INTERNATIONAL STANDARD

ISO/ IEC/IEEE 24748-6

First edition
2023-07

Systems and software engineering — Life cycle management —

Part 6: System and software integration

Ingénierie des systèmes et du logiciel — Gestion du cycle de vie — Partie 6: Intégration du système et du logiciel

Reference number
ISO/IEC/IEEE 24748-6:2023(E)

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

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

Published in Switzerland

Contents

Page

Foreword ..... **v**
Introduction ..... **vii**
1 Scope **1**
2 Normative references **1**
3 Terms, definitions and abbreviated terms **1**
3.1 Terms and definitions 1
3.2 Abbreviated terms 3
4 Conformance **3**
4.1 Conformance to processes 3
4.2 Conformance to information item content 3
4.3 Full conformance 3
4.4 Tailored conformance 4
4.4.1 Processes 4
4.4.2 Information items 4
5 Integration concepts **4**
5.1 General 4
5.2 Interface concept 4
5.3 Aggregation and synthesis concepts 5
5.4 Integration concept 5
5.5 Relationship of integration to life cycle processes 6
5.6 Progressive interface definition 7
5.7 Integration over the life cycle 7
5.8 Iteration and recursion in integration 8
5.8.1 General 8
5.8.2 Iterative application of processes 8
5.8.3 Recursive application of processes 8
5.9 Regression testing in integration 8
5.10 Integration enabling systems 8
6 Integration process planning and application purposes **9**
6.1 General 9
6.2 Integration planning and application guidelines 9
6.2.1 General 9
6.2.2 Integration strategy 9
6.2.3 Efficiency considerations in the integration strategy 10
6.2.4 Integration context 11
6.2.5 Roles and competencies of integration team 11
6.2.6 Methods used to perform integration 12
6.3 Integration process application requirements and guidelines 20
6.3.1 General 20
6.3.2 Purpose 20
6.3.3 Outcomes 22
6.3.4 Activities and tasks 23
6.4 Other processes used in relationship to integration process application 31
6.4.1 General 31
6.4.2 Agreement processes 31
6.4.3 Organizational project-enabling processes 31
6.4.4 Technical management processes 31
6.4.5 Technical processes 32
6.5 Integration of systems-of-systems 37
6.6 Integration throughout a system life cycle 38
7 Information item requirements **38**
7.1 General 38

7.2 Integration plan 38
Annex A (Informative) **Coupling matrixes** **40**
Bibliography **42**
IEEE notices and abstract **43**

Foreword

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

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

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

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

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

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

ISO/IEC/IEEE 24748-6 was prepared by Joint Technical Committee ISO/IEC 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 first edition cancels and replaces ISO/IEC TS 24748-6:2016, which has been technically revised.

The main changes are as follows:

— changed from a Technical Specification to an International Standard;

— updated to reflect the process requirements of ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2023;

— added material specific to software integration and systems-of-systems integration, as well as system aggregation.

A list of all parts in the ISO/IEC/IEEE 24748 series can be found on the ISO and IEC 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 and www.iec.ch/national-committees.

Introduction

Both ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207 include an integration process that focuses on aggregating the elements comprising a system. The integration process depends on a clear understanding of interfaces of all kinds and the use, possibly repeated, of the verification process and the validation process.

Systems that this document are concerned with are as described in ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288, i.e. systems that are human-made, comprised of any mixture of products and services and can be configured with one or more of the following: hardware, software, data, humans, processes (e.g. a review process or processes for providing service to users), procedures (e.g. operator instructions), facilities, materials and naturally occurring entities (e.g. water, organisms, minerals).

The purpose of this document is to elaborate and facilitate the usage of the integration process given in ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 by providing requirements and guidance for the planning and performing of that process, including requirements for the information items to be provided for systems and software integration, considering:

— the underlying concepts of aggregation, integration, interface, synthesis, verification, and validation;

— the possible composition of that human-made system;

— the life cycle stages of a system at which one or more parts of the integration process can occur;

— the context of the domain in which the system functions.

For life cycle process information items (documentation) described in ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207, ISO/IEC/IEEE 15289 summarises requirements for their content and provides guidance on their development. Although this document identifies additional required information items with related content for the integration process, it does not require a specific name, format, recording media or explicit details for population of the information item's required content.

NOTE This document is intended to be consistent with the other parts of the ISO/IEC/IEEE 24748 series.

Systems and software engineering — Life cycle management —

Part 6: System and software integration

1 Scope

This document:

— provides supplemental requirements and guidance for the planning and performing of the integration processes given in ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207;

— provides guidance on the relationship between the integration process and other life cycle processes.

— specifies requirements for information items to be produced as a result of using the integration process, including the content of the information items.

This document is applicable to:

— those who use or plan to use ISO/IEC/IEEE 12207 or ISO/IEC/IEEE 15288, or both, on projects dealing with human-made systems, software-intensive systems, and products and services related to those systems, regardless of the project scope, methodology, size, or complexity;

— anyone planning or performing integration activities to aid in ensuring that the application of the integration process and its relationships to other system life cycle processes conform to ISO/IEC/IEEE 15288 or ISO/IEC/IEEE 12207.

2 Normative references

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

ISO/IEC/IEEE 12207:2017, *Systems and software engineering — Software life cycle processes*

ISO/IEC/IEEE 15288:2023, *Systems and software engineering — System life cycle processes*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC/IEEE 15288, ISO/IEC/IEEE 12207 and the following apply.

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

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

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

— IEEE Standards Dictionary Online: available at: 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 Systems and Software Engineering Vocabulary (SEVOCAB) database and is publicly accessible at www.computer.org/sevocab.

3.1.1
aggregate, noun
result of combining one or more physical, logical, or both, system elements

3.1.2
driver
launcher
external dynamic component or simulator that activates the performing of functions of an *aggregate* (3.1.1) of system elements

3.1.3
interface
point at which two or more logical, physical, or both, system elements or software system elements meet and act on or communicate with each other

3.1.4
integration
process of planning for and aggregating a progressively more complete set of physical, logical, or both, system elements and activating their *interfaces* (3.1.3) to synthesize a system or part of a system whose properties can be verified and possibly validated

Note 1 to entry: An outcome of integration is a system whose properties can be verified and possibly validated.

3.1.5
logical interface
input-output flow between two or more system elements or software system elements and the function that determines it

Note 1 to entry: A logical interface can be functional (as in process steps), informational (as in a communications link), or other.

3.1.6
physical interface
physical link between two or more system elements

Note 1 to entry: A physical interface can be solid, liquid, gas, vacuum, thermal, electromagnetic field, or other.

3.1.7
stub
computer programs and data files built to support software development and testing, but not intended to be included in the final product

3.1.8
synthesis
result of combining the constituent elements of separate physical or logical, or both, entities into a single or unified entity

3.2 Abbreviated terms

API	application program interface
ISC	integration services components
MBSE	model-based systems engineering
NDI	non-developmental item
QA	quality assurance
SDP	software development plan
SEMP	systems engineering management plan
SoI	system-of-interest

4 Conformance

4.1 Conformance to processes

Conformance requirements for the integration process are specified in ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288 and further elaborated in 6.3. One supplemental integration process outcome is added in 6.3.3. If using ISO/IEC/IEEE 15288:2023 outcomes, it would become outcome h). If using ISO/IEC/IEEE 12207:2017 outcomes, it would become outcome i).

4.2 Conformance to information item content

Clause 7 provides requirements for a number of information items to be produced during the life cycle of a system.

A claim of conformance to the information item provisions of this document means that

— the required information items stated in this document are produced, and

— the information items produced demonstrate conformity to the content requirements defined in this document.

Clause 7 contains the requirements for the content of the information items in this document.

In this document, each information item is described as if it were published as a separate document. However, information items are considered to be conforming if the information items are unpublished but available in a repository for reference, divided into separate documents or volumes, or combined with other information items into one document. It is not required to treat every topic in this document in the same order, using the same wording as its title, or with the same level of detail. That depends on the nature of the system, implementation methods, life cycle model, and scope of the project.

4.3 Full conformance

A claim of full conformance to this document is equivalent to claiming conformance to the integration process requirements identified in 4.1 and the requirements for the information items cited in 4.2.

4.4 Tailored conformance

4.4.1 Processes

This document makes the following provision for a claim of tailored conformance to the integration process: ISO/IEC/IEEE 12207:2017, Annex A and ISO/IEC/IEEE 15288:2023, Annex A provide requirements regarding the tailoring of system and software life cycle processes. Those provisions should be used to tailor the integration process requirements identified in 4.1 as the basis for a claim of tailored conformance to the integration process as elaborated in this document.

4.4.2 Information items

When this document is used as a basis for establishing a set of information items that do not qualify for full conformance, the requirements in Clause 7 are selected or modified in accordance with this clause. The tailored text, for which tailored conformance is claimed, shall be declared. Tailored conformance is achieved by demonstrating that requirements for the information items, as tailored, are satisfied.

5 Integration concepts

5.1 General

This clause presents the concepts that apply to system integration, including software systems. These concepts embody the definitions in ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 and set the basis for explaining the application of those concepts. The central concepts that this clause presents are those of interface, aggregation, synthesis, and integration.

There are conceptual relationships between the integration process and system and software life cycle processes of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207, which this clause presents. This clause also presents the concepts associated with system integration as a process over the life cycle of a system, including software systems.

Finally, this clause presents concepts on iteration and recursion in integration, and regression testing.

5.2 Interface concept

An interface has a set of logical or physical characteristics required to exist at a common boundary or connection between system elements, systems, or the environment external to the SoI.

NOTE 1 Throughout this document, statements referring to "system elements" inferentially can include elements of other systems, or the external environment.

A physical interface is a physical link that binds two or more system elements within the SoI, or one system element of the SoI with one or more elements external to the SoI. A physical interface can be considered a system element.

A logical interface consists of an output flow, or an input flow, or a bi-directional, i.e. transactional, flow between two or more elements of the system so that the elements can exchange some mix of energy or information items.

NOTE 2 Table 1 provides examples of system element integration involving physical interfaces (e.g. material and thermal example of Table 1) and logical interfaces (e.g. the software-to-software example of Table 1).

The progressive definition of interfaces is an intrinsic part of applying the technical processes of ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 and is critical to the success of integration. Inadequately defined interfaces are common failure points in complex systems. Such inadequately defined interfaces commonly include the points where independent systems or system elements not necessarily made of the same technology meet and communicate with each other. Consideration for indirectly coupled elements (e.g. via control mechanisms or shared memory in software systems) often involves complex

analysis. Technical processes' activities and decisions should fully define interfaces so that integration, and subsequent verification of system elements' properties, can be planned and performed successfully.

ISO/IEC/IEEE 12207:2017, E.5 and ISO/IEC/IEEE 24748-1:2018, D.5 provide a process view for the management of interfaces and identify the purpose and outcomes with selected life cycle process activities and tasks to facilitate the identification, definition, design, and management of interfaces of the SoI.

5.3 Aggregation and synthesis concepts

The integration of a system is based on the notions of aggregation and synthesis. For example, wings, body, engines, hardware and software controls, air traffic procedures, and information items, are aggregated along with other elements not listed here to synthesize an aircraft operating in its intended environment. A synthesis performs functions not possible from the individual elements that are aggregated. An aggregate has a functional consistency that allows the performance of subsequent verification actions and possibly validation actions. Each aggregate is characterized by a configuration that specifies the implemented system or software elements that are aggregated and their configuration status. Aggregation can be done iteratively or recursively.

5.4 Integration concept

Integration encompasses planning for and aggregating a progressively more complete set of elements and activating their interfaces to synthesize a part of a system that can be verified and possibly validated in keeping with strategy decisions for validation process implementation. Integration enables interoperation among some set of elements to satisfy their requirements and provide a basis for integrating a yet more complete part of the system. The end result is a system or software system that is integrated within itself and at its interfaces for the system's intended purpose and use.

