AS/NZS ISO/IEC/IEEE 24748.3:2022
ISO/IEC/IEEE 24748-3:2020

Australian/New Zealand Standard™

Systems and software engineering — Life cycle management

Part 3: Guidelines for the application of AS ISO/IEC/IEEE 12207:2019 (software life cycle processes)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

AS/NZS ISO/IEC/IEEE 24748.3:2022

This Joint Australian/New Zealand Standard™ was prepared by Joint Technical Committee IT-015, Software and Systems Engineering. It was approved on behalf of the Council of Standards Australia on 11 May 2022 and by the New Zealand Standards Approval Board on 4 May 2022.

This Standard was published on 27 May 2022.

The following are represented on Committee IT-015:

Australian Computer Society
Australian Digital Health Agency
Australian Society for Technical Communication, NSW
Department of Defence (Australian Government)
Engineers Australia
Griffith University
Institute of IT Professionals New Zealand
IT Service Management Forum Australia
NSW Business Chamber
Systems Engineering Society of Australia
Universities New Zealand
University of New South Wales
University of Southern Queensland
University of Technology Sydney

This Standard was issued in draft form for comment as DR AS/NZS ISO/IEC/IEEE 24748.3:2022.

Keeping Standards up-to-date

Ensure you have the latest versions of our publications and keep up-to-date about Amendments, Rulings, Withdrawals, and new projects by visiting:

www.standards.org.au

www.standards.govt.nz

ISBN 978 1 76113 791 4

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

AS/NZS ISO/IEC/IEEE 24748.3:2022
ISO/IEC/IEEE 24748-3:2020

Australian/New Zealand Standard™

Systems and software engineering — Life cycle management

Part 3: Guidelines for the application of AS ISO/IEC/IEEE 12207:2019 (software life cycle processes)

Originated as SA/SNZ TR ISO/IEC 24748.3:2014.
Jointly revised and redesignated as AS/NZS ISO/IEC/IEEE 24748.3:2022.

COPYRIGHT

© ISO/IEC/IEEE 2022 — All rights reserved
© Standards Australia Limited/the Crown in right of New Zealand, administered by the New Zealand Standards Executive 2022

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher, unless otherwise permitted under the Copyright Act 1968 (Cth) or the Copyright Act 1994 (New Zealand).

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Preface

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-015, Software and Systems Engineering, to supersede SA/SNZ TR ISO/IEC 24748.3:2014, *Systems and software engineering—Life cycle management, Part 3: Guide to the application of AS/NZS ISO/IEC 12207 (Software life cycle processes).*

The objective of this document is to provide guidelines for the application of AS ISO/IEC/IEEE 12207:2019, including guidance on a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry. This document also provides guidance on defining, controlling and improving software life cycle processes within an organization or a project, and recommends methods and approaches suitable for a variety of life cycle models. The guidance emphasizes the importance of establishing a strategy, planning, and the involvement of stakeholders, with the ultimate goal of achieving customer satisfaction.

This document applies to the acquisition, supply, design and development, transition, operation, maintenance and disposal of software systems, products and services, and the software portion of any system. This document can also be applied during the acquisition of a system that contains software or as a basis for selecting, establishing and improving organizational environments.

The choice of whether to apply this document for guidance to the software life cycle processes, or AS/NZS ISO/IEC/IEEE 24748.2:2022 depends on the software system-of-interest. Both documents have the same life cycle process framework, but differ in guidance for activities and tasks to perform software engineering or systems engineering, respectively.

Use of the guidance in this document is appropriate regardless of software system size or complexity or organizational size. The process outcomes from the AS ISO/IEC/IEEE 12207:2019 life cycle processes are generic and applicable to the engineering of any software system in any size organization.

This document does not prescribe or detail a specific software life cycle model, development methodology, method, modelling approach, or technique and method. It also does not establish a management system or provide guidance on the use of any management system standard.

However, this document is intended to be compatible with the quality management system specified in AS/NZS ISO 9001:2016, the service management system specified in AS/NZS ISO/IEC 20000.1:2019, the IT asset management system specified in ISO/IEC 19770 (all parts), and the information security management system specified in ISO/IEC 27000.

This document does not repeat the detailed requirements and recommendations for purposes, outcomes, activities, and tasks for each life cycle process found in AS ISO/IEC/IEEE 12207:2019. It also does not detail information items in terms of name, format, explicit content and recording media.

This document is identical with, and has been reproduced from, ISO/IEC/IEEE 24748-3:2020, *Systems and software engineering — Life cycle management — Part 3: Guidelines for the application of ISO/IEC/IEEE 12207 (software life cycle processes)*.

As this document has been reproduced from an International document, a full point substitutes for a comma when referring to a decimal marker.

Australian or Australian/New Zealand Standards that are identical adoptions of international normative references may be used interchangeably. Refer to the online catalogue for information on specific Standards.

The terms “normative” and “informative” are used in Standards to define the application of the appendices or annexes to which they apply. A “normative” appendix or annex is an integral part of a Standard, whereas an “informative” appendix or annex is only for information and guidance.

Documents
Document Sharing Hub

© Standards Australia Limited/Standards New Zealand 2022

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Contents

Preface ii

Foreword v

Introduction vi

1 Scope 1

2 Normative references 2

3 Terms, definitions, and abbreviated terms 2
3.1 Terms and definitions 2
3.2 Abbreviated terms 2

4 Concepts for software and software systems 3
4.1 General 3
4.2 Software system concepts 3
4.3 Organizational concepts 4
4.4 Project concepts 6

5 Process and life cycle concepts 8
5.1 Process concepts 8
5.2 Life cycle concepts 10
5.2.1 Life cycle stages 10
5.2.2 Interrelationships of software processes and stages 11
5.2.3 Life cycle process models for software systems 11
5.3 Process groups 16

6 Software life cycle processes 17
6.1 Agreement processes 17
6.1.1 Acquisition process 17
6.1.2 Supply process 19
6.2 Organizational project-enabling processes 21
6.2.1 Life cycle model management process 21
6.2.2 Infrastructure Management process 22
6.2.3 Portfolio Management process 23
6.2.4 Human Resource Management process 24
6.2.5 Quality Management process 25
6.2.6 Knowledge Management process 25
6.3 Technical Management processes 26
6.3.1 Project Planning process 26
6.3.2 Project assessment and control process 27
6.3.3 Decision Management process 31
6.3.4 Risk Management process 32
6.3.5 Configuration Management process 33
6.3.6 Information Management process 35
6.3.7 Measurement process 36
6.3.8 Quality Assurance process 37
6.4 Technical processes 38
6.4.1 Business or Mission Analysis process 38
6.4.2 Stakeholder Needs and Requirements Definition process 39
6.4.3 System/Software requirements definition process 41
6.4.4 Architecture Definition process 43
6.4.5 Design Definition process 46
6.4.6 System Analysis process 48
6.4.7 Implementation process 50
6.4.8 Integration process 52
6.4.9 Verification process 52
6.4.10 Transition process 54
6.4.11 Validation process 55

AVA Documents Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

6.4.12 Operation process 58
6.4.13 Maintenance process 59
6.4.14 Disposal process 60

Annex A (informative) **Tailoring process** **62**

Bibliography **64**

IEEE notices and abstract **67**

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

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

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

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

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

This document cancels and replaces ISO/IEC TR 24748-3:2011, which has been technically revised.

