6.4.5 Design Definition process

6.4.5.1 Purpose

The purpose of the Design Definition process is to provide sufficient detailed data and information about the system and its elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

For software systems, design activities typically iterate with activities in System/Software Requirements Definition and Architecture Definition. Design definition is typically applied iteratively and incrementally to develop a detailed design, including software elements, interfaces, databases, and user documentation. Software design is usually concurrent with software implementation, integration, verification, and validation. Annex H discusses software design using agile methods. During design and implementation, further process application refines allocation of evolving requirements among software elements.

NOTE 1 The Design Definition process is driven by requirements that have been vetted through the architecture and more detailed analyzes of feasibility. Architecture focuses on suitability, viability, and desirability, whereas design focuses on compatibility with technologies and other design elements and feasibility of implementation and integration. An effective architecture is as design-agnostic as possible to allow for maximum flexibility in the design trade space.

NOTE 2 This process provides feedback to the software system architecture to consolidate or confirm the allocation, partitioning and alignment of architectural entities.

6.4.5.2 Outcomes

As a result of the successful implementation of the Design Definition process:

a) Design characteristics of each system element are defined.

b) System/software requirements are allocated to system elements.

c) Design enablers necessary for design definition are selected or defined.

d) Interfaces between system elements composing the system are defined or refined.

e) Design alternatives for system elements are assessed.

f) Design artifacts are developed.

g) Any enabling systems or services needed for design definition are available.

h) Traceability of the design characteristics to the architectural entities of the system architecture is established.

NOTE Design definition considers applicable technologies and their contribution to the system solution. Design provides the 'implement-to' level of the definition, such as drawings, state diagrams, stories, and detailed design descriptions. For software elements, this process can result in a detailed design description that can be verified against requirements and the software architecture. Even if the software design is not fully specified in a formal description, it is sufficiently detailed to permit software implementation (construction) and test planning.

6.4.5.3 Activities and tasks

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

NOTE The SWEBOK, *Guide to the Software Engineering Body of Knowledge*, provides detailed discussion on software design. This knowledge area addresses fundamentals, key issues, design strategies and methods, and design notations.

a) **Prepare for software system design definition.** This activity consists of the following tasks:

1) Define the design definition strategy, consistent with the selected life cycle model and anticipated design artifacts.

NOTE The software design strategy can include initial or incremental decomposition into system elements; creation of various views of automated procedures, data structures and control systems; selection of design patterns, or progressively more detailed definition of objects and their relationships.

2) Select and prioritize design principles and design characteristics.

NOTE Design principles include controlling ideas such as abstraction, modularization and encapsulation, separation of interface and implementation, concurrency, and persistence of data. Security considerations include the principle of least privilege, layered defenses, restricted access to system services, and other considerations to minimize and defend the system attack surface. Design characteristics include, for example, availability, fault tolerance and resilience, scalability, usability, capacity and performance, testability, portability, and affordability.

3) Identify and plan for the necessary enabling systems or services needed to support design definition.

NOTE This includes identification of requirements and interfaces for the enabling systems. Enabling systems for design definition include selection of software and system platforms, programming languages, design notations and tools for collaboration and design development, design reuse repositories (for product lines, design patterns, and design artifacts), and design standards.

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

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

b) **Establish designs related to each software system element.** This activity consists of the following tasks:

1) Transform architectural and design characteristics into the design of software system elements.

NOTE Characteristics apply to physical and logical system elements, such as database structures, provisions for memory and storage, software processes and controls, external interfaces such as user interfaces, or services. ISO 9241-210 provides human centred design/ergonomic design guidelines.

2) Define and prepare or obtain the necessary design enablers.

NOTE Design enablers include models, equations, algorithms, calculations, formal expressions and values of parameters, patterns, and heuristics, which are associated with design characteristics using adequate representation such as drawings, logical diagrams, flowcharts, coding conventions, logic patterns, information models, business rules, user profiles, scenarios, use cases or user stories, and tables of metrics and their values, e.g., function points or user story points.

