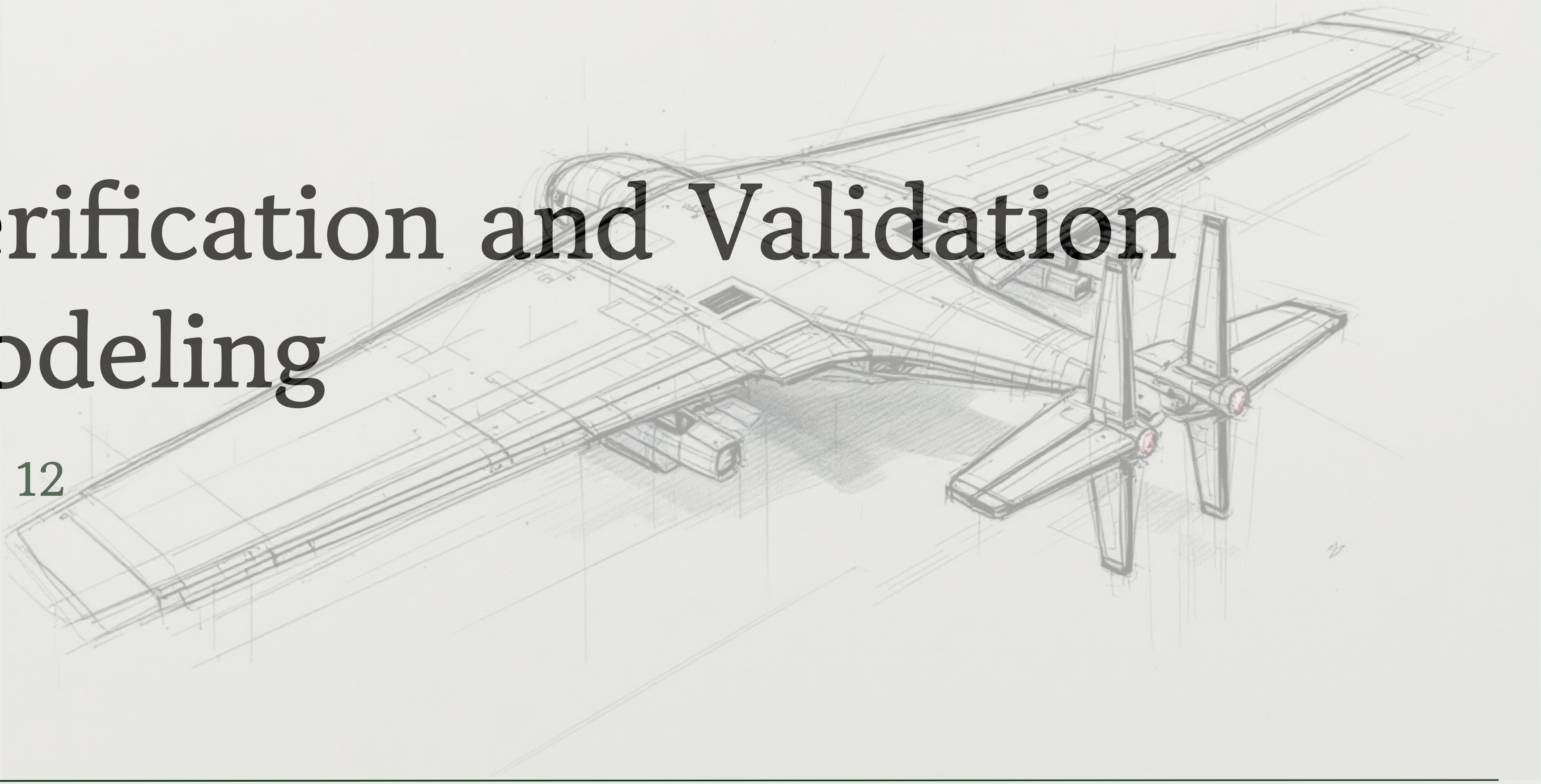


Verification and Validation Modeling

Week 12

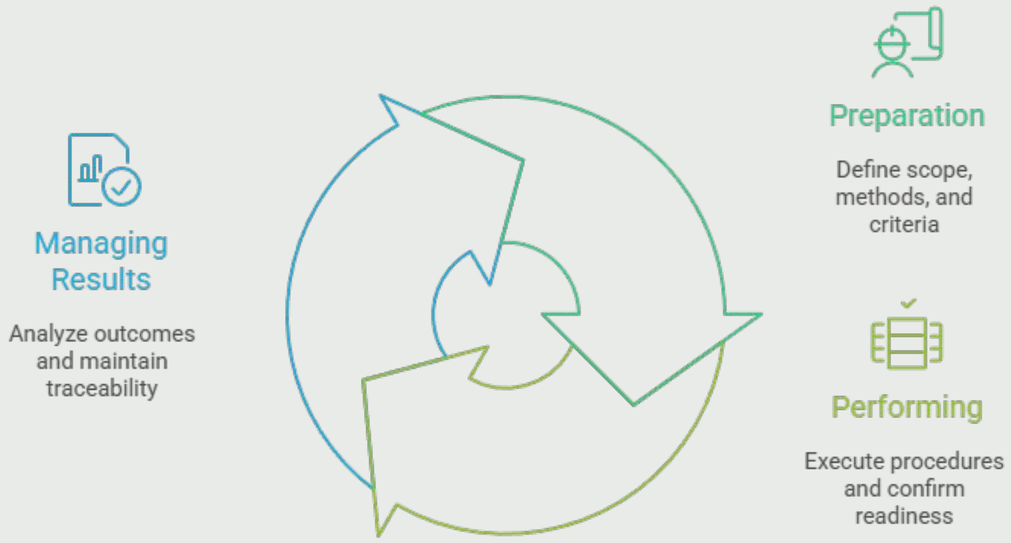
Verification Process

- The Verification Process, as defined in ISO/IEC/IEEE 15288, **provides objective evidence that a system, system element, or artifact fulfils its specified requirements and characteristics.**
- It applies not only to the realized system but also to requirements, models, simulations, architectures, and procedures that contribute to the definition and realization of the system.
- The process ensures that the **artifact has been built according to its requirements**, that no anomalies have been introduced during transformations, and that verification methods are capable of detecting potential errors.
- In SysML v2, this is represented using the verify relationship that links **Verification Cases** to Requirements. Verification results are supported through constraints and calculations that measure conformance, rather than by assertion-based semantics.

“Build the System Right”

Verification Process Activities

The Verification Process consists of preparing for verification, performing verification, and managing the results.


Preparation
Define scope, methods, and criteria
Performing
Execute procedures and confirm readiness
Managing Results
Analyze outcomes and maintain traceability

These activities must be planned early to avoid costly late-stage corrections and to ensure that all requirements can be objectively verified.

Elaboration of the Verification Process

- Verification planning should begin as soon as system requirements are being defined, with success criteria, methods, and strategies established and approved by the acquirer and authority.
- Early planning supports **realistic cost and schedule estimates** and increases the likelihood of full resourcing.