The main changes compared to ISO/IEC TR 24748-3:2011 are as follows:

— revised presentation of concepts, consistent with ISO/IEC/IEEE 12207:2017;

— completely updated presentation of guidance for each life cycle process, including aspects of purpose; outcomes and outputs; activities, tasks, and approaches;

— identified closely related processes;

— identified related international standards for each process, which offer more detailed requirements and guidance.

A list of all parts in the ISO/IEC/IEEE 24748 series can be found on the ISO website.

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.

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2022
Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Introduction

The purpose of this document is to provide guidance on the application of the software life cycle processes standard, ISO/IEC/IEEE 12207:2017. Taken together, the parts of the ISO/IEC/IEEE 24748 series are intended to facilitate the joint usage of the process content of the two high-level life cycle process standards (ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015), which in turn may be used together with various more specialized lower-level process standards. In this way, ISO/IEC/IEEE 24748 (all parts) provides unified and consolidated guidance on the life cycle management of systems and software engineering. Its purpose is to help ensure consistency in system concepts and life cycle concepts, models, stages, processes, process application, key points of view, adaptation and use in various domains as the two standards (and others) are used in combination. It should help an organization to design, develop, and sustain software systems using a life cycle model.

ISO/IEC/IEEE 24748-1 provides guidance for the concepts of life cycle management applicable to both systems and software engineering. It covers fundamental concepts such as system-of-interest, stages, processes, projects, and organizations. This document focuses on and expands the coverage of those aspects and processes most relevant to software systems. A companion guidance document, ISO/IEC/IEEE 24748-2, provides similar guidance for the application of ISO/IEC/IEEE 15288:2015.

In conjunction with ISO/IEC/IEEE 24748-1, this document aids in identifying and planning the use of the life cycle processes described in ISO/IEC/IEEE 12207:2017. Since in many respects the Organizational Project Enabling processes and the Technical Management processes are quite similar for software systems to those used for any type of system, this document concentrates on specific guidance for the Technical processes and how they can be effectively used during the software life cycle. ISO/IEC/IEEE 24748-5 focuses on the Technical Management processes, especially Project Planning and Project Assessment and Control, as applied to software projects. The proper use of these processes can contribute to a project being completed successfully, meeting its objectives and requirements for each stage and for the overall project.

This document elaborates on factors, 'best-practice' or typical approaches and methods that should be considered when applying ISO/IEC/IEEE 12207:2017. It does this in the context of the various ways in which ISO/IEC/IEEE 12207:2017 can be applied. It is intended to be useful in a variety of software life cycle situations, including the use of agile methods, which are the most widely used on all types and sizes of projects.

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Australian/New Zealand Standard

Systems and software engineering — Life cycle management

Part 3: Guidelines for the application of AS ISO/IEC/IEEE 12207:2019 (software life cycle processes)

1 Scope

This document is a guideline for the application of ISO/IEC/IEEE 12207:2017. This document establishes guidance to implement a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry. This document provides guidance on defining, controlling, and improving software life cycle processes within an organization or a project. This document recommends methods and approaches suitable for a variety of life cycle models. The guidance emphasizes the importance of establishing a strategy, planning, and the involvement of stakeholders, with the ultimate goal of achieving customer satisfaction.

This document applies to the acquisition, supply, design and development, transition, operation, maintenance, and disposal (whether performed internally or externally to an organization) of software systems, products, and services (including software as a service (SaaS)), and the software portion of any system. Software includes the software portion of firmware. The guidance on processes, activities, and tasks in this document can also be applied during the acquisition of a system that contains software. The guidance in this document can also be used as a basis for selecting, establishing, and improving organizational environments, e.g., methods, procedures, techniques, tools, and trained personnel.

In the context of this document, 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 guidance to the software life cycle processes, or ISO/IEC/IEEE 24748-2, depends on the SoI. At a high level, both documents have the same life cycle process framework, but differ in guidance for activities and tasks to perform software engineering or systems engineering, respectively.

The processes and process groups in this document are identical in their purpose and outcomes with those in ISO/IEC/IEEE 12207:2017 and in ISO/IEC/IEEE 15288:2015, with one exception: the System/Software Requirements Definition process of ISO/IEC/IEEE 12207:2017 and this document has a different name from the System Requirements Definition process of ISO/IEC/IEEE 15288:2015.

Use of the guidance in this document is appropriate regardless of software system size or complexity or organizational size. The process outcomes from the ISO/IEC/IEEE 12207:2017 life cycle processes are meant to be generic and applicable to the engineering of any software system in any size organization.

This document does not prescribe nor detail a specific software life cycle model, development methodology, method, modelling approach, or technique and method. The variety of ways for implementing software (e.g., development of new code, integration of existing open source components and commercial products, or modifications to existing software, including transition to new platforms) make it impossible to detail specific procedures.

This document does not establish a management system or provide guidance on the use of any management system standard. However, it is intended to be compatible with the quality management system specified by ISO 9001, the service management system specified by ISO/IEC 20000-1, the IT asset management system specified by ISO/IEC 19770 (all parts), and the information security management system specified by ISO/IEC 27000.

Clause 6 provides guidance on aspects of purposes, outcomes, activities, and tasks in ISO/IEC/IEEE 12207:2017. However, this document does not repeat the detailed requirements and recommendations for purposes, outcomes, activities, and tasks for each life cycle process found in ISO/IEC/IEEE 12207:2017. Clause 6 also provides references to specialized standards that provide more

© Standards Australia Limited/Standards New Zealand 2022

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

detailed requirements and guidance for the various processes and information products (information items). This document does not detail information items (process inputs and outputs) in terms of name, format, explicit content and recording media.

NOTE ISO/IEC/IEEE 15289 addresses the content for life cycle process information items (documentation).

2 Normative references

There are no normative references in this document.

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

No terms and definitions are listed in this document.

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

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

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

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

NOTE For additional terms and definitions in the field of systems and software engineering, see ISO/IEC/IEEE 24765, which is published periodically as a "snapshot" of the SEVOCAB (Systems and software Engineering Vocabulary) database and is publicly accessible at www.computer.org/sevocab.

3.2 Abbreviated terms

API	application program interface
CM	configuration management
COTS	commercial-off-the-shelf
FCA	functional configuration audit
IDEF	Integration DEFinition
MOE	measure of effectiveness
MOP	measure of performance
NDI	non-developmental item
PCA	physical configuration audit
PII	personally identifiable information
PRM	process reference model
QA	quality assurance
QM	quality management
SaaS	software as a service

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

SME	subject matter expert
SoI	system-of-interest
SoS	system of systems
TPM	technical performance measure
V&V	validation and verification
VSE	very small entity
WBS	work breakdown structure

4 Concepts for software and software systems

4.1 General

This clause is included to help explain essential concepts as applicable to software and software systems. While understanding concepts does not give the ability to immediately apply them without further thought and work, it is the foundation for their practical use in different project, organizational and life cycle environments.

