;

b) a significant change of state; and

c) meeting of specified constraints, e.g., requirements, goals, etc.

These requirements are met by the process descriptions in Clause 6 of this document. Some outcomes may be interpreted as contributing to levels of capability above level 1 (basic) in some relevant process measurement frameworks. However, conforming implementation of the relevant processes does not require achievement of these higher levels of capability.

C.3 The process reference model

The Process Reference Model (PRM) is composed of the statement of purpose and outcomes of each of the processes included in Clause 6 of this document. The PRM for the software life cycle is composed of the set of processes in Figure 4, shown previously in 5.6.1 of this document.

Annex D
(informative)

Process integration and process constructs

D.1 Introduction

A harmonization project within ISO/IEC JTC 1/SC 7 — a parallel, coordinated revision of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207, and development of guidance in ISO/IEC/IEEE 24748 (all parts), which provides guidelines to both of these documents — is the first, large step towards an integrated set of standards describing system and software life cycles. Concepts of continual process improvement and capability assessment are now well established and recognized, and are being standardized in the ISO/IEC 33000 series of standards (replacing the ISO/IEC 15504 series). The Process Reference Models in Annex C of ISO/IEC/IEEE 15288:2015 and this document are intended to be used in conjunction with ISO/IEC 33020:2015 for capability assessment of the life cycle processes. Capability determination of processes requires that the process descriptions include a clear statement of the purpose of the process and a description of the expected outcomes. Consistent implementation of the processes is aided by having activities, tasks, and implementation notes defined. Thus, the life cycle processes in both life cycle standards have adopted common process constructs, as defined in D.2, Process constructs and their usage, and are consistent with the process definition guidance contained in ISO/IEC/IEEE TR 24774.

D.2 Process constructs and their usage

The process descriptions in this document follow clearly defined rules. First, they were grouped in a logical fashion. Those groupings are dictated by:

a) Logical relations among the processes; and

b) Responsibility for execution of the processes.

This document groups the activities that may be performed during the life cycle of the system into four Process Groups. The top-level description of these groups can be found in 5.6. Each life cycle process within those groups is described in terms of its purpose and desired outcomes and lists the activities and tasks that need to be performed to achieve those outcomes.

c) Agreement processes – two processes (6.1);

d) Organizational Project-Enabling processes – six processes (6.2);

e) Technical Management processes – eight processes (6.3); and

f) Technical processes – fourteen processes (6.455).

Consistent application of process description rules allows for the normalized clause numbering. Within this document a subclause numbered as 6.x denotes a process group and 6.x.y denotes a process within that group. Subclauses numbered as 6.x.y.1 describe the purpose of a process, subclauses numbered 6.x.y.2 describe the outcomes of a process, and subclauses numbered as 6.x.y.3 describe the activities and tasks of a process.

Figure D.1 is a representation of process constructs used in this document and in ISO/IEC/IEEE 15288:2015. These are Process, Activity, Task, and Note.

A process requires a name, purpose and at least one outcome. Each process has at least one activity. A set of processes, with their statements of purpose and outcomes, constitute a Process Reference Model (PRM).

An activity is a construct of related tasks for producing outcomes, Activities provide a means to organize related tasks within the process, to improve understanding and communication of the process. If an activity is cohesive enough, it can be converted to a lower level process by drafting a purpose and a set of outcomes.

A task is a detailed provision for implementation of a process. It may be presented as a requirement (“shall”), a recommendation (“should”) or a permission (“may”).

Notes are used to explain the intent or mechanics of a process, activity, or task. Notes provide insight regarding potential implementation or areas of applicability, such as lists, examples, and other considerations.

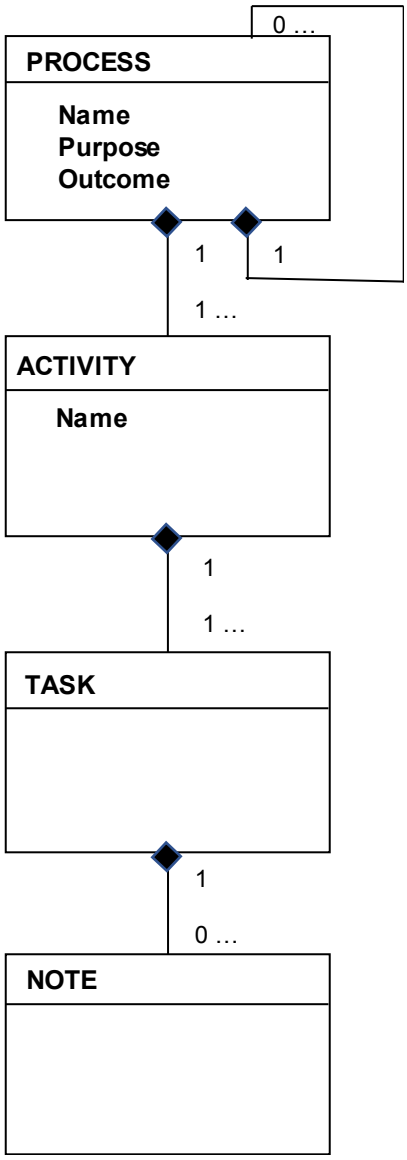

Figure D.1 — ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process constructs

Annex E
(informative)

Process views

E.1 Introduction

A process view allows those representing a particular engineering interest to see gathered in a single place the set of process activities that directly and succinctly address their concern. For such interests, a process view can be developed to organize processes, activities, and tasks selected from ISO/IEC/IEEE 15288 or ISO/IEC/IEEE 12207 to provide a focus to their particular concern in a manner that cuts across all or parts of the life cycle. This annex provides sample process viewpoints that may be used to define process views in these instances.

E.2 The process view concept

There may be cases where a unified focus is needed for activities and tasks that are selected from disparate processes to provide visibility to a significant concept or thread that cuts across the processes employed across the life cycle. It is useful to advise users of this document how to identify and define these activities for their use, even though they cannot locate a single process that addresses their specific concern.

For this purpose, the concept of a process view has been formulated. Like a process, the description of a process view includes a statement of purpose and outcomes. Unlike a process, the description of a process view does not include activities and tasks. Instead, the description includes guidance explaining how the outcomes can be achieved by employing the activities and tasks of the various processes in ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207. Process views can be constructed using the process viewpoint template found in E.3.

E.3 Process viewpoint

A process view conforms to a process viewpoint. The process viewpoint provided here can be used to create process views.

1) The Process viewpoint is defined by:

    1) its stakeholders: users of this document; and

    2) the concerns it frames: the processes needed to reflect a particular engineering interest.

2) The contents of resulting process views should include:

    1) process view name;

    2) process view purpose;

    3) process view outcomes; and

    4) identification and description of the processes, activities and tasks that implement the process view, and references to the sources for these processes, activities and tasks in other standards.

NOTE The requirements for documenting viewpoints are found in ISO/IEC/IEEE 42010:2011, 5.4. This description is consistent with those requirements.

E.4 Process view for specialty engineering

This section provides an example of applying the process viewpoint to yield a process view for specialty engineering, intended to illustrate how a project can assemble processes, activities and tasks of this document to provide focused attention to the achievement of product characteristics that have been selected as being of special interest.

This example treats the cluster of interests, generally called specialty engineering, which includes but is not limited to such areas as availability, maintainability, reliability, safety, security, human factors, and usability. Within this document, these "ilities" requirements are referred to as "critical quality characteristics". These characteristics determine how well the product meets its specified requirements in a specific area selected for focus. See E.6 for a process view relating to (information security assurance for software systems).

NOTE 1 This is a generalized instance of a process view that covers a broad set of functional and non-functional characteristics related to specialty engineering. It provides a broad view across the processes. If a specific critical quality characteristic has a high priority relative to other characteristics, a specific process view can be created for that characteristic, including more detailed information and requirements.

NOTE 2 ISO/IEC 25030, *Software Engineering — Software product quality requirements and evaluation (SQuaRE) — Quality requirements*, can be consulted in specifying software product quality requirements.

NOTE 3 *INCOSE Systems Engineering Handbook* contains descriptions and elaboration about many of the specialty engineering areas and the associated critical quality characteristics.

Name: **Specialty Engineering Process View**

Purpose: The purpose of the Specialty Engineering Process View is to provide objective evidence that the system achieves satisfactory levels of certain critical quality characteristics selected for special attention.

Outcomes:

a) Product critical quality characteristics are selected for special attention.

b) Requirements for the achievement of the critical quality characteristics are defined.

c) Measures for the requirements are selected and related to the desired critical quality characteristics.

d) Approaches for achieving the desired critical quality characteristics are defined and implemented.

e) The extent of achievement of the requirements is continuously monitored.

f) A satisfactory level of the critical quality characteristics is specified and achieved.

The outcomes permit the possibility that the desired critical quality characteristics cannot be directly measured, but instead might be argued and inferred based on other product or process characteristics that can be measured. Measurements can be used to show compliance with established standards. The acquirer and supplier come to agreement on particular standards to be used for this compliance verification.

Specialty Engineering processes, activities and tasks

This process view can be implemented using the following processes, activities, and tasks from this document.

a) The Project Assessment and Control process (6.3.2) provides for monitoring the extent of achievement of the requirements and critical quality characteristics and communicating the results to stakeholders and managers. Relevant activities and tasks include b)6), 7), 9) and 10).

b) The Decision Management process (6.3.3) provides assessment of alternative requirements, architecture characteristics and design characteristics against the decision criteria, including the critical quality characteristics. Results of these comparisons are ranked, via a suitable selection model, and are then used to decide on an optimal solution. Relevant activities and tasks include b) (all tasks); and c)1).

c) The Risk Management process (6.3.4), in its entirety, provides for identifying, evaluating, and handling risks of the system, including those related to meeting the critical quality characteristics.

d) The Information Management process (6.3.6), in its entirety, provides for the specification, development and maintenance of information items for documenting and communicating the extent of achievement. Information items used for the purpose of critical quality characteristics are sometimes specialized in nature. Sources for the description of these information items include industry associations, regulators, and specific standards.

e) The Measurement process (6.3.7), in its entirety, provides for defining an approach that relates measures to the required critical quality characteristics.

f) The Quality Assurance process (6.3.8) addresses identified anomalies (incident and problems) that relate to the achievement of critical quality characteristics.

g) The Business and Mission Analysis process (6.4.1) provides for the definition of the problem space and characterization of the solution space, including the relevant trade-space factors and preliminary life cycle concepts. This includes developing an understanding of the context and any key parameters, such as the critical quality characteristics, e.g., security threats, safety hazards, human interfaces, operational characteristics, and system assurance context. Relevant activities and tasks include b)1) and 2); c)1); and d)1).

h) The Stakeholder Needs and Requirements Definition process (6.4.2) provides for the selection and definition of characteristics, including critical quality characteristics, and associated information items. The activities and the documentation are useful in identifying, prioritizing, defining, and recording requirements for the critical quality characteristics. Relevant activities and tasks include a)1) and 2); b)2),3) and 4); c)1) and 2); d) all tasks; and e)2).

i) The System/Software Requirements Definition process (6.4.2) provides for the specification of parameters for the critical quality characteristics and the selection of measures for tracking the achievement of these requirements with respect to the specific system to be developed. Relevant activities and tasks include a)1); b) all tasks; and c)2).

j) The Architecture Definition process (6.4.4) provides for the identification of stakeholder concerns from an architecture perspective. These concerns often translate into expectations or constraints across the life cycle stages that relate to the critical quality characteristics, such as utilization e.g., availability, security, effectiveness, usability; support, e.g., reparability, obsolescence management; evolution of the system and of the environment, e.g., adaptability, scalability, survivability; production, e.g., manufacturability, testability; retirement, e.g., environmental impact, transportability. This process further addresses those critical quality characteristic requirements that drive the architecture decisions, including the assessment of the architecture with respect to the concerns and associated characteristics. Relevant activities and tasks include a)2) and 4); b)1); c)2), 3), 4), and 5); d)1); and e)2).