- If verification activities must be reduced, this should be guided by a risk-based approach rather than arbitrary cuts, since gaps discovered late in the life cycle are more costly to resolve.
- **A verification action is defined by the entity being verified**, the reference item against which it is compared, the expected result or success criteria, and the chosen strategy and method.
- The execution of a verification action produces results that are **compared** against the expected criteria to determine whether conformance has been achieved with acceptable confidence.

Verification Actions (Requirements and Models)

Verification begins with requirements and models.

- **Stakeholder requirements are verified** to ensure that they are correctly transformed from stakeholder needs and that they satisfy the characteristics of good requirements.
- **System requirements are verified in the same way**, confirming correct transformation from parent requirements and compliance with requirement quality rules.
- **Models and simulations are verified to confirm they meet their intended purpose**, follow syntactic and grammatical standards, and correctly apply modeling methods and heuristics as defined by organizational guidelines.

Verification Actions (Architecture, Design, and Realized System)

- **System architecture verification ensures that the architecture, when realized by design, will result in a system capable of passing system verification**. It also checks correct use of patterns, heuristics, and architecture definition methods.
- **System design verification confirms that design outputs and characteristics meet system requirements** and that accepted trade rules and practices are correctly applied.
- Finally, **verification of the realized system or system elements provides objective evidence** that the product or service conforms to requirements and design characteristics with an acceptable degree of confidence.

