# System Integration





---

***Lead Authors:*** *John Snoderly, Alan Faisandier, Scott Jackson*

---

System integration consists of taking delivery of the implemented system elements which compose the system-of-interest (SoI), assembling these implemented elements together, and performing the verification and validation actions (V&V actions) in the course of the assembly. The ultimate goal of system integration is to ensure that the individual system elements function properly as a whole and satisfy the design properties or characteristics of the system. System integration is one part of the realization effort and relates only to developmental items. Integration should not to be confused with the assembly of end products on a production line. To perform the production, the assembly line uses a different order from that used by integration.

☐

## Contents

# Definition and Purpose

System integration consists of a process that "*iteratively combines implemented system elements to form complete or partial system configurations in order to build a product or service. It is used recursively for successive levels of the system hierarchy.*" (ISO/IEC 15288 2015, 68). The process is extended to any kind of product system, service system, and enterprise system. The purpose of system integration is to prepare the SoI for final validation and transition either for use or for production. Integration consists of progressively assembling aggregates of implemented elements that compose the SoI as architected during design, and to check correctness of static and dynamic aspects of interfaces between the implemented elements.

The U.S. Defense Acquisition University (DAU) provides the following context for integration: *The integration process will be used . . . for the incorporation of the final system into its operational environment to ensure that the system is integrated properly into all defined external interfaces. The interface management process is particularly important for the success of the integration process, and iteration between the two processes will occur* (DAU 2010).

The purpose of system integration can be summarized as below:

- Completely assemble the implemented elements to make sure that the they are compatible with each other.
- Demonstrate that the aggregates of implemented elements perform the expected functions and meet measures of performance/effectiveness.

- Detect defects/faults related to design and assembly activities by submitting the aggregates to focused V&V actions.

Note: In the systems engineering literature, sometimes the term *integration* is used in a larger context than in the present topic. In this larger sense, it concerns the technical effort to simultaneously design and develop the system and the processes for developing the system through concurrent consideration of all life cycle stages, needs, and competences. This approach requires the "integration" of numerous skills, activities, or processes.

# Principles

## Boundary of Integration Activity

Integration can be understood as the whole bottom-up branch of the Vee Model, including the tasks of assembly and the appropriate verification tasks. See Figure 1 below:

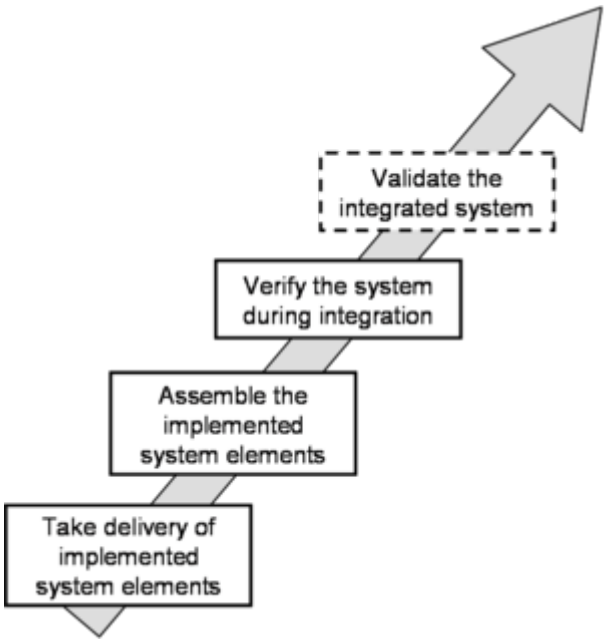


**Figure 1. Limits of Integration Activities.**
(SEBoK Original)

The assembly activity joins together, and physically links, the implemented elements. Each implemented element is individually verified and validated prior to entering integration. Integration then adds the verification activity to the assembly activity, excluding the final validation.

The final validation performs operational tests that

authorize the transition for use or the transition for production. Remember that system integration only endeavors to obtain pre-production prototypes of the concerned product, service, or enterprise. If the product, service, or enterprise is delivered as a unique exemplar, the final validation activity serves as acceptance for delivery and transfer for use. If the prototype has to be produced in several exemplars, the final validation serves as acceptance to launch their production. The definition of the optimized operations of assembly which will be carried out on a production line relates to the manufacturing process and not to the integration process.

Integration activity can sometimes reveal issues or anomalies that require modifications of the design of the system. Modifying the design is not part of the integration process but concerns only the design process. Integration only deals with the assembly of the implemented elements and verification of the system against its properties as designed. During assembly, it is possible to carry out tasks of finishing touches which require simultaneous use of several implemented elements (e.g., paint the whole after assembly, calibrate a biochemical component, etc.). These tasks must be planned in the context of integration and are not carried out on separate implemented elements and do not include modifications related to design.