3) Examine design alternatives and feasibility of implementation.

NOTE 1 For the software system and software elements, typically reuse, adaptation, outsourced service, or new development are examined.

NOTE 2 Assess the feasibility of realizing design characteristics. If warranted by assessment results, examine other alternative design options or perform trade-offs in the architecture or requirements when design characteristics are impractical to implement.

4) Refine or define the interfaces among the software system elements and with external entities.

NOTE Interfaces are identified and defined in the Architecture Definition process (see 6.4.4) to the level or extent needed for the architecture intent and understanding. These are refined in the Design Definition process based on the design characteristics, interfaces, and interactions of software elements with other elements composing the software system and with external entities. Additional interfaces are sometimes identified and defined that were not addressed in the architecture definition.

5) Establish the design artifacts.

NOTE This task formalizes the design characteristics of the software system elements through dedicated artifacts, depending on the implementation technology. Examples of artifacts include prototypes, data models, pseudocode, entity-relationship diagrams, use cases, user role and privilege matrixes, interface specifications, service descriptions, and

– All rights reserved

procedures. Design artifacts are developed, obtained, or modified for selected alternatives. The data is associated with detailed acceptable margins for implementation (if relevant at this process or task iteration).

c) **Assess alternatives for obtaining software system elements.** This activity consists of the following tasks:

1) Determine technologies required for each element composing the software system.

NOTE Several technologies are sometimes used for a given software system element, e.g., internet presence, embedded systems, adaptation of open source software, human operator roles.

2) Identify candidate alternatives for the software system elements.

NOTE Alternatives include newly designed and constructed items; adaptations of existing product lines, components, objects, or services; or acquisition or reuse of Non-Developed Items (NDI). NDI include COTS (Commercial-Off-The-Shelf) or FOSS (Free and Open Source Software) packages or elements, reuse of a previous design, or existing assets, including acquirer provided items.

3) Assess each candidate alternative against criteria developed from expected design characteristics and element requirements to determine suitability for the intended application.

NOTE A make-or-buy decision and resulting implementation and integration approach typically involve trade-offs of the design criteria, including cost. Design choices commonly consider enabling systems required to test the candidate alternative (test-driven design and development) and sustainability over the system life, including maintenance costs. The Maintenance process can be used to determine the suitability of the design for long-term maintenance and sustainability.

4) Choose the preferred alternatives among candidate design solutions for the software system elements.

NOTE The System Analysis process can be used for analyzes and assessments to support the Decision Management process in performing the selection. Design reviews are conducted using the Validation process.

d) **Manage the design.** This activity consists of the following tasks:

1) Capture the design and rationale.

NOTE Commonly captured information includes the software system elements and affiliated requirements and design data, e.g., for software elements, internal and external interfaces, data structures, implementation and test requirements, unit aggregation data for integration, and test cases. Rationale typically includes information about major implementation options and enablers. The resultant design is controlled in accordance with the strategy.

2) Establish traceability between the detailed design elements, the system/software requirements, and the architectural entities of the software system architecture.

NOTE 1 This task facilitates providing feedback to the Architecture Definition process for potential modifications, for example, to modify the allocation of software system elements in order to obtain the expected architectural characteristics; or possibly to modify the expected architectural characteristic due to factors discovered during the design process, or to make stakeholders aware of the potential impacts.

NOTE 2 Through the life cycle, bidirectional traceability is maintained between the design and the verification methods or techniques, and software system element requirements. Allocations and design properties are assigned to software elements, software units and affiliated artifacts, at a detailed enough level to permit software testing and implementation, including construction.

3) Determine the status of the software system and element design.

NOTE 1 The Measurement process is used to establish measures for the completeness and quality of the design as it progresses. The Verification and Validation processes are invoked to verify and validate the detailed design and implementation

NOTE 2 This includes periodic assessment of the design characteristics in case of evolution of the software system and of its architecture, as well as forecasting potential obsolescence of components and technologies, their replacement by others over time in the life cycle of the software system, and the consequences for the design definition. The Risk Management process is typically applied to evaluate risks in the design strategy, initial design, and the evolving design.

