



DPC: 25/30499385 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 10 March 2025

Origin: International

Latest date for receipt of comments: 03 May 2025

Project No. 2024/02432

Responsible committee: IST/15 Software and systems engineering

Interested committees:

Title: Draft BS EN ISO/IEC/IEEE 12207-1 BS ISO/IEC/IEEE 12207 Systems and software engineering. Software life cycle processes.

Please notify admin.start@bsigroup.com if you are aware of any keywords that might assist in clarifying or identifying the standard or of the content of this standard:

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD. THIS DRAFT IS NOT CURRENT BEYOND 03 May 2025

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <https://standardsdevelopment.bsigroup.com/>.

E-mail: admin.start@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/ Figure/Table/Note	Type of comment	Comment (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	



DRAFT International Standard

ISO/IEC/IEEE DIS 12207

Systems and software engineering — Software life cycle processes

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

ICS: 35.080

ISO/IEC JTC 1/SC 7

Secretariat: **BIS**

Voting begins on:
2025-03-10

Voting terminates on:
2025-06-02

This document is circulated as received from the committee secretariat.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENTS AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

Reference number
ISO/IEC/IEEE DIS 12207:2025(en)

Full document details: 25/30499385 DC

© ISO/IEC 2025
© IEEE 2025



Documents

Document Sharing Hub



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2025

© IEEE 2025

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO or IEEE at the respective address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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

Email: stds.ipr@ieee.org
Website: www.ieee.org

Contents

Page

Foreword	v
Introduction	vii
1 Scope	1
1.1 Overview	1
1.2 Purpose	1
1.3 Applicability	1
1.4 Limitations	2
2 Normative references	2
3 Terms, definitions, and abbreviated terms	2
3.1 Terms and definitions	2
3.2 Abbreviated terms	13
4 Conformance	14
4.1 General	14
4.2 Full conformance	15
4.2.1 Full conformance to outcomes	15
4.2.2 Full conformance to tasks	15
4.3 Tailored conformance	15
5 Key concepts and application	15
5.1 General	15
5.2 Software system concepts	15
5.2.1 Software systems	15
5.2.2 Software system structure	16
5.2.3 Interfacing, enabling, and interoperating systems	17
5.2.4 System of systems concepts	20
5.3 Organization and project concepts	21
5.3.1 Organizations	21
5.3.2 Organization and project-level adoption	22
5.3.3 Organization and collaborative activities	22
5.4 Life cycle concepts	23
5.4.1 Software life cycle stages	23
5.4.2 Life cycle models for the software system	23
5.5 Process concepts	25
5.5.1 Criteria for processes	25
5.5.2 Description of processes	25
5.5.3 General characteristics of processes	26
5.5.4 Process views	26
5.5.5 Tailoring	26
5.6 Process groups	26
5.6.1 Introduction	26
5.6.2 Agreement processes	29
5.6.3 Organizational project-enabling processes	29
5.6.4 Technical management processes	30
5.6.5 Technical processes	30
5.7 Process application	31
5.7.1 Selection and timing of process use	31
5.7.2 Process iteration, recursion, and concurrency	32
5.8 Process reference model	32
6 Software life cycle processes	33
6.1 Agreement processes	33
6.1.1 Acquisition process	34
6.1.2 Supply process	37
6.2 Organizational project-enabling processes	39
6.2.1 Life cycle model management process	39

6.2.2	Infrastructure management process.....	41
6.2.3	Portfolio management process.....	42
6.2.4	Human resource management process.....	44
6.2.5	Quality management process.....	45
6.2.6	Knowledge management process.....	46
6.3	Technical management processes.....	48
6.3.1	Project planning process.....	49
6.3.2	Project assessment and control process.....	51
6.3.3	Decision management process.....	54
6.3.4	Risk management process.....	55
6.3.5	Configuration management process.....	58
6.3.6	Information management process.....	62
6.3.7	Measurement process.....	64
6.3.8	Quality assurance process.....	66
6.4	Technical processes.....	68
6.4.1	Business or mission analysis process.....	69
6.4.2	Stakeholder needs and requirements definition process.....	72
6.4.3	System requirements definition process.....	77
6.4.4	System architecture definition process.....	81
6.4.5	Design definition process.....	86
6.4.6	System analysis process.....	90
6.4.7	Implementation process.....	92
6.4.8	Integration process.....	95
6.4.9	Verification process.....	98
6.4.10	Transition process.....	102
6.4.11	Validation process.....	106
6.4.12	Operation process.....	110
6.4.13	Maintenance process.....	113
6.4.14	Disposal process.....	118
Annex A (normative) Tailoring process.....		121
Annex B (informative) Examples of process information items.....		123
Annex C (informative) Process reference model for assessment purposes.....		129
Annex D (informative) Model-based systems and software engineering (MBSSE).....		131
Annex E (informative) Assurance and assurance cases.....		137
Bibliography.....		138
IEEE Notices and Abstract.....		143

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents 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).

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.

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO, IEC, and IEEE take no position concerning the evidence, validity, or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of a patent which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

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

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

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

This second edition of ISO/IEC/IEEE 12207 cancels and replaces ISO/IEC 12207:2017, which has been technically revised.

The main changes are as follows:

- Clarifications and updates to reflect current practices in selected technical processes, including business or mission analysis, system architecture definition, implementation, integration, operations, and maintenance;
- Improvements to selected technical management processes including risk management and configuration management;
- Updates to [Clause 5](#), key concepts, including a more precise description of iteration, recursion, system-of-systems, and quality characteristics;
- New content in [Clause 5](#) on concept and system definition, and expanded content on agile methods, process application, and system concepts.

ISO/IEC/IEEE DIS 12207:2025(en)

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

Introduction

The complexity of software systems continues to increase to unprecedented levels. Technology change has led to new opportunities, but also to increased challenges for the organizations that create and utilise 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 software. This document concerns software systems that can be configured with one or more of the following: hardware elements, data, humans, processes, services, procedures, and facilities,

This document focuses on defining stakeholder needs, concerns, priorities, and constraints for the required functionality early in the life cycle, establishing requirements, and concurrent design synthesis and system validation while considering the complete problem. It integrates disciplines and specialties into a team effort, forming a structured set of process that proceeds from concept through production operation and maintenance sustainment to disposal. It considers both the business and the technical needs of stakeholders with the goal of providing a quality product that meets the needs of users and other applicable stakeholders. It provides the processes for acquiring and supplying systems. It helps to improve communication and cooperation among the parties that create, utilise, and manage modern software systems so they can work in an integrated, coherent fashion. Finally, this document provides the framework for assessment and improvement of the life cycle processes.

There is a wide variety of software systems in terms of their purpose, domain of application, complexity, size, novelty, adaptability, quantity, location, life span, and evolution. 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 fulfil that purpose.

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

- 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 systems through their life cycle stages. In this mode this document is used to assess conformance of a declared, established environment to its provisions. 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.
- 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.
- 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.
- By process assessors — to serve as a process reference model for use in the performance of process assessments that can 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 to be applied during the acquisition of a software system, product, or service and during the supply, development, operation, maintenance, and disposal of software products. This is accomplished through the involvement of stakeholders, with the goal of achieving customer satisfaction. This document includes those aspects of system definition needed to provide the context for software systems and services. This document also provides processes that can be employed for defining, controlling, and improving software life cycle processes within an organization or a project.

1.2 Purpose

The purpose of this document is to provide a defined set of software life cycle processes to facilitate communication among acquirers, suppliers, and other stakeholders in the life cycle of a software system. This document is written for acquirers of software systems, products, and services and for suppliers, developers, integrators, operators, maintainers, managers, quality assurance managers, and users of software systems and products. 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 International Standard 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 Applicability

In the context of this document and ISO/IEC/IEEE 15288, *Systems and software engineering—System life cycle processes*, 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 (SoI) 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 15288:2023, depends on the SoI. 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.

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 applies to one-of-a-kind software systems, software systems for wide commercial or public distribution, and customised, adaptable software systems. Software includes the software portion of firmware. It applies to a complete stand-alone software system and to software systems that are embedded and integrated into larger more complex and complete systems of systems (SoS). 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 15288:2023.

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.

This document may 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.

[Annex A](#) of this document presents the tailoring process which may be applied to software engineering processes. [Annex B](#) identifies examples of artefacts and information items that may be associated with various processes. [Annex C](#) 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. This document can therefore be used as a reference model to support process assessment as specified in ISO/IEC 33002:2015. [Annex D](#) describes model-based systems and software engineering (MBSSE) and the relationships of software engineering, systems engineering, and digital engineering. [Annex E](#) presents the use of assurance claims in system assurance.

1.4 Limitations

This document does not prescribe a specific software life cycle model, development methodology, method, modelling approach, or technique.

This document does not establish a management system, but it is intended to be compatible with the quality management system provided by ISO 9001, the service management system provided by ISO/IEC 20000 series, the IT asset management system provided by the ISO/IEC 19770 series, and the information security management system provided by ISO/IEC 27000.

NOTE ISO/IEC/IEEE 90003 provides guidance for applying the requirements of ISO 9001 to software.

The users of this document are responsible for selecting a life cycle model for the organization or 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.