## Aggregation of Implemented Elements

The integration is used to systematically assemble a higher-level system from lower-level ones (implemented system elements) that have been implemented. Integration often begins with analysis and simulations (e.g., various types of prototypes) and progresses through increasingly more realistic systems and system elements until the final product, service, or enterprise is achieved.

System integration is based on the notion of an aggregate - a subset of the system made up of several implemented elements (implemented system elements and physical interfaces) on which a set of V&V actions is applied. Each aggregate is characterized by a configuration which specifies the implemented elements to be physically assembled and their configuration status.

To perform V&V actions, a V&V configuration that includes the aggregate plus V&V tools is constituted.

The V&V tools are enabling products and can be simulators (simulated implemented elements), stubs or caps, activators (launchers, drivers), harness, measuring devices, etc.

## Integration by Level of System

According to the Vee Model, system definition (top-down branch) is done by successive levels of decomposition; each level corresponds to the physical architecture of systems and system elements. The integration (bottom-up branch) takes the opposite approach of composition (i.e., a level by level approach). On a given level, integration is done on the basis of the physical architecture defined during system definition.

## Integration Strategy

The integration of implemented elements is generally performed according to a predefined strategy. The definition of the integration strategy is based on the architecture of the system and relies on the way the architecture of the system has been designed. The strategy is described in an integration plan that defines the minimum configuration of expected aggregates, the order of assembly of these aggregates in order to support efficient subsequent verification and validation actions (e.g., inspections and/or testing), techniques to check or evaluate interfaces, and necessary capabilities in the integration environment to support combinations of aggregates. The integration strategy is thus elaborated starting from the selected verification and validation strategy. See the System Verification and System Validation topics.

To define an integration strategy, there are several possible integration approaches/techniques that may be used individually or in combination. The selection of integration techniques depends on several factors; in particular, the type of system element, delivery time, order of delivery, risks, constraints, etc. Each integration technique has strengths and weaknesses which should be considered in the context of the SoI. Some integration techniques are summarized in Table 1 below.

**Table 1. Integration Techniques.** (SEBoK Original)

| Integration Technique | Description |
|---|---|

| | |
|---|---|
| **Global Integration** | Also known as *big-bang integration*; all the delivered implemented elements are assembled in only one step.<br>• This technique is simple and does not require simulating the implemented elements not being available at that time.<br>• Difficult to detect and localize faults; interface faults are detected late.<br>• Should be reserved for simple systems, with few interactions and few implemented elements without technological risks. |
| **Integration "with the Stream"** | The delivered implemented elements are assembled as they become available.<br>• Allows starting the integration quickly.<br>• Complex to implement because of the necessity to simulate the implemented elements not yet available. Impossible to control the end-to-end "functional chains"; consequently, global tests are postponed very late in the schedule.<br>• Should be reserved for well-known and controlled systems without technological risks. |
| **Incremental Integration** | In a predefined order, either one or a very few implemented elements are added to an already integrated increment of implemented elements.<br>• Fast localization of faults: a new fault is usually localized in lately integrated implemented elements or dependent of a faulty interface.<br>• Require simulators for absent implemented elements. Require many test cases, as each implemented element addition requires the verification of the new configuration and regression testing.<br>• Applicable to any type of architecture. |
| **Subsets Integration** | Implemented elements are assembled by subsets, and then subsets are assembled together (a subset is an aggregate); could also be called "functional chains integration".<br>• Time saving due to parallel integration of subsets; delivery of partial products is possible. Requires less means and fewer test cases than integration by increments.<br>• Subsets shall be defined during the design.<br>• Applicable to architectures composed of sub-systems. |

| | |
|---|---|
| **Top-Down Integration** | Implemented elements or aggregates are integrated in their activation or utilization order.<br>• Availability of a skeleton and early detection of architectural faults, definition of test cases close to reality, and the re-use of test data sets possible.<br>• Many stubs/caps need to be created; difficult to define test cases of the leaf-implemented elements (lowest level).<br>• Mainly used in software domain. Start from the implemented element of higher level; implemented elements of lower level are added until leaf-implemented elements. |
| **Bottom-Up Integration** | Implemented elements or aggregates are integrated in the opposite order of their activation or utilization.<br>• Easy definition of test cases; early detection of faults (usually localized in the leaf-implemented elements); reduce the number of simulators to be used. An aggregate can be a sub-system.<br>• Test cases shall be redefined for each step, drivers are difficult to define and realize, implemented elements of lower levels are "over-tested", and does not allow architectural faults to be quickly detected.<br>• Mainly used in software domain, but can be used in any kind of system. |
| **Criterion Driven Integration** | The most critical implemented elements compared to the selected criterion are first integrated (dependability, complexity, technological innovation, etc.). Criteria are generally related to risks.<br>• Allows early and intensive testing of critical implemented elements; early verification of design choices.<br>• Test cases and test data sets are difficult to define. |