Integration of the system can involve a mix of the hardware, software, data, humans, processes (e.g. processes for providing service to users), procedures (e.g. operator instructions), facilities, materials and naturally occurring entities of which the system is composed. Hence, system integration can involve interfacing and interoperating such things as mechanical items (e.g. assembly), functions (e.g. software-based functions), people and equipment, thermal flows, sets of processes, human constructs to naturally occurring entities.

NOTE 1 ISO/IEC/IEEE 15288:2023, 5.2.3 further explains concepts for interfacing systems and interoperating systems.

A nuclear power facility is an example of a system that includes a mix of every type of elements considered in ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 and involves both products and services. Table 1 gives illustrative examples of the types of elements that are to be integrated for the system to function properly in its operational environment over its life cycle. Table 1 illustrates the point that verification is typically done following integration, depending on the verification strategy and priorities, whereas validation can depend on factors other than successful confirmation that the interface has been established and checked, and its properties verified.

Table 1 — Integration examples

System elements being integrated	Types of interfaces involved	Verification follows?	Validation follows?
Fuel and cladding to make reactor control rods	Material, thermal	Yes	Yes, but most likely done as part of validating the assembled reactor.
Operator control panels	Mechanical, electrical, functional, human-machine interface, procedural	Yes	Most likely yes. Each type of control can be verified individually, then some combination of controls can be validated, that is, validation can occur at a higher level in the system.

Table 1 *(continued)*

System elements being integrated	Types of interfaces involved	Verification follows?	Validation follows?
Reactor control and monitoring operating system	Software, human-machine interface, procedural	Yes	Most likely yes. Each type of control can be verified individually, then some combination of controls can be validated, that is, validation can occur at a higher level in the system.
Reactor rod control software to flux density monitoring software	Software to software	Yes	No, but validated at a higher level, such as software to display
Facility server farm	Hardware, software, data, thermal, facilities, processes	Yes	Probably, at several points
Reactor to cooling medium, such as a lake or river	Naturally occurring entity, facility, thermal	Yes	Probably
Reactor generator to power grid	Hardware, functional	Yes	Yes

NOTE 2 Some of the integration results in products, some in services. In this example, electrical power to the grid is a product of the reactor, while electrical power from the grid at the point of use is a service. Table 1 is notional, for illustrative purposes only, and is not intended to be exhaustive.

NOTE 3 IEEE 1012 provides requirements and guidance for system, software, and hardware verification and validation processes while the ISO/IEC/IEEE 29119 series provides detailed support for software testing.

Enabling systems are integrated, just as systems-of-interest, and an SoI and its enabling systems are interfaced. If an SoI and its enabling system are interfaced the SoI and its enabling system can be considered as a system-of-systems, or two systems, or as a new SoI, as a matter of perspective, per the treatment in ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 24748-1.

Integration can be considered successfully completed when all of the required interfaces are checked and pass the required interface specifications for the integration step. Additional confirmation of the successful completion of integration is when subsequent verification and possibly validation of the interface properties is successful.

Planning and performing of integration is done in conjunction with other system or software life cycle processes. Integration planning is mostly done in conjunction with requirements, architecture, design, and test processes, while performing integration is mostly done in conjunction with verification and validation processes.

A system can be integrated during a single stage, or involve multiple stages, of its life cycle and can be integrated more than once within a single life cycle stage, such as when a system is repeatedly modified during a lengthy stage of operation and maintenance, or when capabilities are added in discrete increments.

5.5 Relationship of integration to life cycle processes

ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207 recognize that any process can be invoked by any other process and, conceptually, that applies to the integration process. In practice, the extent to which integration is addressed when planning or performing other processes, as well as the extent to which other processes are addressed when planning or performing the integration process, varies extensively. Table 2 shows the relationship of some the other processes to the Integration process at a conceptual level. Clause 6 addresses more specifics to consider as part of the integration planning or performing guidelines.

Table 2 — Conceptual relationship of integration to life cycle processes

ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 process groups	Relationship to integration process when planning integration	Relationship to integration process when performing integration
Agreement processes	Agreements for system or software elements obtained from a supplier, or elements provided to a supplier, include requirements to be satisfied for integration that supports successful verification, validation and deployment. Agreement of interfaces between supplier and customer.	System or software elements obtained from a supplier are checked as specified in the agreement before acceptance for integration with the SoI.
Organizational project-enabling processes	Provisions of the life cycle model management process, infrastructure management process, quality management process, and knowledge management process include capabilities for any interaction with the Integration process during preparation, performing, or outcome handling.	Integration draws on the infrastructure and provisions of the quality management process. The lessons learned from a specific integration effort are captured and made use of through the knowledge management process.
Technical management processes	Management of the intended integration effort is reflected in the specific framework of each of these processes: integration uses or provides input to each of these processes.	Management of the actual integration effort uses or provides input to each of these processes.
Technical processes	Integration concepts, requirements, and solutions emerge from the business or mission analysis, systems requirements, architecture definition, and design definition processes, drawing on the system analysis process.	Implementation, verification, transition, validation, operation, maintenance, and disposal processes all draw on or are supported by the integration process over the life cycle of the system.

NOTE The relationships in Table 2 also apply to enabling systems.

5.6 Progressive interface definition

Interfaces are progressively defined, with the first descriptions at a high level commonly occurring with application of the business or mission analysis process, then throughout the application of the requirements, architecture and design processes. The integration process coordinates with those processes to check that the interface definitions, as implemented and integrated, are adequate and that the processes consider the integration needs.

5.7 Integration over the life cycle

Life cycle concepts and models, as discussed in ISO/IEC/IEEE 24748-1 and ISO/IEC/IEEE 24748-3, relate directly to the integration process in two ways.

a) Integration planning should be done with a consideration of the system's or software system's entire life cycle, or remaining life cycle, in mind. There can be a need to integrate some part of the system's elements at any stage of the life cycle.

NOTE 1 For example, integration in the concept stage can be done on conceptual models.

b) Integration can also be done at any stage of a system's life cycle.

NOTE 2 In particular, system modifications to extend or improve utilization and support often drive a requirement to integrate the changed parts of the software or system.

A system's complexity can impact integration planning and stage planning. A system can have any mix of products and services and be composed of one or more of the following: hardware, software, data, humans, processes (e.g. processes for providing service to users), procedures (e.g. operator instructions), facilities, materials and naturally occurring entities. Since those various products and services can have different life cycle models, the timing and outputs associated with each model should support integration of the overall system at all affected stages of the system's life cycle.

5.8 Iteration and recursion in integration

5.8.1 General

Two forms of process application – iterative and recursive – are useful for applying the Integration process. Iteration and recursion concepts are provided in ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 and explained in detail in ISO/IEC/IEEE 24748-1.

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

5.8.2 Iterative application of processes

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 integration process is used iteratively starting from a first aggregate of system elements, which can include software unit or software item configurations for software system elements, until the completion of the SoI. The last successful iteration of the integration process results in the entirely integrated SoI.

The verification process is used with the integration process to perform the planned verification procedures related to the integration iteration.

The validation process can be used iteratively when performing the planned validation procedures related to a set of integration and verification actions.

5.8.3 Recursive application of processes

When the same set of processes or the same set of process activities are applied to successive levels of system elements within the system structure, the application method is referred to as recursive.

The integration process is used by any system-of-interest at any level of the decomposition of the system, hence can be used recursively.

The verification and validation processes can be applied following, as part of, or in conjunction with, the integration process in each recursion.

5.9 Regression testing in integration

It is often necessary to re-run tests previously completed on a lesser set of a system's elements. Element interfaces or simulated equivalents remain available for regression testing to be feasible. These considerations can be carried over to ensuing life cycle stages.

5.10 Integration enabling systems

The purpose of an integration enabling system is to provide an environment that serves to prepare for and perform the integration of any SoI for one or more life cycle stages. It can be viewed as a facility including the necessary services, and the resources and tools for integration. Each integration enabling

system is a system in its own right, in which its life cycle activities interact with those for the SoI to enable successful integration of the SoI.

NOTE 1 The enabling system is defined in ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 and can apply in the case of integration. An integration enabling system has its own life cycle composed of generic stages and uses the system or software processes of ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288, but is instantiated for each SoI to be integrated.

NOTE 2 For a given SoI, the composition of the enabling system can be different at different life cycle stages, considering the particular composition of system elements that are being integrated at that stage.

NOTE 3 Other independent enabling systems can exist for verification, validation and delivery purposes, or be combined with the integration enabling system.

ISO/IEC/IEEE 15288:2023, Annex D discusses system life cycle process implementation using an approach applicable for both MBSE and model-based systems and software engineering. When using this approach, early model and method selection and potential compatibility issues for integration are important planning considerations. The timing and content of the outputs of models (e.g. UML/SysML diagrams, 3D models) and simulations (e.g. electrical loads, material flow) that enable the realization of a system should be integrated among themselves and any activities to integrate the system itself.

NOTE 4 ISO/IEC/IEEE 24641 specifies a reference framework for systems and software engineering with a model-based approach

6 Integration process planning and application purposes

6.1 General

This clause presents guidelines for planning and applying the integration process stated in ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207. Guidelines for planning integration that are given in this clause include material on strategy, integration resources, and integration plan development.

In this clause, application guidelines are first given in the context of the integration process of ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288. Additional discussion focuses on the relationship of other processes from ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 to application of the integration process. Finally, this clause presents scenarios and approaches for system and software integration.

6.2 Integration planning and application guidelines

6.2.1 General

This clause presents guidelines on integration planning, including integration strategy, efficiency of the strategy, integration context, and roles and competencies of the integration team. This clause also presents guidelines for selection of methods used to apply the integration and guidelines for the development of the integration plan.

6.2.2 Integration strategy

The integration strategy defines which activities are performed, where, when, and how the activities are performed, and who or what performs them. Some factors influencing the choice of integration strategy include scalability and the choice of custom or ready-for-use integration solutions and tools. Until obtaining the complete integrated system that conforms to its architecture and design characteristics, system element integration activities include:

— obtaining each implemented system element that is to be integrated, including enabling system elements;

— activation of logical and physical interfaces appropriate to the elements being integrated;

— checking each interface to confirm that a basis for verification has been established;

— checking each applicable interface to confirm that a basis for any corresponding validation has been established.

The integration strategy is typically described in an integration plan that identifies configurations of expected aggregated elements that synthesize a part of the SoI, ordering of aggregate integration, and the integration testing to be performed. Much of this information is commonly supplied via reference to enabling systems and repositories, especially with regard to software integration strategy. The integration strategy and the selected integration techniques are thus developed in coordination with the verification and validation strategies.

The integration strategy is based on the architecture of the system and relies on the way the architecture of the system has been defined. Project objectives, such as delivery time, project cost, system security and safety, also influence the integration strategy, considering criteria such as deadlines, integration cost, risk, availability of resources, and skills. The integration strategy should be as flexible as possible. For example, in the case of large systems, the strategy can need revision when an initial operational capability is required by stakeholders to reduce delays from need definition to partial realization of benefits before the final operational capability is available. Similarly, integration strategy revision can be warranted if system rollback is necessitated when encountering unexpected anomalies during integration of new elements into an operational software system.

6.2.3 Efficiency considerations in the integration strategy

Efficiency of the integration strategy commonly involves optimizing the mix of workload, time and risk to obtain a complete integrated system operating in its context of use. It often involves defining the appropriate order for:

— developing integration supporting infrastructure and resources;

— aggregating the implemented system elements using defined procedures;

— performing verification actions against system requirements to check physical and logical interfaces before, during, and after the aggregation of implemented system elements, as well as the functionality of the resulting aggregate;

— performing any validation actions against stakeholder requirements to check system functionalities and requirements within the resulting aggregate.

Efficiency of the integration of the system is determined by considering the strategies for integration, verification and validation together. That is, an efficient integration strategy can also support efficient verification and efficient validation.

Efficiency considerations can be less clear when developing software integration strategy. Software integration can be driven by other priorities, e.g. need for a function, creating difficult elements first, or availability of existing source code. Continuous integration and automated testing of software system elements and other characteristics of modern software approaches and enabling systems can tend to merge individual process instantiation and impact element synthesis and strategy.