This document does not detail information items in terms of name, format, explicit content, and recording media. ISO/IEC/IEEE 15289 identifies 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, IEC, and IEEE maintain terminology databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org/>
- IEEE Standards Dictionary Online: available at: <https://dictionary.ieee.org/>

NOTE Definitions for other system and software engineering terms can be found in ISO/IEC/IEEE 24765, available at www.computer.org/sevocab.

3.1.1

acquirer

stakeholder (3.1.73) that acquires or procures a *system* (3.1.75), *product* (3.1.46) or *service* (3.1.62) from a *supplier* (3.1.74)

Note 1 to entry: Other terms commonly used for an acquirer are buyer, *customer* (3.1.17), owner, purchaser, or internal *organizational* (3.1.39) sponsor.

3.1.2

acquisition

process (3.1.45) of obtaining a *system* (3.1.75), *product* (3.1.46) or *service* (3.1.62)

3.1.3

activity

set of cohesive *tasks* (3.1.80) of a *process* (3.1.43)

3.1.4

agile development

development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which *requirements* (3.1.56) and solutions evolve through collaboration in cross-functional teams and through continual *stakeholder* (3.1.73) feedback

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

fundamental concepts or properties of an entity in its environment and governing principles for the realization and evolution of this *system* (3.1.75) and its related *life cycle* (3.1.34) *processes* (3.1.43)

[SOURCE: ISO/IEC/IEEE 42010:2022, 3.2]

3.1.7

architecture description framework

conventions, principles, and practices for the description of *architectures* (3.1.6) established within a specific domain of application or community of *stakeholders* (3.1.73)

[SOURCE: ISO/IEC/IEEE 42010:2022, 3.5]

EXAMPLE Generalised Enterprise Reference Architecture and Methodologies (GERAM) [ISO 15704, Reference Model of Open Distributed Processing (RM-ODP) [ISO/IEC 10746], Reference Model of Open Distributed Processing (RM-ODP), Unified Architecture Framework (UAF), and NATO Architecture Framework (NAF)

3.1.8

architecture view

information part comprising portion of an *architecture* (3.1.6) description

[SOURCE: ISO/IEC/IEEE 42010, 3.7]

3.1.9

architecture viewpoint

set of conventions for the creation, interpretation, and use of an *architecture view* (3.1.88) to frame one or more *concerns* (3.1.16)

[SOURCE: ISO/IEC/IEEE 42010, 3.8]

3.1.10 artefact

work *product* (3.1.46) that is produced and used during a *project* (3.1.47) to capture and convey information

EXAMPLE Models, *stakeholder* (3.1.73) *requirements* (3.1.56), *system* (3.1.75) requirements, *architecture* (3.1.6) descriptions, *design* (3.1.18) descriptions, source code, implemented *system elements* (3.1.76), verified or validated system

3.1.11 audit

independent examination of a work *product* (3.1.46) or set of work products to assess compliance with contractual *agreements* (3.1.5) or conformance to specifications, standards, or other criteria

3.1.12 baseline

formally approved version of a *configuration item* (3.1.17), regardless of media, formally designated and fixed at a specific time during the configuration item's *life cycle* (3.1.34)

[SOURCE: IEEE Std 828-2012]

3.1.13 business process

partially ordered set of enterprise *activities* (3.1.3) that can be executed to achieve some desired end-result in pursuit of a given objective of an *organization* (3.1.39)

3.1.14 capability

ability to do something useful under a particular set of conditions

[SOURCE: ISO/IEC/IEEE 24641:2023, 3.1.3, modified — Note to entry has been removed.]

3.1.15 concept of operations

verbal and graphic statement, in broad outline, of an *organization's* (3.1.39) assumptions or intent regarding an operation or series of operations of new, modified, or existing organizational *systems* (3.1.75)

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 to achieve an organizational performance objective. *See also operational concept* (3.1.37).

Note 2 to entry: The concept of operations provides the basis for bounding the operating space, system capabilities, *interfaces* (3.1.31) and operating environment (3.1.21).

[SOURCE: ANSI/AIAA G-043B-2018, 5.2, modified — The second definition has been used; the last two sentences of Note 1 to entry have been removed; Note 2 to entry has been added.]

3.1.16 concern

matter of interest or importance to a *stakeholder* (3.1.73)

Note 1 to entry: A concern pertains to any influence on a *system* (3.1.75) in its environment (3.1.21), including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ethical, ecological, and social influences.

[SOURCE: ISO/IEC/IEEE 42020:2019, 3.8, modified — EXAMPLE has been removed; Note 1 to entry has been added.]

3.1.17 configuration item

item or aggregation of hardware or *software* (3.1.63) with associated information and data

Note 1 to entry: *information items* (3.1.29) can be considered as part of the software and managed as part of the configuration item, or can be managed using the *information management* (3.1.36) *process* (3.1.43).

3.1.18**constituent system**

independent *system* ([3.1.75](#)) that forms part of a *system of systems (SoS)* ([3.1.78](#))

Note 1 to entry: Constituent systems can be part of one or more SoS. Each constituent system is a useful system by itself, having its own development, *management* ([3.1.36](#)), utilization, goals, and *resources* ([3.1.57](#)), but interacts within the SoS to provide the unique *capability* ([3.1.14](#)) of the SoS.

[SOURCE: ISO/IEC/IEEE 21839:2019, 3.1.1]

3.1.19**customer**

organization ([3.1.39](#)) or person that receives a *product* ([3.1.46](#)) or *service* ([3.1.62](#))

EXAMPLE Consumer, client, *user* ([3.1.85](#)), *acquirer* ([3.1.1](#)), buyer, or purchaser.

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

3.1.20**design**

specification of *system elements* ([3.1.76](#)) and their relationships, that is sufficiently complete to support a compliant implementation of the *architecture* ([3.1.6](#))

Note 1 to entry: Design provides the detailed implementation-level physical structure, behaviour, temporal relationships, and other attributes of system elements.

3.1.21**design characteristic**

design ([3.1.20](#)) attribute or distinguishing feature that pertains to a measurable description of a *product* ([3.1.46](#)) or *service* ([3.1.62](#))

3.1.22**digital engineering**

application of *software engineering* ([3.1.66](#)) *processes* ([3.1.43](#)) using software modelling, simulation, and analytical functions for physical or logical *systems* ([3.1.75](#)) conceptualization, development, and integrated control of operational performance

3.1.23**enabling system**

system ([3.1.75](#)) that supports a *system-of-interest* ([3.1.77](#)) during its *life cycle* ([3.1.34](#)) *stages* ([3.1.72](#)) 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* ([3.1.34](#)) of its own.

3.1.24**environment**

context determining the setting and circumstances of all influences upon a *system* ([3.1.75](#))

3.1.25**facility**

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

3.1.26**firmware**

ordered set of instructions and associated data stored in a way that is functionally independent of main storage, usually in read-only memory

3.1.27**governance**

practice of establishing and enforcing strategic goals and objectives, organizational policies, and performance parameters

3.1.28**incident**

anomalous or unexpected event or set of events at any time during the *life cycle* (3.1.34) of a *project* (3.1.47), *product* (3.1.46), *service* (3.1.62), or *system* (3.1.75)

Note 1 to entry: An incident is elevated and treated as a *problem* (3.1.42) when the cause of the incident needs to be analysed and corrected to prevent recurrence, or to avoid or minimise loss of life, or damage to property or natural resources (3.1.57).

3.1.29**information item**

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

[SOURCE: ISO/IEC/IEEE 15289:2019, 3.1.12,

3.1.30**infrastructure**

hardware and *software* (3.1.63) *environment* (3.1.24) to support *system* (3.1.75) and *software design* (3.1.20), development, operation, and modification

3.1.31**interface**

point at which two or more logical or physical *system elements* (3.1.76) meet to act on or communicate with each other

3.1.32**interoperating system**

system (3.1.71) that exchanges information with the *system-of-interest* (3.1.77) and uses the information that has been exchanged

3.1.33**iteration**

repeating the application of the same *process* (3.1.43) or set of processes at the same level of abstraction on the same *system* (3.1.75) or *system element* (3.1.76)

3.1.34**life cycle**

evolution of a *system* (3.1.75), *product* (3.1.46), *service* (3.1.62), *project* (3.1.47) or other human-made entity from conception through *retirement* (3.1.58)

3.1.35**life cycle model**

framework of *processes* (3.1.43) and *activities* (3.1.3) concerned with the *life cycle* (3.1.34) which can be organized into *stages* (3.1.72), acting as a common reference for communication and understanding

3.1.36**management**

oversight and direction of activities and controls to achieve the strategic objectives set by the *organization's* (3.1.39) governing body

3.1.37**operational concept**

verbal and graphic statement of an *organization's* (3.1.39) assumptions or intent in regard to an operation or series of operations of a specific *system* (3.1.75) or a related set of specific new, existing, or modified systems. *See also concept of operations* (3.1.15).

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* (3.1.24) from the *users'* (3.1.85) and *operators'* (3.1.38) perspectives.

Note 2 to entry: The operational concept is about systems, while a concept of operations typically refers to organizations.

[SOURCE: ANSI/AIAA G-043B-2018, 5.2, modified — The third definition has been used; the first sentence in Note 1 to entry has been removed; Note 2 to entry has been added.]

