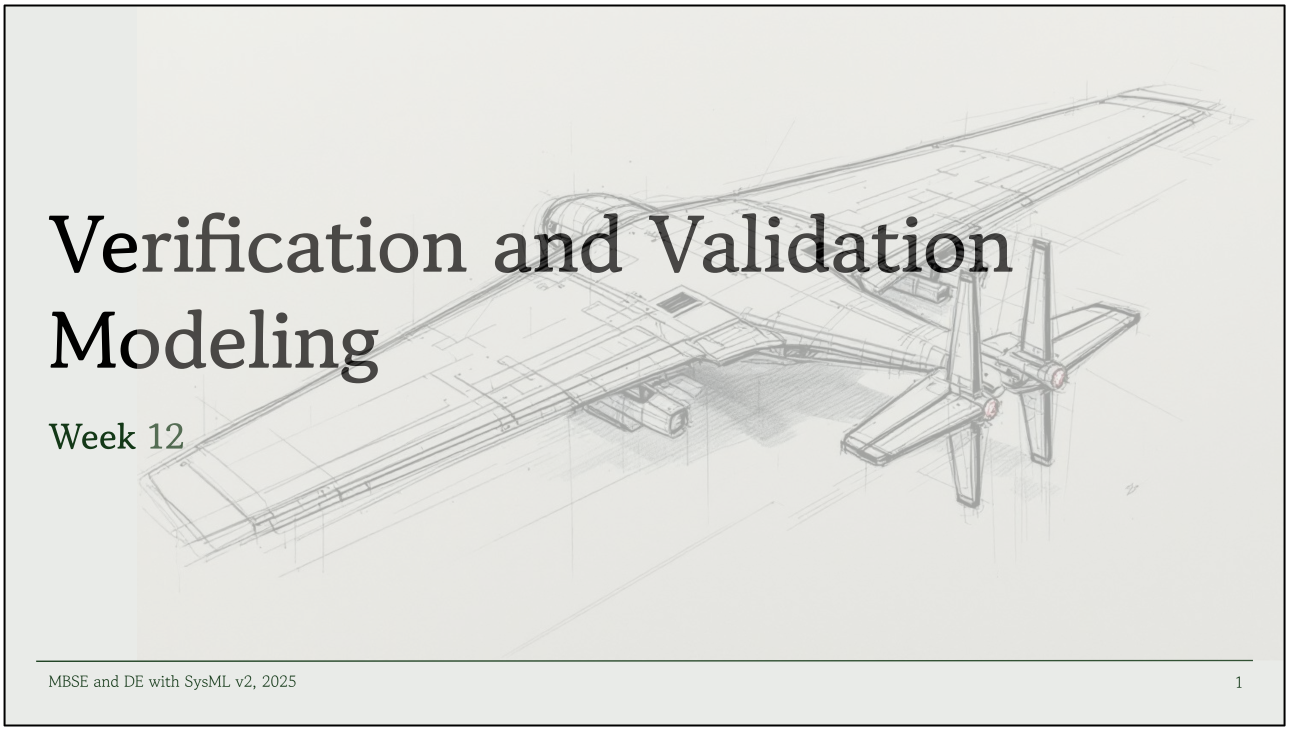
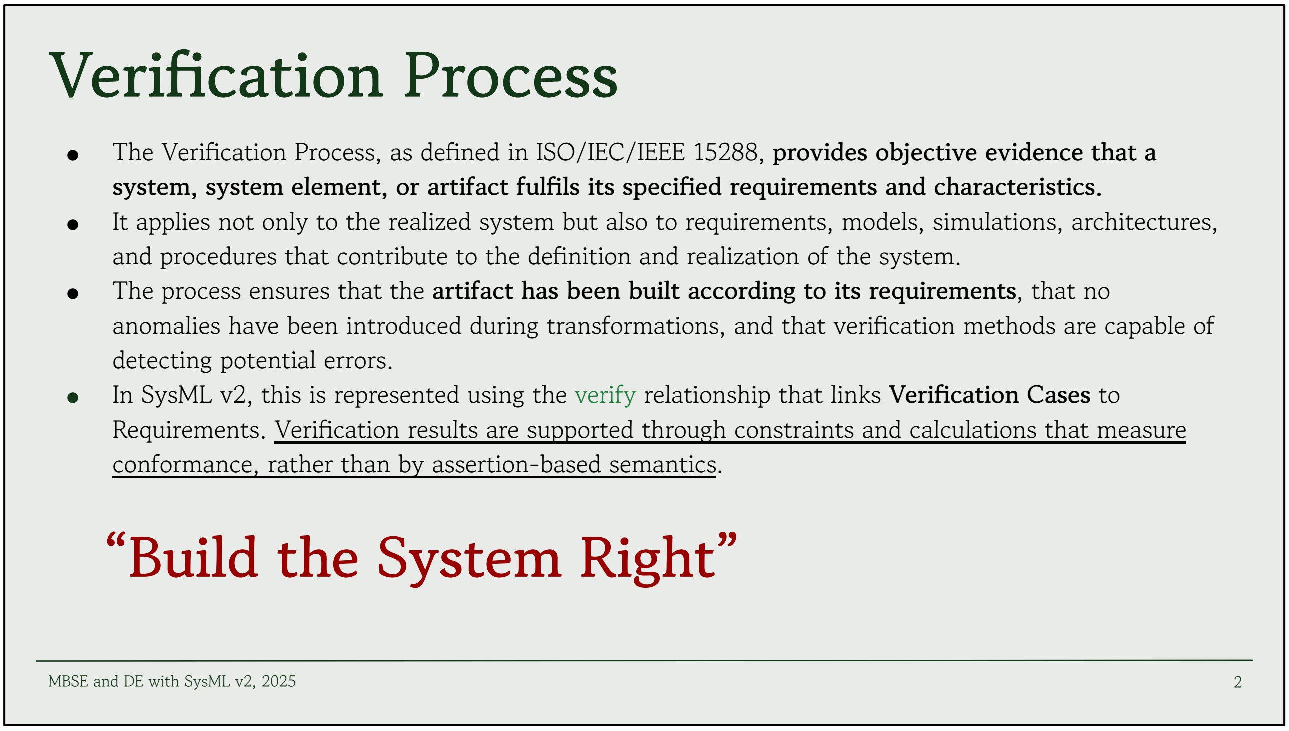