NOTE 1 Efficiency and effectiveness of integration can depend on the extent to which an obtained or acquired enabling system is capable of providing automated integration services and testing environment, especially for performing continuous integration and DevOps of the software systems.

NOTE 2 For efficient and effective interface checking, an approach can be adopted that integrates software system elements up to a specific extent in the SoI. Then interfaces within the integrated aggregate are checked via top-down execution using drivers and stubs for the elements that are still to be supplied.

6.2.4 Integration context

Integration also includes the integration of the SoI in its context of existence and use, which can include physical, organizational, social, legal, and environmental conditions. When the SoI has been formed and subsequently verified, it has to be integrated with the context and transitioned into use.

The context of use is the environment in which the SoI is intended to operate alongside the contextual constraints and enablers that are present, including other systems that may already be in place. Incorporating context of use follows the same approach as an integration step that consists of connecting the implemented elements of the SoI to the elements of the context, and checking that the SoI can operate under specified conditions.

The physical, organizational, social, legal, and environmental conditions, in which the SoI is intended to be used, are identified and described as applicable stakeholder requirements and the impact of these are derived to the obtain the corresponding applicable system requirements. These contextual conditions are then considered as factors during architecture and design definition to endow the SoI with the required system characteristics. These system characteristics can have a significant effect on the effectiveness of the SoI's mission. For example, the physical context can include thermal conditions, lighting, noise, or spatial layout. The organizational and social aspects of the context can include factors such as work practices, organizational structure and attitudes. Safety aspects can include factors such as hazards to life and property while security aspects can include factors such as data integrity assurance and vulnerability to threats.

NOTE Elements for the context of use description are provided in ISO/IEC 25063 which focuses mainly on the usability of an interactive system, including human operators or human users.

The following are examples of systems or products and their environmental conditions of use.

— Equipment for use in a cold store is designed to consider the need for the workers to wear insulated protective gloves or additional protective wear for overhaul and reassembly.

— A ticket machine, which is to be installed for use in an outdoor car park, is designed to accommodate the range of varying environmental conditions in which it is used or maintained, for example darkness to bright sunlight, precipitation, dry and dusty, or temperature extremes.

— Working in cosmic space, in a vacuum, with debris floating, or in an environment with high radiation, limits ands restricts the time and actions that people can take for integration tasks. For example, in space stations and nuclear facilities, humans and manipulators or robots often work together to perform integration tasks.

— An airline seat reservation system is designed to provide users with on-line service 24 hours a day across all time zones, while accommodating integration of system or software maintenance updates within designated timing requirements of a service level agreement.

— Application software on smart-phones that are deployed as a client terminal of a system (e.g. smart self-register shopping, on-line payment, on-line ticketing, route map guidance) are expected to be integrated for update with incremental functions or services through remote broadcast on the communication network environment during system operation or maintenance.

Information items regarding taking physical context, i.e. environmental, factors into account can be found in numerous standards, such as ISO 8995-1, ISO 15265 for workplaces, and in ISO 24500 for elderly and disabled persons.

These last considerations are more part of operational concept and system requirements and so subject to validation activity. The considerations can be seen as constraints to define and perform some integration activities, that is, aggregation and/or verification.

6.2.5 Roles and competencies of integration team

The following roles and associated competencies should be included as project team members involved in integration. In applying this document, one person can assume multiple roles and one role

can be held by multiple individuals or subgroups within the organization, considering workload and competence. There are no requirements for independence of roles in this document. For these roles and competencies, it is not necessary to have one unique team member for each role. It is possible that one person can be competent in more than one of the following areas, so that the project team size can vary, depending on system complexity, organizational processes, roles, and staff. Larger projects can require more than one person in each role.

— Requirements engineers, who provide the link to concepts, needs, expected operations, problem analysis and requirements change resolution with respect to multiple system impacts.

— Architects, who have the knowledge of the system architecture definition and description, particularly in regard to interfaces. The architects give advice and expertise to team integrators concerning the composition of aggregates; architects analyse problem reports and participate in resolving them.

— Designers, who have knowledge of system element design, system element testing and technologies that compose system elements. The designers give advice and expertise to the integrators concerning the involved system elements. Designers also analyse problem reports and participate in resolving them.

— Integrators, who define the integration strategy, the integration procedures or operations, take delivery of system elements, tools and resources, perform the integration of system elements and enabling systems, check the interfaces, confirm that the resulting aggregate of elements is ready for verification, and write the integration reports and problem reports.

NOTE 1 Software integration is typically performed continuously by software developers, who use automated tools to integrate, i.e. merge, new software branches into the baseline.

— Verification technologists, who apply tests to determine that the interfaces function to system requirements.

— Validation technologists, who apply tests to determine that the verified interface functionality meets stakeholder requirements.

— Quality assurance technologists, who confirm that the results of the integration at each step have produced the required outcomes.

NOTE 2 There are, of course, a number of functions, with their own roles and competencies, that can support integration, but which are not listed above on the integration team. For example: configuration management, knowledge management, risk management, for all the ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 processes.

6.2.6 Methods used to perform integration

6.2.6.1 Interface definition and analysis

To integrate two or more system elements, the interfaces between the elements are defined. The definition starts with an analysis of what the interfaces are, identification of interface linkages, identification of what passes across each interface or any other required properties. These are determined and identified through the architecture definition and design definition processes.

A further part of interface definition and analysis is to determine possible relationships among a set of interfaces. One method of analysis is the coupling matrix, or N square representation, which is described in detail in Annex A.

The requirements for each interface are commonly captured in an interface requirements document, or the equivalent. Once the interface has been put under configuration control using the configuration management process, the properties to be controlled are captured in an interface control document, or equivalent. The information item can also be incorporated into the integration plan.

6.2.6.2 Integration scenarios

Sometimes, the term integration is interpreted based on a restrictive definition consisting of integrating existing elements. That can be the case, for example, when dealing with industrial practices that consist of using existing products, or NDI, that are to be incorporated into a given system at a particular life cycle stage.

Three example cases are given below that demonstrate the importance of the system architecture definition, beyond considerations of physical integration.

— Planned integration. This is the classic approach used in the development stage of a system in which the system elements to be integrated are defined up front. Those system elements are defined, or identified as NDI, during architecture definition, to perform functions that are allocated to them to satisfy system requirements. These system elements can be developed specifically for the system-of-interest, can be existing and re-used as-is, or can be modified. The system elements can also be NDI evolved by external parties. The selection of the system elements is an architectural and development decision. Planned integration can be achieved at design and build time and can also be achieved through self-configuration during operation.

— Evolutionary integration (evolution/extension or contraction/adaptation). This is the approach used when an existing in-service system (e.g. in its utilization stage) needs to interoperate with another one, existing or not. The integration cannot occur without performing upstream definition tasks, because that case can be considered as a request that includes specific requirements. In particular, some architecture and evaluation studies are performed, including interoperability analyses (e.g. compatibility of protocols, interfacing capability), integrability analyses (e.g. feasibility of functional exchanges or physical connection), and verification and validation feasibility. These analyses are necessary inputs to the architecture definition or modification in order to either modify or not modify the interfacing system elements, or add new system elements. To do these analyses, it is assumed that the engineering artefacts of the concerned systems or system elements are available. Otherwise, a reverse engineering step has to be performed to characterize the interfacing systems or system elements.

— Capability extension, integrating existing system elements or NDI. This case often corresponds to a decision to develop a new capability of an in-service system (e.g. in its utilization stage), incorporating an existing system element, or an NDI. This case is generally highly constrained by a fast, narrowly bounded scope effort and low-cost implementation, giving a limited set of available options. Also, these system elements can be significantly complex. This case can be considered as an extension request that includes specific requirements. Architecture and evaluation studies should be performed upstream, based on the requirements. The studies should include in particular interfacing feasibility, interoperability analysis, integrability analysis (i.e. feasibility of functional exchanges and of physical or other connection), and verification and validation feasibility. An "open" architecture, which includes interfaces that conform to accepted standards and modules with strongly cohesive and loosely-coupled properties, or a service-oriented architecture can be used in addressing this situation. With such architectural characteristics, the integration is facilitated, even if the characteristics are not immediately apparent.

NOTE 1 Adoption of NDI is driven by time-to-market constraints and the higher level of technology readiness of the NDI products. The use of NDI constrains the architecture of a system. Architecture decisions about NDI have a significant impact on integration. Deciding early on in the system development to use NDI vs a “built in-house” solution aids identification of the appropriate interface and performance of adequate verification and validation actions.

NOTE 2 A SoI is not necessarily static in its environment. The environment, or context of use, can change over time, so the SoI has to adapt to new operating conditions, possibly by adding or modifying capabilities. Any type of system is considered in this document, including evolving systems having dynamic self-configuration capability, as discussed in ISO 15704. ISO/IEC/IEEE 15288, ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 24748-1 explain that a system can evolve over time and that every life cycle process can be applied at any time during the system life cycle. That is the case, in particular, with the integration process and related processes.

NOTE 3 An extension can involve a mix of products, services, or both, that are not possible in the existing system. Further, the extension can involve hardware, software, humans, data, processes (e.g. review process), procedures (e.g. operator instructions), facilities and naturally occurring entities (e.g. water, organisms, and minerals).

6.2.6.3 Alternative integration methods

Alternative integration methods are summarised in Table 3.

Table 3 — Integration methods

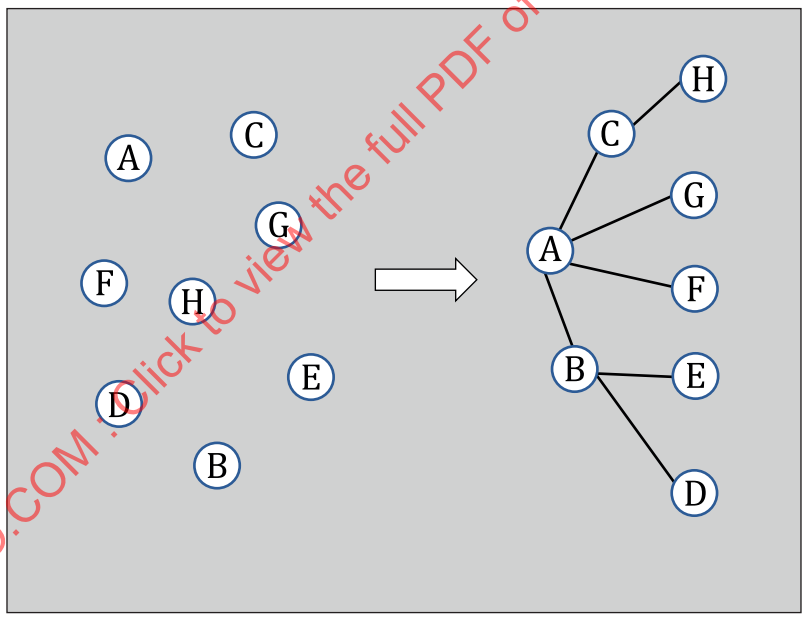
Integration method	Brief description
Global integration	**Principle**: Also known as big-bang integration. All the implemented system elements are aggregated in only one step. After each implemented system element has been integrated individually, the system elements are verified concurrently. **Advantage**: No need for drivers or stubs to simulate absent system elements for verification purposes. **Disadvantage**: Interface anomalies are detected late. It can be difficult to detect, diagnose, and locate remaining anomalies in the global system. Expensive and inefficient to fix any issues. **Application**: This method should be reserved for simple systems with few interactions and few system elements, where the technological risk is low. 
Key X system element transition stub or driver	