4.2 Software system concepts

The application of ISO/IEC/IEEE 12207:2017 presupposes an understanding of system concepts. A system is a combination of interacting elements organized to achieve one or more stated purposes. Software is the subsystems or elements of a system consisting of computer programs, related procedures, associated documentation, and data pertaining to the operation of the subsystem or element. Software occurs in most systems, even if it is not the predominant element of interest as it is in a software system.

For the purposes of this document, software systems are considered as created by humans and utilized to provide services in defined environments for the benefit of users and other stakeholders. These systems may be configured with one or more of the following: hardware, software, services, humans, processes (e.g. review process), procedures (e.g. operator instructions), facilities, and naturally occurring entities (e.g. water, organisms, minerals). A system may be considered as a product or as the services it provides. A system element is a member of a set of elements that constitutes a system. A system element is a discrete part of a system that can be implemented to fulfil specified requirements.

NOTE 1 Additional discussion regarding systems, systems of systems, and system structure, is provided in ISO/IEC/IEEE 24748-1. ISO/IEC/IEEE 24748-2 provides more information on concepts related to system life cycle management.

System concepts are directly applicable to software systems. An underlying principle of ISO/IEC/IEEE 12207:2017 is that software engineering applies similar processes to systems engineering, but that software is the leading method for system requirements realization. Consequently, processes are aligned and adapted for methods and approaches relevant for software.

NOTE 2 As applicable to all systems and projects, guidance to concepts is found in ISO/IEC/IEEE 24748-1. ISO/IEC/IEEE 24748-2 includes guidance more specifically applicable to systems where hardware or other non-software elements are the primary concern.

Typically, the system engineering approach to develop a system design is described as a hierarchical, top-down process of systematic decomposition of the system into its subsystems and elements (components). This top-down approach has been traditionally applied to the architecture of software systems as well, from the top-level system-of-interest down through the lowest system element level of the software system structure, such as a line of code. However, the malleable, easily refactored nature of software leads to a different way of considering the software structure. The common practice of

Documents Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2022
Full document details: AS NZS ISO IEC IEEE 24748.3-2022

producing a minimum of new code and integrating a software system from available components (open source modules, application program interfaces (APIs), software services, support software such as a database management system and a web browser) leads to a more holistic view of its structure. Software systems include a number of components for processing information that can be directly or indirectly related to the software functions and requirements. In compartmentalized software, often none of the sub-systems or components can be considered "top-level" in a hierarchy. Traceability of a high-level stakeholder requirement to each specific subsystem can be difficult.

Characteristic properties at the boundary of a SoI arise from the interactions between subordinate systems. Whatever the boundaries chosen to define the software system, the concepts and models in this document are generic and permit a practitioner to correlate or adapt individual instances of life cycles to its software concepts and principles.

Enabling systems are required for each life cycle process and are typically integrated as an infrastructure for concurrent performance of multiple development, test, and operational processes. Enabling systems are deployed throughout the software life cycle to provide the SoI with support as needed. Each life cycle stage can require one or more enabling systems. An enabling system has its own life cycle; when ISO/IEC/IEEE 12207:2017 (or ISO/IEC/IEEE 15288:2015, if applicable) is applied to it, it then becomes a SoI.

Selection of methods and tools considers software size and complexity, project duration and the number of contributing organizations. Selection of tools should be based on connectivity to other tools that provide inputs or use its output. Ease of use, need for training before successfully using the tool, and availability of administrative support and enabling systems are also factors in tool selection and use. Use of a method or tool does not replace the standard process to be followed but should support the set of activities of a selected process.

4.3 Organizational concepts

An organization is a person or a group or people and facilities with an arrangement of responsibilities, authorities, and relationships. 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. When an organization, as a whole or a part, enters into a contract, it is a party. Organizations are separate bodies, but the parties may be from the same organization or from separate organizations.

ISO/IEC/IEEE 12207:2017 is intended to be applicable to organizations of all sizes and governance structures, from very small entities (VSE) of fewer than 25 people working on a single project, to large-scale organizations engaged in system of systems (SoS) sustainment for years or decades. The requirements of ISO/IEC/IEEE 12207:2017 are usually stated in terms of "the organization" or "the project," rather than by individual titles or roles. The extent to which tasks and roles can be differentiated depends on the resources available to the organization. Responsibilities that involve checking, validating, or verifying should be assigned to separate persons where the size of the organization permits. In some cases, only one person may be assigned to cover all tasks and take most of the roles and responsibilities.

Modern organizations strive to develop a robust set of life cycle processes that are applied repeatedly to the projects of the organization. To accommodate that need, ISO/IEC/IEEE 12207:2017 is intended to be useful at either the organization level or at the project level. An organization can adopt the standard and supplement it with appropriate policies, procedures, and tools.

A project of the organization typically conforms to the organization's processes rather than conforming directly to the standard. In some cases, projects may be executed by an organization that does not have an appropriate set of processes applied consistently at an organizational level. Such a project may apply the provisions of the standard directly to a project.

An organization may perform one or more processes as part of its services. A process may be performed by one organization or more than one organization, with one of the organizations being identified as the responsible party. A single project may involve multiple organizations working together as

© Standards Australia Limited/Standards New Zealand 2022

AVA Documents Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

partners. Such a project should use ISO/IEC/IEEE 12207:2017 to establish common terminology, as well as information flows and interfaces among the organizations to enhance communication.

In this document, as in ISO/IEC/IEEE 12207:2017, an organization (or a party) derives its name from the process it is currently performing, for example, it is called an acquirer when it performs the Acquisition process. In ISO/IEC/IEEE 12207:2017, the agreeing parties are called the acquirer and the supplier.

The application of ISO/IEC/IEEE 12207:2017 does not require a specific organizational structure for projects. There are numerous organizational models which can be successfully used to perform the processes of ISO/IEC/IEEE 12207:2017—as long as it is clear who is accountable for providing sufficient resources and for producing each outcome. Organizations can use hierarchical management structures or team-focused, agile work groups. The teams may include representatives specializing in each stage of the life cycle or each life cycle process.

Adaptation and application of ISO/IEC/IEEE 12207:2017 on a project, in which many persons may be legitimately involved, depends on responsibility and accountability. The teams or groups need to be given the appropriate responsibility and authority for doing the work required to meet the project requirements, for example the activities and tasks of a process. A party is identified with overall responsibility and accountability for that entire process, even though the execution of individual tasks may be by different people or groups.

Whether the existing basis for processes is an older version of ISO/IEC 12207 or some other reference point, the fundamental starting point is to identify all the changes needed to go from that basis to ISO/IEC/IEEE 12207:2017. If the existing process basis is an older standard, the amount of changes can be noticeably less than if a different process basis is in use.

NOTE ISO/IEC/IEEE 12207:2017, Annex I provides a mapping from the processes of ISO/IEC 12207:2008 to the processes in ISO/IEC/IEEE 12207:2017. See also ISO/IEC/IEEE 12207-2 for more detailed mappings of outcomes, activities, and tasks.