Usually, a mixed integration technique is selected as a trade-off between the different techniques listed above, allowing optimization of work and adaptation of the process to the system under development. The optimization takes into account the realization time of the implemented elements, their delivery scheduled order, their level of complexity, the technical risks, the availability of assembly tools, cost, deadlines, specific personnel capability, etc.

# Process Approach

## Activities of the Process

Major activities and tasks performed during this process include:

- **Establishing the integration plan** (this activity is carried out concurrently to the design activity of the system) that defines:
  - The optimized integration strategy – order of aggregates assembly using appropriate integration techniques.
  - The V&V actions to be processed for the purpose of integration.
  - The configurations of the aggregates to be assembled and verified.
  - The integration means and verification means (dedicated enabling products) that may include assembly procedures, assembly tools (harness, specific tools), V&V tools (simulators, stubs/caps, launchers, test benches, devices for measuring, etc.), and V&V procedures.
- **Obtain the integration means** and verification means as defined in the integration plan. The acquisition of the means can be accomplished through various ways such as procurement, development, reuse, and sub-contracting; usually the acquisition of the complete set of means is a mix of these methods.
- **Take delivery** of each implemented element:
  - Unpack and reassemble the implemented element with its accessories.
  - Check the delivered configuration, conformance of implemented elements and compatibility of interfaces, and ensure the presence of mandatory documentation.
- **Assemble the implemented elements** into aggregates:
  - Gather the implemented elements to be assembled, the integration means (assembly tools, assembly procedures), and the verification means (V&V tools and procedures).
  - Connect the implemented elements to each other using assembly tools to constitute aggregates in the order prescribed by the integration plan and in assembly procedures.
  - Add or connect the V&V tools to the aggregates as

predefined.
    - Carry out eventual operations of welding, gluing, drilling, tapping, adjusting, tuning, painting, parametering, etc.
- **Verify each aggregate**:
    - Check the aggregate is correctly assembled according to established procedures.
    - Perform the verification process that uses verification and validation procedures and check that the aggregate shows the right design properties/specified requirements.
    - Record integration results/reports and potential issue reports, change requests, etc.

## Artifacts and Ontology Elements

This process may create several artifacts such as:

- an integrated system
- assembly tools
- assembly procedures
- integration plans
- integration reports
- issue/anomaly/trouble reports
- change requests (about design)

This process utilizes the ontology elements discussed in Table 2.

**Table 2. Main Ontology Elements as Handled within System Integration.** (SEBoK Original)

| Element | Definition |
|---|---|
| | **Attributes** |
| **Aggregate** | An aggregate is a subset of the system made up of several system elements or systems on which a set of verification actions is applied. |
| | Identifier, name, description |
| **Assembly Procedure** | An assembly procedure groups a set of elementary assembly actions to build an aggregate of implemented system elements. |
| | Identifier, name, description, duration, unit of time |

| | |
|---|---|
| **Assembly Tool** | An assembly tool is a physical tool used to connect, assemble, or link several implemented system elements to build aggregates (specific tool, harness, etc.). |
| **Risk** | Identifier, name, description<br>An event having a probability of occurrence and a gravity degree on its consequence onto the system mission or on other characteristics (used for technical risk in engineering). A risk is the combination of vulnerability and of a danger or a threat. |
| **Rationale** | Identifier, name, description, status<br>An argument that provides the justification for the selection of an engineering element.<br>Identifier, name, description (rationale, reasons for defining an aggregate, assembly procedure, assembly tool) |

Note: verification and validation ontology elements are described in the System Verification and System Validation topics.

The main relationships between ontology elements are presented in Figure 2.

