Module 4

Requirements Verification and Validation

Agenda

- Verification & Validation Overview
- Integration Techniques

V&V Applicability

- V&V is typically applied to
  - Requirements
  - Entities that result from the requirements (component, sub-system, system)

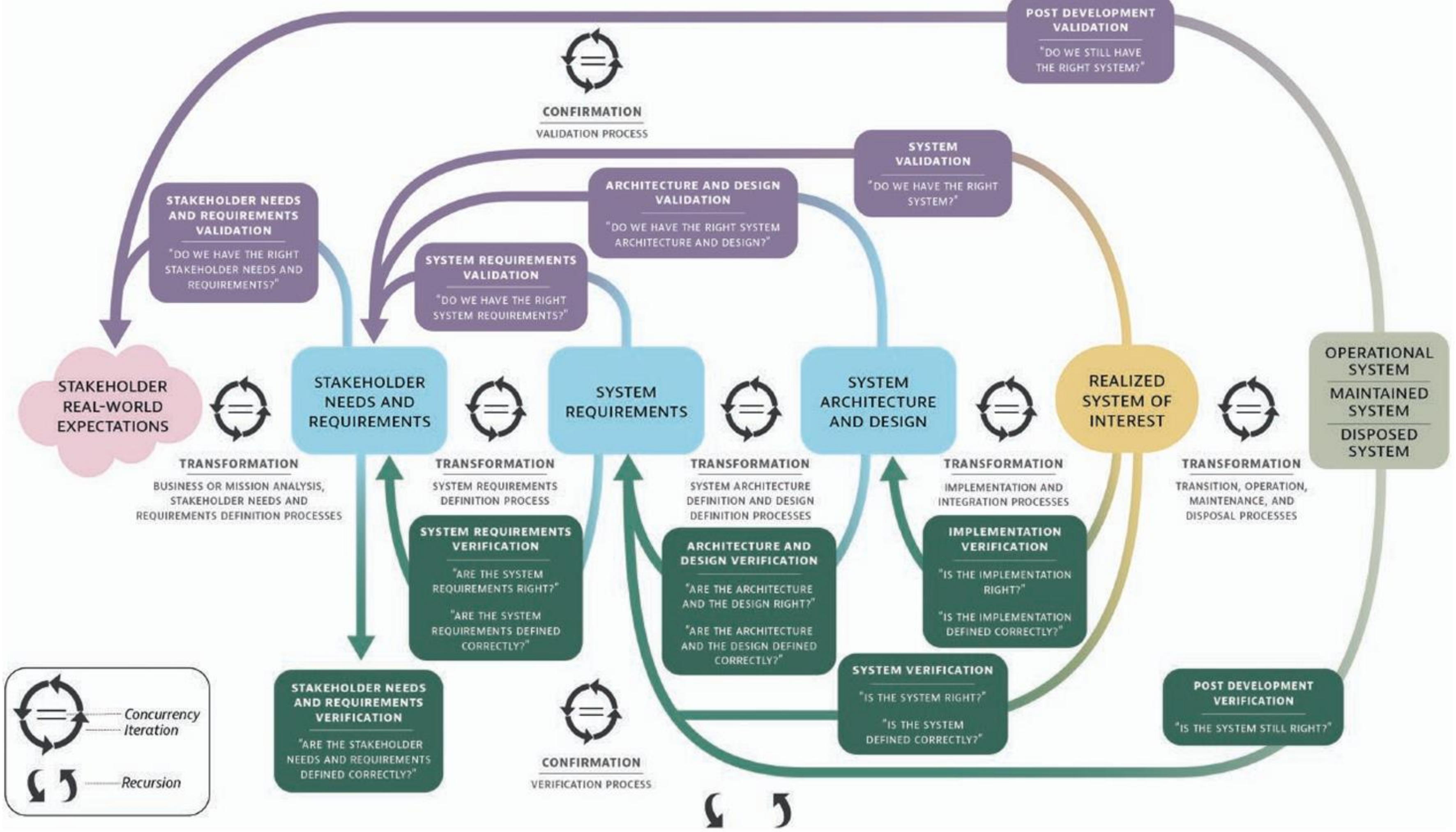

POST DEVELOPMENT VALIDATION
"DO WE STILL HAVE THE RIGHT SYSTEM?"
CONFIRMATION
VALIDATION PROCESS
SYSTEM VALIDATION
"DO WE HAVE THE RIGHT SYSTEM?"
ARCHITECTURE AND DESIGN VALIDATION
"DO WE HAVE THE RIGHT SYSTEM ARCHITECTURE AND DESIGN?"
STAKEHOLDER NEEDS AND REQUIREMENTS VALIDATION
"DO WE HAVE THE RIGHT STAKEHOLDER NEEDS AND REQUIREMENTS?"
SYSTEM REQUIREMENTS VALIDATION
"DO WE HAVE THE RIGHT SYSTEM REQUIREMENTS?"
STAKEHOLDER REAL-WORLD EXPECTATIONS
STAKEHOLDER NEEDS AND REQUIREMENTS
SYSTEM REQUIREMENTS
SYSTEM ARCHITECTURE AND DESIGN
REALIZED SYSTEM OF INTEREST
OPERATIONAL SYSTEM
MAINTAINED SYSTEM
DISPOSED SYSTEM
TRANSFORMATION
BUSINESS OR MISSION ANALYSIS, STAKEHOLDER NEEDS AND REQUIREMENTS DEFINITION PROCESSES
TRANSFORMATION
SYSTEM REQUIREMENTS DEFINITION PROCESS
TRANSFORMATION
SYSTEM ARCHITECTURE DEFINITION AND DESIGN DEFINITION PROCESSES
TRANSFORMATION
IMPLEMENTATION AND INTEGRATION PROCESSES
TRANSFORMATION
TRANSITION, OPERATION, MAINTENANCE, AND DISPOSAL PROCESSES
SYSTEM REQUIREMENTS VERIFICATION
"ARE THE SYSTEM REQUIREMENTS RIGHT?"
"ARE THE SYSTEM REQUIREMENTS DEFINED CORRECTLY?"
ARCHITECTURE AND DESIGN VERIFICATION
"ARE THE ARCHITECTURE AND THE DESIGN RIGHT?"
"ARE THE ARCHITECTURE AND THE DESIGN DEFINED CORRECTLY?"
IMPLEMENTATION VERIFICATION