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

NOTE The Configuration Management process is used to establish and maintain configuration items and baselines for artifacts such as design models. This process identifies candidates for the baseline, and the Information Management process controls the information items, such as design descriptions and specifications.

6.4.6 System Analysis process

6.4.6.1 Purpose

The purpose of the System Analysis process is to provide a rigorous basis of data and information for technical understanding to aid decision-making across the life cycle.

The System Analysis process applies to the development of inputs needed for any technical assessment. It can provide confidence in the utility and integrity of system requirements, architecture, and design. System analysis covers a wide range of differing analytic functions, levels of complexity, and levels of rigor. It includes mathematical analysis, modelling, simulation, experimentation, and other techniques to analyze technical performance, system behavior, feasibility, affordability, critical quality characteristics, technical risks, life cycle costs, and to perform sensitivity analysis of the potential range of values for parameters across all life cycle stages. It is used for a wide range of analytical needs concerning operational concepts, determination of requirement values, resolution of requirements conflicts, assessment of alternative architectures or system elements, and evaluation of engineering strategies (integration, verification, validation, and maintenance). Formality and rigor of the analysis will depend on the criticality of the information need or work product supported, the amount of information/data available, the size of the project, and the schedule for the results.

NOTE The System Analysis process can be employed for the entire software system or any element. This process is often used in conjunction with the Decision Management process.

6.4.6.2 Outcomes

As a result of the successful implementation of the System Analysis process:

a) System analyzes needed are identified.

b) System analysis assumptions and results are validated.

c) System analysis results are provided for decisions.

d) Any enabling systems or services needed for system analysis are available.

e) Traceability of the system analysis results is established.

6.4.6.3 Activities and tasks

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

a) **Define the system analysis strategy and prepare for system analysis.** This activity consists of the following tasks:

1) Identify the problem or question that requires analysis.

NOTE This includes technical, functional, and non-functional objectives of the analysis. Non-functional objectives include critical quality characteristics, various properties, technology maturity, and technical risks. The problem statement or question to be answered by the analysis is essential to establish the objectives of the analysis and the expectations and utility of the results.

2) Identify the stakeholders of the analysis.

3) Define the scope, objectives, and level of fidelity of the analysis.

NOTE The necessary level of fidelity (accuracy or precision) is a factor in determining the appropriate level of rigor.

– All rights reserved

4) Select the methods to support the analysis.

NOTE The methods are chosen based on time, cost, fidelity, technical drivers, and criticality of analysis. Analysis methods have a wide range of levels of rigor and include expert judgment, worksheet computations, parametric estimates and calculations, historical data and trend analysis, engineering models, simulation, visualization, and prototyping. Due to cost and schedule constraints, most projects typically perform system analysis only for critical characteristics.

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

NOTE This task includes identification of requirements and interfaces for the enabling systems. The system analysis enabling systems include the tools, relevant models, and potential data repositories needed to support the analysis. The methods chosen will be a major factor in determining what tools are appropriate to support the analysis. This also includes determining the availability of reusable or other relevant models and data, or resources.

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

NOTE The Infrastructure Management process enables the provision of systems analysis services. The Validation process is used to objectively confirm that the enabling system achieves its intended use for its enabling functions.

7) Collect the data and inputs needed for the analysis.

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

1) Identify and validate contexts and assumptions.

2) Apply the selected analysis methods to perform the required analysis.

3) Review the analysis results for quality and validity.

NOTE The results are coordinated with associated analyzes that have been previously completed.

4) Establish conclusions and recommendations.

NOTE The appropriate subject matter experts and stakeholders are identified and engaged in this task.

5) Record the results of the system analysis,

c) **Manage the system analysis**. This activity consists of the following tasks:

1) Maintain traceability of the analysis results.