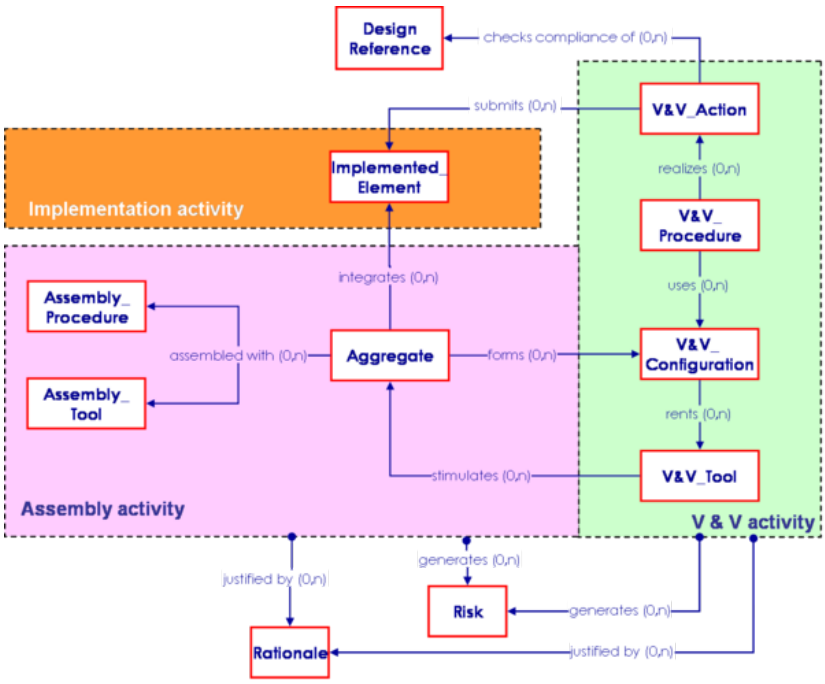


**Figure 2. Integration Elements Relationships with Other Engineering Elements.** (SEBoK Original)

## Checking and Correctness of Integration

The main items to be checked during the integration process include the following:

- The integration plan respects its template.
- The expected assembly order (integration strategy) is realistic.
- No system element and physical interface set out in the system design document is forgotten.
- Every interface and interaction between implemented elements is verified.
- Assembly procedures and assembly tools are available and validated prior to beginning the assembly.
- V&V procedures and tools are available and validated prior to beginning the verification.
- Integration reports are recorded.

## Methods and Techniques

Several different approaches are summarized above in the section Integration Strategy (above) that may be used for integration, yet other approaches exist. In particular, important integration strategies for intensive software systems include: vertical integration, horizontal integration, and star integration.

### Coupling Matrix and N-squared Diagram

One of the most basic methods to define the aggregates and the order of integration would be the use of N-Squared diagrams (Grady 1994, 190).

In the integration context, the coupling matrices are useful for optimizing the aggregate definition and verification of interfaces:

- The integration strategy is defined and optimized by reorganizing the coupling matrix in order to group the implemented elements in aggregates, thus minimizing the number of interfaces to be verified between aggregates (see Figure 3).

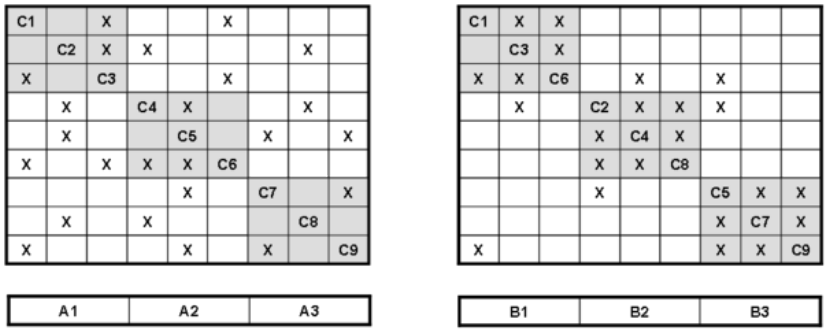


**Figure 3. Initial Arrangement of Aggregates on the Left; Final Arrangement After Reorganization on the Right.** (SEBoK Original)

- When verifying the interactions between aggregates, the matrix is an aid tool for fault detection. If by adding an implemented element to an aggregate an error is detected, the fault can be related to the implemented element, to the aggregate, or to the interfaces. If the fault is related to the aggregate, it can relate to any implemented element or any interface between the implemented elements internal to the aggregate.

## Application to Product Systems, Service Systems, and Enterprise Systems

As the nature of implemented system elements and physical interfaces is different for these types of systems, the aggregates, the assembly tools, and the V&V tools are different. Some integration techniques are more appropriate to specific types of systems. Table 3 below provides some examples.