Verification Actions

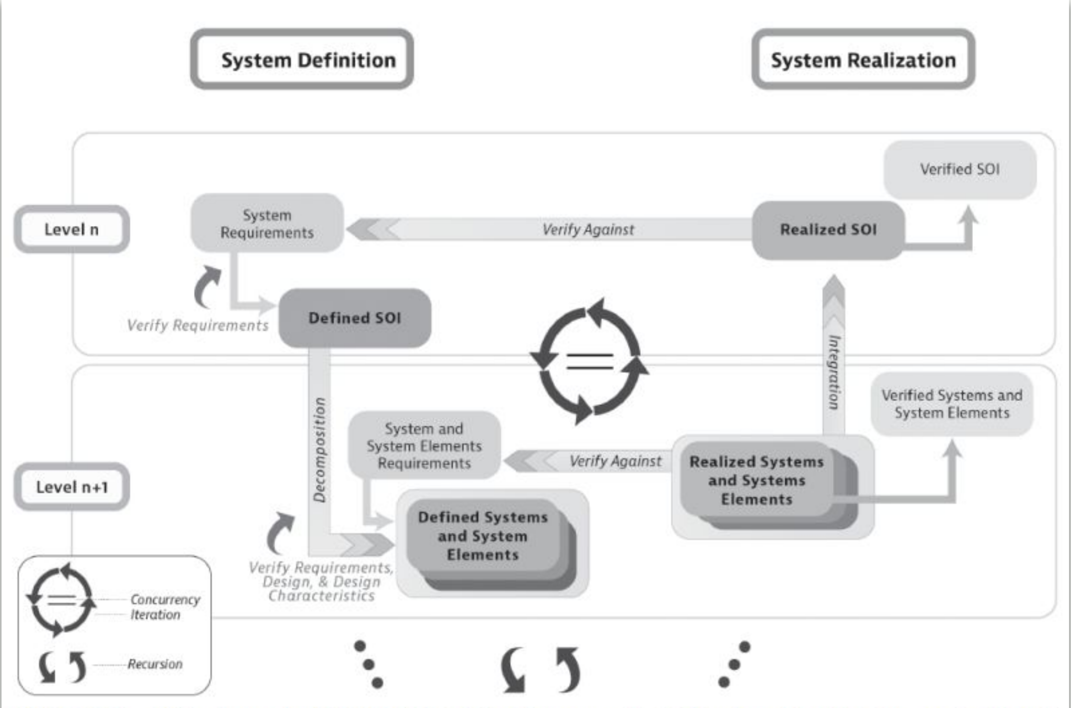
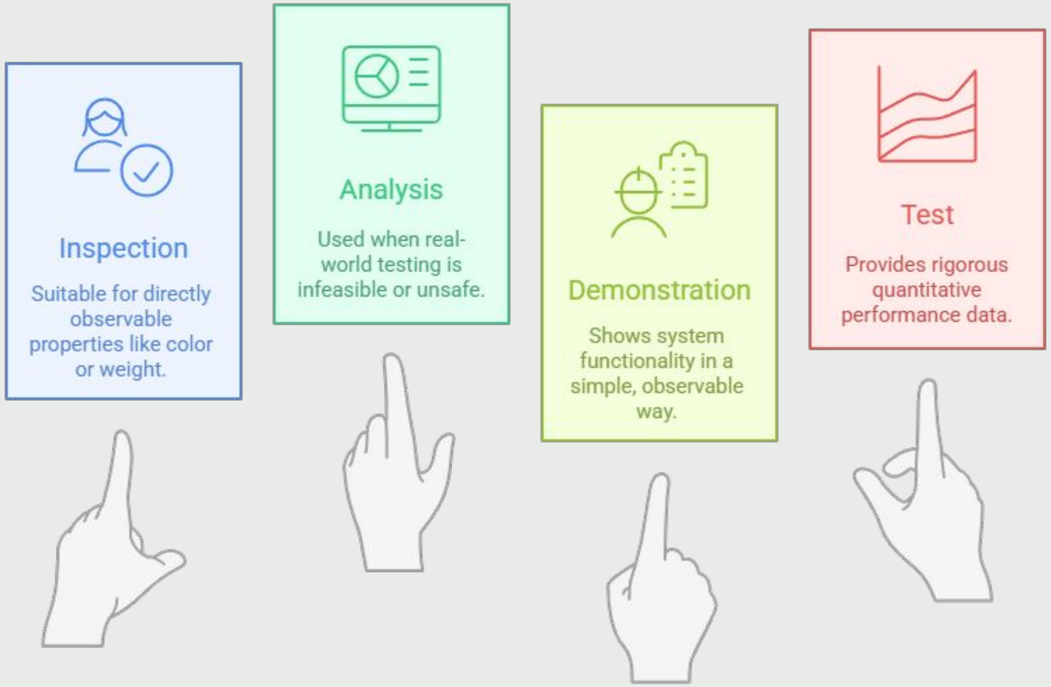


FIGURE 2.50 Verification per level. INCOSE SEH original figure created by Walden from Faisandier. Usage per the INCOSE Notices page. All other rights reserved.

Verification Methods

Verification relies on four fundamental methods.



Inspection
Suitable for directly observable properties like color or weight.
Analysis
Used when real-world testing is infeasible or unsafe.
Demonstration
Shows system functionality in a simple, observable way.
Test
Provides rigorous quantitative performance data.

Verification Methods at a Glance

METHOD	PURPOSE	EVIDENCE TYPE	TYPICAL USE
Inspection	Confirm compliance by direct observation	Qualitative (visual, sensory, simple measurement)	Color, weight, labeling, physical fit, workmanship, simple attributes
Analysis	Demonstrate theoretical compliance	Analytical or simulated data	Performance models, simulations, similarity to verified systems
Demonstration	Show functional performance qualitatively	Observable response to stimuli	Operator usability, simple functional checks, qualitative system behavior
Test	Measure performance under controlled conditions	Quantitative data (instrumented)	Performance, capacity, reliability, environmental tests

Verification Activities to SysML v2

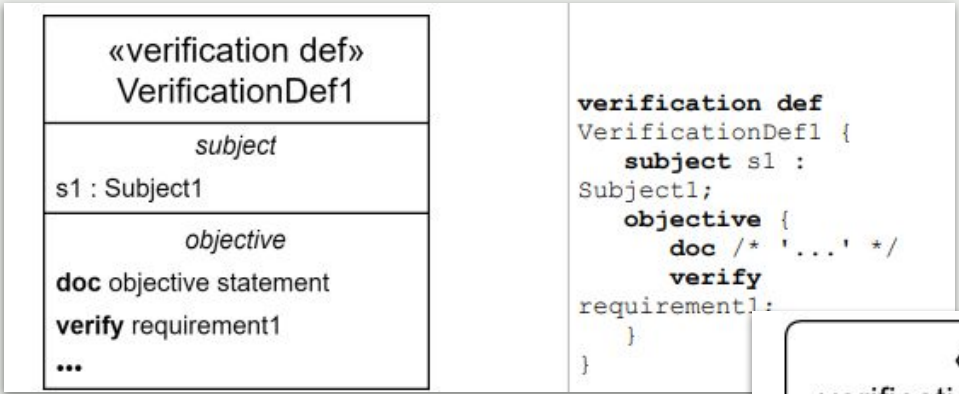
Verification preparation, performance, and results management can be expressed in SysML v2 through structured model elements.