k) The Design Definition process (6.4.5) provides for the determination of necessary design characteristics, which includes the critical quality characteristics, such as security of design criteria for the specialty characteristics and the evaluation of alternative designs with respect to those criteria. Relevant activities and tasks include a)2); b)1), 2), 3) 4), and 6); and c)2).

l) The System Analysis process (6.4.6) provides for the level of analysis needed to understand the trade space with respect to the critical quality characteristics through the conduct of mathematical analysis, modeling, simulation, experimentation, and other techniques. The analysis results are input to trades made through the Decision Management process in support of other Technical processes. Relevant activities and tasks include a) (all tasks); and b) (all tasks).

m) The Implementation process (6.4.7) provides for recording the evidence that critical quality requirements have been met. Relevant activities and tasks include b)3).

n) The Integration process (6.4.8) provides for planning the integration, including the considerations for critical quality characteristics, and the assurance that the achievement of the characteristics is determined and recorded. Relevant activities and tasks include a)1); b)3); and c)1).

o) The Verification process (6.4.9), provides for the planning and execution of a strategy to perform verification, including the critical quality characteristics. The selected verification strategy can introduce design constraints that affect the achievement of the characteristics. Relevant activities and tasks include a)1) and 3); b)1), 2); and c)1) and 2).

p) The Transition process (6.4.10) provides for installing the system in its operational environment. Because some specialty properties involve a trade-off between design constraints and operational constraints, attention to installation is often important. Relevant activities and tasks include a)4); and b) 4), 6), and 7).

q) The Validation process (6.4.11) provides evidence that the services provided by the system meet the stakeholders' needs, including the critical quality characteristics. Relevant activities and tasks include a)1) and 3); b)1) and 2); c)1) and 2).

r) The Operation process (6.4.12) provides for usage of the system. Assuring that critical quality characteristics are appropriately achieved involves monitoring the operation of the system. Relevant activities and task include b)3) and 4); c)1) and 2); and d)1) and 2).

s) The Maintenance process (6.4.13) sustains the capabilities of the system, particularly its ongoing availability to provide its functions, including its critical quality characteristics. This includes failure analysis, maintenance tasks, and logistics tasks needed to assure continued operation of the system. Relevant activities and tasks include b) all tasks; c) all tasks; and d)1) and 2).

t) The Disposal process (6.4.14) ends the existence of a system. The inherent need to anticipate disposal can place constraints on development. In fact, these constraints can be critical quality characteristics. Relevant activities and tasks include a)2); b)1) and 2) and c)3).

E.5 Process view for interface management

This section provides an example of applying the process viewpoint to yield a process view for interface management, intended to illustrate how a project can assemble processes, activities and tasks of this document to provide focused attention to the achievement of product characteristics that have been selected as being of special interest.

This example treats a specific instance of a process view, called interface management, which includes interface definition, design, and change management. Within this document, the tasks that comprise interface management are fully contained within the existing processes.

NOTE For software systems, interfaces with other systems are a typical way of receiving input data or exporting output data (reports). Interfaces with external systems and services allow the software system to function in its operating environment and with enabling systems. The GUI is a special interface for human interaction with the software system.

Name: **Interface Management Process View**

Purpose: The purpose of the Interface Management Process View is to facilitate identification, definition, design and management of interfaces of the software system.

Outcomes:

a) Business or mission needs related to interfaces are identified.

b) Stakeholder needs related to interfaces are identified

c) Requirements for the interfaces are defined.

d) Interfaces between software system elements are identified and defined.

e) Interfaces between the software system and external systems are identified and defined.

f) The extent of realization of the interface requirements is continuously monitored.

Interface Management processes, activities and tasks

This process view can be implemented using the following processes, activities, and tasks from this document.

NOTE *INCOSE Systems Engineering Handbook* contains descriptions and elaboration about interface management.

a) The Project Assessment and Control process (6.3.2**Error! Bookmark not defined.**) provides for monitoring the extent of achievement of the requirements, including interfaces, and communicating the results to stakeholders and decision makers. Relevant activities and tasks include b)6), 7), 9) and 10).

b) The Decision Management process (6.3.3) provides assessing alternative requirements, architecture characteristics and design characteristics against the decision criteria, including the interfaces. Results of these comparisons are ranked, via a suitable selection model and are then used to decide on an optimal solution. Relevant activities and tasks include b) all tasks); and c)1).

c) The Risk Management process (6.3.4), in its entirety, provides for identifying, evaluating, and handling risks of the system, including those related to interfaces.

d) The Configuration Management process (6.3.5) provides for the identification and control of interfaces. It includes management of interface specifications and interface control documents. Internal and external interface requirements and changes are documented in accordance with the project's CM strategy, which is commonly represented in a CM plan. Relevant activities include b)1) and d)1).

e) The Information Management process (6.3.6), in its entirety, provides for the specification, development and maintenance of information items for documenting interfaces and their operational performance.

f) The Measurement process (6.3.7), in its entirety, provides for defining an approach that relates measures to the required interface information needs, and then generating and using those measures to address the identified interface information needs.

g) The Quality Assurance process (6.3.8), in its entirety, addresses identified anomalies (incidents and problems) that relate to the achievement of interface requirements.

h) The Business and Mission Analysis process (6.4.1) provides for the definition of the problem space and characterization of the solution space, including the description of the environment and context, as well as preliminary operational concepts, including software system interfaces and enabling system interfaces. It often identifies external systems that interface with the system-of-interest. Relevant activities and tasks include b)1) and 2); and c)1).

i) The Stakeholder Needs and Requirements Definition process (6.4.2) provides for the identification of stakeholders concerned with interfaces, definition of operational concepts and the interactions of the system with users, existing interfaces to be transitioned, and the intended environment (including other systems). It often identifies external systems that interface with the system-of-interest. Relevant activities and tasks include a)1), c)1) and 2), and d)1) and 3).

j) The System/Software Requirements Definition process (6.4.3) provides for the definition of the system boundary and the interface requirements. Relevant activities and tasks include a)1); b)3) and 4); c) (all tasks), and d) 1) and 3).

k) The Architecture Definition process (6.4.4) provides for the identification of interfaces from an architecture perspective as the architecture models evolve. This process further describes and defines the interfaces to the extent needed for the architecture description. Relevant activities and tasks include a)2); c)1) through 4); d) 2); and f)2) and 6).

l) The Design Definition process (6.4.5) provides for the refinement and full definition of the interfaces and the creation of the information items to specify the interface characteristics and protocols. Relevant activities and tasks include b)2), 5) and 6); c3), and d)2).

m) The System Analysis process (6.4.6) provides for the level of analysis needed to balance interface architecture, design constraints, interface performance requirements, and operational performance measurements, through the conduct of modeling, simulation, and other techniques. The analysis results are input to the Decision Management process in support of other Technical processes. Relevant activities and tasks include a) (all tasks); and (b) (all tasks).

n) The Implementation process (6.4.7) provides for development of the interfaces and recording the evidence that interface requirements for an implemented system element have been met. Relevant activities and tasks include b) 1) and 3).

o) The Integration process (6.4.8) provides for planning the integration, including the considerations for interfaces between software system elements and with external systems. It also includes the integration of systems or system elements and interfaces and confirming the interfaces in the integrated software system. Relevant activities and tasks include a)1), 2), and 5); b) all tasks; and c)1).

p) The Verification process (6.4.9), provides evidence that the services provided by the system meet the system requirements, including the interface requirements. The process provides for the planning and execution of a strategy to perform verification, including the interface requirements. The selected verification strategy may introduce interface constraints that affect their implementation. Relevant activities and tasks include a)1) and 3); b)1), 2); and c)1).

q) The Transition process (6.4.10) provides for planning and migrating the software system or elements and data sets into a different environment using interfaces, and establishing the use of interfaces in the new system. This includes identifying constraints, and checking the installation, activation and operational state of the interfaces. Relevant activities and tasks include a)1) and 3); b) 3), 4), 6), and 7).

r) The Validation process (6.4.11) provides evidence that the services provided by the software system meet the stakeholders' needs, including the interface requirements. The selected validation strategy may introduce interface constraints that affect their implementation. Validation involves communication with stakeholder representatives of the interfacing systems and services. Relevant activities and tasks include a)4); b)1) and 2); c)1) and 2).

s) The Operation process (6.4.12) provides for usage of the software system. There also may be constraints to the interfaces for operations. Confirming that the interface requirements are appropriately achieved involves monitoring the operation of the system, identifying and performing corrective action when interfaces do not function properly, and supporting contact with interface partners (customers). Relevant activities and task include a)1) and 2), b) 1), 3) and 4); c)1) and 2), and d (all tasks).

t) The Maintenance process (6.4.13) sustains the capabilities of the software system, including its interfaces, and their ongoing availability to provide their functions. This includes problem analysis and maintenance tasks, and logistics tasks needed to assure continued and timely operation. Relevant activities and tasks include a)2); b) (all tasks); and d)1), 2), and 3).

u) The Disposal process (6.4.14) ends the existence of a system or an interface. It involves activities to disengage and discontinue interfaces. Relevant activities and tasks include a)2) and 3); and b)1), 2), and 6).

E.6 Process view for software assurance (Information security)

This section provides an example of applying the process viewpoint to yield a process view for software assurance, intended to illustrate how a project can assemble processes, activities and tasks of this document to provide focused attention to the achievement of software assurance characteristics that have been selected as being of special interest. The software assurance characteristics, their extent of achievement, and related information may support a software assurance claim, as described in ISO/IEC/IEEE 15026.

This example is focused on protection against intentional subversion or forced failure due to the software architecture, design, or implementation, especially construction of the code.

Name: **Software Assurance Process View**

Purpose: The purpose of the Software Assurance Process View is to provide objective evidence that the software achieves satisfactory levels of certainty that sufficient protection is achieved against intentional subversion or forced failure due to the software architecture, design, or construction of the code.

NOTE In terms appropriate for conformance to ISO/IEC/IEEE 15026, the assurance claims regarding the software characteristics selected for special attention are achieved and information is provided showing the achievement of those claims.

Outcomes:

a) Product software assurance characteristics are selected for special attention.

b) Requirements for the achievement of the software assurance characteristics are defined.

c) Measures for the requirements are selected and related to the desired software assurance characteristics.

d) Approaches for achieving the desired software assurance characteristics are defined and implemented.

e) The extent of achievement of the requirements is continuously monitored.

f) A satisfactory level of the critical software assurance characteristics is specified and achieved.

The outcomes permit the possibility that the desired software assurance characteristics or claim cannot be directly measured but instead might be inferred based on other product or process characteristics that can be measured. Measurements may be used to show compliance with established standards. The acquirer and supplier agree on particular standards to be used for this compliance verification.