Verification and Validation Modeling

Week 12

Welcome to Week 12. This week is all about **Verification and Validation** — how we **prove** our system meets its requirements.
In traditional engineering, this involves test reports, spreadsheets, and disconnected documents. But in MBSE, we can represent **verification plans directly in the model** — making it traceable, scalable, and integrated.

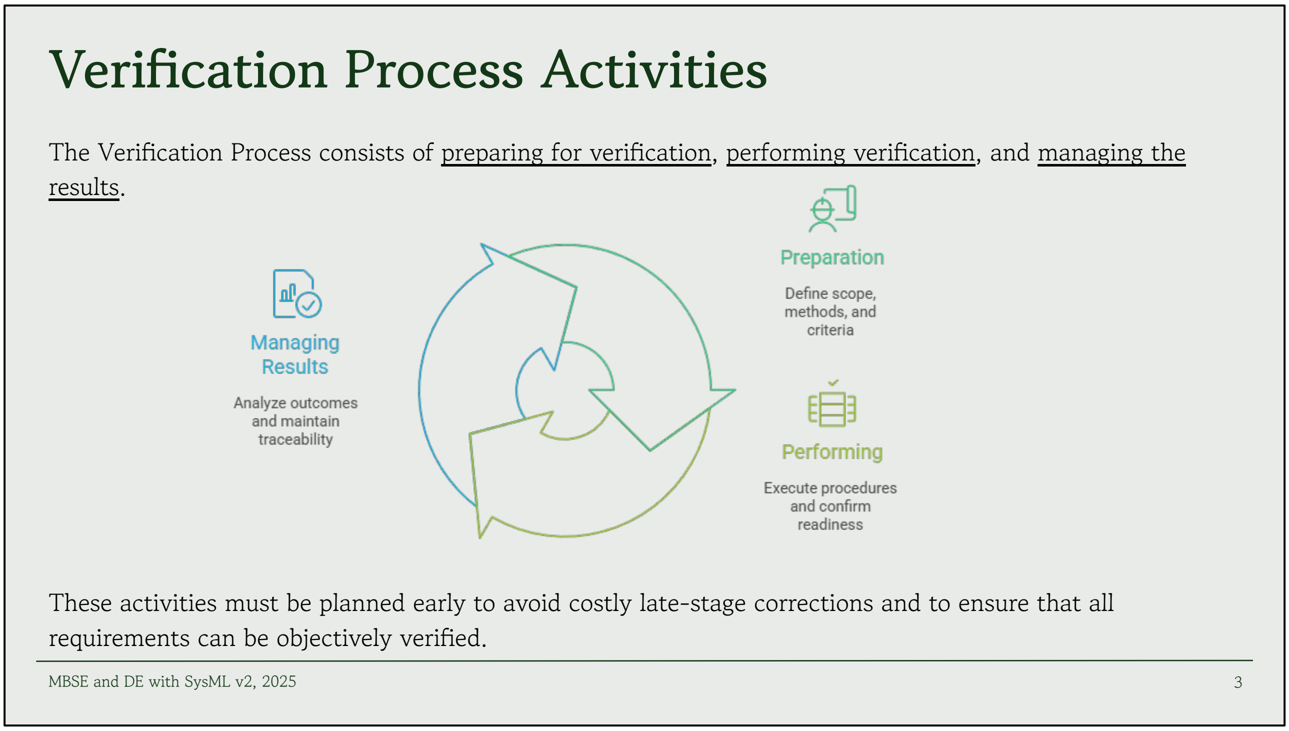


Verification Process

- 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.