- Preparation is captured by defining Verification Cases and linking them to Requirements with the verify relationship.
- Constraints and calculations can be added to express success criteria and measurable conditions.
- Performing verification is modeled by associating Verification Cases with Actions or Scenarios that define how the verification will be executed.
- Results are represented through captured values and metadata, linked back to the requirements, preserving traceability across the system model.

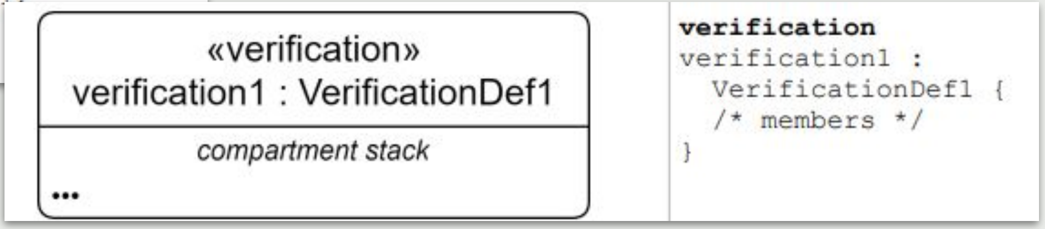
Verification Cases in SysML v2

- A verification def defines a formal **test or validation procedure** as part of the model.
- A verification is a usage of that procedure — applied to specific elements of the system.

Use verifications to prove that **requirements are met** under defined conditions.



```
verification def
VerificationDef1 {
   subject s1 :
Subject1;
   objective {
      doc /* '...' */
      verify
requirement1;
   }
}
```



```
verification
verification1 :
  VerificationDef1 {
  /* members */
}
```

Image capture from OMG Systems Modeling Language™ Standard Specification Beta 3, page 142

Verification Methods to SysML v2

The four basic verification methods can be represented in SysML v2 using specific modeling patterns.

Inspection can be expressed through Verification Cases that reference measured attributes or qualitative constraints.

Analysis, including modeling and simulation, can be described by associating Verification Cases with Calculation Usages or Analysis Cases.

Demonstration can be modeled as Scenarios that link system stimuli to expected observable responses.

Test can be represented as Verification Cases with detailed procedures, parameter constraints, and quantitative measurements linked to system elements.

Verification Across System Levels

Verification is conducted recursively through each level of the system hierarchy.

- Stakeholder requirements are verified against higher-level needs, while system and system element requirements are verified against their parent requirements.
- Architecture and design are verified to confirm that they satisfy the requirements of their corresponding level.
- Each realized system element must pass its verification before integration into the next higher-level system.
- Issues discovered must be resolved before integration proceeds. This cycle of verification and integration continues up the hierarchy until the complete System of Interest passes system verification.

Validation Process

- The Validation Process provides **objective evidence that a system fulfills its business or mission** objectives, stakeholder needs, and intended use in its operational environment.
- Validation can be applied to requirements, models, simulations, architectures, designs, procedures, or realized systems.
- The process confirms that the “right” artifact has been produced in alignment **with stakeholder expectations**, that the resulting SoI will perform correctly in its intended context, and that it cannot be misused by unintended users in ways that compromise its purpose.
- Validation, therefore, ensures that the right system has been built, whereas verification ensures that the system has been built right.

“Build the Right System”

Validation Process Activities (Preparation and Performance)

- The Validation Process begins with **preparation**, defining the scope of what will be validated, the actions to be performed, and the success criteria. Validation planning identifies the artifacts, entities, or information items to be validated, and maps them against stakeholder needs and requirements. Constraints such as contractual, regulatory, or physical limitations must be considered.
- For each validation action, methods such as inspection, analysis, demonstration, or test are selected, and success criteria are established. Validation procedures are then defined, scheduled, and executed, ideally in an operational environment or close to it, with the participation of intended users or qualified surrogates.

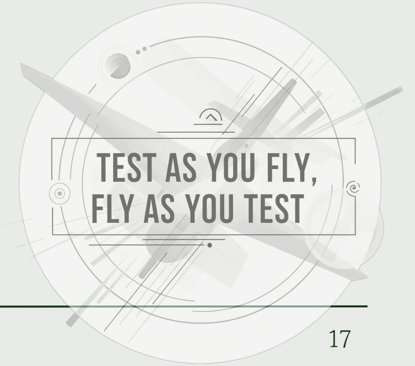
Validation Process Activities (Managing Results)

- Validation results must be recorded, analyzed against success criteria, and approved by the validation authority. Defects or problems identified are tracked to resolution through project assessment and configuration management processes.
- Validation ensures bidirectional traceability between validated entities, their requirements, and the architecture or design.
- Results are combined into a validation package for approval. At higher levels, validation may involve acquirers and end users, while at lower levels it may be performed internally.