Table 3 *(continued)*

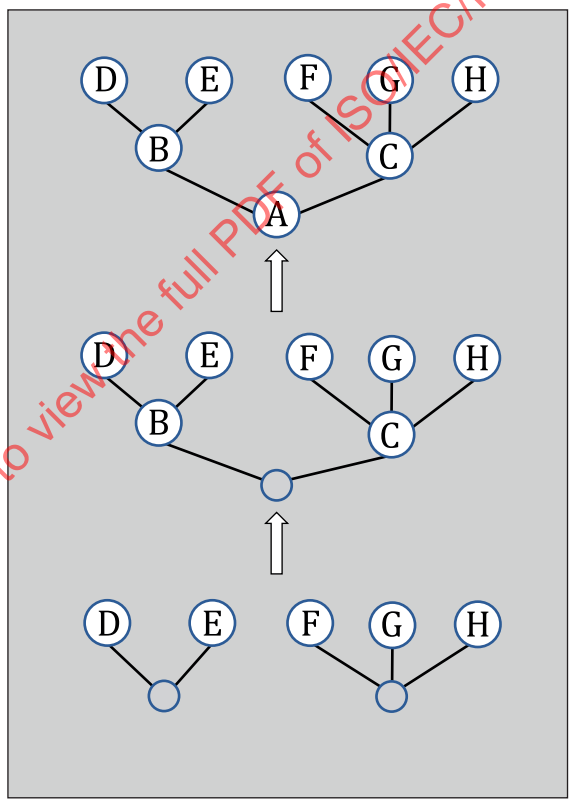
Integration method	Brief description
Bottom-up integration	**Principle**: Starts with system elements performing very detailed functions or dealing with acquisition from sensors and distribution or physical commands onto actuators. Continues with system elements performing detailed or core functions or transformations of the system. Ends with system elements related to overarching framework of the SoI, i.e. close to mission or interfacing operators' or users' commands. **Advantage**: No or few stubs to create. Early detection of implementation anomalies. Test cases to check interfaces and functions are easy to define. **Disadvantage**: Lower-level system elements are solicited a lot by higher-level test cases. Absent system elements from the upper levels are replaced by drivers. For example, when a large number of system elements exist at the same level and the program as an entity does not exist until the last module is added, the bottom up approach becomes very complex since test engineers cannot observe the system level functions from the partly integrated system. **Application**: Bottom-up physical integration is the most common way to integrate hardware. Also applies to software intensive systems. 
Key X system element transition stub or driver	

Table 3 *(continued)*

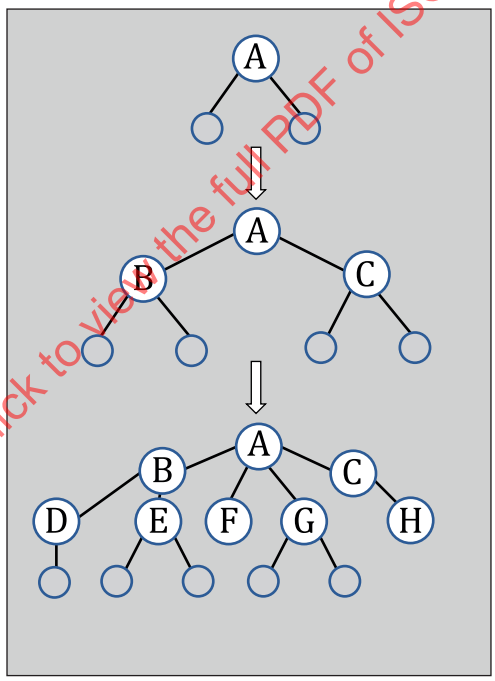
Integration method	Brief description
Top-down integration	**Principle**: Starts with system elements related to the overarching framework of the system-of-interest that are generally close to the mission or users' vision and operations/commands, when there are users and operators in the system-of-interest. Continues with system elements performing core or detailed functions or transformations of the system. Ends with system elements performing more detailed functions or close to acquisition from sensors and distribution or physical commands onto actuators. **Advantage**: Early availability of a skeleton of the system that allows detecting and easily locating remaining anomalies. Test cases for verification actions are close to reality. Does not need many drivers. **Disadvantage**: A lot of stubs to create. Difficult to check efficiently and exhaustively at the lowest level system elements and interfaces with test cases from the operators' or users' level. **Application**: Mainly used in software intensive systems, or in systems with human operators or users that pilot or interface with command-control subsystems. A particular implementation of top-down integration method is the "onion ring approach": the integration starts from a kernel of the system or a backbone layer; then the kernel is used as a live stub or driver to integrate other system elements. The kernel can include one or several functionalities shared by a number of system elements. 
Integration "with the stream"	**Principle**: The system elements are integrated as the elements become available, after each implemented system element has been verified individually before aggregation. **Advantage**: Allows for quick start of integration. **Disadvantage**: Can necessitate use of drivers, stubs, or simulation for absent system elements. Inefficiencies can result if modification of the integration plan is necessary every time a system element is integrated. **Application**: Should be reserved for well-known and controlled systems without technological risk.
Key X system element ⇨ transition stub or driver	

Table 3 *(continued)*

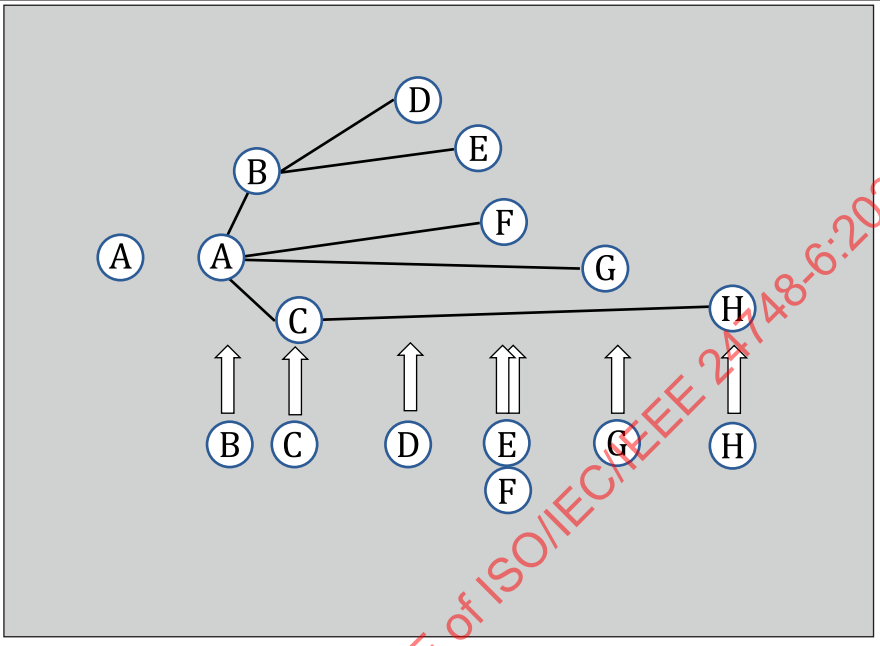
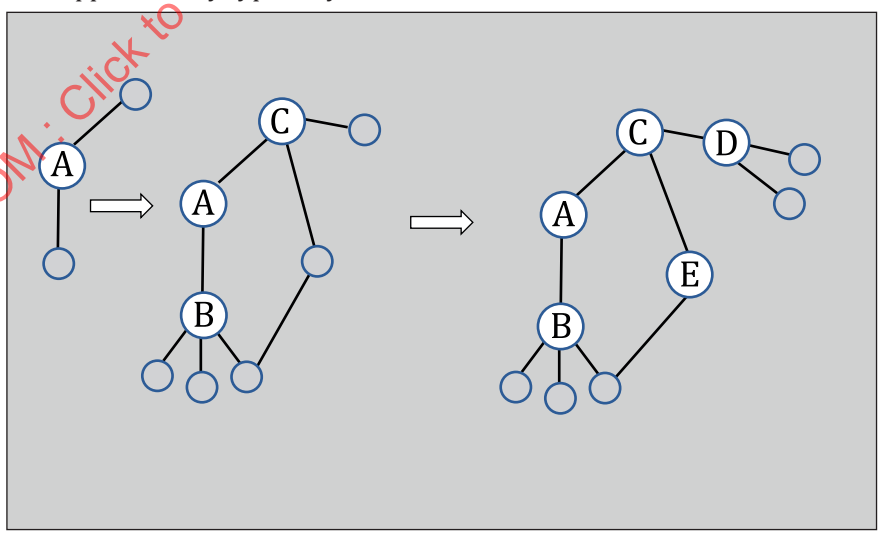
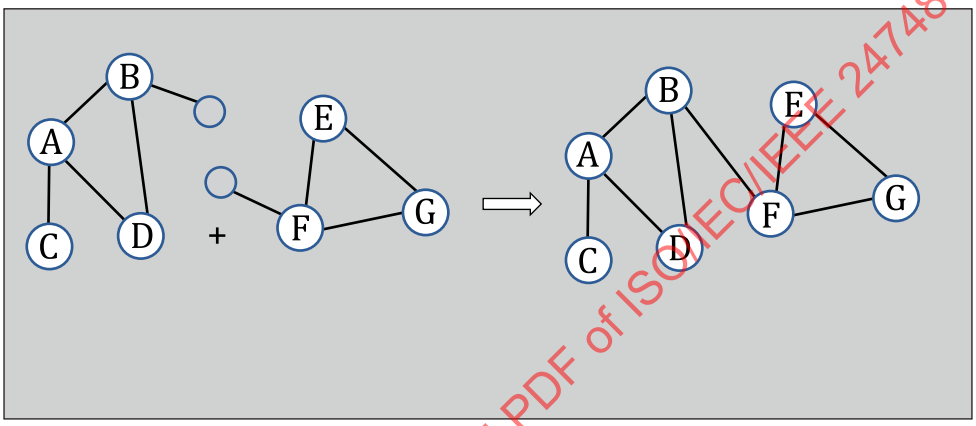
Integration method	Brief description
	
Incremental integration	**Principle**: In a predefined order, one or a very few implemented system elements are added to an already integrated increment of system elements. **Advantage**: Allows for early detection of interface anomalies and facilitates locating remaining anomalies in the new increment. **Disadvantage**: Necessitates a larger number of drivers or stubs to simulate absent system elements. **Application**: Applies to any type of system architecture. 
Key X system element transition stub or driver	

Table 3 *(continued)*

Integration method	Brief description
Subset-by-subset integration	**Principle**: Implemented system elements are synthesized by subset, and then subsets are synthesized together. **Advantage**: Parallel integration of aggregates is possible. Delivery of partial products or services is possible. **Constraint**: Necessitates a system architecture defined with subsystems, i.e. collections of aggregates. Better suited to systems with less complex interfaces. **Application**: Applies to system architectures composed of subsystems. Applies well to products with functional or transactional chains. 
Criterion-driven integration	**Principle**: The most critical implemented system elements with respect to the selected criterion are integrated first. Criteria come from system requirements and are typically related to technical risk or other priorities. Criteria commonly address dependability, security, complexity, performances or effectiveness, usability, technological innovation, availability, etc. The aggregates are defined around system elements that correlate with technical risks or other prioritised criteria. **Advantage**: Allows early and intensive testing of critical system elements. Early check of design choices. **Disadvantage**: Drivers and stubs can often be difficult to define and realize. **Application**: Applies to any system architecture that includes critical system elements.
Key X system element ⟹ transition stub or driver	

NOTE 1 Drivers and stubs are of particular use in the verification and validation processes and can already be defined to support testing. By using integration process drivers and stubs for verification and validation, the cost burden for each process can be decreased and the overall utility can be increased.

The selection of integration methods depends on several factors, such as the type of system elements, their criticality, the delivery time, the order of delivery, technical risks, constraints, the verification strategy, the validation strategy, the type of project and organization. Each integration method has strengths and weaknesses that should be considered when defining the integration strategy.

Often, a mix of integration methods is selected as a trade-off between the methods listed above, in order to optimise work and to adapt the process to the system under development. A "middle-out"

solution is commonly selected, where the most suitable aspects of top-down and bottom-up methods are combined and applied to address a project's specific integration needs.

NOTE 2 Integration process activity is not a phase of the project fixed in time; integration activity can also be performed dynamically during operation through self-configuration arrangements of the SoI architecture. The top-down integration method can be used in this case; the presence of stubs enables progressively building a system driven by, for example, availability, scalability and adaptability requirements.