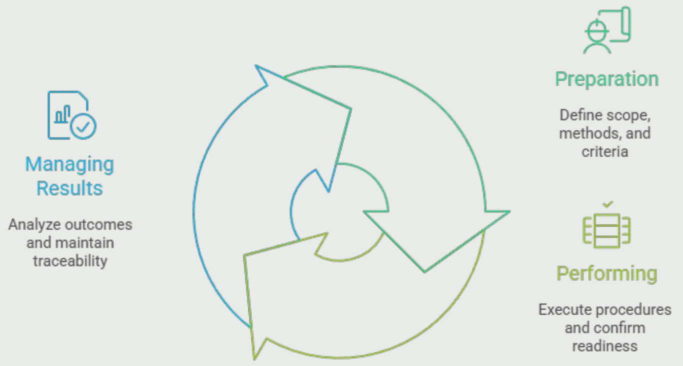
“Build the System Right”

The key idea of the Verification Process is that we confirm “the system was built right.” It is broader than final system testing—it also includes verification of requirements, models, and design elements. The process emphasizes the need for objective evidence and confidence that requirements have been satisfied and no errors were introduced. In SysML v2, this process is modeled through Verification Cases connected to Requirements by the verify relationship. Rather than using assertions, SysML v2 employs constraints and calculations to provide measurable results that demonstrate compliance.



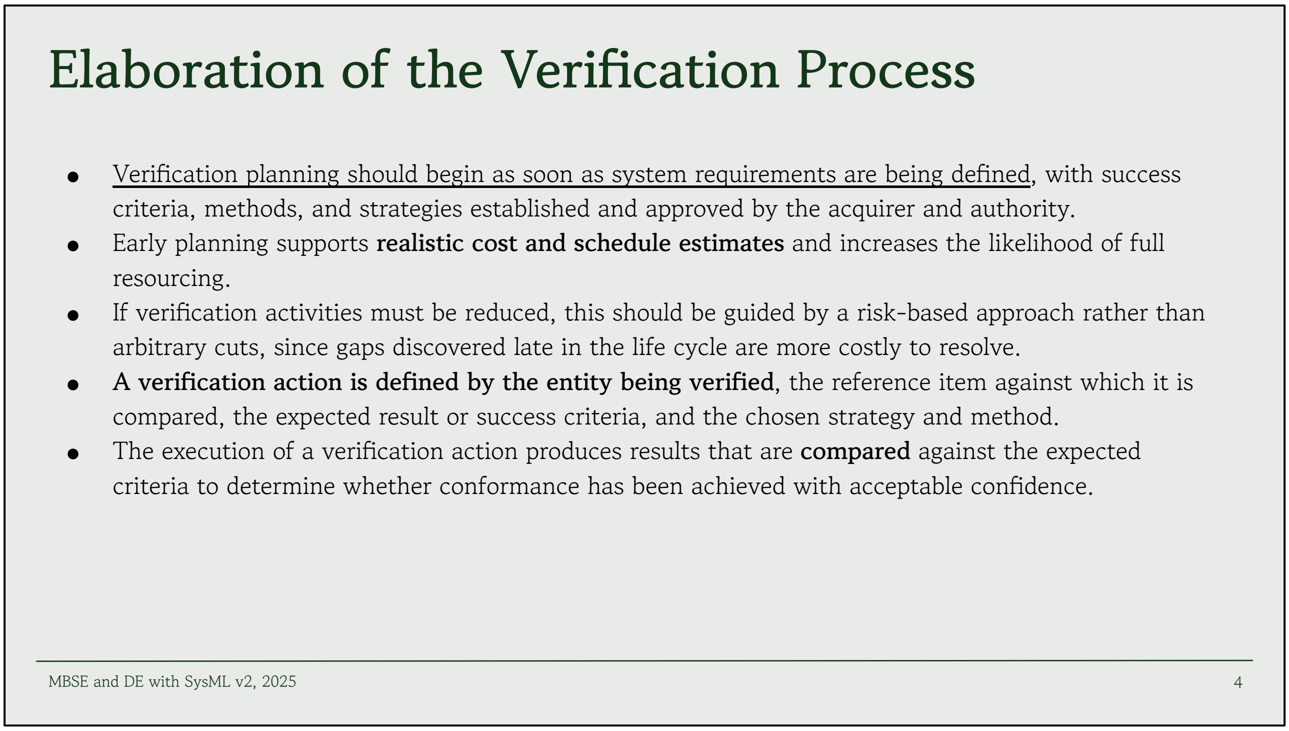
Verification Process Activities

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.