NOTE Through the life cycle, bidirectional traceability is maintained between the analysis results and any software system item for which the analysis is supporting a decision or providing rationale (e.g., system/software requirement values, architecture alternatives). This is often facilitated by an appropriate data repository.

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

NOTE The Configuration Management process is used to establish and maintain configuration items and baselines. This process identifies candidates for the baseline, and the Information Management process controls the information items. For this process, the analysis results or reports are typical information items that are managed.

6.4.7 Implementation process

6.4.7.1 Purpose

The purpose of the Implementation process is to realize a specified system element.

This process transforms requirements, architecture, and design, including interfaces, into actions that create a system element according to the practices of the selected implementation technology, using appropriate technical specialties or disciplines. This process results in a system element that satisfies specified system requirements (including allocated and derived requirements), architecture, and design.

For software systems, the purpose of the Implementation process is to realize a software system element.

Software system elements can include hardware, software, and services. For software implementation, this process transforms specified designs, behavior, interfaces and implementation constraints into actions that create a software system element implemented as a software product or service, also known as a "software item". Software implementation results in a software element that satisfies specified requirements through verification and stakeholder requirements through validation. Software implementation includes various combinations of construction (coding of newly built software elements), acquisition of new software packages (e.g., from open source or a commercial or organizational source) or re-use of existing elements (with or without modification).

Software implementation commonly involves use of the Agreement processes to obtain non-developmental items (NDI), such as hardware and operating systems (the platform) or enabling systems and services. Software implementation is usually performed concurrently with software integration. Implementation is typically performed along with all of the Technical Management processes and many of the Technical processes, especially:

a) The Verification process, which provides objective evidence that the software implementation fulfills its specified requirements and identifies anomalies (errors, defects, faults) in implementation-related information items, (e.g., system/software requirements, architecture, design, or other descriptions), processes, software elements, items, units;

b) The Validation process, which confirms that the implementation fulfils requirements for a specific intended use of a software work product.

6.4.7.2 Outcomes

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

a) Implementation constraints that influence the requirements, architecture, or design are identified.

b) A system element is realized.

c) A system element is packaged or stored.

d) Any enabling systems or services needed for implementation are available.

e) Traceability is established.

6.4.7.3 Activities and tasks

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

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

1) Define an implementation strategy, with consideration of the following:

i) development policies and standards, including standards that govern applicable safety, security, privacy and environmental practices; programming or coding standards; unit test policies; and language-specific standards for implementing security features;

ii) For reused or adapted software, methods to determine the level, source, and suitability of the reused system elements and security of the supply chain;

iii) procedures and methods for software development (construction) and development of unit tests; and the use of peer reviews, unit tests, and walkthroughs during implementation;

iv) use of CM control during software construction;

v) change management considerations for manual processes;

vi) implementation priorities to support data and software migration and transition, along with retirement of legacy systems;

– All rights reserved

vii) creation of manual or automated test procedures to verify that a software unit meets its requirements before creation of the software unit (test-driven development); and

viii) comprehensive or specialized life cycle development and support environments for realizing and managing requirements, models and prototypes, deliverable system or software elements, and test specifications and test cases.

NOTE The implementation strategy is commonly recorded in a project's SDP or SEMP, or sometimes in a PMP.

2) Identify constraints from the implementation strategy and implementation technology on the system/software requirements, architecture characteristics, design characteristics, or implementation techniques.

NOTE 1 Constraints include current or anticipated limitations of the chosen implementation technology (e.g., for software, the operating system, database management system, web services), acquirer furnished materials or system elements for adaptation, and limitations resulting from the use of required implementation-enabling systems.

NOTE 2 The implementation strategy for software typically identifies and allocates 'implement-to' criteria, e.g., software architecture and design characteristics, system/software requirements including software assurance, usability considerations, configuration management, traceability, or other conditions to be satisfied. These criteria can clarify appropriate unit aggregation levels, specifications, and constraints.

3) Identify and plan for the necessary and distinct software environments, including enabling systems or services needed to support development and testing.