Sometimes critical factors elevate the need for method selections where integration becomes a key driver and focus for life cycle stage, process and environmental decisions. Continuous integration is a major aspect and characteristic of agile and DevOps approaches for software systems and is highlighted in this document. In both approaches merged process strategies, instantiations and enabling systems occur early-on in the software system's life cycle.

ISO/IEC/IEEE 12207:2017, Annex H discusses application of agile methods and ISO/IEC/IEEE 24748-3 contains more detailed discussion of agile methods and life cycle model aspects.

DevOps provides a customer-focused, risk-managed approach driven by need for time-to-market, secure and dependable software systems. DevOps practices are employed in highly automated environments that support and are especially integrated with the technical and technical management processes described in ISO/IEC/IEEE 12207. IEEE 2675 provides an in-depth presentation of DevOps concepts, including principles, organizational culture, and roles, and DevOps practices as applied through ISO/IEC/IEEE 12207 software life cycle processes. Besides 'continuous integration', IEEE 2675 presents a concept of 'Left-shift and Continuous Everything' to highlight the need for early performance of practices commonly held until later in the life cycle, e.g. continuous delivery, testing, and QA during design, using the same practices in development as in operations and sustainment stages. IEEE 2675 indicates that the DevOps-based elaboration of ISO/IEC/IEEE 12207's process, activity, and task requirements can be used across a software system's life cycle.

6.2.6.4 Integration by layers of systems

Definition of a system, as shown on the left side of Figure 1, is performed on successive layers of abstraction; each layer corresponds to an architecture of systems and non-decomposable system elements. The integration, shown on the right side of Figure 1, consists of following the opposite way of composition layer by layer.

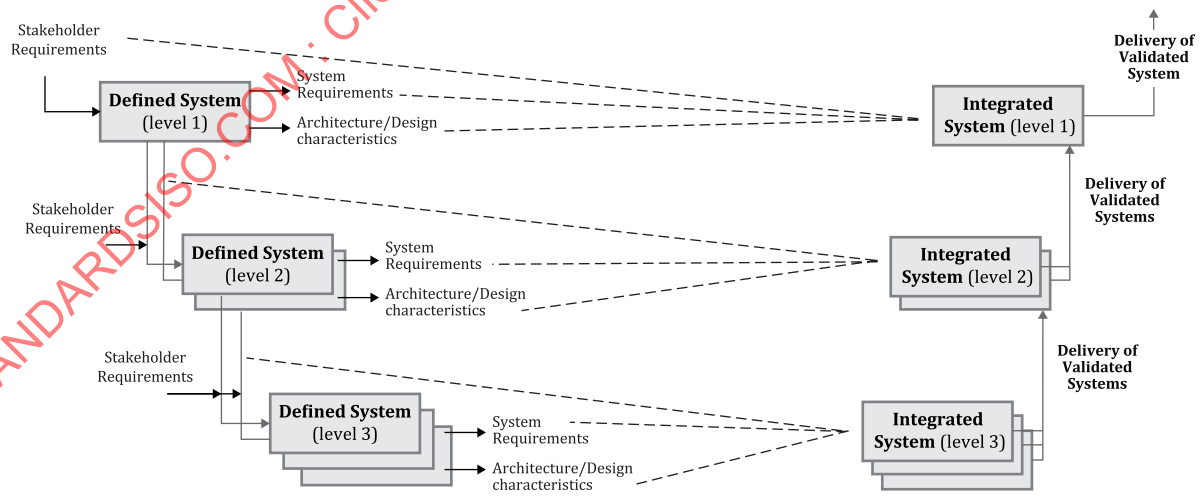


Figure 1 — Integration by layer of systems

On a given layer, integration of implemented system elements can be done on the basis of the physical architecture as established during the system definition. However, the functional or logical architecture

can also be used as the basis for selection of the system elements that are integrated to perform a function of the system.

NOTE 1 The use of physical, functional, logical, or other architecture at a given layer is driven by the nature of the system elements. For example, a system of services, with process and human elements, can be integrated within itself, then can be integrated with physical elements.

The goal is to progressively form the SoI per specified system layer by forming aggregates, such that a set of critical system elements or specific characteristics requiring to be integrated, are synthesized, and verified and validated, with the optimum balance of time, cost, and risk obtainable. In that case, the criterion driven method of integration can be the best approach. Mixing several methods, such as using different integration methods for different layers, or parts, of the system, can make the global integration and validation of a SoI more efficient.

NOTE 2 Many integrated systems are systems-of-systems, and the integration starts with combination of existing validated, possibly NDI, systems. In that case the system-of-systems is the SoI; and each existing system is a system element within the SoI. That holds true for the enabling systems as well. Regression testing can become increasingly necessary as systems-of-interest are combined to make an increasingly complex system-of-systems, due to the phenomenon of emergent behaviour. That is, system responses that are not foreseen based on the behaviour of the individual systems-of-interest before their combination into the system-of-systems.

6.3 Integration process application requirements and guidelines

6.3.1 General

Any additional requirements and guidelines for the integration process in ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2023 are described in 6.3.2, 6.3.3, and 6.3.4, which address the purpose, outcomes, and activities and tasks, respectively, of the integration process. The treatment in each subclause starts with a presentation of the integration process purpose, outcomes, or activities and tasks, as described in ISO/IEC/IEEE 15288:2023, 6.4.8 and ISO/IEC/IEEE 12207:2017, 6.4.8.

Provisions from ISO/IEC/IEEE 15288:2023 and ISO/IEC/IEEE 12207:2017 are highlighted in one or more boxes. The process purpose statements in ISO/IEC/IEEE 12207:2017 are harmonized with those in ISO/IEC/IEEE 15288:2015, so there is very little difference between those two standards in that respect. The outcome statements in ISO/IEC/IEEE 12207:2017 are also based on those in ISO/IEC/IEEE 15288:2015. Many of the process purpose, outcome activity and task statements in ISO/IEC/IEEE 15288:2023 have been updated, so at this point the ISO/IEC/IEEE 12207:2017 text can seem more aligned with systems than software. The minor differences between the text in ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015 are also identified. These differences are shown within the ISO/IEC/IEEE 12207:2017 text sections enclosed in braces, [system/software requirements].

In the cases of purpose and outcomes, the ISO/IEC/IEEE 12207:2017 text is boxed and displayed first in 6.3.2 and 6.3.3. The boxed text for ISO/IEC/IEEE 15288:2023 follows to show the refinements that have been made to those purpose and outcome statements. A supplemental outcome has been added in 6.3.3 for elaboration of the integration process. For activities and tasks in 6.3.4, there are more differences, so for better clarity, the texts of ISO/IEC/IEEE 15288:2023 and ISO/IEC/IEEE 12207:2017 are shown side-by-side, again with the differences highlighted as stated above.

Following the side-by-side presentation of activities and tasks in 6.3.4, aligned groups of tasks are discussed.

Notes that occur in ISO/IEC/IEEE 12207:2017 or ISO/IEC/IEEE 15288:2023 are not included in the boxed material in this document. Most of the guidance given in those notes has been integrated into the guidance given in this document.

6.3.2 Purpose

The differences between the purpose statements of ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2023 are shown in the two text boxes below.

The purpose of the Integration process is to synthesize a set of system elements into a realized system (product or service) that satisfies [system/software] requirements, architecture, and design.

This process assembles the implemented system elements. Interfaces are identified and activated to enable interoperation of the system elements as intended. This process integrates the enabling systems with the system-of-interest to facilitate interoperation.

ISO/IEC/IEEE 12207:2017, 6.4.8.1

NOTE For any type of system element other than a physical element, the term assemble can be considered as meaning to combine the elements, which can have no physical attributes, e.g. elements that are logical, procedural, thermal.

ISO/IEC/IEEE 15288:2023 is updated as indicated.

The purpose of the integration process is to synthesize a set of system elements into a realised system that satisfies the system requirements.

This process encompasses planning for, preparing for, and aggregating a progressively more complete set of system elements. Interfaces are identified and activated to enable interoperation and subsequent verification and possibly validation of the requirements (including characteristics) of the system elements as intended. This process also connects and checks out interfaces of the SoI with enabling systems for which there is direct interaction.

ISO/IEC/IEEE 15288:2023, 6.4.8.1

The goal of the integration process is to progressively aggregate the implemented system elements, perhaps iteratively or recursively, in order to synthesize the system that is made of products, or provides services, or both, that conforms to its architecture and design characteristics, its system requirements, or system/software requirements, and its stakeholder requirements, and achieves the intended system purpose in its intended context of use.

The central thrust of the integration process is to check that the implemented system elements that are intended to interface do so and, through subsequent verification, do so correctly. In other words, to check that the interfaces between the implemented system elements correspond to their definition obtained during the performing of the architecture and design definition processes. In performing the check, the interfaces on the implemented system elements are first identified, then activated to confirm that there is a basis for successful performance of verification actions. These checks demonstrate that the system elements interoperate as intended in the system definition.

Regarding enabling systems, to facilitate the progressive aggregation of system elements and the subsequent performing of verification actions, different kinds of enabling resources (e.g. products, systems, services) are permanently or temporarily integrated with the concerned aggregate. That principle is extended to the integration of the complete SoI in its operational environment in which other necessary enabling systems are interfaced with the SoI, for example, a maintenance system or a logistics support system, so that the SoI can achieve its expected mission.

Software system integration iteratively combines implemented software system elements to form complete or partial system configurations in order to build a product or service. In software, the linking involves merging source code of implemented elements that are then compiled into an informal software engineering build. Software integration is typically performed daily or continuously during development and maintenance stages, possibly even with operation, often using automated tools. Continuous integration involves frequent inclusion, or replacement and archiving of items, in software libraries under the configuration management process control.

Enabling systems for software continuous integration can provide automatic analysis, testing and merging of program code, selecting components and generating suitable configurations for delivery to individual user environments.

6.3.3 Outcomes

The differences in wording for outcomes between ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2023 are shown in the two text boxes below.

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

a) Integration constraints that influence system requirements, architecture, or design, including interfaces, are identified.

b) Approach and checkpoints for the correct operation of the assembled interfaces and system functions are defined.

c) Any enabling systems or services needed for integration are available.

d) A system composed of implemented system elements is integrated.

e) The interfaces between the implemented system elements that compose the system are checked.

f) The interfaces between the system and the external environment are checked.

g) Integration results and anomalies are identified.

h) Traceability of the integrated system elements is established.

ISO/IEC/IEEE 12207:2017, 6.4.8.2

ISO/IEC/IEEE 15288:2023 outcomes contain minor differences and the prior outcomes e) and f) were combined as indicated in the box below.

As a result of the successful performance of the integration process:

a) integration constraints that influence system requirements, architecture, or design, including interfaces, are identified;

b) approaches and checkpoints for the correct activation of the identified interfaces and system functions are defined;

c) enabling systems or services needed for integration are available;

d) a system composed of implemented system elements is integrated;

e) the system external interfaces (system to external environment) and system internal interfaces (between implemented system elements) are checked;

f) integration results and anomalies are identified;

g) traceability of the integrated system elements is established.

ISO/IEC/IEEE 15288:2023, 6.4.8.2

For the purposes of elaboration of the integration process, a supplemental outcome is added to the above sets. The following outcome would become outcome i) if using the ISO/IEC/IEEE 12207:2017 set or outcome h) if using the ISO/IEC/IEEE 15288:2023 set:

The strategy or strategies, as defined for the integration process, are established.

The definition of the system is impacted by technologies that support the system elements, how to combine or connect the implemented elements and what verification actions to perform. Those impacts should be identified and translated into system requirements, corresponding architectures or characteristics and design properties, using the applicable corresponding technical processes. This progressively drives and constrains where the interfaces are and what is supposed to go across them.