Early identification of enablers, broad s**takeholder participation**, and inclusion of system interactions with enabling and interfacing systems are vital. Validation should provide insight as early as possible, including through analysis, modeling, and simulation of operational scenarios.

Validation – General Considerations

- Stakeholder needs and requirements validated against the SoI are **derived from the mission, goals, objectives, constraints, risks, and life cycle concepts.**
- These life cycle concepts include scenarios and use cases across production, operation, support, and retirement, all exercised in the intended operational environment by intended users.
- Validation must consider not only nominal operations but also alternate, off-nominal, misuse, and loss scenarios.
- A positive validation result in one environment or user group may not hold if either changes, requiring adaptation of needs and requirements through the acquirer and developer processes.
- <u>Involving intended users and operators directly during validation is essential, ensuring acceptance tests reflect real operational conditions.</u>



Validation Planning and Risk Considerations

- Validation planning should start early, as stakeholder needs and requirements are defined.
- Success criteria, methods, and strategies must be established with approval from acquirers and authorities to ensure resources are allocated.
- Early planning improves cost and schedule estimates and maximizes the chance of full plan execution. If validation activities must be reduced, this should be done through a risk-based approach, never by arbitrarily cutting the number or costliest actions.
- Gaps left unresolved can become costly at final acceptance. If additional resources are later available, they should be used to expand validation to lower-risk needs, increasing confidence and reducing project risk.

Validation Actions (Requirements and Models)

Validation actions confirm that the “right” requirements and artifacts have been defined.

- **Stakeholder requirements** are validated to ensure they accurately reflect stakeholder needs, are written in stakeholder language, and are actionable.
- **System requirements** are validated to ensure they express stakeholder intent in technical terms and can be transformed into architecture and design.
- **Models and simulations** are validated to confirm that they reflect intended behavior in the operational environment and that they meet the purpose for which they were developed.

Validation Actions (Architecture, Design, and Realized System)

- **System architecture** is validated to ensure that it is the right architecture, capable of guiding a design that will meet stakeholder needs and requirements.
- **The system design** is validated to confirm that its design characteristics will result in a SoI that achieves its intended purpose when operated in its real environment by intended users.
- Finally, **the realized system** is validated to ensure that it performs its mission as intended, does not enable misuse by unintended users, and provides confidence that the SoI fulfills stakeholder needs and requirements in its operational environment.