Whatever the reason for an organization's application of ISO/IEC/IEEE 12207:2017 is, a suggested implementation and transition strategy consists of the following:

a) Identify the desired life cycle model and goals for the organization and the project. Implementing a single process, without considering how it relates to other explicit or de facto processes, is less likely to be beneficial. The Life Cycle Management and Business and Mission Analysis processes can be useful for this effort. If no obvious link is established between this project and the organization's business focus, then lasting commitment to achieve the project goals will be difficult if not impossible to maintain.

b) Plan the implementation. The Transition and Project Planning process can be useful. Identify roles and responsibilities of the project team/organization, assigning a single point of responsibility for each process. In many cases, one individual or organization may be responsible for more than one process, particularly in small projects or organizations.

c) Tailor ISO/IEC/IEEE 12207:2017, if applicable.

d) Conduct pilot project(s).

e) Formalize the approach. A project management plan or change management plan can be useful.

f) Institutionalize the approach, so that the process is supported by top management commitment, organizational policies and procedures, and is used consistently and automatically throughout the project or organization. This also involves measuring performance and implementing process improvement again as necessary.

This strategy is typical of a Change Management approach to introduce changes into an organization or project. The strategy described above may be repeated several times within a project or across an organization as additional processes are implemented or improved.

As with any program which results in changes to work practices, it is essential that the top management within the affected organization is visibly committed to implementing and supporting the changes. It

© Standards Australia Limited/Standards New Zealand 2022

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

a two-party situation, this can be initiated by a contract and then, as for general organizational use, policies are established with support of top management of both parties.

Bringing all the stakeholders together in this effort is critical: even one area left out that should have been in the planning can materially disrupt applying the new basis. One way of proceeding is for a small group to develop a checklist of transition activities, such as the following:

a) documentation changes, including flow and nomenclature of policies and procedures;

b) staff training needs;

c) responsibility changes, including need for new agreements;

d) impacts on tools and databases; and

e) changes in the inputs required by and outputs from each process.

The initial checklist should then be used by an immediately following, larger, group of all stakeholders to work through what other items need to be added and what the specific changes are for each item on the checklist. Repeated reviews of checklist drafts should be held to find the final few surprises.

Once there is a detailed listing of the changes derived in this, or equivalent, manner, the time and cost impact of each needs to be assessed. Then further analysis of the sequence of implementing the changes is necessary. The group should explore phasing in changes in a way that minimizes cost, project disruption and the potential for adverse human reactions. Readiness criteria should be developed for starting each step of the transition, as well as checks for successful completion after each step. Quantitative metrics should be developed and used.

Throughout, a core group should be maintained to oversee the change from one basis to another, with periodic meetings of the entire group of stakeholders.

When a project or organization is already in a steady state, i.e. where the processes have been established and institutionalized, then the implementation strategy can be shortened, considering the risks and opportunities of the project.

4.4 Project concepts

A project is an endeavour with defined scope and start and finish dates, undertaken to work on a product or service in accordance with specified resources and requirements. Typically, a project exists to satisfy an agreement by providing the desired deliverables to the quality expected. For this purpose, ISO/IEC/IEEE 12207:2017 provides a set of Organizational Project-Enabling processes.

Any project is assumed to be conducted within the context of an organization. This is important because a software 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.

To perform needed operations and transformations upon software systems during their life cycles, the organization creates and monitors projects. The organization both constrains and supports a project. Examples of such organizational constraints and supports are the following:

— sets standards, policies, and procedures by which projects are carried out within the organization;

— initiates, redirects or terminates projects according to business opportunities and strategies;

— provides requested resources including physical and human within availability and financial constraints;

— provides infrastructure support;

— manages the overall quality of software systems produced by a project for internal or external customers.

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Projects have defined scope, resources (including time) and focus. The organization can establish projects that are contiguous with stages in the software life cycle (for example, a project to develop software concepts and requirements, followed by another project to integrate, validate, and verify the software system). Projects can also align with the various organizations responsible for different software components. The scope can involve managing all of the stages of the life cycle, a subset of the stages, one or more defined processes, or one or more process activities. Although any software system should be sustained over its full life cycle, it is common for projects to span only portions of that life cycle.

The time scale of projects can vary, for example, one day or several years. The focus of the project is related to the software and its elements in some form of system structure or stage partitioning. The project places certain demands on the organization and the organization places demands on the project. The project requires physical infrastructure, financial and human resource support to carry out project work.

Relationships can exist between a project and other projects, and subprojects. A subproject is a set of resources and tasks organized to undertake a portion of a project. A subproject may be considered a project by those assigned the work.

Many software "projects" involve continuing delivery of capabilities to sustain a software system. Ongoing sustainment efforts, responsive to changes in the environment of the SoI or to new functional and non-functional requirements, handle a continuous stream of work packages, with the timespan limited only by established budgetary or financial reporting restrictions.

Figure 1 illustrates the concept of a software system and its relationship to an organization and its projects.