3.1.38

operator

individual or *organization* (3.1.39) that performs the operations of a *system* (3.1.75)

Note 1 to entry: The role of operator and the role of *user* (3.1.85) 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 may perform operations on a system that is operated, or of a system that is operated, depending on whether operating instructions are placed within the system boundary.

3.1.39

organization

person or group of people that has its own functions with responsibilities, authorities, and relationships to achieve its objectives

EXAMPLE Company, corporation, firm, enterprise, manufacturer, 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.

3.1.40

party

organization (3.1.39) entering into an *agreement* (3.1.5)

Note 1 to entry: In this document, the agreeing parties are called the *acquirer* (3.1.1) and the *supplier* (3.1.74).

[SOURCE: ISO 9000:2015, 3.2.1, modified — Notes to entry have been removed]

3.1.41

portfolio

collection of *projects* (3.1.47) or *systems* (3.1.75) that address the strategic or business objectives of the *organization* (3.1.39)

3.1.42

problem

difficulty, uncertainty, or otherwise realised and undesirable event, set of events, condition, or situation that is investigated and can receive corrective action

3.1.43

process

set of interrelated or interacting *activities* (3.1.3) that transform inputs into outputs

3.1.44

process outcome

observable result of the successful achievement of the *process* (3.1.42) *purpose* (3.1.39)

3.1.45

process purpose

high level objective of performing the *process* (3.1.42) 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* (3.1.73)

3.1.46

product

result of a *process* ([3.1.42](#))

Note 1 to entry: ISO/IEC/IEEE 15288:2023 defines a product as "output of an *organization* ([3.1.39](#)) that can be produced without any transaction taking place between the organization and the *customer* ([3.1.19](#))"

Note 2 to entry: There are four generic product categories: hardware, e.g., engine mechanical part; *software* ([3.1.63](#)), e.g., computer program procedures, and possibly associated documentation and data; *services* ([3.1.62](#)), e.g., transport; and processed materials, e.g., lubricant. Hardware and processed materials are generally tangible products, while software or services are generally intangible.

3.1.47

project

endeavour with defined start and finish criteria undertaken to create a *product* ([3.1.46](#)) or *service* ([3.1.62](#)) in accordance with specified *resources* ([3.1.57](#)) and *requirements* ([3.1.56](#))

Note 1 to entry: A project is sometimes viewed as a unique *process* ([3.1.42](#)) comprising coordinated and controlled *technical management* ([3.1.82](#)) and *technical activities* ([3.1.3](#)).

Note 2 to entry: Continuous development approaches such as agile and DevOps can create products and services without the defined start and finish criteria of projects.

3.1.48

qualification

process ([3.1.42](#)) of determining whether an entity is capable of fulfilling specified *requirements* ([3.1.56](#))

3.1.49

quality assurance

QA

process ([3.1.42](#)) that is focused on providing confidence that *quality requirements* ([3.1.53](#)) are fulfilled

3.1.50

quality characteristic

inherent characteristic of a *product* ([3.1.46](#)), *service* ([3.1.62](#)), *process* ([3.1.42](#)), or *system* ([3.1.75](#)) related to a *requirement* ([3.1.56](#))

Note 1 to entry: Critical quality characteristics commonly include those related to health, *safety* ([3.1.60](#)), *security* ([3.1.61](#)) assurance, reliability, availability, and supportability.

[SOURCE: ISO 9000:2015, 3.10.2, modified — 'object' has been replaced with 'product (3.1.46), service, process, or system;' notes to entry have been removed.]

3.1.51

quality control

tasks ([3.1.80](#)) to evaluate the quality of *services* ([3.1.62](#)) as delivered or the quality of developed or manufactured *products* ([3.1.46](#))

3.1.52

quality management

coordinated *activities* ([3.1.3](#)) to direct and control an *organization* ([3.1.39](#)) with regard to quality

3.1.53

quality requirement

requirement ([3.1.56](#)) for quality properties or attributes of a *product* ([3.1.46](#)), data, or *service* ([3.1.62](#)) that satisfy needs associated with its the purpose

3.1.54

recursion

repeating the application of the same *process* ([3.1.42](#)) or set of processes to progressively more detailed *system elements* ([3.1.76](#))

3.1.55 release

version of a configuration item ([3.1.16](#)) that is made available for a specific purpose

EXAMPLE Test release

3.1.56 requirement

statement which translates or expresses a need and its associated constraints and conditions

[SOURCE: ISO/IEC/IEEE 29148:2018, 3.1.19, modified — Notes to entry have been removed.]

3.1.57 resource

asset that is utilised or consumed during the execution of a *process* ([3.1.42](#))

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

Note 2 to entry: Resource includes diverse entities such as funding, personnel, facilities, capital equipment, tools, and utilities such as power, water, fuel, and communication *infrastructures* ([3.1.30](#)).

3.1.58 retirement

withdrawal of active support by the operation and maintenance *organization* ([3.1.39](#)), partial or total replacement by a new *system* ([3.1.76](#)), installation of an upgraded system, or final decommissioning and disposal

3.1.59 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* ([3.1.60](#)), and environmental goals] and can apply at different levels, such as strategic, organization-wide, *project* ([3.1.47](#)), *product* ([3.1.46](#)) and *process* ([3.1.43](#)).

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, 1.1, modified — The last sentence in Note 1 to entry has been added.]

3.1.60 safety

expectation that a *system* ([3.1.76](#)) does not, under defined conditions, lead to a state in which human life, health, property, or the *environment* ([3.1.24](#)) is endangered

3.1.61 security

protection against intentional subversion or forced failure

Note 1 to entry: Security includes authenticity, accountability, confidentiality, integrity, availability, non- repudiation, and reliability, all of which have the related issue of their assurance.

[SOURCE: NATO AEP-67, modified — Note 1 to entry has been updated.]

3.1.62

service

performance of activities, work, or duties

Note 1 to entry: The dominant elements of a service are generally intangible.

Note 2 to entry: A service is coherent, discrete, and can be composed of other services.

Note 3 to entry: In ISO/IEC/IEEE 15288:2023, service is defined as "output of an *organization* (3.1.39) with at least one *activity* (3.1.3) necessarily performed between the organization and the *customer* (3.1.19)

3.1.63

software

computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer *system* (3.1.75)

3.1.64

software bill of materials

SBOM

list of uniquely identified *software* (3.1.63) components, including their sources in the supply chain, version, license and copyright data, and possibly *security* (3.1.61)-related data

Note 1 to entry: Each SBOM item can have its own SBOM.

3.1.65

software element

system element (3.1.76) that is *software* (3.1.63)

3.1.66

software engineering

application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of *software* (3.1.63); that is, the application of engineering to software

3.1.67

software item

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

Note 1 to entry: A *software* (3.1.63) item can be viewed as a *system element* (3.1.76). Software items are typically *configuration items* (3.1.17).

3.1.68

software product

set of computer programs, procedures, and possibly associated documentation and data

Note 1 to entry: A *software* (3.1.63) *product* (3.1.46) is a *software system* (3.1.69) viewed as the output product resulting from a *process* (3.1.43).

3.1.69

software system

system (3.1.75) for which *software* (3.1.63) is of primary importance to the *stakeholders* (3.1.73)

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).

3.1.70

software system element

member of a set of elements that constitute a *software system* (3.1.69)

Note 1 to entry: A *software system element* (3.1.76) can include one or more *software units* (3.1.71), *software elements* (3.1.65), hardware units, hardware elements, *services* (3.1.62), and other system elements and *systems* (3.1.75).

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

3.1.71

software unit

atomic-level *software* (3.1.63) component of the software *architecture* (3.1.6) 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.]

3.1.72

stage

period within the *life cycle* (3.1.34) of an entity that relates to the state of its description or realization

Note 1 to entry: Stages relate to major progress and achievement milestones of the entity through its life cycle.

Note 2 to entry: Stages often overlap.

3.1.73

stakeholder

individual or *organization* (3.1.39) having a right, share, claim, or interest in a *system* (3.1.75) or in its possession of characteristics that meet their needs and expectations

EXAMPLE Users, end user organizations, supporters, developers, customers, producers, trainers, maintainers, disposers, acquirers, suppliers, regulatory bodies, and people influenced positively or negatively by a system

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

Note 2 to entry: Stakeholders include those who are affected or impacted by a system.

3.1.74

supplier

organization (3.1.39) or an individual that enters into an *agreement* (3.1.5) with the *acquirer* (3.1.1) for the supply of a *product* (3.1.46) or *service* (3.1.62)

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

3.1.75

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* (3.1.46) or as the *services* (3.1.62) it provides.

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

Note 3 to entry: A system can include the associated equipment, facilities, material, *software* (3.1.62), *firmware* (3.1.26), technical information, services and personnel required for operations and support to the degree necessary for use in its intended *environment* (3.1.24).

Note 4 to entry: In ISO/IEC/IEEE 15288:2023, system is defined as "arrangement of parts or elements that together exhibit a stated behaviour or meaning that the individual constituents do not."

3.1.76

system element

discrete part of a *system* (3.1.75) that can be implemented to fulfil specified *requirements* (3.1.56)

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.