"IS THE IMPLEMENTATION RIGHT?"
"IS THE IMPLEMENTATION DEFINED CORRECTLY?"
SYSTEM VERIFICATION
"IS THE SYSTEM RIGHT?"
"IS THE SYSTEM DEFINED CORRECTLY?"
POST DEVELOPMENT VERIFICATION
"IS THE SYSTEM STILL RIGHT?"
STAKEHOLDER NEEDS AND REQUIREMENTS VERIFICATION
"ARE THE STAKEHOLDER NEEDS AND REQUIREMENTS DEFINED CORRECTLY?"
CONFIRMATION
VERIFICATION PROCESS
Concurrency
Iteration
Recursion

FIGURE 2.38 Technical Processes in context. INCOSE SEH original figure created by Roedler, Walden, and Wheatcraft derived from INCOSE NRM (2022). Usage per the INCOSE Notices page. All other rights reserved.

V&V of Requirements

- Verification of Requirement
  - Involves verifying that the statements and sets of statements have the characteristics of well-formed requirements statements resulting from following requirements writing rules

- Validation of Requirement
  - Determines whether the requirement statement clearly communicates the intent of the need or parent requirement from which it was derived or transformed

V&V of the Entity

- Verification of the Entity
  - Results in collected evidence that can be used to prove that the entity is being/has been “formed” in the right way as defined by the set of system requirements with the required level of confidence

- Validation of the Entity
  - Results in collected evidence to be used for the acceptance, certification, and qualification of the SOI that proves the right “entity” is being/has been “formed” as defined by the set of stakeholder requirement with the required level of confidence