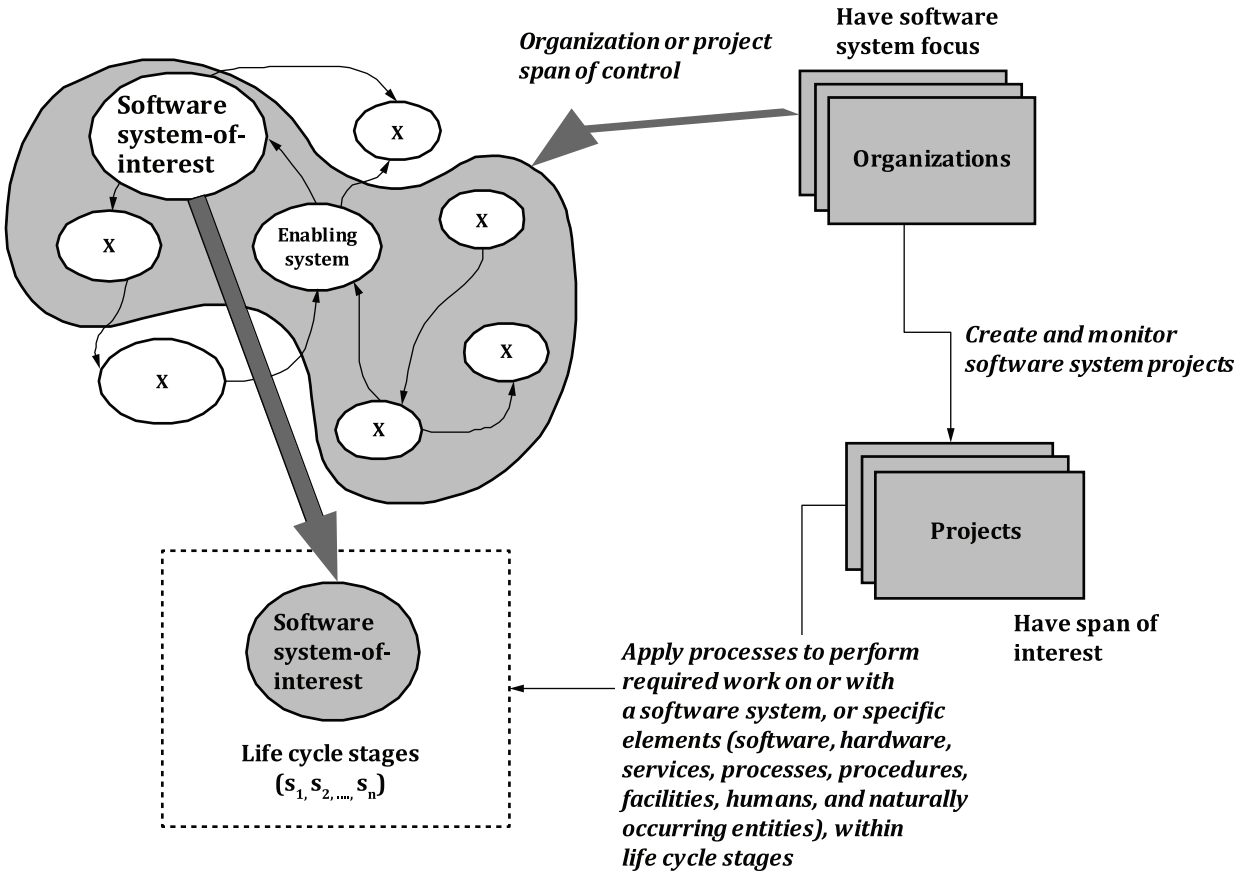

Figure 1 — Organizations, projects, and software systems

Documents Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2022
Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Some enabling systems are under direct control of the project. The software and those enabling systems make up the project span of control.

5 Process and life cycle concepts

5.1 Process concepts

A process is an integrated set of activities that transform inputs (for example, a set of data such as requirements) into desired outputs (for example, a set of data describing a desired solution). The software engineering processes in ISO/IEC/IEEE 12207:2017 are defined so that each process has a distinct purpose and outcomes (results). The purpose statement provides the overall rationale for the use of the process. The outcomes are the expected observable results from carrying out the activities of a process. The outcomes provided for each process in ISO/IEC/IEEE 12207:2017 provide a benefit that motivates selection and execution of that process. Attainment of the results fulfills the purpose for performing the process. However, outcomes are not identical with process outputs, which are tangible artifacts and services that can be evaluated. Processes are managed through controls and performed using enabling mechanisms (Figure 2).

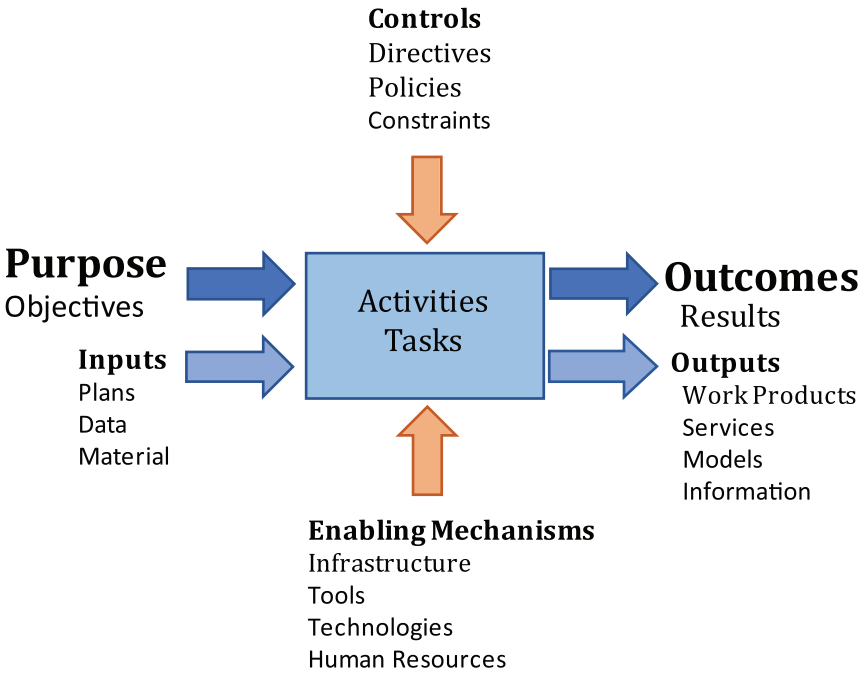

Figure 2 — Process elements

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

Inputs can come from outside an organization or project, or from other processes that precede or accompany the process being examined. Outputs can go to other processes or back to the same process (recursive processing) inside the organization, project (or both), or they can go outside the project or organization, or both. Examples of inputs to, and outputs from, a process include the following:

a) information, such as requirements, interface or architecture definitions;

b) data, such as measurements and test reports;

c) material that either ends up in the output or is consumed in producing the output;

d) services that are part of a chain of services, such as setting up a computer prior to, or coincident with establishing an account.

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

The source of process controls and constraints include organizational or organization management directives and constraints; governmental regulations and laws; project agreements; interfaces with processes used on other systems for which the project is responsible; and internal standard policies and procedures of the organization. Government regulations and laws do not require any specific software design or implementation. They can be viewed as constraining the software system requirements. For consistent application, controls are typically automated through the enabling systems, or applied using methods, procedures, and techniques by the workforce.

ISO/IEC/IEEE 12207:2017 describes the set of processes that are applied to the life cycle of any software system. Therefore, ISO/IEC/IEEE 12207:2017 is designed so that its processes can be applied for a software project of any type, size and complexity, whether focused on tangible products, services, or a mix of both. The processes are designed to be used whether the software is considered as a stand-alone element, or a part of the total software system.

While the processes identified in ISO/IEC/IEEE 12207:2017 are intended to model all the work of software engineering, they do not form a closed loop model. The outputs of one process are not directly identified as inputs for the other processes; the inputs of one process are not always outputs of the other processes. Controls, directives, governance, policy, and organizational strategy and vision drive each process.

Another reason why it is not feasible to create a cohesive, closed-loop model of the software engineering processes is that they are not necessarily sequential. Cohesiveness of a process does not imply that the process can only be performed by one part of an organization at one stage of the software life cycle. Performing a process is not locked to any one time, part of the life cycle, or party. Each process can be applied at any time during the system life cycle and repeated using available inputs at different stages in the life cycle.

The life cycle processes of ISO/IEC/IEEE 12207:2017 can be specialized into process views to reflect specific activities, tasks, and methods appropriate to particular concerns, such as information assurance (security), interfaces, ethics, safety, or various business domains.

NOTE 2 ISO/IEC/IEEE 12207:2017, Annex E presents several examples of process views.

Process performance is accomplished through activities and tasks. An activity is a set of cohesive tasks. A task is a requirement, recommendation, or permissible action, intended to contribute to the achievement of one or more outcomes of a process. A task is expressed in the form of a requirement, self-declaration, recommendation, or permissible action.

The processes, activities, and tasks in ISO/IEC/IEEE 12207:2017 are arranged in a general sequence for ease of presentation. This positional sequence does not dictate the life cycle model sequence. It is intended that the software project select, order, adapt, and iterate the processes, activities, and tasks as applicable or appropriate, using the Life Cycle Model Management process (see 6.2.1).

The processes in ISO/IEC/IEEE 12207:2017 typically follow a similar pattern in the presentation of their activities. It is helpful to be aware of this pattern if an organization wants to combine or subdivide processes or to create new specialty processes. It is more effective when combining tasks from different processes if the tasks relate to the same generic activity. When developing sub-processes, it is better to include at least one activity of each generic type, or to refer to another process where the generic activity is handled for that sub-process.