Software Assurance processes, activities and tasks

This process view can be implemented using the following processes, activities, and tasks from this document.

a) The Agreement processes (6.1) provide for the establishment of expectations and responsibilities related to software assurance, including legal agreements and licensing requirements, the protection of organizations' assets that are accessible by suppliers, the possibility of compromise during delivery, detection of anomalies, detection of counterfeits in the application element at arrival, expectations for defect resolution, patch management, service level agreements, and safeguards against supply chain threats. Relevant activities and tasks include in the Acquisition process: a)1) and 2) and c)1); and in the Supply Process, c)1), e)1 and 2).

b) The Life Cycle Model Management process (6.2.1) provides for the establishment and maintenance of software assurance policies and procedures including a software security development lifecycle and the controlled use of outsourced code. Relevant activities and tasks include a)2 and 3), b)1, and c) all tasks.

c) The Infrastructure Management process (6.2.2), in its entirety, provides for secure development and operational environments, software assurance tools, timely patch management, and code libraries that are properly maintained and improved based on lessons learned from the projects, organization and industry.

d) The Human Resource Management process (6.2.4), in its entirety, provides for relevant screening of employees and suppliers with access to the software or software development environment and defines specific training related to software assurance based on roles and responsibilities across the life cycle.

e) The Knowledge Management process (6.2.6) provides for collecting and maintaining knowledge and informing the project about changes to the threat landscape, evolutions in software assurance practices, mitigations for software vulnerabilities, lessons learned from incidents and incident responses, and evolutions in software assurance testing tools. Relevant activities and tasks include b) (all tasks) and d)3).

f) The Decision Management process (6.3.3), in its entirety, assesses alternative requirements, architecture characteristics and design characteristics against the decision criteria, including the software assurance characteristics. Results of these comparisons are ranked and recorded along with the selection model, and are then used to decide on an optimal solution. Stakeholders can make decisions based on the justification provided.

g) The Risk Management process (6.3.4), in its entirety, provides for the evaluation of the potential for not being able to achieve the necessary application security, resulting in a risk to the users including the interface partners, or resulting in the software not being used as intended. Software security is a risk category in each risk analysis.

h) The Configuration Management process (6.3.5) provides for the establishment and maintenance of the integrity of all identified outputs of a project or process, including the monitoring of assets and security of designated storage systems, and makes them available to authorized parties. This includes records of changes made to the software and software releases, as well as control and audit of all accesses to the software items that handle security functions. Relevant activities and tasks include a)1), d)1), and f)2).

i) The Information Management process (6.3.6), in its entirety, provides the information about the achievement of software assurance to the relevant stakeholders, including regulatory or approval authorities. The quantifiable information about the software is collected to support a set of arguments that justify a claim about the software assurance of a system. Due to the sensitivity of the data, additional care is given to identify the appropriate audiences for the various assurance measures. Information management includes the protection of sensitive information items regarding software assurance.

j) The Measurement process (6.3.7), in its entirety, provides a common platform for collecting information about the software assurance claims, strategies, and evidence, sometimes referred to as an assurance case.

k) The Quality Assurance process (6.3.8) evaluates project and supplier processes for conformance to software assurance requirements and procedures. It addresses identified anomalies (incident and problems) that relate to the achievement of software assurance characteristics. Relevant activities and tasks include c) and e) (all tasks).

l) The Business and Mission Analysis process (6.4.1) provides for understanding the operating environment for the software system under development and laws, policies, risks, and constraints related to software assurance. Relevant activities and tasks include b)1) and c)1).

m) The Stakeholder Needs and Requirements Definition process (6.4.2) provides for the selection and definition of risks and threats to missions or information. It incorporates this knowledge in defining requirements relating to software assurance (information security), including confidentiality, availability and integrity in the context of how the software is intended to function; and in consideration of misuse and abuse scenarios. Stakeholders need to agree upon what aspects of software assurance are sufficient. Relevant activities and tasks include a)2); b)1) and 2); and c) (all tasks).

n) The System/Software Requirements Definition process (6.4.3) provides for the selection and definition of software assurance (information security) related requirements, including confidentiality, availability and integrity in the context of how the software is intended to function; and requirements for software integrity in the event of misuse and abuse. Relevant activities and tasks include b)3) and 4); and c) 3).

o) The Architecture Definition process (6.4.4) provides for the identification of stakeholder concerns from an architecture viewpoint through threat modeling and assessing the vulnerability of the product architecture and design to potential attacks to gain an understanding of the threat landscape and the relevant architecture elements. Relevant activities and tasks include a)2) and 4); b)1); c) (all tasks), d)5), and f)10, 20, and 5).

p) The Design Definition process (6.4.5) provides for the determination of necessary design characteristics, which include attack surface reduction, including the location of components; software assurance design patterns; and avoidance of anti-patterns. Relevant activities and tasks include a)2) and 3); b)2),3), and 6); c)2) and d)1).

q) The Implementation process (6.4.7) provides for the use of secure coding practices to avoid common coding errors that lead to exploitable product vulnerabilities, and the use of a variety of testing techniques including inspection for competent authenticity, fuzz testing, static analysis testing, and dynamic testing to identify and address software weaknesses and vulnerabilities. Relevant activities and tasks include a)1); and b)1), 2) 3) and 4).

r) The Integration process (6.4.8) provides for planning the integration, including the considerations for software assurance characteristics, and the assurance that the achievement of the characteristics is determined and recorded. Implementation of interface standards wherever practical promotes system and element sustainability and element reusability. Relevant activities and tasks include a)2) and 5); b)3); and c)1) and 2).

s) The Verification process (6.4.9) provides for the planning and execution of a strategy to verify that software assurance requirements, including the software assurance characteristics, have been achieved. The selected verification strategy can introduce testing for code weaknesses in the development process or during sustainment. Threat analysis provides input into the creation of test plans and cases. The results include the information required to effect the remedial actions that correct nonconformances in the software or the processes that act on it and account for uncertainty in verification activities, such as test tool reliability and level of the uncertainty in results (i.e., rates of false positives and false negatives).

Additional considerations include the resiliency of the software to function. Relevant activities and tasks include a)1), 4), 5) and 6: b) (all tasks); c)1) and 2).

t) The Transition process (6.4.10) provides for installing the software system or element in a different environment. Because some software assurance properties involve a trade-off between design constraints and operational constraints, attention to installation and user documentation is often important. Relevant activities and tasks include a)2); b)1) and 4), 5), and 6); and c)1) and 2).

u) The Validation process (6.4.11) provides evidence that the services provided by the software system meet the stakeholders' needs, including the software assurance characteristics. Validation methods include vulnerability scans, code assessment and validation using a variety of tools and techniques such as static code analysis, dynamic code analysis, binary code analysis, code coverage tools, stress testing, and the use of tools to gather evidence of changes resulting from remote maintenance activities. Additional considerations include the resiliency of the software to function under use and abuse cases. Relevant activities and tasks include, a)1), 3, and 4), b (all tasks), and c)2) and 3).

v) The Operation process (6.4.12) provides for usage of the software system. Assuring that software assurance characteristics are appropriately achieved involves monitoring the operation of the system to deliver its services in its intended environment and provides support to the customers of the software product. Plans for this process consider achievement of application security throughout the life of the system, operational restrictions such as access control, and consistency of assumptions made during earlier stages regarding application security with the operational environment. The process includes establishing reporting systems and procedures for investigation and disposition of application security-related incidents. Relevant activities and tasks include a)1), b) (all tasks); c)1), 2), and 3); and d)1).

w) The Maintenance process (6.4.13) sustains the capabilities of the software system, particularly its ongoing availability to provide its software assurance characteristics. This includes failure analysis, maintenance tasks, and logistics tasks needed to assure continued operation of the system. The maintenance process provides for evaluating the effect on software assurance from changes made to the software during maintenance and maintaining appropriate evidence for software assurance. It includes a documented process for patching and remediating software, detecting and removing or inactivating unauthorized and malicious software, and a process for informing an acquirer or interface partner of notification and remediation mechanisms. Remediation of vulnerabilities is prioritized based on a variety of factors, including risk. Documented development and sustainment practices are followed when implementing software remediation. Relevant activities and tasks include a)1); b) (all tasks); c) (all tasks); and d)1) and 2).

x) The Disposal process (6.4.14) ends the existence of a software system. The inherent need to anticipate disposal can place constraints on development and management of data contained in the software system. In fact, these constraints can be software assurance characteristics. Relevant activities and tasks include a)1), 2), and 5); b) (all tasks); and c)1) and 3).

Annex F
(informative)

Software system architecture modelling

F.1 Introduction

In this document, architecture and design activities are described as separate processes. Several iterations of the Architecture Definition, Design Definition, and Implementation processes can be involved when evolving a software element's architecture e.g., to determine whether an entity or function will be realized through integration of existing software, adaptive reuse, or newly constructed software. In the systems and software engineering communities dealing with complex systems, architecture can be followed by different designs for different systems in different product lines. In this case it is important to perform these two processes in a separate manner. Furthermore, architecture is often done for other reasons than as the immediate basis for design, such as to drive technology investments, to achieve consistency or reduce complexity in an organization's product line or portfolio of projects, or to guide acquirer-supplier decisions.

The architecture of a software system can be understood as a set of structured architectural entities and their relationships, chosen to achieve characteristics such as interoperability, scalability, environmental resilience, encapsulation, availability, affordability, robustness, execution efficiency, or mission effectiveness (fitness for use). Software system architecture deals with relationships among a variety of entities, such as scenarios, functions, function flows, interfaces, resource flow items, information or data elements, objects, physical components and environments, containers, nodes, links, communication resources, constraints, equations and parametric models.

This Annex describes some of the models (model kinds) that are used in creating and evaluating the architecture of software systems.

NOTE ISO/IEC/IEEE 15288:2015, Annex F describes how models and views are applied to the architecture of systems in general, and has additional information relating to views and modelling of the architecture of physical products, such as mass models and layout models.

F.2 Views, models and model kinds used in software system architecture

The Architectural Definition process uses a variety of models for software systems, including the example models listed in the following section. Model kinds specify the languages, notations, conventions, modelling techniques, analytical methods or other operations to be used on models of that kind. (Traditional system engineering practice classifies some of these models as "logical models" or "physical models", but the taxonomical distinction is unnecessary in the application of this document.) A variety of views are used to represent how the system architecture addresses stakeholder concerns. Views are composed of models. For example, a logical view of software can represent business processes in its functions; a process view can represent the events and transformations occurring within different states of the software and can include concurrency and timing concerns; a structural view represents the different system components, which can be associated with physical or virtual system elements, an information view represents the relationships between data elements contained in and transformed by the software.

Refer to ISO/IEC/IEEE 42010 for definitions of architecture terms and additional detail on architecture concepts and models.

F.2.1 Functional model

A functional model of the system is a representation of a set of functions that defines the transformations of inputs into outputs performed by the system to achieve its mission or purpose. These functions are determined by how the system is expected to behave when used as intended. Consequently, every system function is associated with an interaction between the system and its environment. Functional, performance, non-functional, and constraint requirements are usually analyzed to determine functions and input-output flows. When functions are associated with system elements, the design definition process will need to determine if each system/software element has

been sufficiently specified to build or buy it. If the system element is further resolved in order to achieve this sufficiency, then the functions associated with the system element are also further resolved and properly associated with the sub-elements. Typically, there are multiple ways to decompose the functions that contribute to the definition of multiple candidate architectures.

F.2.2 Static model

A static model describes the structure of a software system. In object-oriented programming, it is represented through a set of objects (classes) and their relationships (inheritance, association, and dependency), depicted as nodes and links.

F.2.3 Data model

A data model (semantic or information model) represents the data elements and their relationships and properties (attributes) that the software system will handle. Logical data models use schema to reflect structural relationships between data entities which can be implemented in databases. Data models reflect different types of data (text, graphics, geographic data, images, general objects) and their use in the system functions (frequency of change, data volume, use in searching) as well as the logical relationships among data elements. Data models are applied in the development of interfaces and software services, data analysis, and data reporting. Physical data models reflect the schema for storage and retrieval of data records.

F.2.4 Behavioral model

A behavioral model (dynamic model) is an arrangement of functions and interfaces (internal and external) that defines how the system or its elements act under conditions to sustain the operational scenarios, including the execution sequencing, synchronization, and concurrency, the conditions for behavioral change and the performance. Behavioral models are applicable to software control systems. A behavioral model can be described with a set of interrelated scenarios. This includes identifying the behavioral elements (e.g., modes/states, transitions, trigger events, and operational scenarios) through the life cycle.

F.2.5 Temporal model

A temporal model of the system is a representation that expresses how the time is taken into account in the behavior the system or its elements that presents levels of execution frequency of functions (e.g., strategic level, tactical level, operational monitoring level, regulation level) corresponding to levels of decision that enable humans and program logic to monitor and control the system operations. This includes identifying temporal elements (e.g., duration, frequency, response time, triggers, timeout, stop conditions), from the operational concept and system requirements.

F.2.6 Structural model

A structural model of the system is a representation that shows the arrangement of elements with respect to each other and where necessary shows the interfaces between elements and with external entities. Such a model enables consolidating or identifying physical interfaces between system elements in a level of the system hierarchy and between levels of the system hierarchy, as well as those with external entities to the concerned system (in its environment/context). Structural models can be hierarchical decompositions or object-oriented.

F.2.7 Network model

A network model defines an arrangement of nodes and links to help understand how resources (e.g., information and people) traverse from one node to another. A network model can be used to determine constraints such as throughput, latency, and congestion points. A network model is sometimes modelled along with a protocol stack to understand how layers in a network interact vertically up and down the stack.