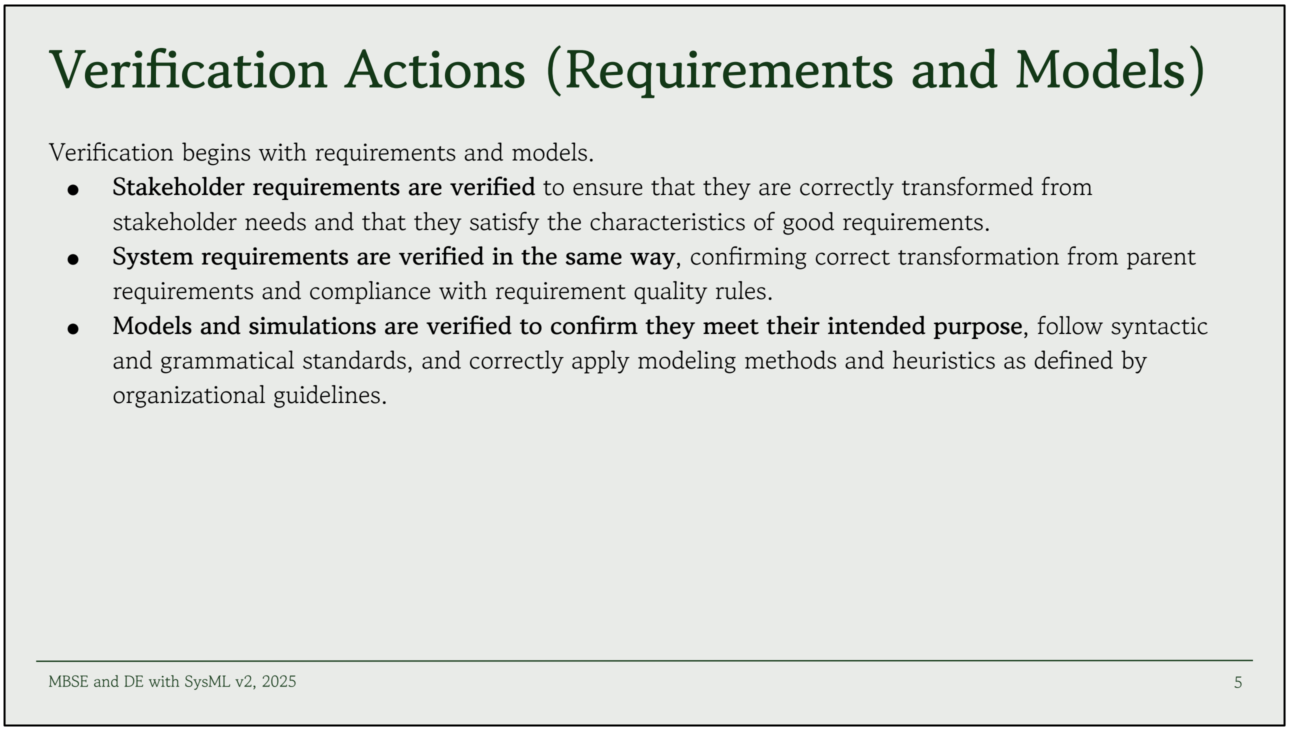
The Verification Process is operationalized in three stages.
First, preparation establishes the scope, entities, constraints, methods, and success criteria, ensuring resources and enablers are ready.
Second, the performance stage focuses on executing planned procedures in a controlled and scheduled manner, with readiness checks in place.
Third, the results stage emphasizes recording evidence, analyzing results against criteria, resolving problems, and gaining formal approval from verification authorities. A key best practice is to identify enabling systems and materials early and to integrate verification throughout the life cycle, not only at the end. This reduces the risk of late discoveries and ensures every requirement is verifiable.



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.

This elaboration highlights the “how-to” aspects of verification. Planning starts in parallel with requirement definition, where methods, criteria, and strategies should already be agreed with stakeholders and approval authorities. Reductions in verification activities must always be based on risk analysis, never on cost or schedule pressures alone, because unresolved verification gaps discovered at higher integration levels can delay and increase costs significantly. Each verification action is clearly defined: it identifies what entity is verified, what requirement or standard it is compared to, what success criteria define compliance, and which method and strategy will be applied. The obtained results are compared with the expected ones, providing a judgment of conformance with measurable confidence.