Table 1 presents the model with examples from a couple of processes.

Table 1 — Model of generic activities within a process

Generic activity	Example activities/task in the Implementation process	Example activities in the Design definition process
Strategize and plan (Plan)	6.4.7.3, a) Prepare for implementation	6.4.5.3, a) Prepare for software system design definition
Perform (Do)	6.4.7.3, b) Perform implementation	6.4.5.3, b) Establish designs related to each software system element

© Standards Australia Limited/Standards New Zealand 2022

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

Table 1 *(continued)*

Generic activity	Example activities/task in the Implementation process	Example activities in the Design definition process
Evaluate and decide (Check/Act)	6.4.7.3, b) 4) Evaluate software unit and affiliated data or other information according to the implementation strategy and criteria.	6.4.5.3, c) Assess alternatives for obtaining software system elements
Manage outcomes and outputs: Preserve and present artifacts and information items	6.4.7.3, c) Manage results of implementation	6.4.5.3, d) Manage the design.

5.2 Life cycle concepts

5.2.1 Life cycle stages

A software SoI has a life cycle that consists of multiple stages through which the system passes during its lifetime. Within a life cycle stage, processes are performed as required to achieve stated objectives. The progression of a system through its life is the result of actions managed and performed by people in one or more organizations using the processes selected for a life cycle stage. Stages may be interdependent and overlapping, may be of differing durations, and may iterate or be applied recursively.

The size and complexity of a software system affect the work of an organization or project. For example, the tasks performed to accomplish an activity of a system life cycle process or the type and form of work products from application of the processes can be affected. The users of this document are responsible for selecting a life cycle model for the project and mapping the processes, activities, and tasks in ISO/IEC/IEEE 12207:2017 into that model. The organization is responsible for selecting and applying appropriate methodologies, methods, models, and techniques suitable for the project. An organization (for example, an automobile company or medical equipment supplier) or a domain group of an organization (for example, a government defense agency or industry group) often has a unique view of the software system life cycle to control the passage from one life cycle stage to the next. The organization view includes management-focused activities that are used to form both milestones and decision gates. The organization uses these milestones and gates as decision points where investment decisions can be made as to whether a software system should be continued to the next life cycle stage; be modified, cancelled or retired; or be deferred while the plans for the next stage are revised. These milestones and decision gates can be used by organizations to contain the inherent uncertainties and risks associated with costs, schedule, and functionality when a system is created or utilized. To meet the exit criteria of a decision gate, a software system has to be appropriately engineered and the appropriate work products (artifacts and information items) need to be produced to provide decision-making information and required deliverables. Thus, planned engineering activities need to take place during each system life cycle stage to obtain the outcomes and meet the purpose of the stage or a set of stages.

Per 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. For software, production of multiple copies is usually not so significant a step as for hardware systems, and other life cycle stages are more common. 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 (Figure 3). Another typical set of stages is Conceptualization of a Need, Realization, Utilization, Evolution, and Disposal. A third example of typical stages is Concept, Development, Operation and Maintenance. Another example of software system life cycle stages includes Needs determination, Concept exploration and definition, Demonstration and evaluation, Engineering and development, Production and manufacturing, Deployment and sales, Operations, Maintenance and support, and Retirement.

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

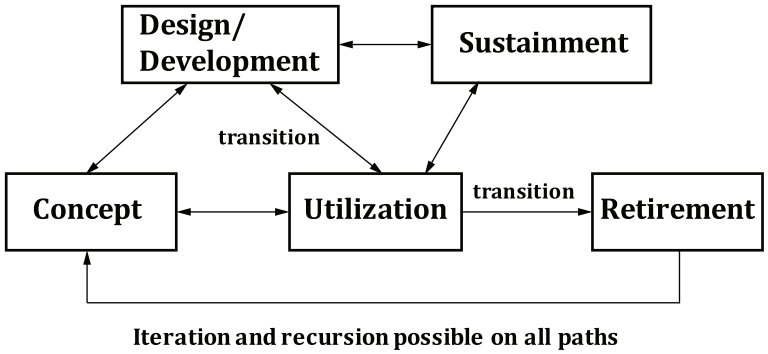

Figure 3 — Typical software system life cycle stages

5.2.2 Interrelationships of software processes and stages

For many reasons, there is not one process per stage. Some processes can be started in early stages for risk mitigation and optimisation of the project. Some complementary processes are performed concurrently to achieve a life cycle stage. With incremental and iterative life cycle models, stages occur several times in the life cycle associated with the same processes.

A manager of a system life cycle stage typically selects the appropriate set of life cycle processes to meet the exit criteria and other stage objectives. For example, the Business or Mission Analysis process, Stakeholder Needs and Requirements Definition process, and System/Software Requirements Definition process are used concurrently and iteratively to develop the concept for a software system. The Architecture and Design Definition processes are used to design the solution for each element in the system. Application of these processes can be highly iterative in order to arrive at the desired design solution. At the Realization stage, the Implementation, Integration, Transition, Verification, and Validation processes are used in a highly iterative way to produce the software deliverable. During any of the later life cycle stages, a manager can use the Operation process, Maintenance process, and Disposal process to manage the software system while it performs its required functions or is serviced to meet requirements. During earlier life cycle stages, the same processes can be used to help manage the implementation of the system as well as affect the disposal of work products that are no longer needed. The Operation process and Maintenance process should be used to allow the stakeholders to achieve the service or product benefits sought from the system, at the level desired and over the extent of time intended, when they initially defined their stakeholder requirements.

5.2.3 Life cycle process models for software systems

5.2.3.1 General

The life cycle process model of ISO/IEC/IEEE 12207:2017 is intentionally identical to that of ISO/IEC/IEEE 15288:2015 in process names, purposes, and outcomes. To some extent, their activities and tasks are also similarly worded. This approach has advantages in enabling consistent engineering project planning for all kinds of systems. However, this identical model can over-emphasize or under-emphasize some technical aspects of software engineering. The rigorous presentation of numerous processes can be misinterpreted, implying that each process on every project has to be invoked, documented in a written plan, performed, accepted in one or more gated reviews, and results retained in detailed written artifacts. This heavyweight view of the process model has led some software development organizations to avoid the use of the standards, considering them too costly and time-consuming to apply. However, ISO/IEC/IEEE 12207:2017 is intended to be usable in developing process models suitable for all types and sizes of software engineering efforts.

NOTE 1 ISO/IEC/IEEE 24748-7 provides a more rigorous set of activities and tasks for the system life cycle.

Typically, organizations choose sequential, incremental, or evolutionary life cycle models, as described further in 5.2.3.2 to 5.2.3.4.

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2022
Full document details: AS NZS ISO IEC IEEE 24748.3-2022

The processes, activities, and tasks in ISO/IEC/IEEE 12207:2017 are presented in sequence for reference. In some cases, the sequence represents a possible order of performing processes in stages and activities and tasks within processes. However, this positional sequence does not dictate the life cycle model sequence. The organization or software project should select, order, adapt, and iterate the processes, activities, and tasks as applicable or appropriate.