F.3 Other model considerations

Stakeholder life cycle concerns, such as maintenance, evolution, disposal, potential changes of environment, obsolescence management, and other non-functional requirements, are addressed by defining architectural characteristics such as modularity, relative independence, scalability, upgradability, adaptation to several

environments, level of effectiveness, reliability, robustness, and resilience. Other necessary models can include some of these characteristics or other critical quality characteristics. For example, a software assurance case, regarded as a model, can help in deducing potential architectural mitigations to minimize operational risks (mission loss due to exploited security vulnerabilities) related to critical concerns and functions.

Determination of which models to use in system definition can be based on examination of stakeholder concerns. The models and the resulting views can be used to express how the system architecture and design addresses their concerns and to gain better understanding of their actual needs, wants and expectations.

Furthermore, models can be used in other life cycle processes besides architecture and design definition. Model-Based Systems Engineering (MBSE) is the formalized application of modelling to support system requirements, architecture, design, analysis, and verification and validation activities throughout the life cycle.

NOTE A verification and validation model defines representations of test information, which can support the verification of the architecture. Verification and validation models can generate test analyzes, data, cases, and other information.

Annex G
(informative)

Application of software life cycle processes to a system of systems

G.1 Introduction

A system of systems (SoS) is a system-of-interest (SOI) whose elements are themselves systems. A SoS brings together a set of systems for a task that none of the systems can accomplish on its own. Each constituent system keeps its own management, goals, and resources while coordinating within the SoS and adapting to meet SoS goals. In the context of terminology discussed in 5.2.3 (as shown in Figure 3), the composite set of systems including the original SOI, enabling systems and interacting systems together constitute an SoS. Where there are concerns that affect the composite set, the system of systems becomes the SOI, which is considered to satisfy some business or mission objective that cannot be satisfied by the individual constituent systems, or to understand emergent behavior of the combination.

This annex addresses the application of system life cycle processes to such SoS. It describes general characteristics, the common types of SoS, and the implications throughout the life cycle.

G.2 SoS characteristics and types

SoS are characterized by managerial and operational independence of the constituent systems, which in many cases were developed and continue to support originally identified users concurrently with users of the SoS. In other contexts, each constituent system itself is a SOI; its existence often predates the SoS, while its characteristics were originally engineered to meet the needs of their initial users. As constituents of the SoS, their consideration is expanded to encompass the larger needs of the SoS. This implies added complexity, particularly when the systems continue to evolve independently of the SoS. The constituent systems also typically retain their original stakeholders and governance mechanisms, which limits alternatives to address the needs of the SoS.

SoS have been characterized into four types based on the governance relationships between the constituent systems and the SoS (Table G.1). The strongest governance relations apply to directed system of systems, where the SoS organization has authority over the constituent systems despite the fact that the constituent systems were not originally engineered to support the SoS. Somewhat less control is afforded for acknowledged SoS, where allocated authority between the constituent systems and the systems of systems has an impact on application of some of the systems engineering processes. In collaborative SoS, which lack system of systems authorities, application of systems engineering depends on cooperation among the constituent systems. Virtual systems of systems are largely self-organizing and limit opportunity for systems engineering of the SoS.

Table G.1 — System of Systems types

Type	Characteristic
Virtual	• Decentralized management authority • No explicit, centrally agreed-upon purpose • Emerging behaviors that rely on relatively invisible mechanisms for continuity
Collaborative	• Constituent systems interact voluntarily to fulfill agreed-upon purposes • Collectively decide how to interoperate, enforcing and maintaining standards
Acknowledged	• Recognized objectives, a designated manager and resources for the SoS • Constituent systems retain their independent ownership, management, and resources
Directed	• Integrated SoS built and managed to fulfill specific purposes • Centrally managed and evolved • Constituent systems maintain ability to operate independently • Normal operational mode is subordinated to central purpose

A key characteristic of SoS is *emergence* — the unanticipated effects at the SoS level attributed to the complex interaction dynamics of the constituent systems. However, in SoS, constituent systems are intentionally considered in their combination, so as to obtain and analyze outcomes not possible to obtain with the systems alone. The complexity of the constituent systems and the fact they perhaps were designed without regard to their role in the SoS, can result in new, unexpected behaviors. Identifying and addressing unanticipated emergent results is a particular challenge in engineering SoS.

NOTE Some of the largest software SoS, such as the Internet, are virtual SoS in which the constituent systems are engineered to follow common recommendations and communication protocols. Virtual SoS can exhibit beneficial emerging behaviors such as redundancy, dynamic reconfiguration, collaboration and resilience.

G.3 SE processes applied to systems of systems

G.3.1 General

The above characteristics of SoS have implications on the application of each of the four types of system life cycle processes.

G.3.2 Agreement processes

Agreement processes are crucial for SoS because they establish the modes of developmental and operational control among the organizations responsible for the SoS and the often independent constituent systems. Constituent systems, which are acquired and managed by different organizations, sometimes hold original objectives that do not align with those of the SoS. Except in the directed SoS case, the SoS organization cannot task a constituent system organization without their cooperation. In an acknowledged or collaborative SoS, these tasks are balanced against the tasks of the constituent system as a SOI in its own right. For virtual SoS, agreement processes may be informal, or considered only for analysis purposes.

Even in agreements among owners of constituent systems, there is still an acquirer and a supplier. A system owner can be both an acquirer and a supplier for another constituent system.

G.3.3 Organizational project enabling processes

In a typical system-of-interest, Organizational Project-Enabling processes establish the environment in which projects are conducted. The organization establishes the processes and life cycle models to be used by projects; establishes, redirects, or cancels projects; provides resources required, including human and financial; and sets and monitors the quality measures for systems and other deliverables that are developed by projects for internal and external customers (6.2).

In an SoS, the owners of the constituent systems usually retain responsibility for engineering their systems and they each have their own Organizational Project-Enabling processes. Depending on the SoS type, the SoS also applies these Organizational Project-Enabling processes to the particular considerations of the SoS: planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into a SoS capability.

Consequently, in SoS these Organizational Project-Enabling Processes are implemented at two levels. The organizations responsible for the constituent systems implement these processes for their SOI independent of the SoS. The SoS organization (or in collaborative systems of systems by agreement of the SoS) implement these processes for the SoS for those considerations that apply to the overall SoS. For example, Human Resource Management is addressed by each constituent system organization for the engineering of their system. The SoS organization addresses Human Resources Management only for the systems engineering activities that apply across the constituent systems to the SoS.

A particular challenge in SoS engineering is the lack of alignment among the constituent system Organizational Project-Enabling Processes and those of the SoS. Constituent systems processes are designed to meet their own outcomes and sometimes do not align with those of the SoS. For example, Portfolio Management can be a constituent system responsibility in cases where the constituent system organization has full control over the constituent system and other systems and projects in its portfolio, distinct from the portfolio management of the SoS organization.

G.3.4 Technical management processes

In a typical system-of-interest, Technical Management processes are concerned with managing the resources and assets allocated by organization management and with applying them to fulfill the agreements into which the organization or organizations enter. They relate to the management of projects, in particular to planning in terms of cost, timescales and achievements, to the checking of actions for compliance with plans and performance criteria, and to the identification and selection of corrective actions that recover shortfalls in progress and achievement. They are used to establish and perform technical plans for the project, manage information across the technical team, assess technical progress against the plans for the system products or services, control technical tasks through to completion, and to aid in the decision-making process (6.3).

The Technical Management processes are also implemented at the level of the SoS and that of the constituent systems. Technical Management processes are applied to the particular considerations of SoS engineering — planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into a system-of-systems capability. In parallel, the constituent systems organizations retain responsibility for engineering their systems and for their own Technical Management processes.

The SoS organization addresses the Technical Management processes as they apply across the SoS, while the processes are also implemented independently in the constituent system organizations. For configuration management, for instance, constituent systems manage their own configurations while the SoS addresses configuration management as it applies to the mix of systems in the SoS. Risk is managed by the constituent systems based on assessment of risk as it applies to their outcomes, while the SoS risk management looks at risk to the SoS.

Planning and Assessment and Control (6.3) are key to all management practices; a key challenge in SoS engineering is the lack of control by the SoS organization over the processes for the constituent systems (particularly for acknowledged and collaborative SoS). Driven by its own organizational requirements, each of the constituent systems can be on a development or upgrade schedule that differs from the schedules of other constituent systems. The SoS organization plans an integrated life cycle that recognizes the independent changes in the constituent systems, in addition to the SoS-initiated changes in a life cycle that treats the SoS as the SOI. This often involves the definition of stable intermediate forms that punctuate the SoS evolution with incremental capabilities added among the constituent systems.

G.3.5 Technical processes

Technical processes are concerned with technical actions throughout the life cycle. They transform the needs of stakeholders first into a product and then, by applying that product, provide a sustainable service, when and where needed in order to achieve customer satisfaction. The Technical processes are applied in order to create and use a system, whether it is in the form of a model or is a finished product, and they apply at any level in a hierarchy of system structure (6.4).

As with the other processes when applied to SoS, Technical processes are implemented for both the SoS and constituent systems; in some cases, the SoS implementation is by means of conduct of the constituent system processes rather than for the SoS as a whole.

Business or Mission Analysis for an SoS looks across the full SoS business and mission environment. To the degree the constituent system was developed to operate in that space, the Business or Mission Analysis for the systems of system and constituent systems will be largely shared. The objective is to determine the best means to provide the desired capability.

Stakeholder Needs and Requirements Definition focuses on the top level SoS, but also considers how the disparate needs of the stakeholders for the individual systems can lead to constraints on the SoS.

System/Software Requirements Definition for the SoS tends to be defined at the level needed to satisfy stakeholder needs and mission objectives, to be translated into requirements for the constituent systems with the SoS serving as “stakeholder” for new requirements for the constituent systems.

The architecture for the SoS is a framework for organizing and integrating the capabilities of a mix of existing and new systems into a SoS capability, leaving the architectures of the constituent systems to their organizations. Because the constituent systems in an SoS usually predate the SoS, SoS Architecture Definition often begins with the de facto architecture of the SoS. Architecture alternatives are then examined in order to frame stakeholder

concerns and meet top level SoS requirements, and to recognize the effect of new requirements for the constituent systems and accommodate the constituent system architecture constraints.

The Design Definition process provides sufficient detailed data and information to enable the SoS implementation. This involves collaboration with the constituent systems organizations who will conduct their own design trades to identify the approach to address SoS requirements as they apply to their system. These are the responsibility of the constituent system organizations. Implementation is done by the constituent systems with the SoS organization in a monitoring role.

Integration, Verification, Transition, and Validation are done by the constituent systems for the changes they implement to support requirements generated by the SoS. These processes also apply to the SoS when the upgraded constituent systems are integrated into the SoS and performance is verified and validated. The independent and asynchronous nature of constituent systems in an SoS poses challenges to effective implementation of these processes as implemented in a traditional SOI. In some cases, the SoS-level evaluations can only be performed in the operational environment, in which case precautionary measures should be considered to avoid adverse SoS-behavior.

Finally, the Operations, Maintenance and Disposal processes tend to be implemented by the constituent systems, given their management and operational independence. SoS-level interactions can facilitate interoperability and reduce duplicate effort for those processes.

Annex H
(informative)

Application of Agile

This document is intended to be applied in organizations and software projects using agile approaches and methods as well as in those using other formal engineering approaches. Agile development is one of the most widely used approaches for software development (including software maintenance) because it is believed to be more affordable and to deliver usable products more quickly. In large software development efforts as well as in smaller projects, many Agile methods can be used with various life cycle models, and different methods can be used at different points in the lifecycle. This annex points out interpretations of the process requirements in this document that are appropriate for commonly used agile techniques.

As discussed in 5.4.2, the life cycle models used in agile projects are often highly incremental and evolutionary. However, organizations that use agile methods do apply the life cycle processes identified in this document, including organizational, technical management, and technical processes (and may operate under the agreement processes). As stated in 5.4.1, this document does not prescribe any particular sequence of processes within the life cycle model. The sequence of the processes is determined by project objectives and by selection of the life cycle model. An agile project, because it transforms or combines activities while creating or improving working software, may find it more appropriate to claim full conformance to outcomes (4.2.1) rather than full conformance to activities and tasks (4.2.2).