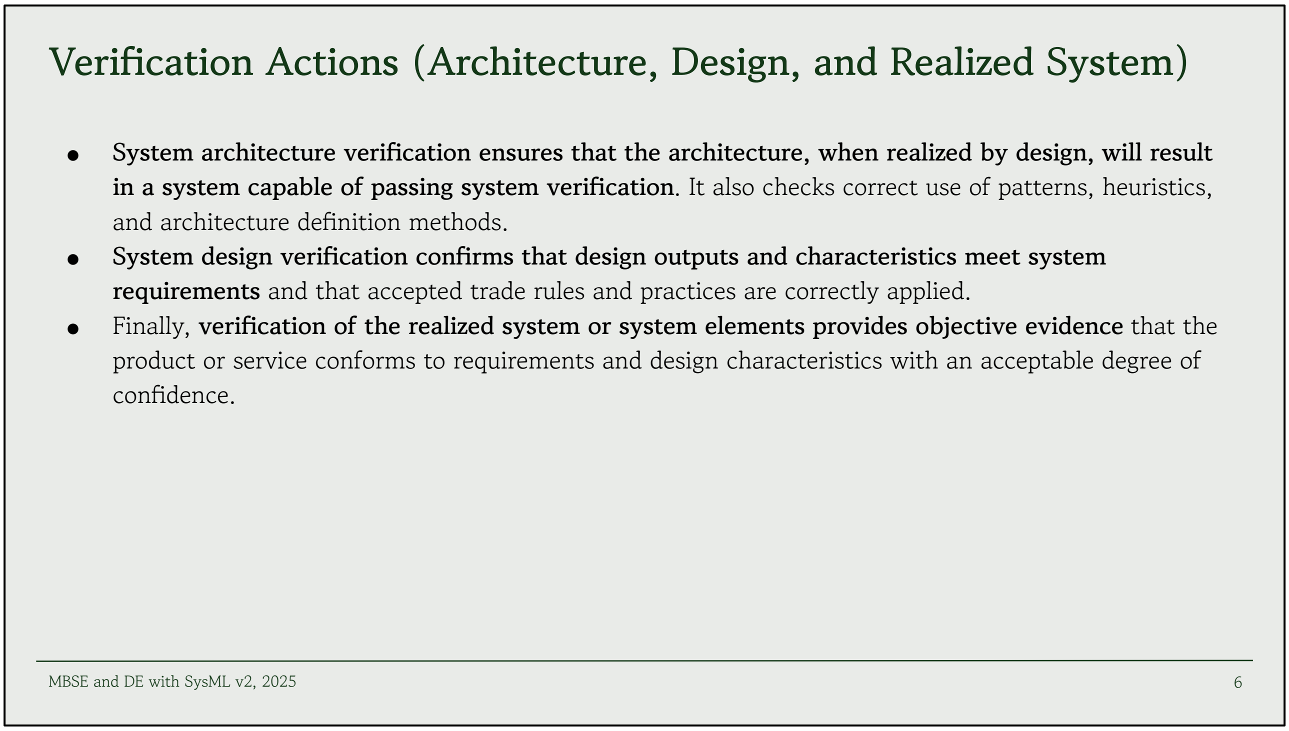


Verification Actions (Requirements and Models)

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.

This first group of verification actions emphasizes early artifacts. Requirements verification ensures that both stakeholder and system requirements are well-formed, properly derived, and testable. Model and simulation verification ensures correctness not only in purpose but also in form, adherence to standards, and use of appropriate modeling techniques. By applying verification early at this level, errors can be detected before they propagate into architecture and design.



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.

This second group of verification actions focuses on later stages of the life cycle. Architecture verification ensures design feasibility and proper use of architectural methods. Design verification links design characteristics directly to system requirements and checks against domain best practices. The final system or system element verification provides the confidence that the SoI has indeed been “built right,” fulfilling the requirements and design specifications established earlier.