The reasons for applying ISO/IEC/IEEE 12207:2017 processes within an organization can include such situations as the following:

— validating the adequacy of an existing method, e.g., a method that was developed in-house or adopted and extensively modified;

— adapting an existing method to cater for the risks associated with moves into new market sectors where more rigor is required because of perceived risks;

— developing a new method, e.g. to meet the needs of a new organization; this includes organizations created through mergers or business alliances where several process models and procedures already exist;

— managing the introduction of new technology; examples include the automation of existing manual processes, or a change in the methods and tools used to implement a software product;

— evaluating an internal capability of a party to meet agreed criteria, e.g. as part of an acquisition review process;

— establishing a benchmark upon which improvement programs can be developed through an audit.

Implementation of the processes in ISO/IEC/IEEE 12207:2017 can have various scopes:

— use on a single project, either internal to an organization or as part of a two-party contract;

— concentration on some key processes or even a single process where there is expected to be some gain or reduction in risk for an organization;

— adoption across a range of projects in a phased introduction; or

— adoption across all projects and within all parts of the organization; this approach is more feasible for a very small organization.

In practice, some of the life cycle processes, system analysis, measurement, and decision management, are never performed independently but are always an adjunct to another software engineering process. Two processes are often inseparable: it is considered essential to perform software verification (often by testing) at each stage of software development (implementation). In another example, software design, integration, implementation, and validation are usually performed simultaneously and often by the same team members, and the distinction of separately performed processes can be overlooked: all these processes are often considered as parts of a larger stage of software development.

NOTE 2 In ISO/IEC 12207:2008, Validation and Verification activities which are now rigorously separated into distinctive processes were sometimes inserted as part of the same Integration process. That approach reflected that the tasks were closely associated.

The use of DevOps has confirmed the tendency to merge various processes that in the past were more likely to be assigned to distinct teams with separate controls. (DevOps is expanding the work of a software development team and cooperation with the operations team to include management of the software release (Transition) process and the sustainment of software in the operational environment).

Moreover, support for these closely coupled life cycle processes is increasingly automated, and some processes require a significantly reduced level of human intervention compared to practices detailed in previous versions of ISO/IEC 12207 and ISO/IEC TR 24748-3:2011. For example, a human may set up the configuration management (CM) system controls, but how a new day's work is integrated with other branches of the software code being produced that day does not involve human interaction to perform the formal Implementation, Integration, or CM processes of ISO/IEC/IEEE 12207:2017.

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

The life cycle models for software systems development are not necessarily linked to the software delivery or release schedule. The effort required for transition, the need to utilize software revisions immediately, and the ability of the software system and the customers to handle frequent incremental changes influence whether incrementally produced or evolutionary changes are released frequently or on a regular (e.g., quarterly or annual) schedule.

5.2.3.2 Sequential life cycle model

For new or replacement systems that have extended development cycles (several years) before delivery of the operational system, a sequential approach can be appropriate. The sequential approach has defined decision gates so that an organization can manage an orderly progression of the system from conception through retirement. Many systems, such as those produced for the aircraft or automotive industries, use a similar approach, with development and thorough test before a new automobile model or software system is introduced. Projects using this approach face many challenges, including cost control, funding changes, technology changes, workforce retention, and final customer or acquirer requirements satisfaction. These challenges are created because of the long period from establishing the initial requirements for the embedded software system to the deploying of the system in the marketplace.

This sequential approach can also be applicable to modernization of legacy systems. The software engineering is done on the system being enhanced and its related elements. The impact on the SoI does need to be analysed and where conflicts are revealed, the changes to higher-level systems and the SoI need to be made or the requirements for the applicable system need to be revised.

Because of the long duration of implementation using the sequential approach, several risks should be considered and resolved:

a) expectations and requirements related to the system can change over the years of implementation;

b) knowledgeable workers on teams can turn over;

c) decision making personnel in the organization can change;

d) customer personnel in the acquirer's organization can change;

e) suppliers of system elements and related services can go out of business or change technologies;

f) technical risks can be unresolvable for extended periods;

g) technical obsolescence can arise during a long implementation.

Opportunities can be associated with the sequential approach:

a) the deliberate, stepwise refinement approach whereby the progress of system implementation is carefully evaluated at each milestone allows system quality and risks to be evaluated and investment decisions confirmed before progressing to the next stage of implementation, production lot, or delivery to market;

b) all system capabilities can be delivered at the same time;

c) in-service modification decisions allow determination of whether to do maintenance, a major modification, or to retire the system from service;

d) old systems can be simultaneously retired from service or withdrawn from the market.

In summary, the sequential approach can be effective and efficient for engineering systems where the requirements are well known and stable, deliveries or releases are infrequent, and the system has an extended life span.

Documents
Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2022
Full document details: AS NZS ISO IEC IEEE 24748.3-2022

5.2.3.3 Incremental life cycle model

The incremental approach applies to organizations that develop (and usually, release) new versions of a product at more frequent planned intervals. The system realized as a result of the concept stage can be a first version. However, it is possible to apply incremental approaches to the software development stage while still using a sequential approach to infrequent software releases.

In this document and in ISO/IEC/IEEE 12207:2017, agile is considered as a method, not as a life cycle process model. Agile can be described as the use of highly iterative methods for the development of requirements and working system increments.

NOTE ISO/IEC/IEEE 12207:2017, Annex H has additional discussion on the application of agile to software engineering. ISO/IEC/IEEE 26515 describes the work of the information developer on an agile team.

Typically, the overall capabilities of the eventual software product are envisaged at the start of system implementation. However, a limited set of capabilities is allocated to the first version. With each successive version, more capabilities are added until the last release fully incorporates the overall capabilities.

Iterative life cycle models involve the operation and support of each version in parallel with the implementation, utilization and support of successive versions. Early versions of the system and support for those versions can be phased out as newer versions are released and used by the customer base or a block modification to earlier versions can be made to incorporate the new capabilities of a later version.

This approach applies mainly to systems that rely on new, enhanced capability versions of the system to be introduced in short intervals so as to remain competitive in the marketplace. Examples include information technology systems such as business systems, medical systems, and routing and firewall systems. Iterative approaches often begin with an 'Iteration Zero" for such non-deliverable tasks as infrastructure establishment, collecting and prioritizing features to be developed, and developer training.

The incremental approach has associated risks that should be considered and resolved before adopting this approach:

a) Initial versions of the system can have such a limited set of capabilities that customers can be dissatisfied and not be interested in buying the next version.

b) Versions marketed with too short an interval can cause customer dissatisfaction with the cost to upgrade or the retraining costs.

c) Costs for developing and frequently updating training and information for users (time and money) can increase.

d) Expectations may not be met if customers desire the full capabilities in the first version.

e) Poor results may be realized if requirements are not as well understood as originally thought.

f) Unplanned technology changes or competitor system capabilities can require re-direction of the implementation and have a significant impact upon costs and schedule for subsequent versions.

g) The customer may change the requirements as the implementation progresses.

Opportunities can be associated with the incremental approach:

a) Acquirer requirements for early capabilities can be satisfied.

b) The prototypes developed for each early milestone can have a place in the market.

c) Early introduction of the system, even with limited capabilities, can enable exploitation of the marketplace.