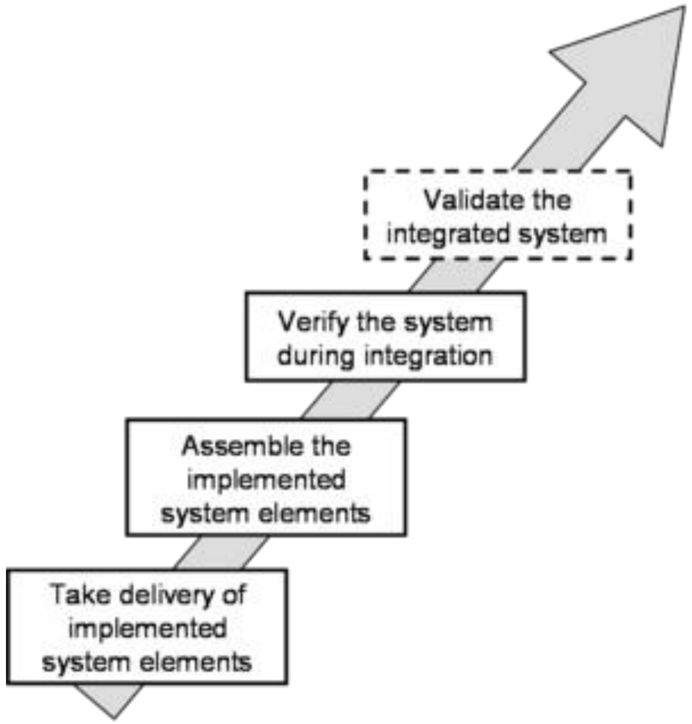
System Integration

System Integration

- 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)

https://www.sebokwiki.org/wiki/System_Integration

Boundary of Integration Activities


Validate the integrated system
Verify the system during integration
Assemble the implemented system elements
Take delivery of implemented system elements

SEBOK wiki

Integration Process

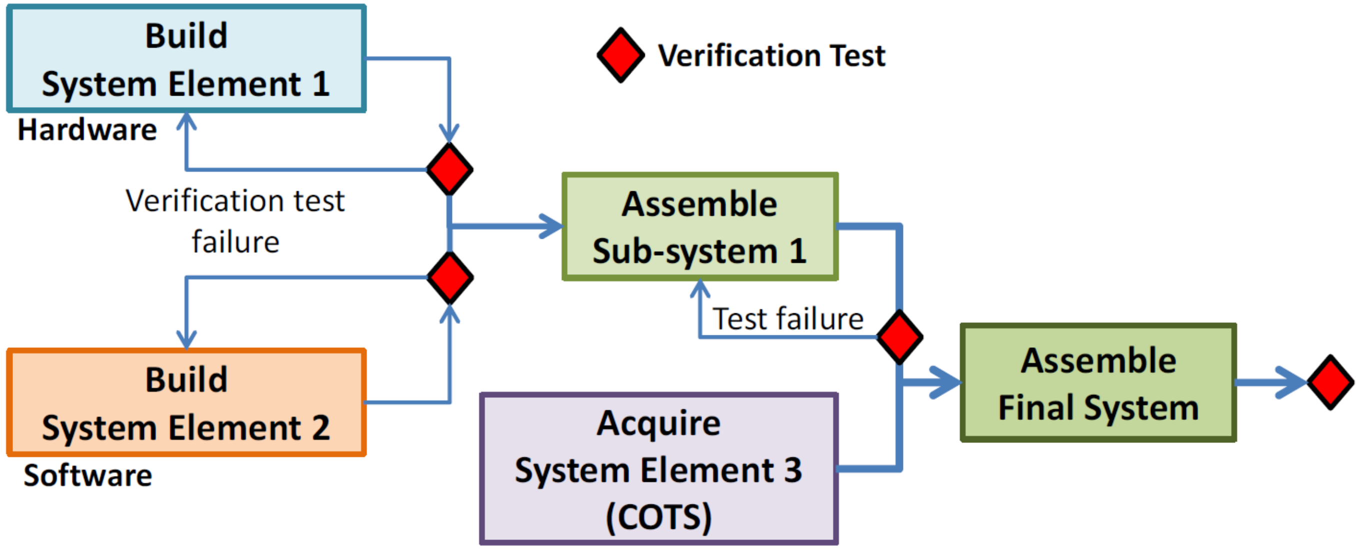


Figure 11: Notional Integration Process

Source: FAA SE Manual

Integration Strategies

Integration Technique	Description
Global Integration	•**Also known as *big-bang integration*; all the delivered implemented elements are assembled in only one step.**This technique is simple and does not require simulating the implemented elements not being available at that time. •Difficult to detect and localize faults; interface faults are detected late. •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.** •Allows starting the integration quickly**.** •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. •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**. •Fast localization of faults: a new fault is usually localized in lately integrated implemented elements or dependent of a faulty interface. •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. •Applicable to any type of architecture.