3.1.77**system-of-interest****SoI**

system ([3.1.75](#)) whose *life cycle* ([3.1.34](#)) is under consideration

3.1.78**system of systems****SoS**

set of *systems* ([3.1.75](#)) or *system elements* ([3.1.76](#)) that interact to provide a unique *capability* ([3.1.14](#)) that none of the *constituent systems* ([3.1.18](#)) can accomplish on its own

Note 1 to entry: Each constituent system is a useful system by itself, having its own *management* ([3.1.36](#)), goals, and *resources* ([3.1.57](#)), but coordinates within the SoS to provide the unique *capability* ([3.1.14](#)) of the SoS.

[SOURCE: ISO/IEC/IEEE 21839:2019, 3.1.4]

3.1.79**systems engineering**

transdisciplinary and integrative approach to enable the successful realization, use, and *retirement* ([3.1.58](#)) of engineered *systems* ([3.1.75](#)) using systems principles and concepts and scientific, technological and *management* ([3.1.36](#)) methods

3.1.80**task**

required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a *process* ([3.1.42](#))

3.1.81**technical debt**

deferred cost of work not identified or not done at an earlier point in the *product* ([3.1.46](#)) *life cycle* ([3.1.34](#))

3.1.82**technical management**

application of technical and administrative *resources* ([3.1.57](#)) to plan, organize and control engineering functions

3.1.83**traceability**

association established between two or more logical entities, especially entities having a predecessor-successor or primary-secondary relationship to one another, such as *requirements* ([3.1.56](#)), *system elements* ([3.1.76](#)), *verifications* ([3.1.87](#)), or *tasks* ([3.1.80](#))

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

3.1.84**trade-off**

decision-making actions that select from various *requirements* ([3.1.56](#)) and alternative solutions on the basis of net benefit to the *stakeholders* ([3.1.73](#))

3.1.85**user**

individual or group that interacts with a *system* ([3.1.75](#)) or benefits from a system during its utilization

Note 1 to entry: The role of user and the role of *operator* ([3.1.38](#)) are sometimes vested, simultaneously or sequentially, in the same individual or organization ([3.1.33](#)).

3.1.86 validation

confirmation, through the provision of objective evidence, that the *requirements* (3.1.56) for a specific intended use or application have been fulfilled

Note 1 to entry: In a *life cycle* (3.1.34) context, validation involves the set of *activities* (3.1.3) for gaining confidence that a *system* (3.1.75) is able to accomplish its intended use, goals, and objectives in an *environment* (3.1.24) like the operational environment. The right system was built.

[SOURCE: ISO 9000:2015, 3.8.13, modified — Notes 1 to 3 to entry have been removed; a new Note 1 to entry has been added.]

3.1.87 verification

confirmation, through the provision of objective evidence, that specified *requirements* (3.1.56) have been fulfilled

Note 1 to entry: Verification is a set of *activities* (3.1.3) that compares a *system* (3.1.75) or *system element* (3.1.76) against the required characteristics. Verification includes specified requirements, *design* (3.1.20) descriptions, and the system itself. The system was built right.

[SOURCE: ISO 9000:2015, 3.8.12, modified — Notes 1 to 3 to entry have been removed; a new Note 1 to entry has been added.]

3.1.88 view

representation of a *system* (3.1.75) from the perspective of a related set of *concerns* (3.1.16)

Note 1 to entry: A view can be an operational, functional, or architectural representation of a system.

[SOURCE: ISO/IEC/IEEE 24774:2021, 3.21, modified — removed ‘whole’ from the definition and replaced the original Note 1 to entry]

3.1.89 viewpoint

specification of the conventions for constructing and using a *view* (3.1.88)

[SOURCE: ISO/IEC/IEEE 24774:2021, 3.22, modified — Notes 1 to 3 to entry have been removed.]

3.2 Abbreviated terms

CM	configuration management
COTS	commercial-off-the-shelf
FCA	functional configuration audit
GUI	graphical user interface
NDI	non-developmental items
QA	quality assurance
PCA	physical configuration audit
PMI	project management institute
PMP	project management plan
PRM	process reference model
SBOM	software bill of materials

SCM	software configuration management
SDP	software development plan
SEMP	systems engineering management plan
SoI	system-of-interest
SoS	system of systems
WBS	work breakdown structure

4 Conformance

4.1 General

[Clause 6](#) contains requirements for software engineering processes. Projects or organizations can use selections from 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 the required activities and tasks of the declared set of processes are achieved. Alternatively, claiming “full conformance to outcomes” asserts that all 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 ISO/IEC/IEEE 24774 Clause 7 describes how to state claims of conformance to process outcomes or to tasks.

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.

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, can have innovative process variants that achieve the objectives (i.e. the outcomes) of the declared set of processes without implementing all 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.

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.

An organization (e.g. 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.

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 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 the ISO/IEC 33000 family of standards. 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 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 processes in [Clause 6](#) of this document are selected or modified in accordance with the tailoring process prescribed in [Annex A](#). The tailored set of processes, 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.

Tailoring can diminish the perceived value of a claim of conformance to this document because it is difficult for other organizations to understand the extent to which tailoring can have deleted desirable provisions.

An organization asserting a claim of conformance to this document can find it advantageous to claim full conformance to a smaller list of processes rather than tailored conformance to a larger list of processes.

An organization can choose to claim full conformance to a selected set of processes as well as tailored conformance to other processes.

5 Key concepts and application

5.1 General

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/IEEE 24748-1 and ISO/IEC/IEEE 24748-2 on the application of life cycle processes and systems.

5.2 Software system concepts

5.2.1 Software systems

The software systems considered in this document are human-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 or significant importance to the stakeholders. It is based upon the general principles of systems engineering and software engineering. A fundamental premise of this document is 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 the platform, infrastructure systems or external systems in the operating environment. In this case, the software is the system-of-interest (SoI).

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 SoI can be viewed as a system element in another stakeholder's SoI. Furthermore, an SoI can be viewed as being part of the environment for another stakeholder's SoI.

The following are key points regarding the characteristics of SoI:

- a) Defined boundaries encapsulate meaningful needs and practical solutions;
- b) There is a logical, functional, or other relationship between system elements;
- c) An entity at any level in the SoI 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 of a system and as system elements (i.e., operators) within a system;
- f) A system can be viewed in isolation as an entity, i.e., a product; or as means of producing products, 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](#)).

NOTE Responsibility for the implementation of any system element can be delegated to another party through an agreement.

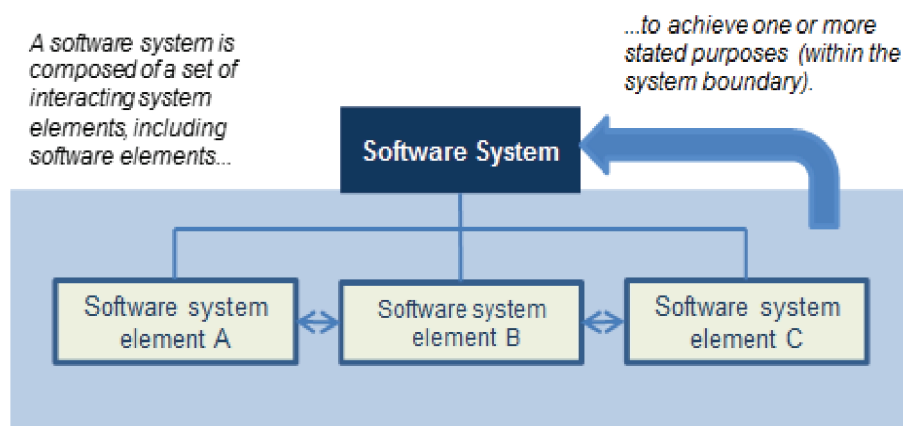


Figure 1 — Software system and software system element relationship

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](#)). The appropriate system life cycle processes are applied recursively to an SoI to resolve its structure to the point where understandable and manageable software system elements can be implemented (created, adapted, acquired, or reused). [Annex D](#) contains more information on modelling software systems.

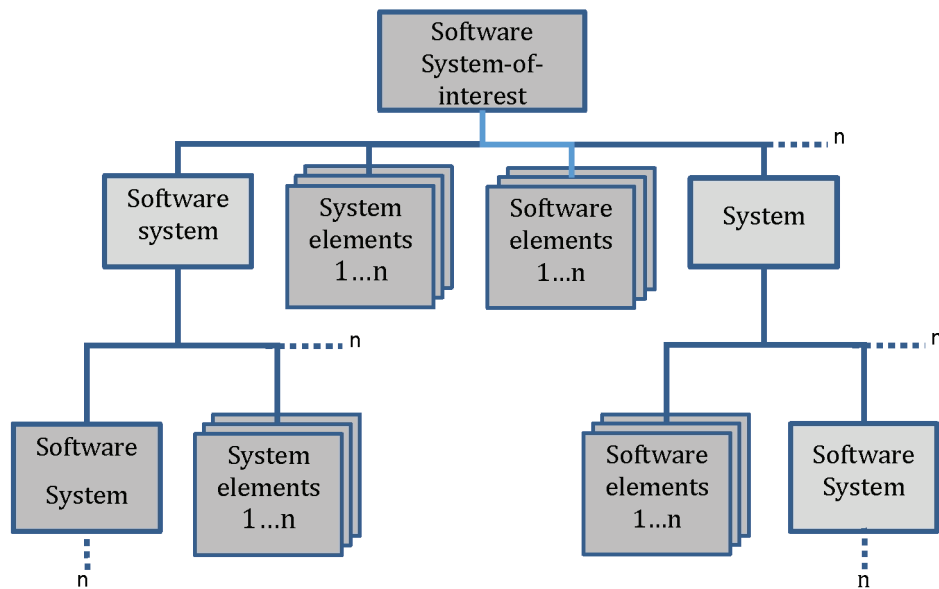


Figure 2 — Example of software system-of-interest structure

While [Figures 1](#) and [2](#) imply a hierarchical relationship among software system elements, many software systems, from one or more aspects, are not hierarchical, such as networks, digital twins, and other distributed or loosely coupled systems. Systems of systems (SoS) usually do not have a hierarchical structure.