**Table 3. Different Integration Elements for Product, Service, and Enterprise Systems.** (SEBoK Original)

| Element | Product System | Service System | Enterprise System |
|---|---|---|---|
| **System Element** | Hardware Parts (mechanics, electronics, electrical, plastic, chemical, etc.)<br>Operator Roles<br>Software Pieces | Processes, data bases, procedures, etc.<br>Operator Roles<br>Software Applications | Corporate, direction, division, department, project, technical team, leader, etc.<br>IT components |

| | | | |
|---|---|---|---|
| **Physical Interface** | Hardware parts, protocols, procedures, etc. | Protocols, documents, etc. | Protocols, procedures, documents, etc. |
| **Assembly Tools** | Harness, mechanical tools, specific tools<br>Software Linker | Documentation, learning course, etc. | Documentation, learning, moving of office |
| **Verification Tools** | Test bench, simulator, launchers, stub/cap | Activity/scenario models, simulator, human roles rehearsal, computer, etc.<br>Skilled Experts | Activity/scenario models, simulator, human roles rehearsal |
| **Validation Tools** | Operational environment | Operational environment | Operational environment |
| **Recommended Integration Techniques** | Top down integration technique<br>Bottom Up Integration technique | Subsets integration technique (functional chains) | Global integration technique<br>Incremental integration |

## Practical Considerations

Key pitfalls and good practices related to system integration are described in the next two sections.

## Pitfalls

Some of the key pitfalls encountered in planning and performing SE Measurement are provided in Table 4.

**Table 4. Major Pitfalls with System Integration.** (SEBoK Original)

| Pitfall | Description |
|---|---|
| **Delay of expected element** | The experience shows that the implemented elements always do not arrive in the expected order and the tests never proceed or result as foreseen; therefore, the integration strategy should allow a great flexibility. |
| **Big-bang not appropriate** | The "big-bang" integration technique is not appropriate for a fast detection of faults. It is thus preferable to verify the interfaces progressively all along the integration. |

| | |
|---|---|
| **Integration plan too late** | The preparation of the integration activities is planned too late in the project schedule, typically when first implemented elements are delivered. |

## Good Practices

Some good practices gathered from the references are provided in Table 5.

**Table 5. Proven Practices with System Integration.**
(SEBoK Original)

| Practice | Description |
|---|---|
| **Start earlier development of means** | The development of assembly tools and verification and validation tools can be take as long as the system development itself. It should be started as early as possible as soon as the preliminary design is nearly frozen. |
| **Integration means seen as enabling systems** | The development of integration means (assembly tools, verification, and validation tools) can be seen as enabling systems, using system definition and system realization processes as described in this SEBoK, and managed as projects. These projects can be led by the project of the corresponding system-of-interest, but assigned to specific system blocks, or can be subcontracted as separate projects. |
| **Use coupling matrix** | A good practice consists in gradually integrating aggregates in order to detect faults more easily. The use of the coupling matrix applies for all strategies and especially for the bottom up integration strategy. |
| **Flexible integration plan and schedule** | The integration process of complex systems cannot be easily foreseeable and its progress control difficult to observe. This is why it is recommended to plan integration with specific margins, using flexible techniques, and integrating sets by similar technologies. |
| **Integration and design teams** | The integration responsible should be part of the design team. |

# References

## Works Cited

DAU. February 19, 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition

University (DAU)/U.S. Department of Defense (DoD).

Faisandier, A. 2012. *Systems Architecture and Design*. Belberaud, France: Sinergy'Com.

ISO/IEC/IEEE. 2015.*Systems and Software Engineering - System Life Cycle Processes*.Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 15288:2015.

## Primary References

INCOSE. 2010. *Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities*. Version 3.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.

NASA. 2007. *Systems Engineering Handbook*. Washington, D.C.: National Aeronautics and Space Administration (NASA), NASA/SP-2007-6105.

## Additional References

Buede, D.M. 2009. *The Engineering Design of Systems: Models and Methods.* 2nd ed. Hoboken, NJ, USA: John Wiley & Sons Inc.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense. February 19, 2010.

Gold-Bernstein, B. and W.A. Ruh. 2004. *Enterprise integration: The essential guide to integration solutions*. Boston, MA, USA: Addison Wesley Professional.

Grady, J.O. 1994. *System integration*. Boca Raton, FL, USA: CRC Press, Inc.

Hitchins, D. 2009. "What are the General Principles Applicable to Systems?" INCOSE *Insight* 12(4):59-63.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions*. Hoboken, NJ, USA: John Wiley & Sons.

Reason, J. 1997. *Managing the Risks of Organizational Accidents.* Aldershot, UK: Ashgate Publishing Limited.

< Previous Article | Parent Article | Next Article >
**SEBoK v. 2.11, released 25 November 2024**