Verification Actions

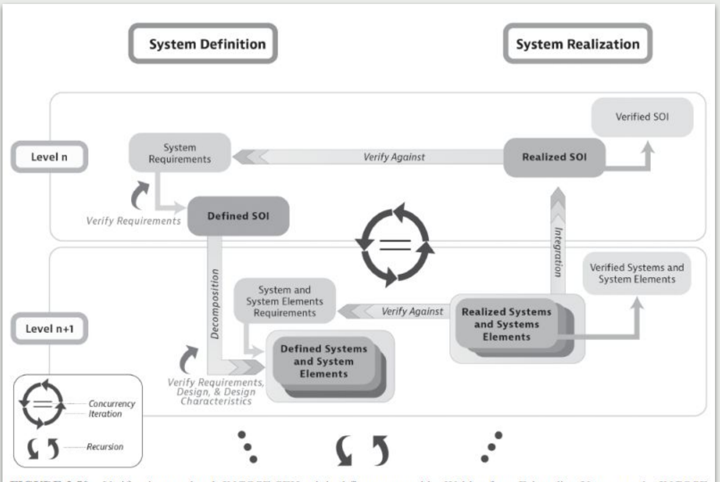
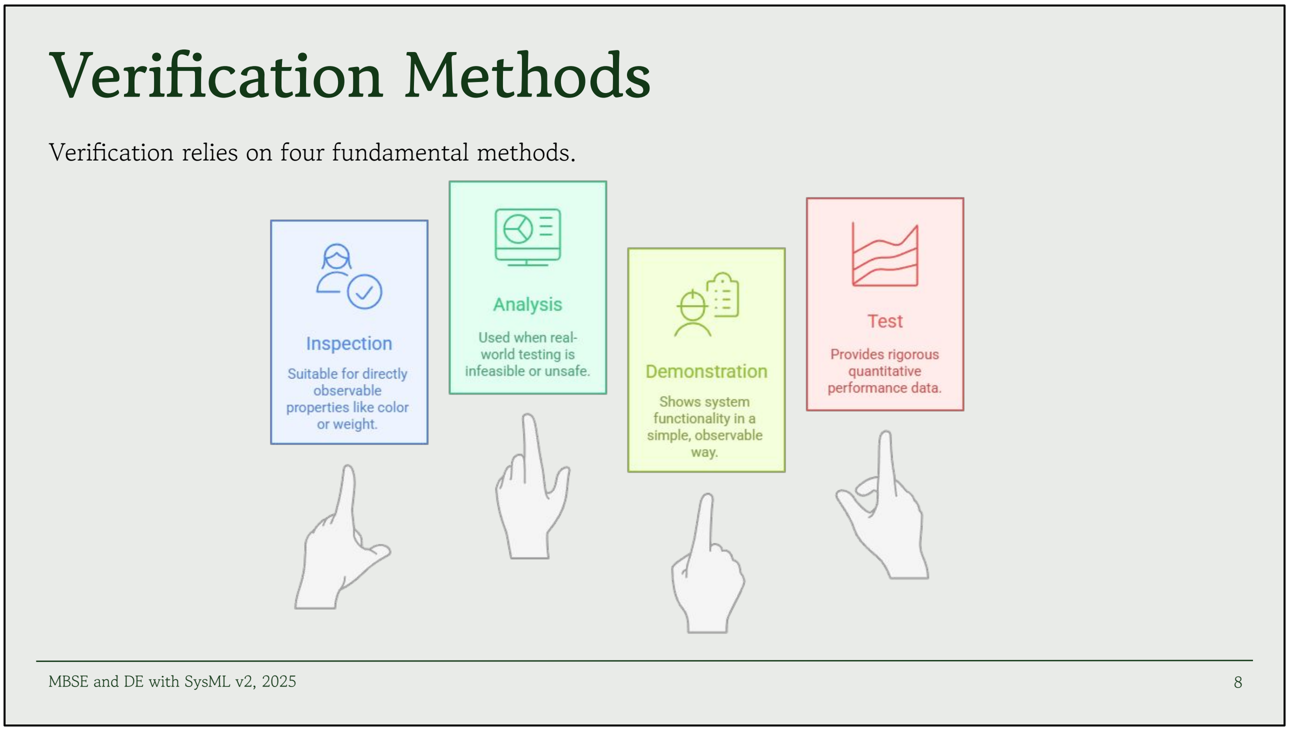


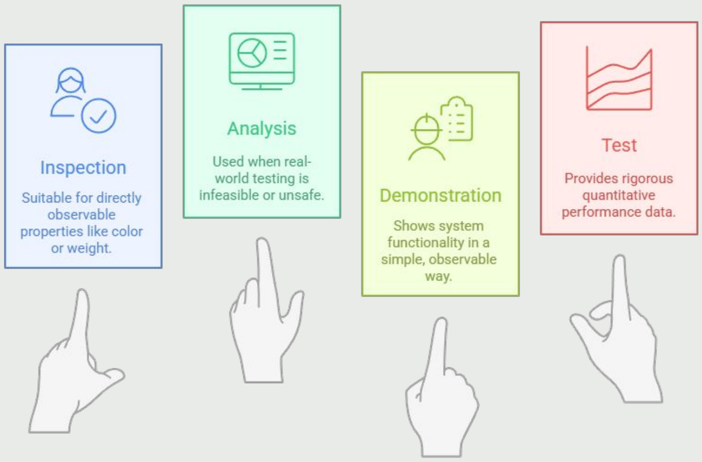
FIGURE 2.50 Verification per level. INCOSE SEH original figure created by Walden from Faisandier. Usage per the INCOSE Notices page. All other rights reserved.