NOTE Decomposition, or analysis, 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 Interfacing, enabling, and interoperating systems

Interfacing, enabling, and interoperating systems are outside the boundaries of the SoI, but are directly involved with it.

Interfaces among software systems are primarily sending, receiving, or exchanging data or information or controlling or monitoring functional performance. The most common types of software system interfaces are graphical user interfaces (GUI) and application programming interfaces (API). Indeed, a major part of software implementation and integration is accomplished through interfaces between existing software objects, libraries, programs, and systems. Interfaces with humans can be external or internal interfaces, i.e. human operators considered as part of the system.

Interoperating systems are a subset of interfacing systems. Systems that interact to perform a function are called interoperating systems. Interoperating systems can perform communication and networking functions, or exchange information to enable an SoI to operate reliably, securely, usefully, or efficiently, or to improve accessibility or usability. Interoperability is an important aspect in the context of SoS (see [5.2.4](#)). An

interoperating system can also receive information from the SoI for use by other systems or SoS. In software systems, interoperation typically involves the exchange and use of information, or the ability of multiple software applications to function on the same software platform and operating system. Interoperating systems, including both software and hardware, can form the infrastructure for the software SoI. Infrastructure management is so important that it is one of the organizational project-enabling processes in this document (6.2.2).

Throughout the life cycle of an SoI, essential services are obtained from systems that are not directly a part of the SoI, e.g., a modelling system, training system, release management system, security monitoring system, or diagnostic maintenance system. Enabling systems in software development include software development and test environments for target platforms. Enabling systems comprise the infrastructure and facilitate progression of the SoI through one or more stages in its life cycle. Almost all the life cycle processes in this document include activities to obtain the needed enabling systems.

An SoI and its enabling systems are normally thought of as a solution addressing stakeholder concerns. The various concerns of different stakeholders, based on their business models, drive the need to use different enabling systems to make the SoI viable in their own system solution context. For example, the concerns of an SoI supplier, an acquirer, and a user are different.

The relationship between the services delivered to the operational environment by the SoI and the services delivered by the enabling systems to the SoI is shown in [Figure 3](#). Enabling systems can be seen to contribute indirectly to the services provided by the SoI. The interrelationships between the SoI and the enabling systems can be bidirectional or a one-way relationship. In addition to interacting with enabling systems, the SoI can also interact with interoperating and interfacing 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 SoI.

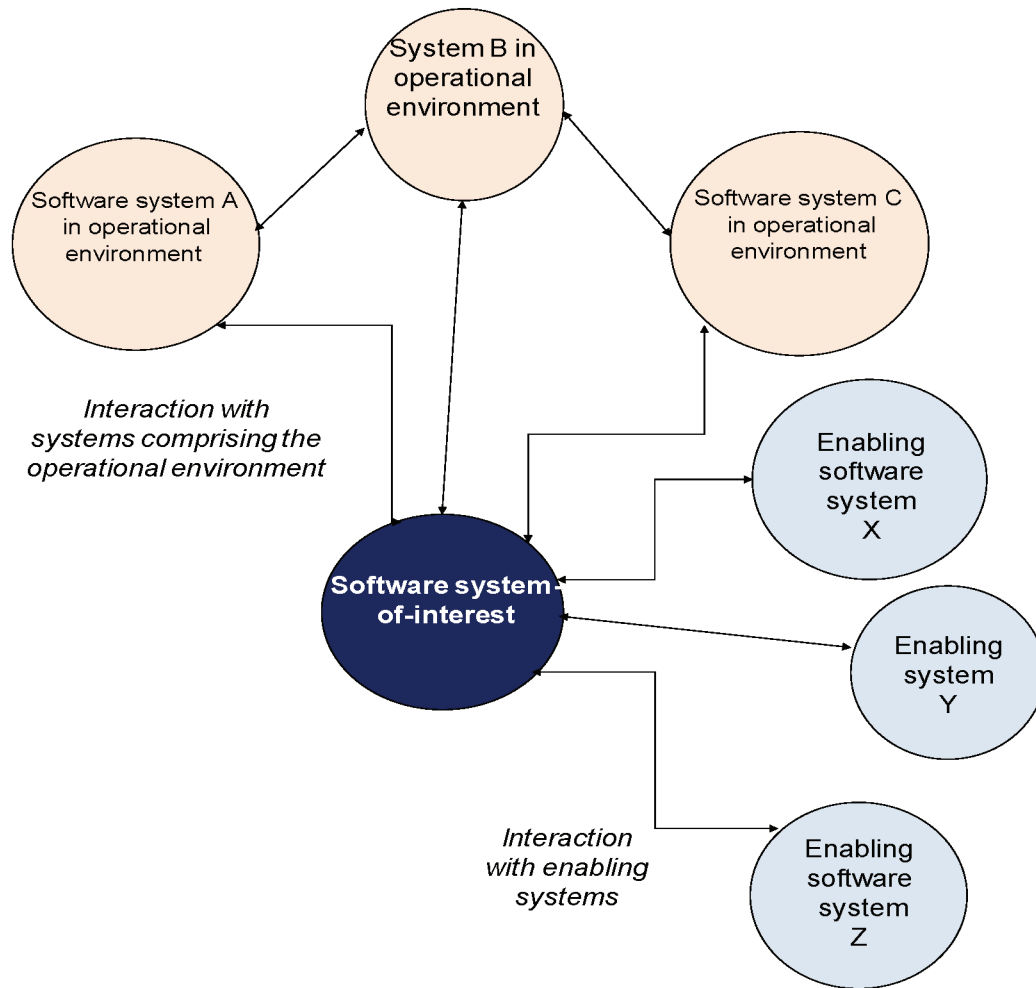


Figure 3 — Software system-of-interest, its operational environment and enabling systems

During a stage in the software life cycle, the relevant enabling systems and the SoI are considered together. Since they are interdependent, they can also be viewed as a system. Creating the enabling system can be viewed as a separate project and subsequently as another SoI. When the software is provided as a service (SaaS) it can be considered as a distinct software product (system) with its own enabling systems, or the software with its infrastructure can be part of the deliverable product or service.

Variants and options can be specified for the SoI to address the set of solution contexts. Products and services are often grouped in system families and product lines with identification of elements common to different projects, and variants and options per project (see ISO/IEC 26550 for more details). Product line engineering has more nuanced concepts of system elements and system boundaries. A product line is comprised of a combination of common core elements along with product-specific configurations and components. For example, a software product line can include product variants to operate on different operating system platforms or include different functional options for different missions or market segments. Product line engineering is about managing those core components, such as choosing to bring a specific implementation from a product into the core set.

Development of systems, products, and services often benefits from the identification of reuse opportunities between projects, including the establishment of product lines, families of products, systems, and SoS. These assets available for projects are managed through the application of the configuration management, information management, and knowledge management processes.

5.2.4 System of systems concepts

5.2.4.1 Differences between systems and SoS

An SoS is a set of systems that interact to provide a unique capability that none of the constituent systems can accomplish on its own. In the context of SoS, the relevant pieces of the Sol are, by definition, systems themselves. An SoS consists of some number of constituent systems, plus any inter-system infrastructure, facilities, and processes necessary to enable the constituent systems to integrate or interoperate.

Within an SoS, each constituent system is an independent system that forms part of an SoS. A constituent system can interact as part of one or more SoS in support of multiple capabilities. An SoS is often comprised of systems to enable it to interact in the anticipated SoS configurations. Each constituent system is a useful system by itself, having its own development, management, utilization, goals, and resources, but interacts within the SoS to provide the unique capability of the SoS. These additional attributes are what distinguish SoS from a collection of systems.

In this document, when the interaction of an Sol with an SoS is discussed, this may include one or more SoS in support of one or more capabilities. The differences between systems and SoS are not in the structure or arrangement of the individual elements, but rather in the behavioural and managerial characteristics of those elements.

5.2.4.2 Managerial and operational independence

Systems operate within a context of managerial control subject to governance. Organizations govern a portfolio of programmes through goals and objectives, subject to laws, regulations, and external agreements such as contracts. Programmes manage some number of projects to achieve those goals and objectives. Relationships between constituent systems affect the SoS. Systems that do not have any interactions with the constituent systems of a subject SoS are not part of that SoS.