Agile development succeeds in part because of the nature of software, which accommodates flexibility in design while the software is being constructed. In agile practice, software design, implementation (construction), and continuous integration are frequently performed concurrently. This practice is contrasted with a formal top-down approach to traceability in which construction cannot begin until design is approved so that the constructed software is traced to a previously approved detailed design. Thus, agile projects take full advantage of the approach in this document in which processes occur concurrently, as contrasted with sequentially staged (idealized waterfall) projects.

In agile projects, concept exploration, development, construction, testing, transition, and retirement of previous software can be performed concurrently for successive iterations. Agile projects often perform replanning concurrently with the activities mentioned above. In such approaches, using the end of a stage as a management checkpoint or control is not very useful. Other agile approaches perform replanning at points between designated iterations (e.g., sprints or pre-defined time-boxed cadences) so that each of these iterations can be treated as a stage.

Agile projects can have closely tied cycles for development and software release (e.g., with daily, weekly, monthly scheduled releases) or can separate completion of development iterations from management of scheduled software releases, for the convenience of the customer or according to organizational strategy.

Besides applying a highly iterative and evolutionary life cycle model, agile organizations have specific practices for the Project Planning and Project Assessment and Control processes. Rather than establishing major control points at the transition between stages or processes, agile projects often hold less formal checkpoints or retrospective reviews at the end of a time-boxed cycle to agree on improvements for the next cycle. Each iteration includes design, development, and test activities (test-driven development). After a sprint of approximately one to four weeks or longer, new working software elements are accepted as “done” — completely developed, verified (tested) and validated. Lessons learned and process improvements are identified, and work begins on another sprint. Continuing learning, risk management, and process improvement can be facilitated by planning meetings that initiate each iteration and retrospective meetings held at the end of each iteration.

Agile methods emphasize the Stakeholder Needs and Requirements Definition process, facilitating change through a high degree of ongoing stakeholder involvement. In agile projects, key stakeholders, such as the acquirer or user representatives, are not just approvers of information, measurements, and evaluation reports. Continuous stakeholder involvement is consistent with this document, which identifies the involvement of stakeholders in every technical process as well as in the Tailoring process (Annex A). They are closely involved in requirements management (6.4.3.3.d) at each iteration, by bringing new requirements and changes in priorities and participating when prioritized requirements are selected from a backlog of undeveloped stories or functions and further refined for development. The iterative approach encourages flexibility to add, reprioritize, or defer

requirements which are recognized as being within the general scope of the project. Also, stakeholder involvement in the approval of tested software in each iteration means that validation is continual throughout the project.

With the incremental definition of evolving requirements, the concept of project scope differs in an agile project from projects where scope is defined by a predetermined baseline of specified requirements. When an Agile project requires a defined product, its scope is initially tied to high-level or fundamental requirements. More detailed levels of product definition are expected to emerge as additional knowledge is gained throughout construction. Agile work without pre-defined products (e.g., level of effort maintenance) may control scope through time-boxed schedules or resource-limited teams. This approach is particularly appropriate for software maintenance efforts, where the extent or content of corrective or adaptive work is not fully specified initially.

Specification of baselines also differs in degree and timing for agile development projects compared to more traditional development efforts. The baseline of requirements may initially comprise high-level user stories ("epics") and key performance measurements, including standards for usability. Agile projects take full advantage of the task to "Define baselines through the life cycle" (6.3.5.3.b)4). During development, new baselines are agreed upon and built at least daily under configuration control. A software element is generally traceable to a high-level functional requirement and closely traceable to the use case it implements and the test case used to verify its functionality and performance. Rather than being traceable to a previously approved, baselined design document, a new software element may simply be traced to a design element or object that was, in fact, created during the construction of the software and only then placed under configuration control.

The preparation of specifications, design artifacts, and information items or documentation during agile projects is often limited, while software developers apply their time and skills to transform a scenario or narrative of a function ("user story") into a working, testable software element. Rather than preparing elaborate review packages for briefing at infrequent major milestone reviews, the team meets with stakeholders frequently to present informal evidence of completing a set of functions and to agree on the content of the next iteration. Documented information items focus on what will be needed for transition, operation and maintenance, such as operator and end-user documentation and baselines of tested and released versions of software with test plans and test cases. Projects reuse organizational procedures for configuration and release management, verification, and incident and problem management. Where possible, bidirectional traceability is enabled and enforced by integrated automated systems and procedures for requirements management, architecture and design, configuration management, measurement, and information management.

The incremental and iterative nature of Agile development can facilitate efficient technical and management processes and practices to reduce the cost associated with change. In comparison, projects managed at the waterfall end of the continuum seek to reduce total rework cost by minimizing the number of changes, limiting the number of control points, and baselining detailed specifications which are reviewed and traced throughout the project.

Agile projects for complex systems attempt to manage cost by prioritizing the most important capabilities for early realization. If an organization manages the development and maintenance of its entire portfolio of software systems as a single system, managed by spend rate rather than total spending, then the organization can, in principle, manage that portfolio of systems as a continuing agile development, analogous to managing a highly iterative "Kanban" maintenance effort.

Annex I
(informative)

Process Mapping from ISO/IEC/IEEE 12207:2008

This document adopts a process model that is identical with ISO/IEC/IEEE 15288:2015. To enable traceability and ease transition for users of the previous (2008) edition of ISO/IEC/IEEE 12207, Table I.1 presents a cross-reference of processes that shows the primary process alignments between the two versions. Table I.1 does not imply that process definitions are identical between the 2008 and the 2017 editions of this document. In some cases, processes in the earlier version are now addressed through activities and tasks or described in notes. Other, more detailed mappings to align process purpose, outcomes, activities, or tasks of the two versions are possible. One such mapping is depicted in Table I.2.

Table I.1 — Comparison of processes in ISO/IEC/IEEE 12207:2017 and the previous edition

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process		ISO/IEC 12207:2008 (IEEE Std 12207:2008) process	
Subclause	**Title**	**Subclause**	**Title**
6	Software life cycle processes	6	System Life Cycle Processes
		7	Software Specific Processes
6.1	Agreement processes	6.1	Agreement Processes
6.1.1	Acquisition process	6.1.1	Acquisition Process
6.1.2	Supply process	6.1.2	Supply Process
6.2	Organizational project-enabling processes	6.2	Organizational Project-Enabling Processes
6.2.1	Life cycle model management process	6.2.1	Life Cycle Model Management Process
6.2.2	Infrastructure management process	6.2.2	Infrastructure Management Process
6.2.3	Portfolio management process	6.2.3	Project Portfolio Management Process
6.2.4	Human resource management process	6.2.4	Human Resource Management Process
6.2.5	Quality management process	6.2.5	Quality Management Process
6.2.6	Knowledge management process	6.2.4.2.e	Human Resource Management Process Knowledge Management activity
		6.2.4.3.4	Knowledge management (activity)
		7.3	Software Reuse Processes
		7.3.1	Domain Engineering Process
		7.3.2	Reuse Asset Management Process
		7.3.3	Reuse Program Management Process
6.3	Technical management processes	6.3	Project Processes
		7.2	Software Support Processes
6.3.1	Project planning process	6.3.1	Project Planning Process
6.3.2	Project assessment and control process	6.3.2	Project Assessment and Control Process
		7.2.6	Software Review Process
6.3.3	Decision management process	6.3.3	Decision Management Process
6.3.4	Risk management process	6.3.4	Risk Management Process
6.3.5	Configuration management process	6.3.5	Configuration Management Process
		7.2.2	Software Configuration Management Process

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process		ISO/IEC 12207:2008 (IEEE Std 12207:2008) process	
Subclause	**Title**	**Subclause**	**Title**
6.3.6	Information management process	6.3.6	Information Management Process
		7.2.1	Software Documentation Management Process
6.3.7	Measurement process	6.3.7	Measurement Process
6.3.8	Quality assurance process	7.2.3	Software Quality Assurance Process
		7.2.7	Software Audit Process
		7.2.8	Software Problem Resolution Process
6.4	Technical processes	6.4	Technical Processes
6.4.1	Business or mission analysis process		NA, [but related to some outcomes of 6.4.1 Stakeholder needs and requirements definition process]
6.4.2	Stakeholder needs and requirements definition process	6.4.1	Stakeholder Requirements Definition Process
6.4.3	System/Software requirements definition process	6.4.2	System Requirements Analysis Process
		7.1.2	Software Requirements Analysis Process
6.4.4	Architecture definition process	6.4.3	System Architectural Design Process
		7.1.3	Software Architectural Design Process
6.4.5	Design definition process	6.4.3	System Architectural Design Process
		7.1.4	Software Detailed Design Process
6.4.6	System analysis process		NA, [but analysis activities are found in many processes, especially Decision Management]
6.4.7	Implementation process	6.4.4	Implementation Process
		7.1	Software Implementation Processes
		7.1.1	Software Implementation Process
		7.1.5	Software Construction Process
6.4.8	Integration process	6.4.5	System Integration Process
		7.1.6	Software Integration Process
6.4.9	Verification process	6.4.6	System Qualification Testing Process
		7.1.7	Software Qualification Testing Process
		7.2.4	Software Verification Process
6.4.10	Transition process	6.2.4	Human Resource Management Process
		6.4.7	Software Installation Process
6.4.11	Validation process	6.4.8	Software Acceptance Support Process
		7.2.5	Software Validation Process
6.4.12	Operation process	6.4.9	Software Operation Process
6.4.13	Maintenance process	6.4.10	Software Maintenance Process
6.4.14	Disposal process	6.4.11	Software Disposal Process
Annex A	Tailoring process	Annex A	Tailoring Process

Table I.2 provides a mapping of outcomes of processes in this document with outcomes of selected software-specific implementation, reuse and support processes in ISO/IEC/IEEE 12207:2008. Users of 12207:2008 can identify outcomes of this document related to previous versions of lower level software-specific processes, activities or tasks for implementation, support or reuse of software elements in the software system. Table I.2 does not imply that process outcomes are identical between the 2008 and the 2017 editions of this document. Not all process outcomes of this document are explicitly identified in the previous edition.