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC IEEE 24748.3-2022

d) Consumers and other customers have come to expect frequent improvements in software products, e.g., to respond to security issues, and be wary of products that have not been updated for extended periods.

5.2.3.4 Evolutionary life cycle model

The evolutionary approach also applies to organizations that market new versions of a product at regular or planned intervals. The major difference of this approach with the incremental approach is that the full capabilities of the future versions of the system are not known when an evolutionary implementation is undertaken.

Initially the requirements for the system are partially defined and then refined with each successive version of the system as lessons learned from the use of an early version are translated into new desired capabilities. Often, a new version with enhanced capabilities can replace an earlier version, or a block modification can be made to the earlier version to incorporate the new capabilities of a later version. In this case, implementation of new versions can be done serially or in parallel with partial overlapping. As with versions developed using the incremental approach, different versions can be operated and supported in parallel. Particular care should be taken, however, to maintain configuration information for each version so that users can find operation, training and support procedures appropriate to the version being used.

This approach applies to complex systems for which requirements are not well understood even though the need for the system is understood and approved. These are typically one-of-a-kind or low-quantity production systems. Example systems can include custom information technology systems, military information technology systems, and specific information technology security systems. This approach is also useful for systems that have to achieve quality in use.

During the production stage, either one or a few systems can be produced and delivered, or a large quantity production can be initiated that can continue into the utilization and support stages. The utilization and support stages are typically the longest period of this life cycle and can last for many years. Major systems realized using this approach often have an operation life of decades with modifications using technology refreshments and technology insertions made to sustain the system and lengthen its useful life.

The evolutionary approach has associated risks that should be considered and resolved before adopting this approach:

a) Full capabilities can be preferred at the same time.

b) Training costs can be unacceptable for moving to the next version.

c) There can be uncertainties related to determining future requirements.

d) There can be uncertainties with respect to planning the schedule release of the next version.

e) Configuration control can be a problem.

f) Inappropriate early use of a product prototype in production can cause issues.

Opportunities can be associated with the evolutionary approach:

a) Acquirer requirements for an early capability can be satisfied.

b) Customer feedback can be used to enhance the capabilities of a future version of the system.

c) The prototypes developed to satisfy an early milestone can have a use in the market.

d) Early introduction of a limited capability system can enable countering a competitor threat.

e) Emerging technologies can be exploited.

Documents Document Sharing Hub
© Standards Australia Limited/Standards New Zealand 2022
Full document details: AS NZS ISO IEC IEEE 24748.3-2022

5.3 Process groups

The software life cycle processes are presented in four process groups: Agreement, Organizational Project-Enabling, Technical Management, and Technical processes. These groups characterize the relationship of the process to the software system and to the software engineering project (Figure 4).

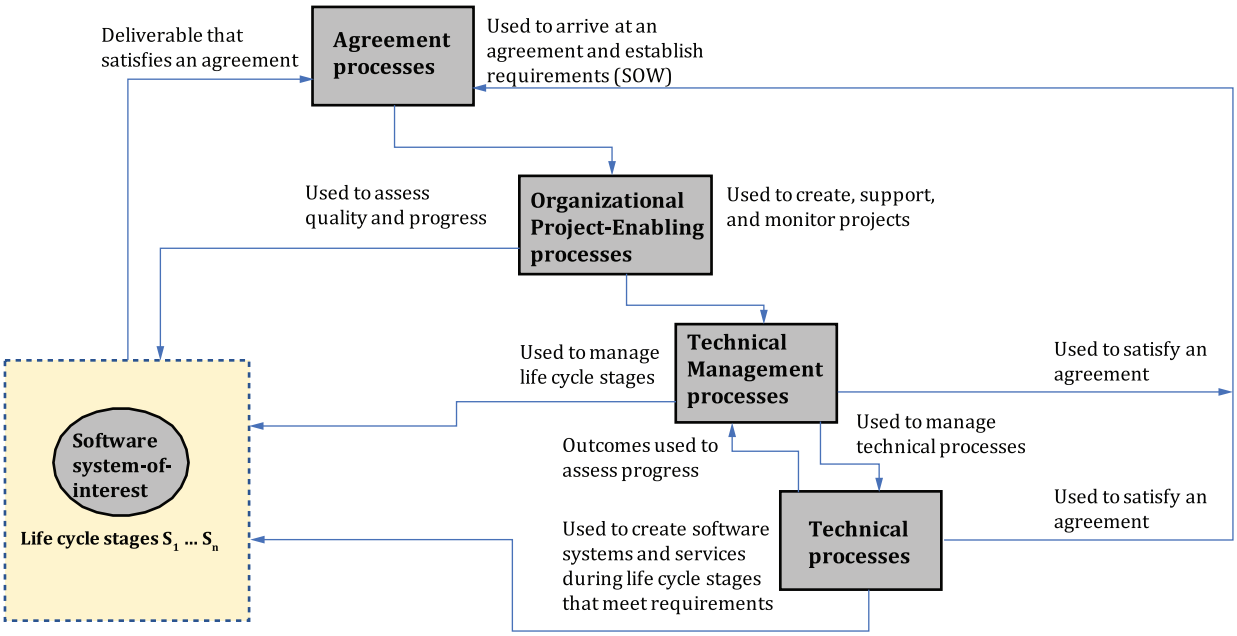

Figure 4 — Relationship of software engineering process groups with the software system

Each of the process groups is involved during the course of a typical project. The project performs the Project Planning process that can consist of updating the plans used to form the response to the acquisition request or the applicable plans from the previous life cycle stage. Appropriate teams are assigned the work required to meet planned requirements. These teams do the work associated with application of the Technical processes to obtain the work products required. The Project Assessment and Control process, Decision Making process, Risk Management process, CM process and Information Management process should be used to monitor, control, and assess the Technical process outcomes to be able to keep work within acceptable cost, schedule, and risk in order to meet performance requirements for the system. The Organizational Project-Enabling processes and Technical Management processes are implemented to provide support to the project, and to review the project, as appropriate.

The Organizational Project-Enabling processes are not assumed to be adequate to operate a business, nor are the Technical Management processes assumed to be sufficient to manage a project. Technical management focuses on managing the technical aspects of the software engineering work, rather than performing general project management. The Organizational Project-Enabling and Technical Management processes, considered as a collection, are intended to state the minimum set of outcomes that the software engineering project requires from the organization. They focus on the tasks and activities related to software engineering, not on the overall governance, business, financial, human resource, or information management policies and procedures of the organization.

Organizations generally use ISO/IEC/IEEE 12207 as part of an effort to improve software products and services through improving their software engineering processes. This may be in conjunction with other process assessment and capability determination methods. As explained in ISO/IEC/IEEE 12207:2017, Annex C, it can be used as a process reference model (PRM) for process assessment. However, neither ISO/IEC/IEEE 12207 nor this document provide a maturity model. All the activities and tasks in ISO/IEC/IEEE 12207 are presumed to be achieved at a level 2 of a typical 5-level maturity

© Standards Australia Limited/Standards New Zealand 2022

Documents
Document Sharing Hub
Full document details: AS NZS ISO IEC IEEE 24748.3-2022