For the correct operation of interfaces and system functions, the potential causes of any anomalies in aggregating system elements that can alter their integrity, or their interface properties, and introduce potential system malfunctions, should be identified. Anomalies can happen because of wrong system elements, wrong configurations, wrong interfaces, or having several possible right or wrong ways to connect the interfaces of system elements. Anomalies can affect security, safety of operators, operability, interoperability, etc. To avoid such constraining issues, an approach and checkpoints should be defined early during the system definition stage.

Enabling systems or services are made available prior to performing integration. Consequently, the enabling systems or services are defined, developed, verified and validated prior to using them.

Regarding the traceability of the integrated system elements, the integration process establishes the configuration status of each implemented system element that is integrated, i.e. which version is integrated, the history, rationale and impacts of modifications or changes, using the configuration management process. The results of performing integration are recorded as well.

6.3.4 Activities and tasks

6.3.4.1 Integration process activity and task comparisons

ISO/IEC/IEEE 15288:2023 and ISO/IEC/IEEE 12207:2017 introduce integration process activities and tasks with the statements shown in the boxed text below.

ISO/IEC/IEEE 15288:2023, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
The following activities and tasks shall be implemented in accordance with applicable organization policies and procedures with respect to the integration process.	The project shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the Integration process.

The activities and tasks for each standard are discussed in the following clauses of this document. The task statements under 6.3.4.2 and 6.3.4.4 are better aligned and presented in side-by-side boxed text followed by the guidance information. The ISO/IEC/IEEE 15288:2023 task statements have changed in 6.3.4.3. In 6.3.4.3, the tasks statements are presented in a single side-by-side box. The task groupings in the guidance that follows are intended to highlight the task alignments.

6.3.4.2 Prepare for integration

There are five tasks under the "prepare for integration" activity, each of which is discussed in this subclause.

a) Tasks 1 and 2--Identify checkpoints and define strategy

The order of the tasks to define criteria and checkpoints versus defining the integration strategy is reversed in ISO/IEC/IEEE 12207:2017 or ISO/IEC/IEEE 15288:2023, as shown in the boxed text below.

ISO/IEC/IEEE 15288:2023, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
1) Identify and define checkpoints for the correct activation and integrity of the interfaces and the selected system functions as the system elements are synthesized. 2) Define the integration strategy	1) Define the integration strategy. 2) Identify and define criteria for integration and points at which the correct operation and integrity of the interfaces and the selected software system functions will be verified.

In actuality, the tasks are highly iterative, so either can be taken as first, knowing that the results change as soon as the second task unfolds: both are done and the tasks strongly interact.

Generally, the SoI cannot be aggregated in a single round of combining elements. As mentioned in 6.2.2, a strategy has to be defined describing the way the integration is to be prepared for and performed.

The complete strategy is defined applying the following considerations and dispositions, which implicitly require identifying and defining criteria for integration and points at which the correct operation and integrity of the interfaces and the selected system and software system functions are verified, on an ongoing basis:

- analyse the system architecture description, in particular physical and logical models, in order to:
  - identify the system elements and their physical, logical or other interfaces;
  - identify the system elements that perform the core functioning of the system;
  - identify the system elements that perform functionalities related to the core functioning of the system;
  - identify the system elements that interface with the external elements to the system;
- analyse system requirements and architecture descriptions to identify priorities for combining system elements;
- study alternative integration techniques or methods, shown in 6.2.6.3, that can be relevant to the system to be integrated;
- study enabling system alternatives, that can be relevant to the system to be integrated;
- study alternative aggregates and sets of aggregates that can predefine the order of combination;
- analyse the verification and validation strategies to define which system elements and which aggregates of system elements are necessary to perform the selected subsequent verification, and possibly validation, actions; the strategies contain the list of subsequent verification and validation actions that should be performed; the list is the result of trade-offs between what is desired to be verified and validated and what can be verified and validated considering multiple constraints such as cost, deadlines, feasibility, security, safety, risk;
- perform hazard analysis, security analysis, safety analysis, and risk analysis, to prevent unintended or undesired consequences that can appear during, or be generated by, integration performance; the consequences concern the integrity of the system itself, the safety of operators inside or outside the system, the integrators, and the environmental conditions;
- Evaluate need for anti-counterfeit, anti-tamper, system and software assurance and interoperability elements when identifying and defining checkpoints.

NOTE 1 The ISO/IEC/IEEE 15026 series and the ISO/IEC 27000 family of standards give information on assurance, integrity, and security.

- assess and select one integration technique or method, or define a mix of these, considering the preceding studies;
- define the order for combining the system elements through the set of aggregates previously defined;
- consider project parameters and constraints to minimise integration time, cost and risk; synchronise integration tasks with the master schedule of the project.

The integration strategy commonly allows for sequencing the aggregation order for implemented system elements based on the priorities of the system requirements and architecture definition focusing on the interfaces, and for minimizing integration time, cost, and risks. The strategy often provides for subsequent verification against a sequence of progressively more complete system element configurations. It is dependent on system element availability and is consistent with a fault isolation and diagnosis strategy. The integration strategy often is consistent with a regression strategy, particularly for software, which is applied for re-verifying software elements when related software units and potentially associated requirements, design artefacts, and information for users, are changed.

Defining the integration strategy commonly accompanies defining the strategy for other processes that occur, such as implementation, transition, operation and maintenance, and disposal.

For software systems, ISO/IEC/IEEE 12207:2017 reflects that software elements can occur both in software systems, that is, systems where software is the predominant concern and the primary product is data or information, and also in most other systems, where the primary output is a result other than information and the software is not the primary concern. Often software system integration process strategy definition involves different approaches for system or software element integration and that for integration of software at an atomic level. More modern software methods and techniques typically involve automated and 'continuous' software construction, integration, verification and storage, which tends to blur the lines for instantiation of the implementation, integration, verification and configuration management, or other, processes. The defined criteria for software integration can allow for different treatment of unit composition, interface and other aspects of developing software that fall outside the system or software element points identified for verification of the correct operation and integrity of the interfaces and the selected software system functions.

Integration strategy for software systems is typically determined by the architecture and design decisions. Conversely, architecture and design decisions are made with software system integration in mind. Software system integration often involves reuse of existing open source or proprietary libraries, as well as integration of NDI or COTS software packages, since it is often more cost-effective, quicker, and more reliable to integrate proven software elements than to redevelop them. For example, software systems managing data are usually distinct from, and integrated with, transactional applications. Software integration works within the design perspective of containerization or encapsulation, meaning that executable software units or elements have distinct boundaries with well-defined interfaces, often APIs.

Containerization improves portability of software to be integrated on different operating systems. Containerization can also reduce the threat of unwanted integration when malware threatens multiple software elements. This approach simplifies troubleshooting analysis, maintenance, and verification, as well as integration. During development or maintenance of software code, software integration is performed continuously with software implementation and unit test. Configuration management and version control tools allow developers to branch code for refactoring, development, and unit test, and then merge (integrate) the new code back into the mainline or baseline version. The same procedure can be used for continuous delivery of new versions for testing or production use.

NOTE 2 IEEE 2675 provides an elaboration of the ISO/IEC/IEEE 12207:2017 integration process focused on continuous integration, which is a key concept for DevOps. IEEE 2675, ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 24748-3 each provide additional information and guidance that support integration process strategy development for software as well as strategies for the other closely related processes.

NOTE 3 Typical software integration techniques include APIs or webhooks, as well as use of ISC and use of orchestration to handle integration with cloud or virtual services. See ISO/IEC 18384-1 and ISO/TS 23029 for additional information.

The strategy can include early dispositions and tasks related to other development activities in order to facilitate the integration. In particular, consider using simulators, mock-ups, prototypes, or virtual environments during architecture and design activities to anticipate potential integration difficulties, hazards or threats and reduce integration time, cost and risk. The simulators, mock-ups, etc., can also be used in integration of system elements not yet implemented (e.g. receiving data from interfacing systems), in order to predefine and perform the sequence of aggregation. Wherever possible, an integrated configuration includes its human operators.

The integration strategy should be refined to facilitate the maintainability of the SoI during integration and subsequent activities, such as verification, validation, transition, operation, maintenance, and disposal, and to help efficiently analyse causes of problems encountered during integration. To support the foregoing, consider adding functions in the SoI, or in any enabling system, such as log data gathering and recording, help to diagnosis. Throughout, version control is maintained through the configuration management process.

For projects involving multiple sources of system elements, such as suppliers, or development teams, the availability of system elements for integration is typically part of the project schedule with milestones under the project assessment and control process. At major integration points (e.g. completion of a stage, element, or version) checkpoints for reviews and validation with stakeholders are typically held. The frequency of these reviews is related to the selected life cycle model, development method, and number of system elements.

The integration strategy is documented in the integration plan, described in 7.2. The integration plan can be a standalone document or a portion of a project's SDP or SEMP.

The integration plan also indicates where, when, how, and who performs the integration activities. For example, this can be site locations, schedules, aggregation approach detail, and skills.

b) Tasks 3, 4, and 5 –Identify and plan for enabling systems or services, acquire access to them and identify constraints and objectives from integration

The order of tasks 3, 4 and 5 is changed in ISO/IEC/IEEE 15288:2023 from that in the previous edition and ISO/IEC/IEEE 12207:2017. The order and minor differences in the text of the tasks are shown in the boxed text below. Guidance discussion in this subclause follows the task order in ISO/IEC/IEEE 12207:2017.

ISO/IEC/IEEE 15288:2023, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
3) Identify constraints and objectives from integration to be incorporated in the system requirements, architecture or design.	3) Identify and plan for the necessary enabling systems or services needed to support integration.
4) Identify and plan for the necessary enabling systems or services needed to support integration.	4) Obtain or acquire access to the enabling systems or services to be used to support integration.
5) Obtain or acquire access to the enabling systems or services, and materials to be used.	5) Identify constraints for integration to be incorporated in the system/software requirements, architecture or design.

1) Task – Identify and plan for necessary enabling systems or services

Integration planning, including integration strategy, efficiency of the strategy, and the roles and competencies of the integration team, are documented in an integration plan to be used by all participants in the integration activities. The structure of the integration plan and its requirements are defined in 7.2.

ISO/IEC/IEEE 15288:2023, 5.2.3 and ISO/IEC/IEEE 12207:2017, 5.2.3 explain the concept of enabling systems instantiated to the specific integration requirements and resources. Those subclauses provide information on setting up integration enabling systems and their relationships with the SoI to be integrated. 6.2.3 discusses efficiency considerations for integration enabling systems and 6.2.4 addresses considerations for the SoI in its context of existence and use. ISO/IEC/IEEE 15288:2023, 5.2.3 also explains concepts for interfacing and interoperating systems.

Just as for the SoI itself, requirements and interfaces are identified for the enabling systems.

Enabling systems for integration commonly include integration facilities, specialised equipment, training systems, discrepancy reporting systems, simulators, measurement devices, and facility and environmental security.

The usage of local or distributed test environments and tools are identified in support of the three tasks described in b). For software, regression test suites and configuration management systems for the integrated testing of software systems, incident and problem reporting systems, simulators representing external systems or undeveloped elements, and software library management systems for development operations, can be involved.

Changes or specializations needed for the enabling systems to support the integration tasks need to be identified and defined. The needs for these changes are provided to the stakeholders that govern the enabling systems, including identification of requirements and interfaces for the enabling systems. Enabling systems for integration commonly include integration facilities, specialised equipment, training systems, discrepancy reporting systems, simulators, measurement devices, and physical, information, or other, security. Especially for software, regression test suites and CM systems for the integrated testing of software systems, incident and problem reporting systems, simulators representing external systems or undeveloped elements, and software library management systems for development operations can be involved.

Changes or specializations needed for the enabling systems to support the integration tasks need to be identified and defined. Typically, the enabling systems or services used for integration during development stages can also help support system element integration as the software system and enabling environments evolve to operational status. The “DevOps” approach supports iterative software system implementation, integration, verification, transition, validation, operation and maintenance processes.

2) Task --Obtain or acquire access to the enabling systems or services

ISO/IEC/IEEE 15288:2023, 5.2.3 and ISO/IEC/IEEE 12207:2017, 5.2.3 give information on setting up integration of enabling systems and their relationships with the SoI to be integrated, in particular about the acquisition and validation of integration resources.