This figure illustrates how verification applies at different stages of the system life cycle.
At the **requirements level**, stakeholder and system requirements are verified against their sources and quality characteristics. At the **model and simulation level**, verification checks purpose, syntax, and correct method usage.
At the **architecture level**, the defined SoI architecture is verified to ensure feasibility and compliance with accepted architectural methods.
At the **design level**, the SoI design and design characteristics are verified against requirements and industry rules.
Finally, at the **realization level**, the realized system and its elements are verified against system requirements and design characteristics to confirm that the system has been built right.



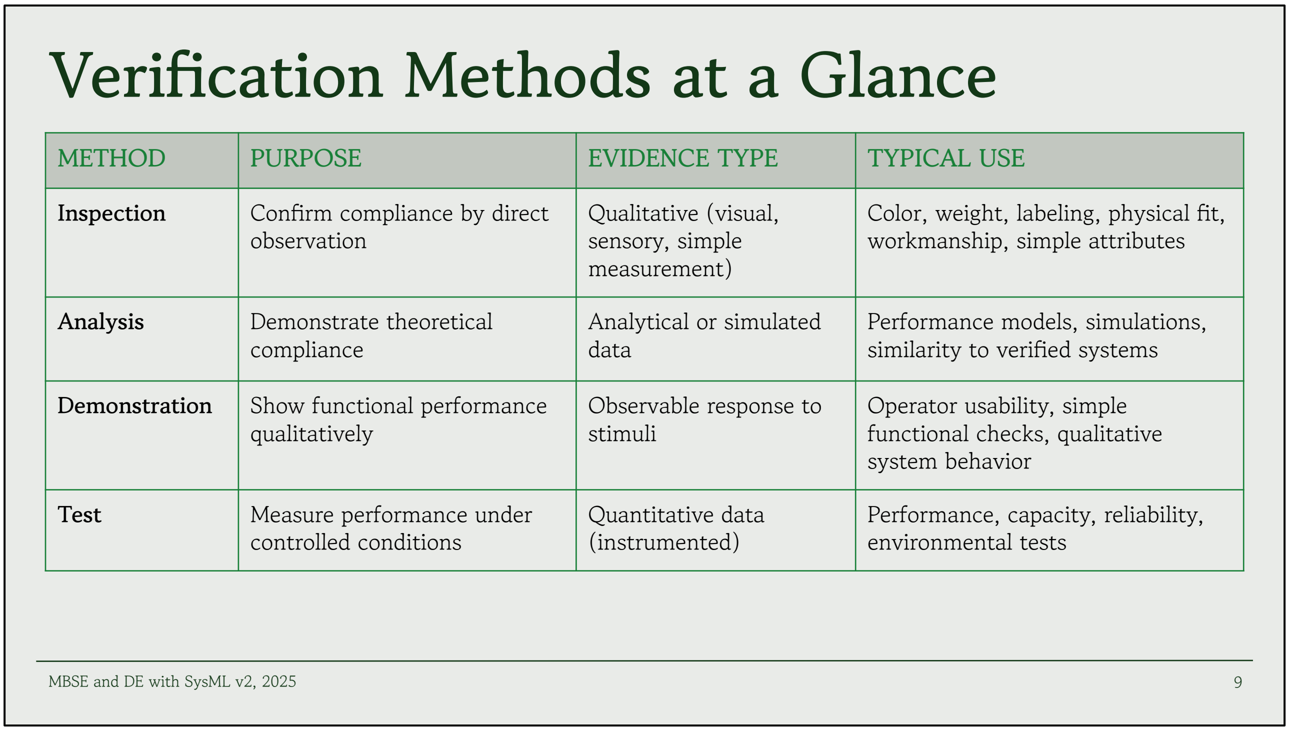
Verification Methods

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 provide the “how” of gathering objective evidence.