An essential characteristic of the SoS is that constituent systems within the SoS are operationally independent. That is, the constituent systems fulfil some purposes on their own, separate from the SoS. While constituent systems operate independently from each other for their own purposes, they also operate interdependently with each other and other elements to produce the SoS outputs. Constituent systems are never totally independent, yet they are also never totally subservient to the SoS.

Another essential characteristic is that constituent systems within the SoS are both managerially independent and interdependent. Managerial independence suggests that the constituent systems can be managed by organizations that retain some degree of independence even though they are interdependent while participating in SoS. The implication is that these organizations can have goals and objectives for the constituent systems that differ from those of the SoS and the other constituent systems. If so, there is likely some degree of independence and interdependence of governance, as well as some degree of independence and interdependence of management.

Regardless of the means of managing the organizations, alignment (or lack thereof) in the goals and objectives affects the SoS. While some constituent systems are directed or influenced to belong to SoS, owners of some constituent systems can be unaware of the SoS. Owners of some constituent systems choose to belong on a cost/benefits basis, also to cause greater fulfilment of their own purposes, and because of their belief in the overarching SoS purpose.

5.2.4.3 Taxonomy of SoS

ISO/IEC/IEEE 21841 can be useful when describing an SoS or comparing SoS. Using essential characteristics to partition the various types of SoS provides an abbreviated nomenclature for thinking about SoS. ISO/IEC/IEEE 21841 defines a normalised taxonomy for SoS to facilitate communications among stakeholders. It also briefly explains what a taxonomy is and how it applies to the SoS to aid in understanding and communication. There are many characteristics such as scale and scope, around which taxonomies can be derived.

The SoS taxonomy in ISO/IEC/IEEE 21841 organizes the relevant aspects or essential characteristics of SoS, providing specific viewpoints that align with stakeholder concerns. This organization facilitates

communications between the various stakeholders that are involved with activities like governance, engineering, operation, and management of these SoS, and provides a reference for other related standards.

5.2.4.4 SoS considerations in life cycle stages of a system

When the SoI is part of an SoS, the application of the processes in this document is dependent on the type of SoS life cycle and impact of the SoS interactions on the SoI. ISO/IEC/IEEE 21839 can be useful as an augmentation to this document when the SoI is a constituent system within an SoS, which is the typical situation for many systems. ISO/IEC/IEEE 21839 provides a set of SoS considerations to be addressed at key points in the life cycle of an SoI which is part of an SoS. The considerations and key points align with those which are introduced in this document. Selected subsets of these considerations can be applied throughout the life of systems through the involvement of stakeholders.

SoS considerations for the SoI include the technical impact on interacting systems and their stakeholders and infrastructure. ISO/IEC/IEEE 21839 states that “this includes both systems/services on which the SoI depends and systems/services that depend on the SoI”. The SoS configurations can impose new requirements or constraints on the SoI in support of required or desired business or mission capabilities. The goal is to achieve stakeholder satisfaction for the overall mission, so that when delivered, the SoI operates effectively in the operational environment which is typically characterized as one or more SoS.

5.2.4.5 Application of this document to SoS

ISO/IEC/IEEE 21840 provides guidance for the utilization of life cycle processes in the context of SoS. However, ISO/IEC/IEEE 21840 is not a self-contained SoS replacement for this document. ISO/IEC/IEEE 21840 can be used in conjunction ISO/IEC/IEEE 15288, ISO/IEC/IEEE 21839, and ISO/IEC/IEEE 21841.

In some cases, there can be “waves” of SoS revision. In other cases, SoS changes occur continually, with many processes operating on a continuous basis to implement evolutionary change.

5.3 Organization and project concepts

5.3.1 Organizations

For the purpose of this document, any project is assumed to be conducted within the context of an organization. This context 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.

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, integrator, 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” can refer to the user or to an acquirer, from the supplier’s perspective. “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. An organization that implements the life cycle processes in this document has the responsibility of defining a suitable structure for the organization and assigning 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.

5.3.2 Organization and project-level adoption

An organization can choose to establish one or more projects for the life cycle of a system, product, or group of systems or product lines. Each project has its own life cycle of initiation, operation, and completion or transition.

Organizations using this document can strive to develop a robust set of life cycle processes that are applied repeatedly to their projects and services. 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, or whose organizational processes are specialized for non-software production, such as hardware security or manufacturing. Such a project may apply the provisions of this document directly to the project.

5.3.3 Organization and collaborative activities

Ongoing and recurring activities for one or more life-cycle processes can also be handled by a team or group on a level-of-effort basis without established projects. Organizations that use agile methods can apply the life cycle processes identified in this document, including organizational, technical management, and technical processes (and can operate under the agreement processes). Due to the increasing complexities of system solutions, it is often useful to employ collaborative and concurrent engineering approaches across the system life cycle. The following are some considerations:

- System life cycle activities are performed in a collaborative manner by involving stakeholders and subject matter experts concurrently, as practical.
- A collaborative framework can be defined within an organization or between organizations to facilitate the involvement of the range of stakeholders and experts. A collaborative framework includes shared methods, tools, and other resources, and establishes an environment for better communication and shared vision and values.
- Collaborative engineering is more important for iterative or incremental development approaches to help ensure timely feedback and communication across the stakeholders.

DevOps can be considered as an extended use of collaboration, supported by integrated tools, among development and operational groups, which can increase the velocity and the reliability of software. Site reliability engineering implements DevOps goals of collaboration and tooling, but is concerned with reliability for very large operational environments, where reliability applies to the entire infrastructure as an SoS, rather than to individual SoI.

NOTE ISO/IEC/IEEE 32675 includes principles for the implementation of DevOps to define, control, and improve software life cycle processes.

5.4 Life cycle concepts

5.4.1 Software life cycle stages

A system life cycle model is segmented by stages to facilitate planning, provisioning, operating, and supporting the SoI. This segmentation by stages provides an orderly progression of a system through established decision-making gates. 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. The stages 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. 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. In ISO/IEC/IEEE 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. Often, the development stage is subdivided more finely and in different ways. Life cycle models vary according to the nature, purpose, use and prevailing circumstances of the software system.

This document does not prescribe a sequence of processes or stages within the life cycle model. The sequence of the stages and processes is determined by project objectives and by selection of the software system life cycle model. Using stages concurrently and in different orders can lead to life cycle forms with distinctly different characteristics.

The life cycle processes defined in this document are not aligned to any specific stage in a software life cycle. All the life cycle processes involve planning, performance, and evaluation activities that should be considered for activities in 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 models 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. Software system elements, hardware system elements, and constituent systems can have different life cycles and life cycle models. A software system progresses through its life cycle as the result of activities, performed and managed by people in organizations, using life cycle processes for defining the activities. 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 specific life cycle model. Instead, it defines a set of processes that can be used in the definition of the system's life cycle. The processes described in this document support sequential as well as iterative and incremental development models.

NOTE 1 ISO/IEC/IEEE 24748-1 includes a set of steps and guidance for life cycle model selection and adaptation.

NOTE 2 ISO/IEC/IEEE 21839 provides additional guidance, grouped by life cycle stage, for SoIs that are an SoS or a constituent system in an SoS.

One oft-cited set of stages associated with software development includes elicitation, requirements, design, construction, and testing, known as the predictive sequential, or “waterfall” model. If the stages are considered as sequential, then each stage is expected 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 3 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. Royce argued that a highly sequential model is one that “invites failure.”^[95]

To deal with conditions such as incompletely known requirements and inaccurate estimates, other types of life cycle 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 life cycle models can affect software release strategies and acquisition strategies for software services.

The incremental life cycle 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 life cycle model orders the development and utilization of functionality based on risk, with the riskiest problems considered in the early increments. This approach provides some protection against cost and schedule surprises occurring late in the development cycle.

When the same process or set of processes is repeated on the same system element, the process model is referred to as iterative. The iterative life cycle model performs initial planning and then consists of a cyclic process of prototyping, testing, analysing, and refining the requirements and the solution. Iterative life cycle models repeatedly apply the life cycle processes to deliver prioritized system functions sooner, with refined or more complex elements of the system coming in later iterations.

NOTE 4 Applying the same process or set of processes to progressively more detailed system components is called a recursive process model.

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

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. The process requirements in this document are appropriate for commonly used agile techniques.

Agile methods can be applied within a variety of models. While agile methods are common in executing an evolutionary lifecycle model, they can be used in incremental lifecycle models at various stages. As stated in [5.4.1](#), this document does not prescribe any sequence of processes within a 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](#)).

NOTE 5 ISO/IEC 33202 contains additional information on agile practices.

Agile methods succeed 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.

Similarly, DevOps, which is often combined with agile, emphasizes continuous integration, continuous delivery, and frequent transitions (releases) of software.

In agile projects, concept exploration, development, construction, testing, transition, and retirement of previous software can be performed concurrently for successive iterations. Agile projects typically perform replanning concurrently with development activities. 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, iterations, 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.

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

What the models and 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. These methods help counter the risk of poor communication among multiple teams.

NOTE 6 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 7 ISO/IEC/IEEE 24748-1, ISO/IEC/IEEE 24748-2, ISO/IEC/IEEE 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.

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.

NOTE 1 Additional detail regarding this form of process description can be found in ISO/IEC/IEEE 24774.

NOTE 2 Outputs are an optional attribute for a process description. They are artefacts or information items.