3) Task --Identify system constraints from integration

The text of this task is stated differently in ISO/IEC/IEEE 15288:2023 and ISO/IEC/IEEE 12207:2017. The guidance is specific in saying that the task is to identify system constraints from integration. For example, the integration sequence directly affects verification (test) planning. This is a constraint from integration on another process. As stated above, integration strategies are analysed to define where, when, how, and who performs the integration activities. The analyses also include requirements, such as accessibility, safety for integrators and operators, required interconnections for sets of implemented system elements and for enablers, and interface constraints. Those analyses generate constraints, some of which should be reflected back into changed system requirements baselines, the architecture description and the design description, as feedback.

The constraints to be incorporated in baselines influence the utilization or operation of the system, such as checkpoints to be incorporated in the system, the architecture and design characteristics (e.g. place or arrangement of implemented system elements), input-output flows of information items or commands exchanged with operators or users, interfaces with harnesses or simulators–drivers and stubs.

6.3.4.3 Perform integration

The wording for the "perform integration" activity is different between ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2023, as shown in the boxed text below.

ISO/IEC/IEEE 15288:2023, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
b) Perform integration. This activity consists of the following tasks:	b) **Perform integration**. Successively integrate [software] system element configurations until the complete system is synthesized. This activity consists of the following tasks:

There are three tasks in ISO/IEC/IEEE 12207:2017 under "perform integration". ISO/IEC/IEEE 15288:2023 has refined these into five tasks that more clearly explain the restated outcomes in ISO/IEC/IEEE 15288:2023. The ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2023 tasks are aligned for guidance purposes in the discussions that follow.

ISO/IEC/IEEE 15288:2023, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
1) Check interface availability and conformance of the interfaces in accordance with interface definitions and integration schedules. 2) Perform actions to address any conformance or availability issues. 3) Combine the implemented system elements in accordance with planned sequences. 4) Integrate system element configurations until the complete system is synthesized. 5) Check for expected results of the interfaces, selected functions, and critical quality characteristics.	1) Obtain implemented [software] system elements in accordance with agreed schedules. 2) Integrate the implemented elements. 3) Check that the integrated software interfaces or functions run from initiation to an expected termination within an expected range of data values.

This guidance discussion addresses and aligns tasks 1 and 3 of ISO/IEC/IEEE 12207:2017 with tasks 1, 2 and 5 of ISO/IEC/IEEE 15288:2023.

a) Task --Obtain implemented system elements in accordance with agreements and strategy

Some of the implemented system elements, particularly software and physical elements, can be provided from within the project or received from external suppliers, the acquirer, or other resources and are typically placed under CM control. System elements are handled in accordance with relevant health, safety, security and privacy considerations. As part of the acceptance of the implemented system elements, each element is verified and validated against acceptance criteria specified in an agreement. task 1 of ISO/IEC/IEEE 15288:2023 emphasizes inclusion of checks for interface availability according to plan and conformance to interface definition or other requirements. The delivered configuration, conformance, compatibility of interfaces, the presence of mandatory information items, are checked. Implemented system elements that do not pass checks are identified as such and handled in accordance with defined procedures.

b) Task --Check the interfaces

The goal for these tasks is to check the combination of interfaced system elements and the interfaces or interaction mechanisms between system elements and between the external components of the system. The checking is done in order to establish the conformance of the realized system of products, services or both, to the system architecture description and design descriptions.

Checks of the interfaces can be done before, during, and after the combining of system elements to form an aggregate. The checks are performed to help ensure the operation of the external and internal interfaces, functions, and quality characteristics.

In each integration step, when the implemented system elements are combined to form an aggregate, the aggregate is prepared to check the functions it has to perform, and other architectural characteristics and design properties, or quality characteristics. Preparation can consist of:

— adding and connecting to the system elements, through checkpoints, equipment such as probes, sensors, spies, trackers, specific analysers, in order to observe the behaviour of the aggregate;

— adding and connecting drivers and stubs to simulate missing or absent implemented system elements or external components to the SoI.

Once the aggregate has been checked, it is ready for the planned verification process and possibly the validation process, depending on the characteristics to be checked. Measures and data are collected during or at the end of the operation of the aggregate. In practice, the line between checking the interface and verification can be more theoretical than practical. The point is to check to some extent

and then, when the interfaces between the aggregate appear to be satisfactorily established, to confirm in detail that the interfaces meet all requirements, i.e. perform the verification process.

As part of the acceptance of implemented software system elements, selected elements are checked to help ensure the elements meet acceptance criteria as specified in the integration strategy and applicable agreements. Checking can include conformance to the agreed configuration, compatibility of interfaces, and the presence of mandatory information items. The project assessment and control process can be used in accordance with the integration strategy to plan and conduct technical reviews of the integrated software system elements, for example a test readiness review to help ensure the integrated element or system with its affiliated data and information items is ready for qualification testing.

c) Task --Aggregate system elements

This guidance discussion addresses and aligns task 2 of ISO/IEC/IEEE 12207:2017 with tasks 3 and 4 of ISO/IEC/IEEE 15288:2023.

The combination of the implemented system elements follows the integration strategy as previously defined and documented in the integration plan, in particular the order of combination and any regression testing. The task is performed to achieve a complete or partial system element configuration connecting the implemented system elements as prescribed in the integration strategy, using the defined procedures, interface control descriptions, and the related integration enabling systems.

The integration performance is generally divided into several steps, combining at each step some implemented system elements to form aggregates and integration configurations composed of the aggregates and the necessary related enabling systems, tools or services. An integration step can start and end with two optional milestones that can be called "Integration Step Readiness Review" and "Integration Step Results Review," which identify the start and completion of the aggregating actions, confirming that the system is ready for the verification actions and possibly validation actions that are to follow the integration step.

For each integration step, an integration procedure is defined from the concerned system integration aggregate approach detailed and documented, as stated in 7.2. The integration procedure provides the necessary information items and data to perform the concerned step. These information items are defined and deduced from the integration strategy. The information items can be optional depending on criteria, such as safety, security and complexity of the integration step.

In each step, the integration procedure is used to perform the combining of the concerned elements. The integration procedure explains in particular how the system elements are handled and connected, i.e. physical, logical, or other, interfaces, using the enabling resources tools, systems, i.e. harnesses, simulators, etc., through actions to be directed or done by human operators. The integration operators should be skilled personnel trained to carry out assembly activities on simulators or mock-ups, prior to performing these activities on the target implemented system elements. The disposition is optional depending on criteria, such as safety, security and complexity of integration operations.

In terms of software, integrating the implemented elements can involve linking together pieces of object code or simply bringing together the implemented elements that are part of the software configuration in a methodical piece by piece approach. Software elements are typically compiled into a "build" so that branched units are properly linked or merged in the assembled element. Firmware elements are fabricated, often as prototypes, and installed in hardware elements. If software functions are not yet available for integration, emulated functionality, for example stubs or drivers, can be used to temporarily support integration of software elements or represent input from external interfaces. Successful aggregations result in an integrated software element, that is available for further processing, i.e., additional software system element integration, verification, or validation.

Anti-counterfeit, anti-tamper, system and software assurance and interoperability concerns can arise when performing integration and identifying and defining checkpoints. Integration and verification processes often use fictitious data for security or privacy considerations. The ISO/IEC/IEEE 15026 series and the ISO/IEC 27000 family of standards include information on assurance, integrity, and security considerations affecting integration.

6.3.4.4 Manage results of integration

There are three tasks under the "manage results of integration" activity, each of which is discussed in this subclause.

a) Task 1--Record results and anomalies

ISO/IEC/IEEE 15288:2023 varies from ISO/IEC/IEEE 12207:2017 by one word, which is highlighted in the boxed text below:

ISO/IEC/IEEE 15288:2023, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
1) Record integration results and [any] anomalies encountered.	1) Record integration results and anomalies encountered.

Include anomalies that would preclude or impede moving to subsequent verification and possibly validation actions, such as ones due to the integration strategy, the integration enabling systems, performing the integration, or incorrect system or element definition. Where inconsistencies exist at the interface between the system, its specified operational environment and systems that enable the utilization stage, the deviations lead to corrective actions.

NOTE For modern software development where integration occurs continuously, two strategies are commonly employed – rollback or fix forward.

Anomaly resolution typically involves the technical processes, often repetitive application of the implementation process. The quality assurance and project assessment and control process are used to analyse the data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

b) Task 2--Maintain traceability

ISO/IEC/IEEE 12207:2017 varies from ISO/IEC/IEEE 15288:2023 by one word, which is highlighted in the boxed text below:

ISO/IEC/IEEE 15288:2023, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
2) Maintain traceability of the integrated system elements.	2) Maintain traceability of the integrated [software] system elements.

Bidirectional traceability is maintained between the integrated system elements and the system architecture, design, and system or element requirements, such as use cases for software, and including interface requirements and definitions that are necessary for integration. Integrated software elements and their components are identified by version. Versions of integrated software elements are commonly traceable to implemented units, test procedures, and test cases.

c) Task 3--Provide information items

ISO/IEC/IEEE 12207:2017 varies from ISO/IEC/IEEE 15288:2023 by the words highlighted in the boxed text below:

ISO/IEC/IEEE 15288: —, 6.4.8.3	ISO/IEC/IEEE 12207:2017, 6.4.8.3
3) Provide key artefacts that have been selected for baselines.	3) Provide key artifacts [and information items] that have been selected for baselines.

The integration plan is managed under the information management process, because the integration plan can require changes to address events that can happen before and during integration performance. For example, the order of combination of the implemented system elements can change due to delays in delivery time of system elements or due to non-conformances that arise during verification or validation following an integration step.

The configuration management process is used to establish and maintain configuration items and baselines. The integration process identifies candidates for the baseline, and the information

management process controls the information items. For the information management process, the test cases, regression tests, and automated test scripts are typical artefacts that are baselined.

6.4 Other processes used in relationship to integration process application

6.4.1 General

As both ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 state, any process can invoke any other process. As expected, the integration process can call on some processes more often than others. The material following gives summaries of the nature of the interactions between the integration process and other processes. IEEE 2675, where continuous integration is a key concept, presents a DevOps elaboration of the ISO/IEC/IEEE 12207 processes, outcomes, activities and tasks.

6.4.2 Agreement processes

The agreement between the acquirer and the supplier should include clauses that define and detail the responsibilities of the parties concerning delivery, integration and warranty of the system elements composing the SoI. The justification is that the system element participates in the achievement of the purpose, objectives and mission of the SoI during operation.

Actual integration of a supplied system element within the larger SoI can occur long after delivery and acceptance of the supplied system element. Supplier participation can be crucial during and after SoI integration. This often includes analysis and correction of potential anomalies and can involve elements from multiple suppliers.

6.4.3 Organizational project-enabling processes

The same processes that were applied during integration planning can come into play during application of the integration process. The life cycle model management process, infrastructure management process, quality management process, and knowledge management process should include capabilities for interaction with the integration process while it is being performed, or for integration process outcome handling. These provisions should apply across the stream of projects the organization engages in. Integration draws upon the infrastructure and provisions of the quality management process. The lessons learned from a specific integration effort should be captured and made use of through the knowledge management process.

6.4.4 Technical management processes

Integration management uses the same processes as for project management, i.e. project planning process, project assessment and control process, but instantiated to accord with integration activities. The project assessment and control process can be used in accordance with the integration strategy to plan and conduct technical reviews of the integrated software system elements, for example, a test readiness review to help ensure the integrated element or system with its affiliated data and information items is ready for verification and validation.

The project manager often delegates the management of integration to an integration manager, because of the integration specificities at the right level of detail. The integration manager is responsible for the integration team, discussed in 6.2.5.

Configuration management supports necessary configuration identification, control, and change management.

The quality assurance process supports work product audits and inspections, and problem, non-conformance, or incident reporting and handling.

The information management process supports required integration-related information items.