Validation Actions

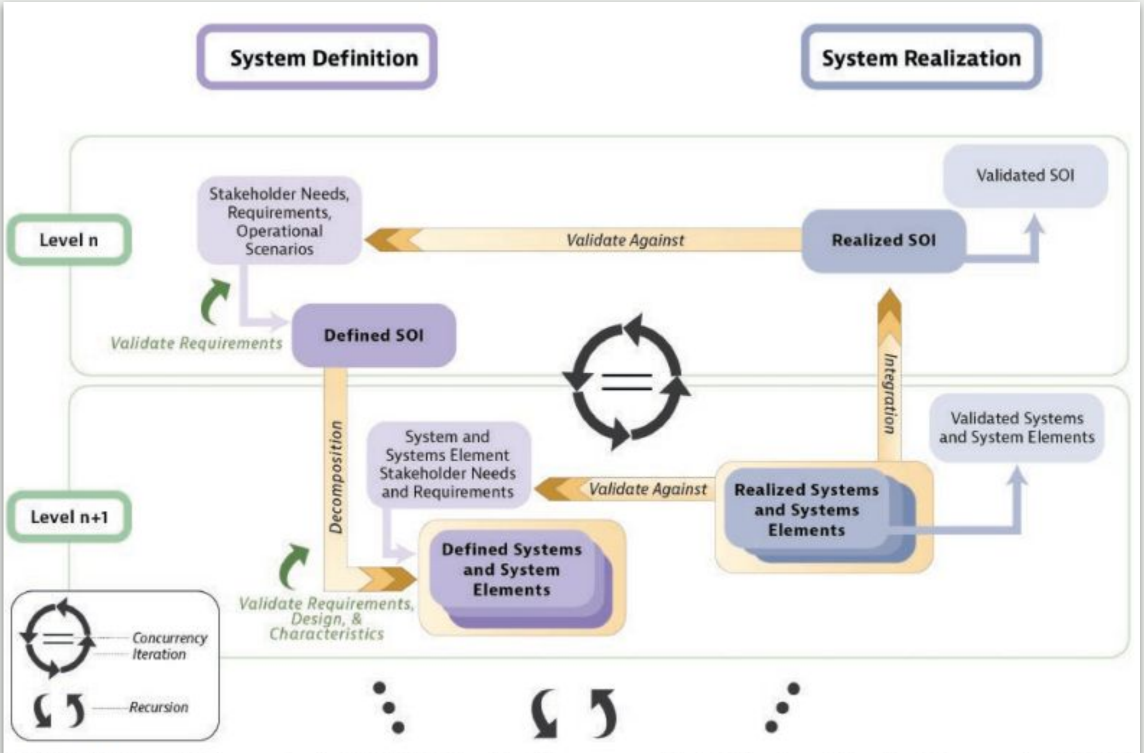
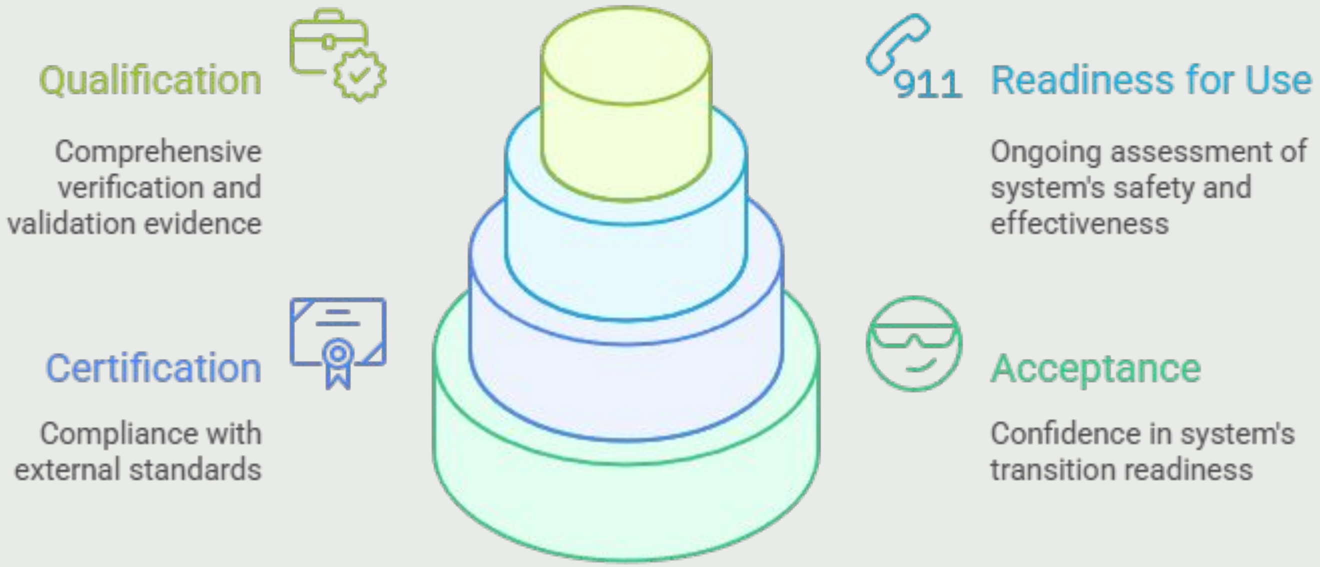

System Definition
System Realization
Level n
Stakeholder Needs, Requirements, Operational Scenarios
Validate Against
Realized SOI
Validated SOI
Validate Requirements
Defined SOI
Integration
Decomposition
System and Systems Element Stakeholder Needs and Requirements
Validated Systems and System Elements
Validate Against
Realized Systems and Systems Elements
Level n+1
Defined Systems and System Elements
Validate Requirements, Design, & Characteristics
Concurrency
Iteration
Recursion

FIGURE 2.53 Validation per level. INCOSE SEH original figure created by Walden from Faisandier. Usage per the INCOSE Notices page. All other rights reserved.

Validation Outcomes

Validation leads to several key outcomes that support system acceptance and deployment.


Qualification
Comprehensive verification and validation evidence
911
Readiness for Use
Ongoing assessment of system's safety and effectiveness
Certification
Compliance with external standards
Acceptance
Confidence in system's transition readiness

Validation Outcomes

Outcome	Purpose	Typical Authority/Context
Acceptance	Confirms system is suitable for transition to acquirer. Often based on operational validation actions or supplier validation results.	Acquirer, project team, validation authority
Certification	Provides written assurance system complies with legal/industry standards and can perform intended functions.	Independent regulatory or certification authority (e.g., FAA, CE, UL)
Readiness for Use	Determines if the system is safe and effective to enter or reenter service. Occurs at delivery, production completion, after maintenance, or field trials.	Project team, validation authority, sometimes end users
Qualification	Confirms all verification and validation activities are completed, documented, and system (including interfaces) meets requirements with margins.	Supplier organization with acquirer oversight; concluded by acceptance/operational readiness review

Validation to SysML v2 (Activities → Model)

- Validation is represented by modeling stakeholder **Needs/Concerns**, **Use Cases/operational scenarios**, and **Requirements** that are traced to those needs.
- Plan validation by defining **Cases/Analysis Cases** that reference the intended users, operational environment, and success criteria derived from needs.
- Perform validation by binding scenarios to the SoI configuration and recording measured outcomes as values linked to the Case.
- Manage results by keeping bidirectional traceability among
  Needs/Concerns ⇄ Requirements ⇄ Cases ⇄ Evidence,
  and capturing approvals as metadata.