The processes and process groups in this document are identical in their purpose and outcomes with those in ISO/IEC/IEEE 15288:2023, to assist in the application of life cycle processes in software engineering and systems engineering. Software-specific activities, tasks, and work products (artefacts and information items) are applied to achieve the outcomes of the processes in this document.

To emphasize this harmonization of systems and software system processes, the process purposes and outcomes are presented in boxes in [Clause 6](#).

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.

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.

5.5.4 Process views

There are 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. 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 unique activities and tasks. Instead, the description includes guidance explaining how the outcomes can be achieved by employing the activities and tasks of the various life cycle processes. The detailed information about process views is available in ISO/IEC/IEEE 24748-1 and ISO/IEC/IEEE 24774. The following International Standards provide details on some of the technical viewpoints:

- ISO/IEC/IEEE 24748-1: specialty engineering, interface management, and security;
- ISO/IEC/IEEE 15026-4: systems assurance and software assurance.

5.5.5 Tailoring

[Annex A](#), which is normative, defines the basic activities needed to perform tailoring of a process. Tailoring can diminish the perceived value of a claim of conformance to this document, 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.

NOTE See ISO/IEC/IEEE 24748-1 for additional information about selection and adaptation of live cycle models as well as selection and tailoring 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,
- d) Technical Processes.

A description of each process group is provided in [5.6.2](#), [5.6.3](#), [5.6.4](#), and [5.6.5](#).

NOTE Additional interpretation of software concepts and processes can be found in ISO/IEC/IEEE 24748-3. More information about process application for the SoS is provided in ISO/IEC/IEEE 21839 and ISO/IEC/IEEE 21840.

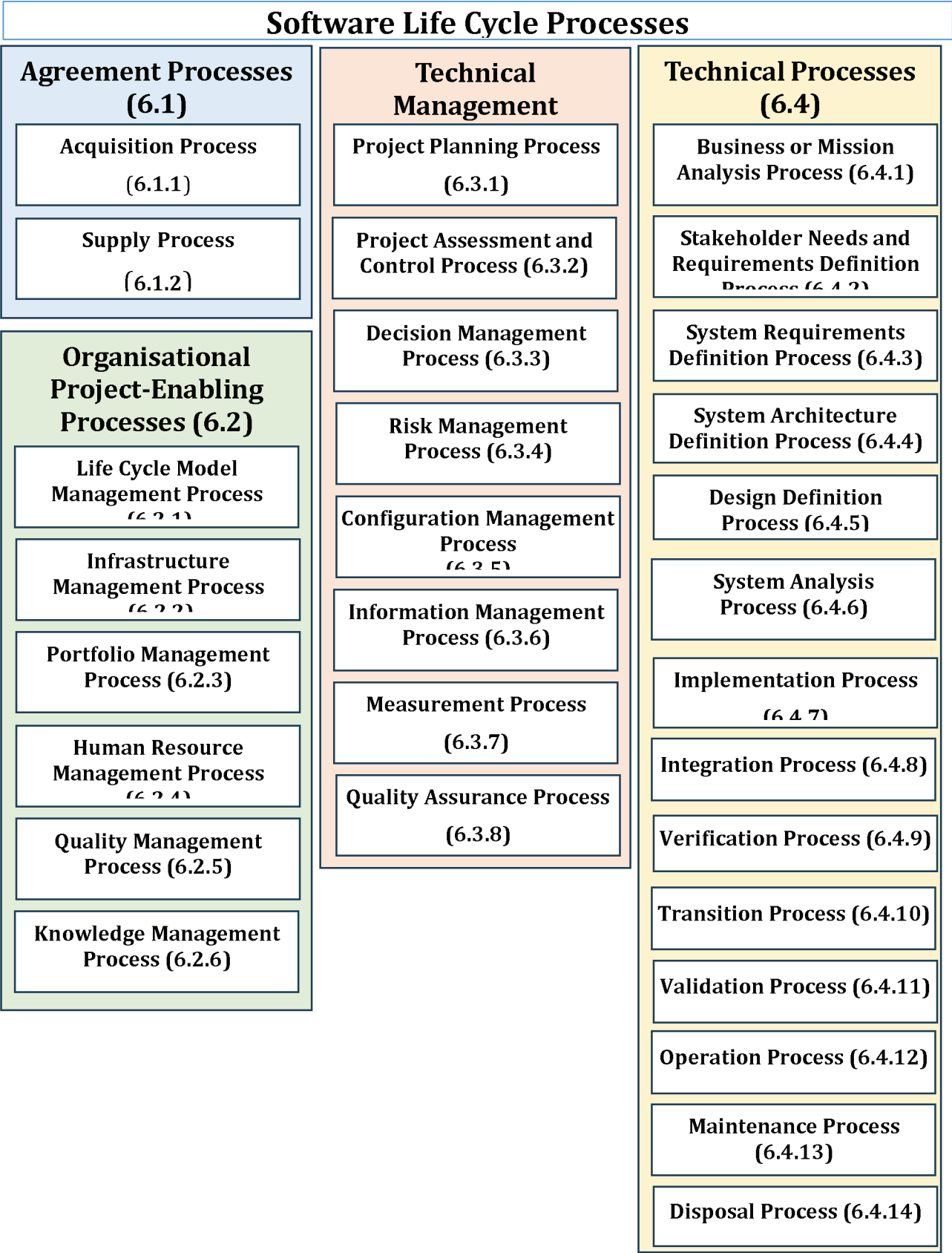


Figure 4 — Software life cycle processes

5.6.2 Agreement processes

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

Generally, organizations act simultaneously or successively as both acquirers and suppliers of 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.

These processes define the activities necessary to establish an agreement between two organizations. If the acquisition process is invoked, it provides the means for interacting with a supplier. The agreement can include products that are supplied for use as an operational system, services in support of operational activities, or elements of a system being provided by a supplier. If the supply process is invoked, it provides the means for an agreement for a product or service that is provided to the acquirer.

NOTE 1 Security is an increasing concern in systems engineering. See the ISO/IEC 27036 series 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 and ISO/IEC 27036-4.

NOTE 2 ISO/IEC/IEC 41062 includes detailed processes for software acquisition.

When an SoI participates as part of a managed SoS, its resource and capability dependencies can be reflected in the agreement processes. As needed, the agreements can include clauses for the SoS interactions and dependencies. The processes apply for the agreements between the stakeholders of the SoI and interoperating systems. If there is an external entity with some type of responsibility that spans an SoS in which the SoI is a constituent system, then management and support arrangements can be required with that external entity. The agreements establish the responsibilities and modes of support and control across the life cycle stages among the stakeholder organizations from the context of the SoI's participation in the SoS. This participation is of greater importance if the organization holds primary objectives for their constituent system that are not directly aligned with those of the SoS.

5.6.3 Organizational project-enabling processes

The organizational project-enabling processes establish the environment and activities supporting software engineering. This process group is 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 are not intended to be a comprehensive set of organizational processes that enable strategic management or ongoing governance or administrative operations of the organization.

Concepts, needs, and requirements evolve at various levels resulting from the establishment of or changes to the organizational concept of operations, strategy, or environment. The concept of operations addresses the leadership's intended way of operating the organization. This evolution is accomplished through the application of the technical processes: business or mission analysis, stakeholder needs and requirement definition, system requirements definition, system architecture definition, and design definition processes, with the support of other processes as needed. Through the iterative and concurrent use of all these processes, stakeholders are identified and insights are gained into the relationships between the concepts, needs, and requirements of the organization, stakeholders, and systems, as well as the emergent properties and behaviours of the system that arise from the interactions and relations among the system elements and environment.

The organizational project-enabling processes do not necessarily imply commercial or profit-making motives. 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.

In a typical Sol, organizational project-enabling processes establish the environment and provide the necessary resources for the conduct of projects to address system solutions. The organization establishes the processes and life cycle models to be used by projects; establishes, redirects, or cancels projects; provides resources required, including human, material and financial; and sets and monitors the quality measures for systems and other deliverables that are developed by projects for internal and external customers. These processes also provide the environment and resources for the Sol to be able to support capabilities provided by an SoS in which the Sol participates. The organizations responsible for the constituent systems implement these processes for their Sol independent of the SoS. These processes can be influenced by regulations, interface standards, or agreements for those parts of the Sol that contribute to the overall SoS capabilities.

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. Technical management processes are applied during the performance of each technical process. 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 activities 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.

In this document, the project is chosen as the context for describing processes. The same processes can also be applied in the development and performance of services, the development and support of products and product lines, or ongoing level-of-effort activities. 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 agreements as well as risk and complexity of the project.

The technical management processes do not comprise a management system or a comprehensive set of processes for project management; they are included so that system-specific technical processes can be conducted effectively.

Typically, several projects co-exist in any one organization. The Technical Management processes can be employed at various organizational levels to meet internal needs. [Figure 4](#) lists the processes contained in this process group.

Project planning and project assessment and control are key to all management practices. Software projects or approaches applying the technical management processes all have plans and are driven by those plans, including agile approaches with planning for each sprint and each release. 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 performing to a specialised management objective. The technical management processes are applied to the management of any undertaking, ranging from a complete organization down to a single life cycle process and its tasks.