NOTE Implementation of software commonly uses distinct environments that are separated under configuration control from the operational (production) environment. Common Implementation process, enabling systems, and services include comprehensive or specialized life cycle development and support environments for realizing and managing requirements, models and prototypes, deliverable elements, and test environments, specifications and test cases; simulators for external systems, training systems; and content management systems for user documentation.

4) Obtain or acquire access to the software environments and other enabling systems or services.

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

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

NOTE Throughout the Implementation process the Verification process is used to objectively confirm the system elements conform to requirements. The Validation process is used to objectively confirm the element is suitable to be used in its intended operational environment according to stakeholder requirements.

1) Realize or adapt software elements, according to the strategy, constraints, and defined implementation procedures.

NOTE 1 Software elements are acquired, identified for reuse from organizational assets, or developed (constructed). Software elements that are acquired can range from a simple product purchase in accordance with organizational or project purchasing rules to a complex acquisition of a software system that involves the Acquisition and Supply processes. Adaptation includes configuration of software elements that are reused or modified. Construction can involve software coding, adaptive reuse and integration of existing units, refactoring, database development, and construction of manual or automated test procedures for each unit.

NOTE 2 For software elements that are developed, at the lowest level of implementation executable software units are constructed (often with associated data structures, application programming interfaces, service descriptions, user documentation, test cases, or other elements), controlled, made available to authorized roles, and stored according to the CM procedures for development artifacts.

NOTE 3 The *SWEBOK, Guide to the Software Engineering Body of Knowledge* provides detailed discussion on Software Construction. This knowledge area addresses fundamentals, management, measurement, practical considerations (e.g., construction design, languages, testing, reuse and integration), construction technologies (e.g., object oriented, error and exception handling, executable models, distributed software), and tools and environments.

2) Realize or adapt hardware elements of software systems.

NOTE Hardware elements are acquired or fabricated using applicable techniques relevant to the physical implementation technology and materials selected. As appropriate, hardware elements are verified for conformance to specified system requirements and critical quality characteristics. In the case of repeated system element implementation (e.g., mass production, replacement system elements) the implementation procedures and fabrication processes are defined and can be automated to achieve consistent and repeatable producibility. Some common hardware elements in software systems include integrations of acquired COTS systems, special modifications, e.g., for test or operational environments, and hardware controls with embedded software.

3) Realize or adapt service elements of software systems.

NOTE Service elements include a set of services to be provided. ISO/IEC 20000 (IEEE Std 20000) applies to management of system elements realized in services, including strategy, design, and transition. As appropriate, service elements are verified for conformance to the system requirements and service criteria. For example, operational resource elements are verified for conformance to the system requirements and operational concept. Service elements can include network communications, training, software packaging and distribution services, software customization services for customer-specific needs, operational and security monitoring, and user assistance.

4) Evaluate software unit and affiliated data or other information according to the implementation strategy and criteria.

NOTE 1 Criteria for evaluation commonly include satisfaction of unit requirements and test criteria, unit test coverage, traceability requirements, consistency with software element requirements or design, internal unit requirement consistency, and feasibility for further process activity, e.g., integration, verification, validation, operations and maintenance.

NOTE 2 Use the *Manage results of implementation* activity to record construction and address anomalies.

5) Package and store the software system element.

NOTE Contain the software system element in order to achieve continuance of its characteristics. Conveyance and storage, and their durations, can influence the specified containment. For software, a master copy of the implemented software (electronic or on physical media) is stored in a controlled location and made available to authorized roles (e.g., for use in the Integration and Transition processes). Configuration and product information is captured by the Configuration Management and Information Management processes when the element is stored.

6) Record objective evidence that the software system element meets requirements.

NOTE Evidence is provided in accordance with supply agreements, legislation and organization policy. Evidence includes element modifications made due to processing changes or non-conformances found during the Verification and Validation processes. The objective evidence is part of the element's as-implemented configuration baseline established through the Configuration Management process and includes the results of unit testing, analysis, inspections, walk-through events, demonstrations, product or technical reviews, or other verification exercises.

c) **Manage results of implementation.** This activity consists of the following tasks:

1) Record implementation results and anomalies encountered.