Integration Strategies

Integration Technique	Description
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".** •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. •Subsets shall be defined during the design. •Applicable to architectures composed of sub-systems.
Top-Down Integration	**•Implemented elements or aggregates are integrated in their activation or utilization order.** •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. •Many stubs/caps need to be created; difficult to define test cases of the leaf-implemented elements (lowest level). •Mainly used in software domain. Start from the implemented element of higher level; implemented elements of lower level are added until leaf-implemented elements.

Integration Strategies

Integration Technique	Description
Bottom-Up Integration	**•Implemented elements or aggregates are integrated in the opposite order of their activation or utilization.** •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. •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. •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.Allows early and intensive testing of critical implemented elements; early verification of design choices. •Test cases and test data sets are difficult to define.

How do these strategies inform/influence activities on Left Side of Vee?

- Global integration (big-bang integration)
- Integration with the stream (as elements become available)
- Incremental integration (in predefined order)
- Subset integration (functional chains integration)
- Top-down integration
- Bottom-up integration
- Criterion-driven integration (based on criticality – usually, risk)

Source: SE Handbook

Exercise 4.1

For your BSD,

1. Choose an Integration strategy for your system of Interest
2. Identify some requirements that exist because of integration strategy
3. Classify the levels as System, HW or SW.

Verification and Validation

What’s what?

Verification

- Proving the **system requirements** have been met
- Did we build the system right? (objective)
- Designers and developers perspective & participation
- Glass Box / White Box approach
- Usually initiated/completed first

Validation

- Determining to what extent the **stakeholder requirements** have been met
- Did we build the right system? (subjective)
- Customer or end-user perspective & participation
- Black Box approach
- Usually initiated/completed next

Verification Methods

- Inspection *(Examination against reference)*
- Analysis *(Theoretical proof)*
- Demonstration *(Show functionality)*
- Test *(Controlled environment; special equipment)*
- Analogy / Similarity *(Equivalence)*
- Simulation *(Using “models”)*
- Sampling *(Representation)*

Verification Methods are usually the same as **Validation Methods**, but with a different purpose

How do these methods inform/influence activities on Left Side of Vee?