Table I.2 — Comparison of process outcomes in ISO/IEC/IEEE 12207:2017 and software-related outcomes in the previous edition

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.2.6	**Knowledge Management process**	**7.3.1**	**Domain Engineering process**
6.2.6.2.a	A taxonomy for the application of knowledge assets is identified.	7.3.1.2.a, 7.3.1.2.b, 7.3.1.2.c, 7.3.1.2.d	The representation forms for the domain models and the domain architectures are selected. The boundaries of the domain and its relationships to other domains are established. A domain model that captures the essential common and different features, capabilities, concepts, and functions in the domain is developed. A domain architecture describing the family of systems within the domain, including their commonalities and variabilities, is developed.
6.2.6.2.b	The organizational knowledge, skills, and knowledge assets are developed or acquired.	7.3.1.2.e, 7.3.1.2.f	Assets belonging to the domain are specified. Assets belonging to the domain are acquired or developed and maintained throughout their life cycles.
6.2.6.2.c	The organizational knowledge, skills, and knowledge assets are available.	7.3.1.2.g	The domain models and architectures are maintained throughout their life cycles.
		7.3.2	**Reuse Asset Management process**
6.2.6.2.a	A taxonomy for the application of knowledge assets is identified.	7.3.2.2.a, 7.3.2.2.b	An asset management strategy is documented. An asset classification scheme is established.
6.2.6.2.b	The organizational knowledge, skills, and knowledge assets are developed or acquired.	7.3.2.2.c	Criteria for asset acceptance, certification and retirement are defined.
6.2.6.2.c	The organizational knowledge, skills, and knowledge assets are available.	7.3.2.2.d, 7.3.2.2.f, 7.3.2.2.g	An asset storage and retrieval mechanism is operated. Changes to the assets are controlled. Users of assets are notified of problems detected, modifications made, new versions created and deletion of assets from the storage and retrieval mechanism.
6.2.6.2.d	Knowledge management usage data is gathered and analyzed.	7.3.2.2.e	The use of assets is recorded.
		7.3.3	**Reuse Program Management process**
6.2.6.2.a	A taxonomy for the application of knowledge assets is identified.	7.3.3.2.a, 7.3.3.2.b	The organization's reuse strategy, including its purpose, scope, goals and objectives, is defined. The domains for potential reuse opportunities are identified.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.2.6.2.b	The organizational knowledge, skills, and knowledge assets are developed or acquired.	7.3.3.2.c, 7.3.3.2.d, 7.3.3.2.e	The organization's systematic reuse capability is assessed. The reuse potential of each domain is assessed. Reuse proposals are evaluated to ensure the reuse product is suitable for the proposed application.
6.2.6.2.c	The organizational knowledge, skills, and knowledge assets are available.	7.3.3.2.f	The reuse strategy is implemented in the organization.
6.2.6.2.d	Knowledge management usage data is gathered and analyzed.	7.3.3.2.g, 7.3.3.2h	Feedback, communication, and notification mechanisms that operate between affected parties are established. The reuse program is monitored and evaluated.
6.3.1	**Project Planning process**	**7.2.6**	**Software Review process**
6.3.1.2.a	Objectives and plans are defined.	7.2.6.2.a	Management and technical reviews are held based on the needs of the project.
6.3.1.2.b	Roles, responsibilities, accountabilities, and authorities are defined.	7.2.6.2.a	Management and technical reviews are held based on the needs of the project.
6.3.1.2.c	Resources and services necessary to achieve the objectives are formally requested and committed.	7.2.6.2.a	Management and technical reviews are held based on the needs of the project.
6.3.1.2.d	Plans for the execution of the project are activated.	7.2.6.2.a	Management and technical reviews are held based on the needs of the project.
6.3.2	**Project Assessment and Control process**	**7.2.6**	**Software Review process**
6.3.2.2.a	Performance measures or assessment results are available.	7.2.6.2.c, 7.2.6.2.e	Review results are made known to all affected parties. Risks and problems are identified and recorded.
6.3.2.2.b	Adequacy of roles, responsibilities, accountabilities, and authorities is assessed.	7.2.6.2a 7.2.6.2.b	Management and technical reviews are held based on the needs of the project. The status and products of an activity of a process are evaluated through review activities.
6.3.2.2.c	Adequacy of resources is assessed.	7.2.6.2.a 7.2.6.2.b	Management and technical reviews are held based on the needs of the project. The status and products of an activity of a process are evaluated through review activities.
6.3.2.2.d	Technical progress reviews are performed.	7.2.6.2.a	Management and technical reviews are held based on the needs of the project.
6.3.2.2.e	Deviations in project performance from plans are investigated and analyzed.	7.2.6.2.b	The status and products of an activity of a process are evaluated through review activities.
6.3.2.2.f	Affected stakeholders are informed of project status.	7.2.6.2.c	Review results are made known to all affected parties.
6.3.2.2.g	Corrective action is defined and directed, when project achievement is not meeting targets.	7.2.6.2.d	Action items resulting from reviews are tracked to closure.
6.3.4	**Risk Management process**	**7.2.6**	**Software Review process**

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.3.4.2.a	Risks are identified.	7.2.6.2.e	Risks and problems are identified and recorded.
6.3.5	**Configuration Management process**	**7.2.2**	**Software Configuration Management process**
6.3.5.2.a	Items requiring configuration management are identified and managed.	7.2.2.2.a, 7.2.2.2.b	A software configuration management strategy is developed. Items generated by the process or project are identified, defined and baselined.
6.3.5.2.b	Configuration baselines are established.	7.2.2.2.b	Items generated by the process or project are identified, defined and baselined.
6.3.5.2.c	Changes to items under configuration management are controlled.	7.2.2.2.c, 7.2.2.2.d	Modifications and releases of the items are controlled. Modifications and releases are made available to affected parties.
6.3.5.2.d	Configuration status information is available.	7.2.2.2.e	The status of the items and modifications are recorded and reported.
6.3.5.2.e	Required configuration audits are completed.	7.2.2.2.f	The completeness and consistency of the items is ensured.
6.3.5.2.f	System releases and deliveries are controlled and approved.	7.2.2.2.c, 7.2.2.2.d, 7.2.2.2.g	Modifications and releases of the items are controlled. Modifications and releases are made available to affected parties. The storage, handling and delivery of the items are controlled.
6.3.6	**Information Management process**	**7.2.1**	**Software Documentation Management process**
6.3.6.2.a	Information to be managed is identified.	7.2.1.2.a, 7.2.1.2.c	A strategy identifying the documentation to be produced during the life cycle of the software product or service is developed. Documentation to be produced by the process or project is identified.
6.3.6.2.b	Information representations are defined.	7.2.1.2.b, 7.2.1.2.d	The standards to be applied for the development of the software documentation are identified. The content and purpose of all documentation is specified, reviewed and approved.
6.3.6.2.c	Information is obtained, developed, transformed, stored, validated, presented, and disposed of.	7.2.1.2.d, 7.2.1.2.e, 7.2.1.2.f	The content and purpose of all documentation is specified, reviewed and approved. Documentation is developed and made available in accordance with identified standards. Documentation is maintained in accordance with defined criteria.
6.3.6.2.d	The status of information is identified.	7.2.1.2.f	Documentation is maintained in accordance with defined criteria.
6.3.6.2.e	Information is available to designated stakeholders.	7.2.1.2.e	Documentation is developed and made available in accordance with identified standards.
6.3.8	**Quality Assurance process**	**7.2.3**	**Software Quality Assurance process**

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.3.8.2.a	Project quality assurance procedures are defined and implemented.	7.2.3.2.a, 7.2.3.2.d	A strategy for conducting software quality assurance is developed. Adherence of software products, processes and activities to the applicable standards, procedures and requirements is verified.
6.3.8.2.b	Criteria and methods for quality assurance evaluations are defined.	7.2.3.2.a	A strategy for conducting software quality assurance is developed.
6.3.8.2.c	Evaluations of the project's products, services, and processes are performed, consistent with quality management policies, procedures, and requirements.	7.2.3.2.d	Adherence of software products, processes and activities to the applicable standards, procedures and requirements is verified.
6.3.8.2.d	Results of evaluations are provided to relevant stakeholders.	7.2.3.2.b	Evidence of software quality assurance is produced and maintained.
6.3.8.2.f	Prioritized problems are treated.	7.2.3.2.c	Problems and/or non-conformance with requirements are identified and recorded.
		7.2.6	**Software Review process**
6.3.8.2.a	Project quality assurance procedures are defined and implemented.	7.2.6.2a	Management and technical reviews are held based on the needs of the project.
6.3.8.2.b	Criteria and methods for quality assurance evaluations are defined.	7.2.6.2a	Management and technical reviews are held based on the needs of the project.
6.3.8.2.c	Evaluations of the project's products, services, and processes are performed, consistent with quality management policies, procedures, and requirements.	7.2.6.2.a, 7.2.6.2.b	Management and technical reviews are held based on the needs of the project. The status and products of an activity of a process are evaluated through review activities.
6.3.8.2.d	Results of evaluations are provided to relevant stakeholders.	7.2.6.2.c	Review results are made known to all affected parties.
6.3.8. 2.f	Prioritized problems are treated.	7.2.6.2.d, 7.2.6.2.e	Action items resulting from reviews are tracked to closure. Risks and problems are identified and recorded.
		7.2.7	**Software Audit process**
6.3.8.2.a	Project quality assurance procedures are defined and implemented.	7.2.7.2.a	An audit strategy is developed and implemented.
6.3.8.2.b	Criteria and methods for quality assurance evaluations are defined.	7.2.7.2.a	An audit strategy is developed and implemented.
6.3.8.2.c	Evaluations of the project's products, services, and processes are performed, consistent with quality management policies, procedures, and requirements.	7.2.7.2.b, 7.2.7.2.c	Compliance of selected software work products and/or services or processes with requirements, plans and agreement is determined according to the audit strategy. Audits are conducted by an appropriate independent party.
6.3.8.2.d	Results of evaluations are provided to relevant stakeholders.	7.2.7.2.d	Problems detected during an audit are identified and communicated to those responsible for corrective action, and resolution.
		7.2.8	**Software Problem Resolution process**
6.3.8.2.a	Project quality assurance procedures are defined and implemented.	7.2.8.2.a	A problem management strategy is developed.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.3.8.2.b	Criteria and methods for quality assurance evaluations are defined.	7.2.8.2.a	A problem management strategy is developed.
6.3.8.2.d	Results of evaluations are provided to relevant stakeholders.	7.2.8.2.f	The status of all problems reported is known.
6.3.8.2.e	Incidents are resolved.		
6.3.8.2.f	Prioritized problems are treated.	7.2.8.2.b, 7.2.8.2.c, 7.2.8.2.d, 7.2.8.2.e	Problems are recorded, identified and classified. Problems are analyzed and assessed to identify acceptable solution(s). Problem resolution is implemented. Problems are tracked to closure.
6.4.3	**System/Software Requirements Definition process**	**7.1.2**	**Software Requirements Analysis process**
6.4.3.2.a	The system or element description, including interfaces, functions and boundaries, for a system solution is defined.	7.1.2.2.a	The requirements allocated to the software elements of the system and their interfaces are defined.
6.4.3.2.b	System/software requirements (functional, performance, process, non-functional, and interface) and design constraints are defined.	7.1.2.2.a, 7.1.2.2.e, 7.1.2.2.f, 7.1.2.2.h	The requirements allocated to the software elements of the system and their interfaces are defined. Prioritization for implementing the software requirements is defined. The software requirements are approved and updated as needed. The software requirements are baselined and communicated to all affected parties.
6.4.3.2.c	Critical performance measures are defined.	7.1.2.2.c	The impact of software requirements on the operating environment is understood.
6.4.3.2.d	The system/software requirements are analyzed.	7.1.2.2.b, 7.1.2.2.c, 7.1.2.2.g	Software requirements are analyzed for correctness and testability. The impact of software requirements on the operating environment is understood. Changes to the software requirements are evaluated for cost, schedule and technical impact.
6.4.3.2.e	Any enabling systems or services needed for system/software requirements definition are available.		
6.4.3.2.f	Traceability of system/software requirements to stakeholder requirements is developed.	7.1.2.2.d	Consistency and traceability are established between the software requirements and system requirements.
6.4.4	**Architecture Definition process**	**7.1.3**	**Software Architectural Design process**
6.4.4.2.a	Identified stakeholder concerns are addressed by the architecture.	7.1.3.2.a, 7.1.3.2.c	A software architectural design is developed and baselined that describes the software items that will implement the software requirements. Consistency and traceability are established between software requirements and software design.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.4.4.2.b	Architecture viewpoints are developed.	7.1.3.2.a	A software architectural design is developed and baselined that describes the software items that will implement the software requirements.
6.4.4.2.c	Context, boundaries, and external interfaces of the system are defined.	7.1.3.2.b	Internal and external interfaces of each software item are defined.
6.4.4.2.d	Architecture views and models of the system are developed.	7.1.3.2.a	A software architectural design is developed and baselined that describes the software items that will implement the software requirements.
6.4.4.2.e	Concepts, properties, characteristics, behaviors, functions, or constraints that are significant to architecture decisions of the system are allocated to architectural entities.	7.1.3.2.a	A software architectural design is developed and baselined that describes the software items that will implement the software requirements.
6.4.4.2.f	System elements and their interfaces are identified.	7.1.3.2.a, 7.1.3.2.b	A software architectural design is developed and baselined that describes the software items that will implement the software requirements. Internal and external interfaces of each software item are defined.
6.4.4.2.g	Architecture candidates are assessed.	7.1.3.2.a	A software architectural design is developed and baselined that describes the software items that will implement the software requirements.
6.4.4.2.h	An architectural basis for processes throughout the life cycle is achieved.	7.1.3.2.a	A software architectural design is developed and baselined that describes the software items that will implement the software requirements.
6.4.4.2.i	Alignment of the architecture with requirements and design characteristics is achieved.	7.1.3.2.a, 7.1.3.2.c	A software architectural design is developed and baselined that describes the software items that will implement the software requirements. Consistency and traceability are established between software requirements and software design.
6.4.4.2.j	Any enabling systems or services needed for architecture definition are available.		
6.4.4.2.k	Traceability of architecture elements to stakeholder and system/software requirements is developed.	7.1.3.2.c	Consistency and traceability are established between software requirements and software design.
		7.3.1	**Domain Engineering process**
6.4.4.2.c	Context, boundaries, and external interfaces of the system are defined.	7.3.1.2.b	The boundaries of the domain and its relationships to other domains are established.
6.4.4.2.d	Architecture views and models of the system are developed.	7.3.1.2.a, 7.3.1.2.c	The representation forms for the domain models and the domain architectures are selected. A domain model that captures the essential common and different features, capabilities, concepts, and functions in the domain is developed.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.4.4.2.h	An architectural basis for processes throughout the life cycle is achieved.	7.3.1.2.d	A domain architecture describing the family of systems within the domain, including their commonalities and variabilities, is developed.
6.4.5	**Design Definition process**	**7.1.4**	**7.1.4 Software Detailed Design process**
6.4.5.2.a	Design characteristics of each system element are defined.	7.1.4.2.a	A detailed design of each software component, describing the software units to be built, is developed.
6.4.5.2.b	System/software requirements are allocated to system elements.	7.1.4.2.a, 7.1.4.2.c	A detailed design of each software component, describing the software units to be built, is developed. Consistency and traceability are established between the detailed design and the requirements and architectural design.
6.4.5.2.c	Design enablers necessary for design definition are selected or defined.	7.1.4.2.a	A detailed design of each software component, describing the software units to be built, is developed.
6.4.5.2.d	Interfaces between system elements composing the system are defined or refined.	7.1.4.2.b	External interfaces of each software unit are defined.
6.4.5.2.e	Design alternatives for system elements are assessed.	7.1.4.2.a	A detailed design of each software component, describing the software units to be built, is developed.
6.4.5.2.f	Design artifacts are developed.	7.1.4.2.a	A detailed design of each software component, describing the software units to be built, is developed.
6.4.5.2.g	Any enabling systems or services needed for design definition are available.		
6.4.5.2.h	Traceability of the design characteristics to the architectural entities of the system architecture is established.	7.1.4.2.c	Consistency and traceability are established between the detailed design and the requirements and architectural design.
6.4.7	**Implementation process**	**7.1.1**	**Software Implementation Process**
6.4.7.2.a	Implementation constraints that influence the requirements, architecture, or design are identified.	7.1.1.2.a, 7.1.1.2.b	An implementation strategy is defined. Implementation technology constraints on the design are identified.
6.4.7.2.b	A system element is realized.	7.1.1.2.c	A software item is realized.
6.4.7.2.c	A system element is packaged or stored.	7.1.1.2.d	A software item is packaged and stored in accordance with an agreement for its supply.
6.4.7.2.d	Any enabling systems or services needed for implementation are available.		
		7.1.5	**Software Construction Process**
6.4.7.2.a	Implementation constraints that influence the requirements, architecture, or design are identified.	7.1.5.2.a, 7.1.5.2.d	Verification criteria are defined for all software units against their requirements. Verification of the software units against the requirements and the design is accomplished.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.4.7.2.b	A system element is realized.	7.1.5.2.b, 7.1.5.2.d	Software units defined by the design are produced. Verification of the software units against the requirements and the design is accomplished.
6.4.7.2.e	Traceability is established.	7.1.5.2.c	Consistency and traceability are established between software units and requirements and design.
6.4.8	**Integration process**	**7.1.6**	**Software Integration Process**
6.4.8.2.a	Integration constraints that influence system requirements, architecture, or design, including interfaces, are identified.	7.1.6.2.a	An integration strategy is developed for software units consistent with the software design and the prioritized software requirements.
6.4.8.2.b	Approach and checkpoints for the correct operation of the assembled interfaces and system functions are defined.	7.1.6.2.a, 7.1.6.2.c	An integration strategy is developed for software units consistent with the software design and the prioritized software requirements. Software items are verified using the defined criteria.
6.4.8.2.c	Any enabling systems or services needed for integration are available.		
6.4.8.2.d	A system composed of implemented system elements is integrated.	7.1.6.2.d	Software items defined by the integration strategy are produced.
6.4.8.2.e	The interfaces between the implemented system elements that compose the system are checked.	7.1.6.2.c	Software items are verified using the defined criteria.
6.4.8.2.f	The interfaces between the system and the external environment are checked.	7.1.6.2.c	Software items are verified using the defined criteria.
6.4.8.2.g	Integration results and anomalies are identified.	7.1.6.2.c, 7.1.6.2.e	Software items are verified using the defined criteria. Results of integration testing are recorded.
6.4.8.2.h	Traceability of the integrated system elements is established.	7.1.6.2.f	Consistency and traceability are established between software design and software items.
6.4.9	**Verification process**	**7.1.5**	**Software Construction Process**
6.4.9.2.a	Constraints of verification that influence the requirements, architecture, or design are identified.	7.1.5.2.a	Verification criteria are defined for all software units against their requirements.
		7.1.6	**Software Integration Process**
6.4.9.2.a	Constraints of verification that influence the requirements, architecture, or design are identified.	7.1.6.2.b, 7.1.6.2.g	Verification criteria for software items are developed that ensure compliance with the software requirements allocated to the items A regression strategy is developed and applied for re-verifying software items when a change in software units (including associated requirements, design and code) occur.
6.4.9.2.c	The system or system element is verified.	7.1.6.2.c	Software items are verified using the defined criteria.
		7.1.7	**Software Qualification Testing process**