NOTE This includes anomalies due to the implementation strategy, the implementation enabling systems, or incorrect software system definition. The Project Assessment and Control and Quality Assurance processes are used to analyze the data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

2) Maintain traceability of the implemented software system elements.

NOTE 1 To support traceability throughout the life cycle during operations and maintenance, sources of software licenses and other system assets in the supply chain are recorded. The information management and configuration management processes are used to maintain license and maintenance support terms for a software application and its required infrastructure (host system). The ISO/IEC 19770 standards provide requirements for an IT asset management system.

NOTE 2 Bidirectional traceability is maintained between the implemented elements and the software system architecture; design, and related requirements, including interface requirements and definitions that are necessary for implementation; and validation and verification plans, procedures, and results.

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

– All rights reserved

NOTE The Configuration Management process is used to establish and maintain configuration items and baselines. This process identifies candidates for the baseline, and the Information Management process controls the information items. For this process, the software system elements (e.g., source code), software packages, and unit test results are typical artifacts that are baselined.

6.4.8 Integration process

6.4.8.1 Purpose

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

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

Software system integration iteratively combines implemented software system elements to form complete or partial system configurations in order to build a product or service. Software integration is typically performed daily or continuously during development and maintenance stages, using automated tools. Continuous integration involves frequent inclusion or replacement and archiving of items in software libraries under CM control.

NOTE Interfaces are defined by the Architecture Definition and Design Definition processes. The Integration process coordinates with these other processes to check that the interface definitions, as implemented and integrated, are adequate and that they take into account the integration needs.

6.4.8.2 Outcomes

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

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

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

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

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

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

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

g) Integration results and anomalies are identified.

h) Traceability of the integrated system elements is established.

6.4.8.3 Activities and tasks

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

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

1) Define the integration strategy.

NOTE 1 Integration builds sequences of progressively more complete software system element or software item configurations. It is dependent on applicable software system element availability and is consistent with a fault isolation and diagnosis strategy. Successive applications of the Integration process and the Verification process, and when appropriate the Validation process, are repeated for elements in the system structure until the system-of-interest has been realized. Simulators or prototypes are typically utilized for system elements that are not yet implemented, e.g.,

receiving data from interfacing systems. Integrating the implemented software system elements is based on the priorities of the related requirements and architecture definition, typically focusing on the interfaces, while minimizing integration time, cost, and risks. Software system integration commonly maintains version control through the Configuration Management process for selection of configuration items to be integrated.

NOTE 2 For software integration, the integration strategy typically is consistent with a regression strategy which is applied for re-verifying software elements when related software units (and potentially associated requirements, design and user documentation) are changed.

NOTE 3 Defining a strategy for software unit and element integration commonly accompanies defining the strategy for other processes that occur concurrently, such as:

i) The Implementation process to help ensure timely coordination of Implementation and Integration process tasks and enabling systems, e.g., combined software development and test environments to support automated or continuous implementation and integration of software units and elements.

ii) The Verification process to provide objective evidence that the integrated software fulfils its specified requirements and to identify anomalies (errors, defects, faults) in integration-related information items, (e.g., system/software requirements, architecture, design, test, or other descriptions), processes, software elements, items, units.

iii) The Validation process to confirm that a work product fulfils requirements for a specific intended use of an integrated software function.

iv) The Quality Assurance process to support integration process and work product audits and inspections and to address problem, non-conformance, or incident reporting and handling.

NOTE 4 The integration strategy is commonly recorded in a plan, e.g., an integration plan, or a project's SDP or SEMP.

2) Identify and define criteria for integration and points at which the correct operation and integrity of the interfaces and the selected software system functions will be verified.

NOTE 1 Detailed verification of the interfaces is performed using the Verification process. Software integration typically involves combining software elements, resulting in a set of integrated software elements, that is consistent with the software design, and that satisfies the functional and non-functional system/software requirements on an equivalent of the operational environment.