Systems Engineering V

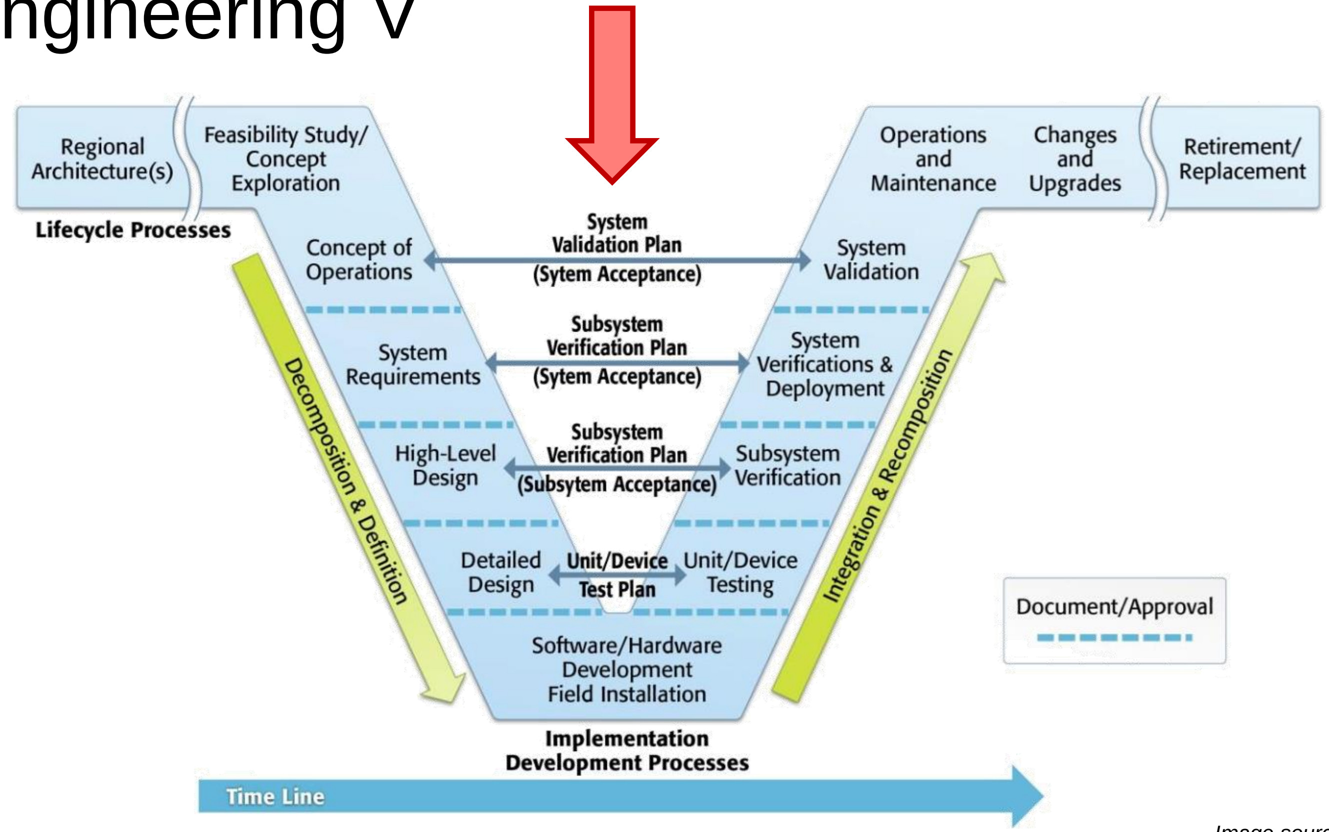

Regional Architecture(s)
Feasibility Study/ Concept Exploration
Lifecycle Processes
Concept of Operations
System Validation Plan (Sytem Acceptance)
System Validation
System Requirements
Subsystem Verification Plan (Sytem Acceptance)
System Verifications & Deployment
High-Level Design
Subsystem Verification Plan (Subsytem Acceptance)
Subsystem Verification
Detailed Design
Unit/Device Test Plan
Unit/Device Testing
Software/Hardware Development Field Installation
Implementation Development Processes
Decomposition & Definition
Integration & Recomposition
Operations and Maintenance
Changes and Upgrades
Retirement/ Replacement
Document/Approval
Time Line

Image source: USDOT

Requirement types and Verification methods

Requirement Types	Requirement type definition	Typical Verification method
Function	➔ What must the system do?	**<u>Demonstration</u>** Use of system, subsystem, or component operation to show that a requirement can be achieved
Performance	➔ How well must it be done?	**<u>Test</u>** Use of system, subsystem, or component operation to obtain detailed data to verify performance or to provide sufficient information to verify performance though further analysis
Design Constraint	➔ What design characteristics must be followed or achieved? ➔ Typically set by a higher authority for business reasons	**<u>Inspection</u>** Visual examination
Quality Attributes	➔ How will the users determine the quality of the system, given the other requirements? ➔ Usability, maintainability, etc. ➔ Specification must include agreement on how it will be measured	**<u>Analysis</u>** Use of mathematical modeling and analytical techniques to predict the compliance of a design to its requirements based on calculated data or data derived from lower level component or subsystem testing

Exercise 4.2

For your BSD,

1. Pick a few Software Requirements already identified, and discuss the following:
   a) Item to be Verified
   b) Verification method
   c) Facility / setup (if applicable)
   d) Conducting the Verification (steps to be followed)
   e) Data to be recorded; Analysis to be performed

End of Module