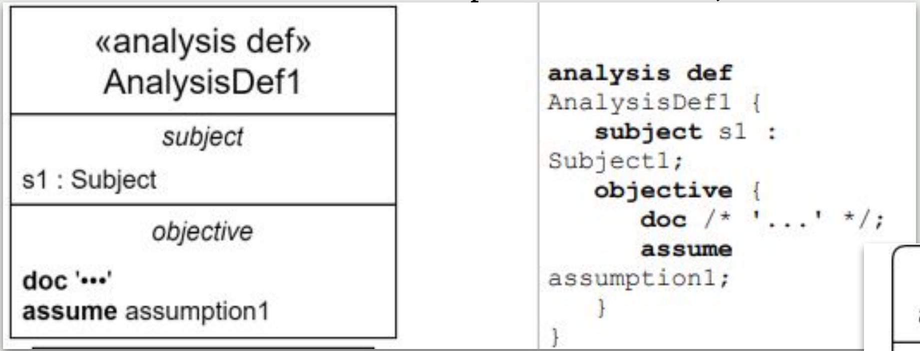
Validation Methods to SysML v2

- **Inspection** is a Case that references observable attributes (e.g., labels, configuration) on the SoI and logs observed values as evidence.
- **Analysis** uses **Calculation Usages/Analysis Cases** tied to scenarios and environment assumptions; similarity arguments are captured as referenced prior artifacts and rationale.
- **Demonstration** is a scenario-centric Case that defines user stimuli and expected observable responses for intended operators in an operational context.
- **Test** is a Case with explicit input conditions, procedure steps, parameter constraints, and quantitative measurements bound to the SoI and environment.

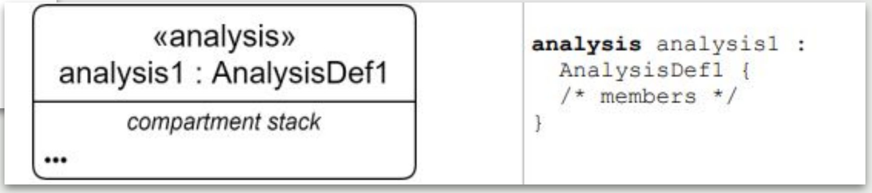
“fitness for purpose”

Analysis Cases in SysML v2

- An analysis case is a specialized case in SysML v2 that defines how an analysis will be performed on a subject.
- Analysis can be expressed as sets of actions with calculations, as specifications for external solvers, or as constraint equations to be solved.
- For example, a *Fuel Economy Analysis* of a drone could define the vehicle as the subject, and return an estimated distance per unit of fuel, evaluated against a fuel economy requirement.



```
analysis def
AnalysisDef1 {
   subject s1 :
Subject1;
   objective {
      doc /* '...' */;
      assume
assumption1;
   }
}
```



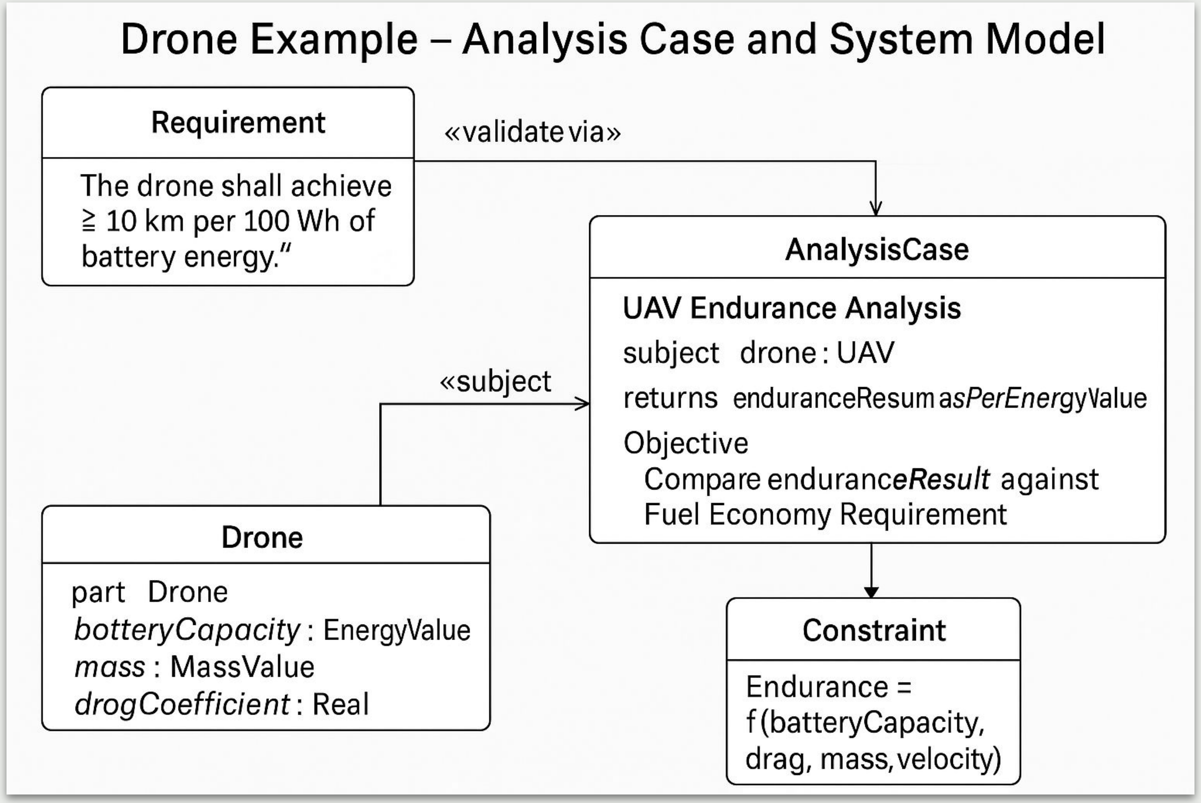
```
analysis analysis1 :
  AnalysisDef1 {
  /* members */
}
```