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.4.9.2.a	Constraints of verification that influence the requirements, architecture, or design are identified.	7.1.7.2.a, 7.1.7.2.d	Criteria for the integrated software are developed that demonstrate compliance with the software requirements. A regression strategy is developed and applied for re-testing the integrated software when a change in software items is made.
6.4.9.2.c	The system or system element is verified.	7.1.7.2.b, 7.1.7.2.d	Integrated software is verified using the defined criteria. A regression strategy is developed and applied for re-testing the integrated software when a change in software items is made.
6.4.9.2.e	Objective evidence that the realized system fulfils the requirements, architecture and design is provided.	7.1.7.2.c	Test results are recorded.
6.4.9.2.f	Verification results and anomalies are identified.	7.1.7.2.c	Test results are recorded.
6.4.9.2.g	Traceability of the verified system elements is established.	7.1.7.2.a	Criteria for the integrated software are developed that demonstrate compliance with the software requirements.
		7.2.3	**Software Quality Assurance process**
6.4.9.2.c	The system or system element is verified.	7.2.3.2.d	Adherence of software products, processes and activities to the applicable standards, procedures and requirements is verified.
		7.2.4	**Software Verification Process**
6.4.9.2.a	Constraints of verification that influence the requirements, architecture, or design are identified.	7.2.4.2.a, 7.2.4.2.b	A software verification strategy is developed and implemented. Criteria for verification of all required software work products are identified.
6.4.9.2.b	Any enabling systems or services needed for verification are available.	7.2.4.2.a	A software verification strategy is developed and implemented.
6.4.9.2.c	The system or system element is verified.	7.2.4.2.c	Required verification activities are performed.
6.4.9.2.d	Data providing information for corrective actions is reported.	7.2.4.2.d	Defects are identified and recorded.
6.4.9.2.e	Objective evidence that the realized system fulfils the requirements, architecture and design is provided.	7.2.4.2.e	Results of the verification activities are made available to the customer and other involved parties.
6.4.9.2.f	Verification results and anomalies are identified.	7.2.4.2.d	Defects are identified and recorded.
6.4.9.2.g	Traceability of the verified system elements is established.	7.2.4.2.b, 7.2.4.2.c	Criteria for verification of all required software work products are identified. Required verification activities are performed.
6.4.10	**Transition process**	**6.2.4**	**Human Resource Management Process**
6.4.10.2.d	Operators, users and other stakeholders necessary to the system utilization and support are trained.	6.2.4.2.c	Skills of personnel are developed, maintained or enhanced.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
		6.4.7	**Software Installation Process**
6.4.10.2.a	Transition constraints that influence system/software requirements, architecture, or design are identified.	6.4.7.2.a, 6.4.7.2.b	A software installation strategy is developed. Criteria for software installation are developed that demonstrate compliance with the software installation requirements.
6.4.10.2.b	Any enabling systems or services needed for transition are available.	6.4.7.2.d	Readiness of the software product for use in its intended environment is assured.
6.4.10.2.c	The site is prepared.	6.4.7.2.d	Readiness of the software product for use in its intended environment is assured.
6.4.10.2.d	The system, as installed in its operational location, is capable of delivering its specified functions.	6.4.7.2.d	Readiness of the software product for use in its intended environment is assured.
6.4.10.2.e	Operators, users and other stakeholders necessary to the system utilization and support are trained	6.4.7.2.d	Readiness of the software product for use in its intended environment is assured.
6.4.10.2.f	Transition results and anomalies are identified.	6.4.7.2.d	Readiness of the software product for use in its intended environment is assured.
6.4.10.2.g	The installed system is activated and ready for operation.	6.4.7.2.c, 6.4.7.2.d	The software product is installed in the target environment. Readiness of the software product for use in its intended environment is assured.
6.4.10.2.h	Traceability of the transitioned elements is established.	6.4.7.2.b	Criteria for software installation are developed that demonstrate compliance with the software installation requirements.
		6.4.8	**Software Acceptance Support process**
6.4.10.2.d	The system, as installed in its operational location, is capable of delivering its specified functions.	6.4.8.2.a, 6.4.8.2.b	The product is completed and delivered to the acquirer. Acquirer acceptance tests and reviews are supported.
6.4.10.2.e	Operators, users and other stakeholders necessary to the system utilization and support are trained	6.4.8.2.c	The product is put into operation in the customers' environment.
6.4.10.2.f	Transition results and anomalies are identified.	6.4.8.2.d	Problems detected during acceptance are identified and communicated to those responsible for resolution.
6.4.10.2.g	The installed system is activated and ready for operation.	6.4.8.2.c	The product is put into operation in the customers' environment.
6.4.11	**Validation process**	**6.4.8**	**Software Acceptance Support Process**
6.4.11.2.b	The availability of services required by stakeholders is confirmed.	6.4.8.2.a, 6.4.8.2.b, 6.4.8.2.c	The product is completed and delivered to the acquirer. Acquirer acceptance tests and reviews are supported. The product is put into operation in the customers' environment.
6.4.11.2.d	The system or system element is validated.	6.4.8.2.b	Acquirer acceptance tests and reviews are supported.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.4.11.2.f	Validation results and anomalies are identified.	6.4.8.2.d	Problems detected during acceptance are identified and communicated to those responsible for resolution.
		6.4.9	**Software Operation Process**
6.4.11.2.d	The system or system element is validated.	6.4.9.2.c	The software is tested and determined to operate in its intended environment.
		7.2.5	**Software Validation Process**
6.4.11.2.a	Validation criteria for stakeholder requirements are defined.	7.2.5.2.a, 7.2.5.2.b	A validation strategy is developed and implemented. Criteria for validation of all required work products are identified.
6.4.11.2.b	The availability of services required by stakeholders is confirmed.	7.2.5.2.c	Required validation activities are performed.
6.4.11.2.c	Constraints of validation that influence the requirements, architecture, or design are identified.	7.2.5.2.a, 7.2.5.2.b	A validation strategy is developed and implemented. Criteria for validation of all required work products are identified.
6.4.11.2.d	The system or system element is validated.	7.2.5.2.c	Required validation activities are performed.
6.4.11.2.e	Any enabling systems or services needed for validation are available.	7.2.5.2.a	A validation strategy is developed and implemented.
6.4.11.2.f	Validation results and anomalies are identified.	7.2.5.2.d	Problems are identified and recorded.
6.4.11.2.g	Objective evidence that the realized system or system element satisfies stakeholder needs is provided.	7.2.5.2.e, 7.2.5.2.f	Evidence is provided that the software work products as developed are suitable for their intended use. Results of the validation activities are made available to the customer and other involved parties.
6.4.11.2.h	Traceability of the validated system elements is established.	7.2.5.2.b	Criteria for validation of all required work products are identified.
6.4.12	**Operation process**	**6.4.8**	**Software Acceptance Support Process**
6.4.12.2.d	System product services that meet stakeholder requirements are delivered.	6.4.8.2.c	The product is put into operation in the customers' environment.
		6.4.9	**Software Operation Process**
6.4.12.2.a	Operation constraints that influence system/software requirements, architecture, or design are identified.	6.4.9.2.a	An operation strategy is defined.
6.4.12.2.b	Any enabling systems, services, and material needed for operation are available.	6.4.9.2.b	Conditions for correct operation of the software in its intended environment are identified and evaluated.
6.4.12.2.e	System product performance during operation is monitored.	6.4.9.2.b	Conditions for correct operation of the software in its intended environment are identified and evaluated.
6.4.12.2.d	System product services that meet stakeholder requirements are delivered.	6.4.9.2.d	The software is operated in its intended environment.
6.4.12.2.f	Support to the customer is provided.	6.4.9.2.e	Assistance and consultation is provided to the customers of the software product in accordance with the agreement.

ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 process outcome		ISO/IEC 12207:2008 (IEEE Std 12207-2008) process outcome	
Subclause	**Outcome**	**Subclause**	**Outcome**
6.4.13	**Maintenance process**	**6.4.10**	**Software Maintenance process**
6.4.13.2.a	Maintenance constraints that influence system requirements, architecture, or design are identified.	6.4.10.2.a, 6.4.10.2.b	A maintenance strategy is developed to manage modification and migration of products according to the release strategy. The impact of changes to the existing system on organization, operations or interfaces are identified.
6.4.13.2.b	Any enabling systems or services needed for maintenance are available.	6.4.10.2.d,6.4.10.2.e	Modified products are developed with associated tests that demonstrate that requirements are not compromised. Product upgrades are migrated to the customer's environment.
6.4.13.2.c	Replacement, repaired, or revised system elements are made available.	6.4.10.c, 6.4.10.d, 6.4.10.e, 6.4.10.f	Affected system and software documentation is updated as needed. Modified products are developed with associated tests that demonstrate that requirements are not compromised. Product upgrades are migrated to the customer's environment. The system software modification is communicated to all affected parties.
6.4.13.2.d	The need for changes to address corrective, perfective, or adaptive maintenance is reported.	6.4.10.2.b	The impact of changes to the existing system on organization, operations or interfaces are identified.
6.4.13.2.e	Failure and lifetime data, including associated costs, is determined.	6.4.10.2.b	The impact of changes to the existing system on organization, operations or interfaces are identified.
		7.3.1	**Domain Engineering Process**
6.4.13.2.c	Replacement, repaired, or revised system elements are made available.	7.3.1.2.f	Assets belonging to the domain are acquired or developed and maintained throughout their life cycles.
6.4.14	**Disposal Process**	**6.4.11**	**Software Disposal Process**
6.4.14.2.a	Disposal constraints are provided as inputs to requirements, architecture, design, and implementation.	6.4.11.2.a, 6.4.11.2.b	A software disposal strategy is defined. Disposal constraints are provided as inputs to requirements.
6.4.14.2.b	Any enabling systems or services needed for disposal are available.	6.4.11.2.a, 6.4.11.2.c	A software disposal strategy is defined. The system's software elements are destroyed or stored.
6.4.14.2.c	The system elements or waste products are destroyed, stored, reclaimed or recycled in accordance with requirements, e.g., safety and security requirements.	6.4.11.2.c	The system's software elements are destroyed or stored.
6.4.14.2.d	The environment is returned to its original or an agreed state.	6.4.11.2.d	The environment is left in an agreed-upon state.
6.4.14.2.e	Records of disposal actions and analysis are available.	6.4.11.2.e	Records allowing knowledge retention of disposal actions and any analysis of long-term impacts are available.

Bibliography

[1] ANSI/AIAA G-043A-2012e, ANSI/AIAA *Guide to the Preparation of Operational Concept Documents*

[2] IEC 61508:2010 (all parts), *Functional safety of electrical/electronic/programmable electronic safety-related systems*

[3] IEEE Std 1012™-2012, *IEEE Standard for System and Software Verification and Validation*

[4] IEEE Std 1062™-2015, *IEEE Recommended Practice for Software Acquisition*

[5] IEEE Std 730™-2014, *IEEE Standard for Software Quality Assurance Processes*

[6] IEEE Std 828™-2012, *IEEE Standard for Configuration Management in Systems and Software Engineering*

[7] INCOSE TP-2003-020-01, *Technical Measurement*

[8] INCOSE.2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc., ISBN: 978-1-118-99940-0

[9] ISO 10004:2012, *Quality management — Customer satisfaction — Guidelines for monitoring and measuring*

[10] ISO 10007:2003, *Quality management systems — Guidelines for configuration management*

[11] ISO 14001:2004, *Environmental management systems — Requirements with guidance for use*

[12] ISO 15704:2000, *Industrial automation systems — Requirements for enterprise-reference architectures and methodologies*

[13] ISO 31000:2009, *Risk management — Principles and guidelines*

[14] ISO 9000:2015, *Quality management systems — Fundamentals and vocabulary*

[15] ISO 9001:2015, *Quality management systems — Requirements*

[16] ISO 9004:2009, *Quality management systems — Guidelines for performance improvements*

[17] ISO 9241-210:2010, *Ergonomics of human-system interaction — Human-centred design for interactive systems*

[18] ISO Guide 73:2009, *Risk management — Vocabulary*

[19] ISO/FDIS 9241-220, *Ergonomics of human-system interaction — Part 220: Processes for enabling, executing and assessing human-centred design within organizations*

[20] ISO TS 18152:2010, *Ergonomics of human-system interaction — Specification for the process assessment of human-system issues*

[21] ISO/IEC 10746-3:2009*, Information technology — Open distributed processing — Reference model: Architecture*

[22] ISO/IEC 15026-3:2011, *System and software engineering — Systems and software assurance — Part 3: System integrity levels*

[23] ISO/IEC 15026-4:2012, *Systems and software engineering — Systems and software assurance — Part 4: Assurance in the life cycle*

[24] ISO/IEC 15939:2007, *Systems and software engineering — Measurement process*

[25] ISO/IEC 16085:2006, *System and Software Engineering — Life Cycle Management — Risk Management*

[26] ISO/IEC 16350:2015, *Information technology — Systems and software engineering — Application management*

[27] ISO/IEC 19770-1:2012, *Information technology — Software asset management — Part 1: Processes and tiered assessment of conformance*

[28] ISO/IEC 20000-1:2011 (IEEE Std 20000-1:2013), *Information technology — Service management — Part 1: Service management system requirements*

[29] ISO/IEC 25010:2011, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*

[30] ISO/IEC 25030:2007, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements*

[31] ISO/IEC 25063:2014, *Systems and software engineering — Systems and software product Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for usability: Context of use description*

[32] ISO/IEC 26550:2013, *Software and systems engineering — Reference model for product line engineering and management*

[33] ISO/IEC 27000:2016, *Information technology — Security techniques — Information security management systems — Overview and vocabulary*

[34] ISO/IEC 27036 (all parts), *Information technology — Security techniques — Information security for supplier relationships*

[35] ISO/IEC 33001:2015, *Information technology — Process assessment — Concepts and terminology*

[36] ISO/IEC 33002:2015, *Information technology — Process assessment — Requirement for performing process assessment*

[37] ISO/IEC 33004:2015, *Information technology — Process assessment — Requirement for process reference, process assessment, and maturity models*

[38] ISO/IEC 33020:2015, *Information technology — Process assessment — Process measurement framework for assessment of process capability*

[39] ISO/IEC TR 19759:2015, *Guide to the Software Engineering Body of Knowledge (SWEBOK) V3, IEEE Computer Society, 2014*

[40] ISO/IEC TR 24748-2:2011, *Systems and software engineering — Life cycle management — Part 2: Guide to the application of ISO/IEC 15288 (System life cycle processes)*

[41] ISO/IEC TR 24748-3:2011, *Systems and software engineering — Life cycle management — Part 3: Guide to the application of ISO/IEC 12207 (Software life cycle processes)*

[42] ISO/IEC TR 24774:2010[1], *Systems and software engineering — Life cycle management — Guidelines for process description*

[43] ISO/IEC TR 25060:2010, *Systems and software engineering — Systems and software product Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for usability: General framework for usability-related information*

[1]The electronic version of this International Standard can be freely downloaded from the ISO/IEC Information Technology Task Force (ITTF) web site.

[44] ISO/IEC TS 24748-1:2016, *Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management*

[45] ISO/IEC/IEEE 14764:2006, *Software Engineering — Software Life Cycle Processes — Maintenance*

[46] ISO/IEC/IEEE 15288:2015, *Systems and software engineering — System life cycle processes*

[47] ISO/IEC/IEEE 15289:2015, *Systems and software engineering — Content of life-cycle information products (documentation)*

[48] ISO/IEC/IEEE 16326:2009, *Systems and software engineering — Life cycle processes — Project management*

[49] ISO/IEC/IEEE 24748-4:2016, *Systems engineering — Life cycle management — Part 4: Application and management of the systems engineering process*

[50] ISO/IEC/IEEE 24748-5[2], *Systems and software engineering — Life cycle management — Part 5: Software development planning*

[51] ISO/IEC/IEEE 24765[3], *Systems and software engineering — Vocabulary*

[52] ISO/IEC/IEEE 26515:2011, *Systems and software engineering: Developing user documentation in an agile environment*

[53] ISO/IEC/IEEE 26531:2015, *Systems and software engineering — Content management for product life-cycle, user, and service management documentation*

[54] ISO/IEC/IEEE 29119 (all parts), *Systems and software engineering — Software testing*

[55] ISO/IEC/IEEE 29148:2011, *Systems and software engineering — Life cycle processes — Requirements engineering*

[56] ISO/IEC/IEEE 42010:2011, *Systems and software engineering — Architecture description*

[57] NATO AEP-67, *Engineering for System Assurance in NATO Programs*

[58] PMI Practice Standard for Work Breakdown Structures-Second Edition

[59] SAE ANSI/EIA 649B, *Configuration Management Standard*

[2] To be published

[3] Database version available at <computer.org/sevocab>

Important Notices and Disclaimers Concerning IEEE Standards Documents

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

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

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.
In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.
Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.
Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants: The list of IEEE participants can be accessed at the following URL: http://standards.ieee.org/downloads/12207/12207-2017/12207-2017_wg-participants.pdf.

Abstract: This document establishes a common process framework for describing the full life cycle of software systems from conception through retirement. It applies to the acquisition or development of software products and services whether performed internally or externally to an organization. Those aspects of software system definition needed to provide the context for software products and services are included. The document also supports the definition, control, assessment and improvement of these processes. These processes can be applied concurrently, iteratively, and recursively to a software system and its elements throughout the life cycle of a software system. Users of the document can apply the processes and terminology to construct a suitable life cycle model, composed of stages that use selected processes. The process purposes and outcomes in ISO/IEC/IEEE 15288:2015 (systems life cycle processes) and this document are fully aligned so that one may select appropriate activities from either document for use in systems with substantial software content.

Keywords: acquisition, agreement, architecture, design, assessment, audit, configuration management, decision management, development, disposal, enabling system, implementation, information management, infrastructure, integration, life cycle, life cycle model, life cycle stages, maintenance, measurement, operation, planning, process, process improvement, process reference model, process tailoring, process view, product, portfolio, quality management, requirements, retirement, risk management, service, software, software system, system-of-interest, stages, stakeholder requirements, supply, system, system structure, system-of-interest, tailoring, transition, validation, verification

ICS 35.080
ISBN 978-1-5044-4253-4 STD 22737 (PDF); 978-1-5044-4254-1 STDPD 22737 (Print)
Price based on 145 pages