NOTE 2 For projects involving multiple suppliers or development teams, the availability of software system elements for integration is typically part of the project schedule with milestones under the Project Assessment and Control process. Integration proceeds as the software is verified in its functionality, performance, and suitability for site-specific or platform-specific environments. At major integration points, e.g., completion of a stage, element, or version, check points for reviews and validation with stakeholders are typically held. The frequency of these reviews is related to the selected life cycle model and development method.

3) Identify and plan for the necessary enabling systems or services needed to support integration.

NOTE This includes identification of requirements and interfaces for the enabling systems. Enabling systems for integration commonly include integration facilities, specialized equipment, training systems, discrepancy reporting systems, simulators, measurement devices, and environmental security. For software, this can involve regression test suites and CM systems for the integrated testing of software systems, incident and problem reporting systems, simulators representing external systems or undeveloped elements, and software library management systems for development operations. Changes or specializations needed for the enabling systems to support the integration tasks need to be identified and defined. Typically, the enabling systems or services used for integration during development stages can also help support system element integration as the software system and enabling environments evolve to operational status. This "DevOps" approach supports iterative software system implementation, integration, verification, transition, validation, operation and maintenance processes.

4) Obtain or acquire access to the enabling systems or services to be used to support integration.

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

5) Identify constraints for integration to be incorporated in the system/software requirements, architecture or design.

– All rights reserved

NOTE This includes requirements such as accessibility, supply chain security, safety for integrators, required interconnections for sets of implemented software system elements and for enablers, and interface constraints.

b) **Perform integration**. Successively integrate software system element configurations until the complete system is synthesized. This activity consists of the following tasks:

1) Obtain implemented software system elements in accordance with agreed schedules.

NOTE The implemented software system elements are provided from the developers or received from suppliers, the acquirer, or other resources and typically placed under CM control. The elements are handled in accordance with relevant health, safety, security and privacy considerations.

2) Integrate the implemented elements.

NOTE 1 This task is performed to achieve software system element configuration (complete or partial) connecting the implemented elements as prescribed in the integration strategy, using the defined procedures, interface control descriptions, and the related integration enabling systems.

NOTE 2 In terms of software, integrating the implemented elements can involve linking together pieces of object code or simply bringing together the implemented elements that are part of the software configuration in a methodical piece by piece approach. Software elements are typically compiled into a "build" so that branched units are properly linked or merged in the assembled element. Firmware elements are fabricated, often as prototypes, and installed in hardware elements. If software functions are not yet available for integration, emulated functionality (stubs or scaffolding) can be used to temporarily support integration of software elements or represent input from external interfaces. Successful aggregations result in an integrated software element, that is stored and available for further processing, i.e., additional software system element integration, verification, or validation.

NOTE 3 Anti-counterfeit, anti-tamper, system and software assurance and interoperability concerns can arise when performing integration and identifying and defining checkpoints. Integration and Verification processes often use fictitious data for security or privacy considerations. ISO/IEC/IEEE 15026 and the ISO/IEC 27000 series include information on assurance, integrity, and security considerations affecting integration.

3) Check that the integrated software interfaces or functions run from initiation to an expected termination within an expected range of data values.

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

c) **Manage results of integration.** This activity consists of the following tasks:

1) Record integration results and anomalies encountered.

NOTE This includes anomalies due to the integration strategy, the integration enabling systems, execution of the integration or incorrect system or element definition. Where inconsistencies exist at the interface between the system, its specified operational environment and systems that enable the utilization stage, the deviations lead to corrective actions. Anomaly resolution typically involves the Technical Processes, often repetitive application of the Implementation process. The Quality Assurance and Project Assessment and Control process are used to analyze the data to identify the root cause, enable corrective or improvement actions, and to record lessons learned.

2) Maintain traceability of the integrated software system elements.

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

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

NOTE The Configuration Management process is used to establish and maintain configuration items and baselines. The Integration process identifies candidates for the baseline, and the Information Management process controls the