Image capture from OMG Systems Modeling Language™ Standard Specification Beta 3, page 139

Relationship of Analysis Cases and the System Model

In SysML v2, an **Analysis Case** is always connected to the system model through its **subject**, which is a system element or the whole System of Interest. The case **uses attributes, constraints, and behaviors from the model as inputs for calculations or simulations**. It produces measurable results that are compared to objectives or requirements. Analysis Cases therefore act as a bridge: they reuse the system model to perform analyses, return results as evidence, and maintain traceability between requirements, analysis activities, and the system elements under study.

Drone Example


Drone Example – Analysis Case and System Model
Requirement
The drone shall achieve ≧ 10 km per 100 Wh of battery energy.”
«validate via»
AnalysisCase
UAV Endurance Analysis
subject drone : UAV
returns enduranceResum *asPerEnergy*Value
Objective
Compare endurance***Result*** against Fuel Economy Requirement
«subject
Drone
part Drone
botteryCapacity : EnergyValue
mass : MassValue
drogCoefficient : Real
Constraint
Endurance = f(batteryCapacity, drag, mass,velocity)

Validation per Level and Early MBSE

Validation occurs recursively across each hierarchical level of a system’s architecture.

- Stakeholder needs and requirements are validated against real-world expectations.
- System requirements are validated to ensure they are the “right” requirements.
- Architecture and design are validated against stakeholder needs.
- Realized systems and system elements are validated in their intended operational environment before integration into higher levels.

Early validation using models and simulations reduces risk, cost, and time by identifying issues before physical implementation. However, final confidence comes only from validation of the realized system in the operational environment. Emergent properties must be assessed carefully, as they may not appear in models alone.

Verification vs Validation in SysML v2 – Drone Example

Verification (Build The System Right)
For the drone's payload capacity, a requirement states: *"The drone shall carry at least 5 kg payload."*
A **Verification Case** applies a test method to measure the maximum payload of Drone X. The result is compared against the requirement limit. If the measured payload ≥ 5 kg, the requirement is verified.

Validation (Build the Right System)
For the drone delivery mission, a stakeholder need states: *"Deliver a 5 kg package across a 10 km urban route."*
A **Analysis Case** demonstrates that Drone X can complete this use case in the intended operational environment, with expected outcome = package delivered.

Summary of Week 12

Verification Process

- **Purpose:** Provide objective evidence that artifacts meet specified requirements and characteristics.
- **Activities:** prepare, perform, and manage verification across requirements, models, architecture, design, and realized systems.
- **Methods:** Inspection, Analysis, Demonstration, and Test.
- **SysML v2:** represented by **Verification Cases** linked to requirements via the verify relationship, using constraints and calculations to show compliance.

Validation Process

- **Purpose:** Provide objective evidence that the system fulfills stakeholder needs, business/mission objectives, and intended use in its operational environment.
- **Activities:** prepare, perform, and manage validation with strong stakeholder and user involvement.
- **Actions:** validation of stakeholder/system requirements, models, architecture, design, and realized system.
- **Outcomes:** Acceptance, Certification, Readiness for Use, Qualification.
- **SysML v2:** represented by **Analysis Cases** and **Use Cases**, validating stakeholder needs and concerns via operational scenarios and outcomes.

Summary of Week 12

Key Distinction

- Verification = *“Build the System Right”* → requirement compliance.
- Validation = *“Build the Right System”* → stakeholder fitness-for-purpose.

Analysis Cases

- Special case type used for performance, trade, or simulation analysis.
- Tightly linked to the system model as the subject, returning measurable results compared to objectives/requirements.
- Example: Drone **Fuel Economy Analysis** binding to UAV attributes and constraints to evaluate endurance.

Drone Example

- Verification: test if drone carries ≥ 5 kg payload.
- Validation: demonstrate drone delivers a 5 kg package across 10 km urban route.
- Analysis Case: calculate UAV endurance based on battery capacity, drag, and mass, compared to fuel economy requirement.

QUESTION!