Technical management processes are concerned with managing the resources and assets allocated by organization management and with applying them to fulfil the agreements into which the organization or organizations enter. The technical management processes need to include the considerations of the expected SoS interactions. The considerations span the planning, assessment, and management of resources, risks, and other factors associated with dependencies from other interacting systems. For example, the planning, assessment, and control activities need to include SoS-related cost and schedule considerations for the Sol, such as monitoring the progress in the areas of cross-system dependencies. The agreement processes are executed to negotiate the required resources across stakeholders.

5.6.5 Technical processes

The technical processes are concerned with technical activities throughout the life cycle of the Sol. 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 to meet the stakeholder requirements and achieve customer satisfaction. The technical processes are applied 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 to any element in a software system and at any stage in the life cycle. [Figure 4](#) lists the processes contained in this process group.

The technical processes define the activities that enable organization and project functions to optimise the benefits and reduce the risks that arise from technical decisions and actions. These activities enable products and services to possess the timeliness and availability, the cost effectiveness, and the functionality, reliability, maintainability, producibility, usability and other quality characteristics required by acquiring and supplying organizations. They also enable products and services to conform to the expectations, ethical perspectives, or legislated requirements of society.

These processes are often performed concurrently, iterating between one another to establish a solution that is balanced with respect to requirements, critical performance measures, critical quality characteristics, and SoS considerations. At any level of abstraction, system requirements and models are made consistent via iterations of applicable technical processes. When requirements and models are not directly capable of being implemented, the same processes are repeated recursively throughout the system.

The technical processes are performed for the provision and support of technical solutions across the life cycle stages. The SoS technical considerations apply to all the technical processes across the life cycle stages and play an especially important role in the business or mission analysis, stakeholder needs and requirements definition, system requirements definition, system architecture definition, and design definition processes.

NOTE Interface management is a set of activities that cut across the systems engineering processes. These are cross-cutting activities of the technical and technical management processes that apply and track as a specific view of the processes and system. See ISO/IEC/IEEE 24748-1 for an example interface management process view and INCOSE-TP-2003-002-5, Part III, section 3.2.4 for more information.

5.7 Process application

5.7.1 Selection and timing of process use

The life cycle processes defined in this document can be used by any organization when acquiring, using, creating, or supplying a software system. 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 [Figures 4](#) and [5](#) 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 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).

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, artefacts, 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 clause on iterative and recursive use of software life cycle processes is not meant to imply any specific hierarchical, vertical, or horizontal structure for the SoI, 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.

5.7.2 Process iteration, recursion, and concurrency

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 tasks and integration tasks 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. Iterative application of a process or set of processes should continue until such questions are resolved

Iteration between business or mission analysis, stakeholder needs and requirements definition, system requirements definition, system architecture definition, and design definition processes can help achieve a common understanding of the problem to be solved and the identification of a satisfactory solution ([Figure 5](#)). These are heavily supported by the system analysis and decision management processes. Information produced by the processes should be shared and used by all other system life cycle processes.

The repeated application of the same process or set of processes applied to successively more detailed system elements is the recursive use of processes. From a view of relations between a system and its system elements, the outputs of processes applied for a system or system element, whether information, artefacts, or services, are inputs to other processes or other system elements for additional analysis or more generalised synthesis of its system elements, to arrive at a more detailed or mature set of outcomes. Such an approach adds value to systems at successive levels in the system.

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

Concurrent use of processes can exist within a project (e.g. when design and preparatory activities for building a 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). All processes can be used in parallel with other processes. As an example, the operation and maintenance processes need to provide input to the system requirements definition, system architecture definition, design definition, and implementation processes.

5.8 Process reference model

[Figure 5](#) illustrates the interrelationships among processes defined in this document. The technical management processes are continually applied to manage and control the processes and life cycle stages.

The system analysis process provides data and information that are essential for each iteration of the systems life cycle processes, which are performed to support any of the technical management processes (6.3) and the technical processes (6.4).

The arrows in Figure 5 show general relationships that include iterative, recursive, and concurrent application. All possible relationships are not included. The actual flows or interactions between the processes for a project are determined by the project tailoring and needs. Also, the arrows are not intended to indicate any specific temporal relationships, sequences, or scheduling. The arrows between process groups or aggregations of processes within process groups are intended to indicate that the project can apply the processes in any order, can iterate between processes, and can implement them concurrently.

The technical processes described in clause 6 interact with other technical and technical management processes to provide necessary inputs and information. For example, the system analysis process provides analysis results to support trade-offs that are managed by the decision management process. Additionally, the establishment of the concepts, requirements, architecture, and design are informed by the other technical processes, such as the operations and maintenance processes.

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”. 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 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.

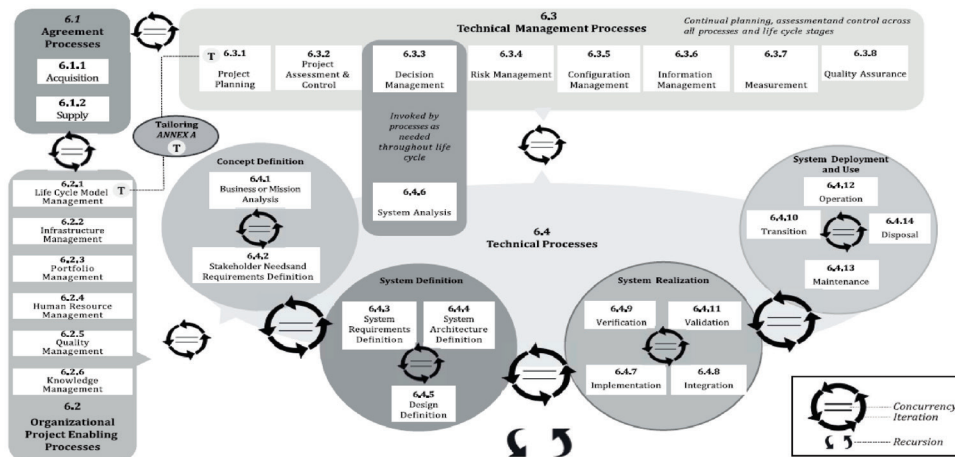


Figure 5 — Interrelationships among processes

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:

- Acquisition process – used by organizations for acquiring products or services;
- 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. Acquisition

can 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.

6.1.1.2 Outcomes

As a result of the successful performance 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.
- f) Responsibility for the acquired product or service is transferred, as directed by the agreement.

NOTE Agreements vary in formality of the agreement (written or not) and procedures for communications between the acquirer and the supplier, and as to whether they are legally enforceable contracts, and whether they are made within an organization or with external suppliers.

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 ISO/IEC/IEEE 41062 contains detailed activities for software acquisition alternatives, including custom-developed, off-the-shelf, and software as a service. It also provides software acquisition guidance for technical data rights and protection of intellectual property, safety assurance or information security.

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

- 1) Define a strategy for how the acquisition is to be conducted.

NOTE 3 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-off factors.

NOTE 4 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 5 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 6 Factors to consider in agreements between suppliers include 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 7 If a supplier is external to the organization, then the request includes the business practices and can refer to the applicable laws and regulations with which a supplier is expected to comply. Suppliers can be requested to supply evidence of their capabilities, experience, and qualifications. The request also indicates the criteria for selecting a supplier.

NOTE 8 The requirements are the defined stakeholder or the system requirements (6.4.2 and 6.4.3), depending on the acquisition approach. Security requirements are typically included, such as processes or features for data integrity, provenance data (software bill of materials, SBOM), and incident response procedures. When an external supplier proposes operations, maintenance, and support services for a proposed software product, the acquirer can agree to acquire the proposed software product with or without the proposed services. If both the proposed product and services are selected, they typically are the subjects of separate related agreements.

NOTE 9 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.

NOTE 10 The supplier's review of the request can result in questions and responses, which can lead to the acquirer's modifying its request or suppliers modifying their proposals.

2) Select one or more suppliers.

NOTE 11 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. If no supplier's proposal is found suitable, the acquisition can be terminated.

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

NOTE 12 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 13 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 security, configuration management, reporting of risks, and reporting of measures and measurement analysis.

NOTE 14 Acceptance criteria, such as acceptance testing, relate to how the product or service satisfies 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 15 In requesting a change to the agreement, the acquirer or the supplier details its specifications, rationale, and background. The agreement is modified when a change request affecting the terms of the agreement is agreed to by both the acquirer and supplier.

3) Evaluate impact of changes on the agreement.

NOTE 16 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 17 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 18 The result of the agreement modification is incorporated into the project plans and communicated to all affected parties.

NOTE 19 Agreements can specify the conditions under which the agreement may 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 20 Suppliers' processes and products can be audited, re-qualified, or periodically reviewed using the quality assurance, verification, and validation processes. This evaluation can include 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.

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 21 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.

NOTE 22 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.¹

NOTE 23 Iterative or incremental delivery and acceptance of and payment for products or services can occur as specified in the agreement

4) Close the agreement.

NOTE 24 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.

6.1.2.2 Outcomes

As a result of the successful performance 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 is transferred as directed by the agreement.

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 1 Potential acquirers can be identified 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 2 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-off 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 3 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. The agreement can define the supplier's responsibility for applying appropriate quality management, assurance, and control processes to produce suitable systems products. or services. 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.