The integration plan, specified in 7.2, is regularly updated, because of events such as system element delivery delays, anomalies that oblige the project to stop combining and subsequently verifying element aggregates, or that necessitate a modification of the concerned aggregate, or part of it.

The integration plan considers the availability of skilled personnel and tools in order to launch the activities and tasks at the right time. The integration is performed within a sequence of steps; each of which concerns one or several aggregates.

Anticipation and flexibility of the integration organization and integration plan are key factors for the successful achievement of integration within schedule and cost. Architecture and design definition, in particular the interfaces, is also critical to controlling integration schedule and cost.

Through the project planning process, the integration team identifies implemented system elements, baselines and information items that are to be controlled as configuration items. Configuration items are given unique identifiers. Configuration items are managed and maintained throughout the entire life cycle in accordance with the organizational configuration management process. Changes to configuration items are maintained.

6.4.5 Technical processes

6.4.5.1 General

The integration process has multiple interactions with the technical processes of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207 over the life cycle of a system. These interactions are examined in this subclause.

6.4.5.2 Business or mission analysis process

Preliminary studies of integration alternative strategies interact with development of a system's operational concept and operational scenarios. The studies should consider such areas as feasibility, effectiveness, cost, and risk analyses concerning the integration of the SoI.

In particular, the process should:

— analyse the concept of operations of the concerned organization and primary options of any potential system-of-interest to identify the necessary corresponding integration concepts, principles and capabilities to be set up;

— perform feasibility, effectiveness, cost and risk analyses concerning the integration approach of the potential SoI.

6.4.5.3 Stakeholder needs and requirements definition process

The process should address identification of stakeholders involved in integration of the SoI, including stakeholders who are impacted by or need to be informed of integration of the SoI, their expectations and constraints, and the definition of high-level objectives in terms of effectiveness, delivery time and cost. It should also include definition of requirements applicable to the integration resources (e.g., tools, simulators, integration procedures, verification procedures), services and their utilization, the refinement of objectives to define effectiveness requirements, and refinement of constraints to define delivery timing and cost limits.

The scope should cover the initiation of acquisition/development of these various resources through the definition of their respective requirements.

In particular, the process should:

— analyse the operational concepts, operational scenarios of the SoI and associated integration concepts and principles in order to identify and define integration stakeholder needs and constraints;

— incorporate, in the stakeholder requirements baseline, integration needs and constraints applicable to the concerned system;

— from the integration strategy definition, provide feedback and recommendations about the ability to integrate, through feasibility analysis of the implementation of the stakeholder requirements of integration type applying to the concerned system;

— when anomalies in stakeholder requirements are identified during performance of the Integration process, apply the stakeholder needs and requirements definition process to correct those requirements.

6.4.5.4 System requirements definition process

The system requirements definition process should include the system requirements for integration, including system requirements for enabling systems used for integration.

In particular, the process should:

— incorporate into the system requirements baseline integration type requirements, i.e. constraints, applicable to the concerned system;

— from the integration strategy definition, provide feedback and recommendations about the ability to integrate, through feasibility analysis of the implementation of the system requirements of integration type applying to the concerned system;

— when anomalies in system requirements are identified during performance of the integration process, the system requirements definition process is applied to correct those requirements.

6.4.5.5 Architecture definition process

The integration of a system pre-supposes that architecture and design activities are performed defining these system elements and their interfaces. The interfaces can pre-exist, be re-used, or can be specifically developed.

NOTE In ISO/IEC/IEEE 15288:2023, the name of the architecture definition process is updated to the system architecture definition process.

Based on stakeholder and system requirements, the definition of the architecture of the system deals with the structure and composition of the system elements and of their interfaces. The design of system elements deals with the detailed description of characteristics and enablers of the necessary technologies for their implementation and interfacing.

Scalability, interoperability, availability, portability, etc. are stakeholder concerns that are addressed during architecture definition to equip the architecture with related architectural characteristics such as modularity, standardized interfaces, encapsulation.

So, the integration activities consist of: defining the order of combining the system elements to the extent that the order is relevant; defining the subsequent verification actions based on architecture and design characteristics; then performing the aggregation of the implemented system elements and performing the verification and/or validation actions. However, note that, particularly in software, it is common to develop, expand, or refactor software units without defining their order of integration into the software product. The order of integrating the system elements can be flexible. Even for hardware items, different subassemblies, i.e. aggregates, can be integrated in various sequences, though there can be a critical path for supply and production of system elements. The critical path consideration for supply and production often identifies integration constraints that influence requirements, architecture, or design, including interfaces; these identified constraints are incorporated in the system requirements, architecture or design.

Nevertheless, architecture and design definition activities and decisions should consider how the integration is to be performed.

In any case the integration of implemented system elements is based on the architecture and design definition, whatever the life cycle scenario. In particular, the process should:

— from the integration strategy definition, provide feedback and recommendations through analysis of logical and physical models of the concerned system to possibly ease the future integration, in particular with interface concerns;

— use architectural characteristics, such as modularity, to facilitate the integration;

— study and identify expected or unexpected potential emergent behaviours or properties that can happen when integrating system-of-systems elements, which is carried out to avoid late detection, in particular, of threats or hazards that can be generated by the system due to interactions of system elements;

— define interfaces in the context of their integration; the physical aspects of interfaces are concerned with physical aggregation of system elements; the logical aspects of interfaces, i.e. input-output flows and interactions, are concerned with functional performance of integration aggregates, that is, sets of system elements;

— define interfaces in the context of use, that is, the existing environment, which influences the interface definition;

— apply the architecture definition process to correct the concerned architecture items when anomalies concerning architecture, particularly anomalies with interfaces, are identified during performance of the integration process.

The architecture definition process for integration enabling systems includes in particular:

— participation in the refinement of the integration strategy, concerning mainly the relevance of aggregates definition, verification actions, and applicable techniques or methods;

— development of a logical view of the integration enabling system, e.g. how aggregation and subsequent verification tasks should be performed and synchronised; scheduling of delivery and assembly of the system elements of the SoI; planning of the verification actions;

— projection or allocation of the logical view onto a physical view including adequate capabilities, e.g. tools, simulators, integration procedures, verification procedures, and resources, such as skills, personnel.

6.4.5.6 Design definition process

The detailed definition of interfaces is an important input for the integration strategy. The use of certain technologies can affect the interfaces between system elements and their design to be able to exchange materials, energy and/or information items. This can influence the order of interfacing system elements, or the selection of integration methods, or both.

For integration purposes, checkpoints should be added in system elements or at their interfaces; these should be incorporated in the design definition. Internal and external interface and integration points and applicable procedures should be specified, consistent with the design solution and system requirements.

Virtual environments or tools, such as MBSE, are increasingly used during design definition of systems and system elements in mechanics, electricity, electronics, software, operator procedures, etc. The MBSE and other virtual environments or tools also can be useful to prepare the future integration, simulating the interfaces of the designed system elements, trading-off alternative definition of aggregates, and simulating combination procedures. The results of using such tools during design definition can influence the design of the system and its requirements.

When anomalies concerning design, in particular with interfaces, are identified during performance of the integration process, the design definition process is applied to correct the concerned design items.

6.4.5.7 System analysis process

The system analysis process refines approaches and methods for integration. In particular, the process should:

— analyse pre-defined system elements, including NDI, of the architecture of the SoI to determine the capability to integrate a selected system element. The analyses can include interoperability evaluation;

— evaluate the integration strategy alternatives to improve effectiveness, reduce time and cost of integration;

— evaluate the verification strategy alternatives, from an integration viewpoint, to reduce risk, time, and cost, of the integration;

— evaluate the validation strategy alternatives, from an integration viewpoint, to reduce risk, time, and cost, of the integration.

6.4.5.8 Implementation process

Since the purpose of the implementation process is to realize a specified system element, the integration process is used to confirm that intended interfaces are established and that their outputs conform to requirements, setting the basis for elements to be combined and then checked together, that is, integrated. Any changes in the requirements for an element are checked for possible impact on interface characteristics, the subsequent integration, and following verification and possibly validation.

System element design and system architecture definition can involve virtual environments, modelling activities (e.g. MBSE), mock-ups, or any other convenient tools to model or simulate element aggregations. Those activities provide an early check on the feasibility of integration, the definition of necessary checkpoints, and an opportunity to simulate the interface properties.

The technique of using early checks allows identifying and correcting integration mistakes earlier in development and more easily mitigating risks. The result of the checking activity can lead to modifying the architecture and/or modifying the design of system elements and/or their interfaces.

Implementation and integration are commonly performed as one process during software development, along with verification.

6.4.5.9 Verification process

Integration checks are followed by performance of verification actions applied onto the aggregates to check the conformance of the assembly to the system requirements, architecture and design.

For example, the verification process can include:

— inspection of dimensions of the physical faces of each system element to be connected;

— obtaining objective evidence that the integrated software fulfils its specified requirements and identifies anomalies in integration-related information items, e.g. system/software requirements, architecture, design, test, or other descriptions, and processes, software elements, items, and units;

— establishing that an expected service product is created and is ready for planned verification;

— checking the voltage that is required between the delivered and used power;

— checking the position and/or polarity of pins;

— checking the compatibility of interface protocols in the software domain.

Integration activity is not limited to physically, logically, or both, combining implemented system elements. When an implemented system element is combined with another, the interfaces are checked. Verification actions are performed at selected points in combining the implemented system elements.

6.4.5.10 Transition process

Transition moves a system, or one or more parts of a system, into a new environment. The moves in themselves can impact at least those interfaces that bear on the operational environment and new interfaces can be activated that were not in use before. Consequently, integration planning for transition, should consider the need for additional interface checking and other potential verification. Moving into the operational environment implies that the system, or the parts being transitioned, can require validation as well. Overall, transition can require careful attention to maintaining integration of the elements that are affected.

For software systems, the purpose of the transition process is to establish a capability for a system to provide services in a different environment.

The transition process is often used for recurring deployments of software to different environments, for example from a development environment to an integration or test environment, or between various test environments. Transition for software systems can involve the physical relocation of hardware, the installation and activation or deactivation of physical or virtual infrastructure or enabling systems in different locations, or no change to the physical infrastructure. Transition can involve transfer between organizations. Integration of humans and procedures into the system is typically performed during the transition process.

6.4.5.11 Validation process

Validation of the implementation of certain stakeholder and system requirements are not always possible using the complete system, due, for example, to security, safety, physical, or economic constraints. Such validation is addressed by forming temporary aggregates to exercise the concerned requirements.

The validation process is regularly used by the system integration activities. The purpose of validation is to obtain the maximum confidence in the use of the system, before its final delivery, i.e. transfer of property from supplier to acquirer. Because of the number of requirements to validate, time and cost can exceed expectations, so a validation strategy has to be established.

A validation strategy is based on the balance between what is desired to be checked, that is, stakeholder requirements, using relevant validation techniques (e.g. demonstration, test), and what can be checked considering all constraints or limits (e.g. technical feasibility, cost, time, availability of validation capabilities and qualified personnel, contractual constraints). The difference between what should be checked and what can be checked for the complete end-to-end SoI reflects potential risks. The selection of validation actions should be made according to acceptable risks if validation actions are dropped out.

NOTE Concerning the integration of the SoI in its environmental context, or context of use, attributes to be validated include in particular the end-to-end performance, the usability, the interoperability with the other systems, existing or not, and the physical environment resistance thresholds (e.g. mechanical, climatic, chemical, bacteriological, electromagnetic). Operational scenarios, as defined during the concept definition activities early in the development stage, are used in particular to validate the operational concept on which the SoI lies. The validation process confirms that a work product fulfils requirements for a specific intended use of an integrated software function.

The validation process is linked to the integration process. In particular, validation should:

— provide potential inputs from the validation strategy to elaborate the integration strategy, in particular those inputs that address concerns that can best be examined in the intended operational environment, such as safety and security;

— perform validation actions to check the implementation of stakeholder requirements that are relevant with concerned submitted aggregate of system elements;

— when anomalies are identified, apply the validation process to correct the concerned processes.