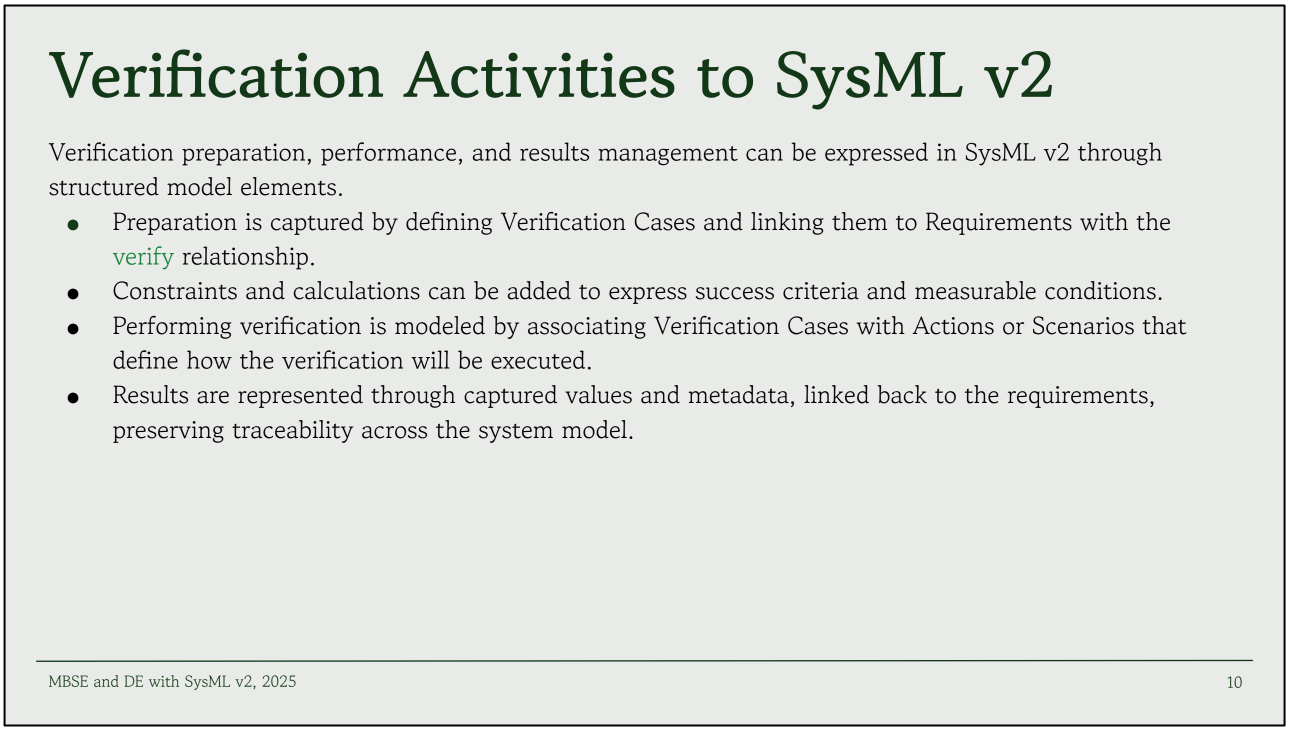
- **Inspection** confirms compliance by direct examination, often using human senses, gauges, or measurements, and is typically non-destructive.
- **Analysis**, including modeling and simulation, demonstrates theoretical compliance when physical testing is impractical or costly, and may also rely on similarity with prior verified systems under equivalent conditions.
- **Demonstration** provides qualitative evidence of functional performance by exercising the system with minimal instrumentation and observing expected responses.
- **Test** is a quantitative method, using controlled conditions and specialized equipment to measure operability, supportability, or performance and produce data for analysis. Together, these methods provide complementary ways to build objective evidence that requirements have been satisfied.



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

This table contrasts the four primary verification methods.
Inspection relies on direct sensory observation and simple measurement, making it effective for basic properties.
Analysis provides theoretical evidence, often through modeling, simulation, or similarity to existing verified systems.
Demonstration shows that a system or element can perform its intended function in a straightforward, observable way, without heavy instrumentation.
Test is the most quantitative, measuring performance under controlled, often instrumented conditions. Together, these methods form a toolkit: each requirement should be matched with the method that yields reliable and cost-effective evidence of compliance.



Verification Activities to SysML v2

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

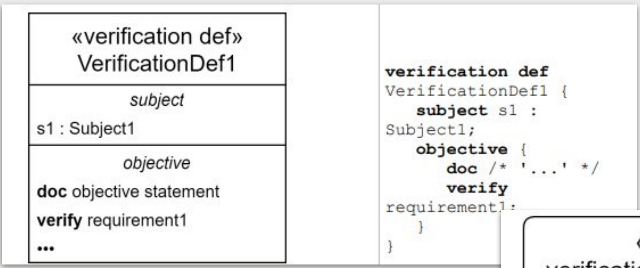
- Preparation is captured by defining Verification Cases and linking them to Requirements with the verify relationship.
- Constraints and calculations can be added to express success criteria and measurable conditions.
- Performing verification is modeled by associating Verification Cases with Actions 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.

SysML v2 does not execute verification but provides a framework to describe it. Verification Cases define the scope, success criteria, and context for verification actions. These are linked to requirements with the verify relationship. Constraints and calculations replace assertion-style semantics, providing measurable criteria. When performing verification, Scenarios or Actions define the procedure. Results are documented through values and metadata, ensuring traceability between requirements, verification actions, and outcomes.

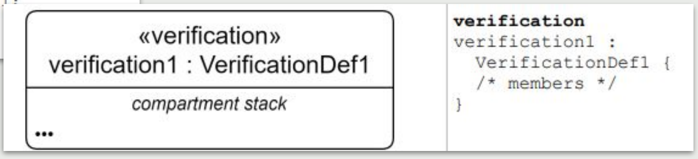
Verification Cases in SysML v2

- 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.



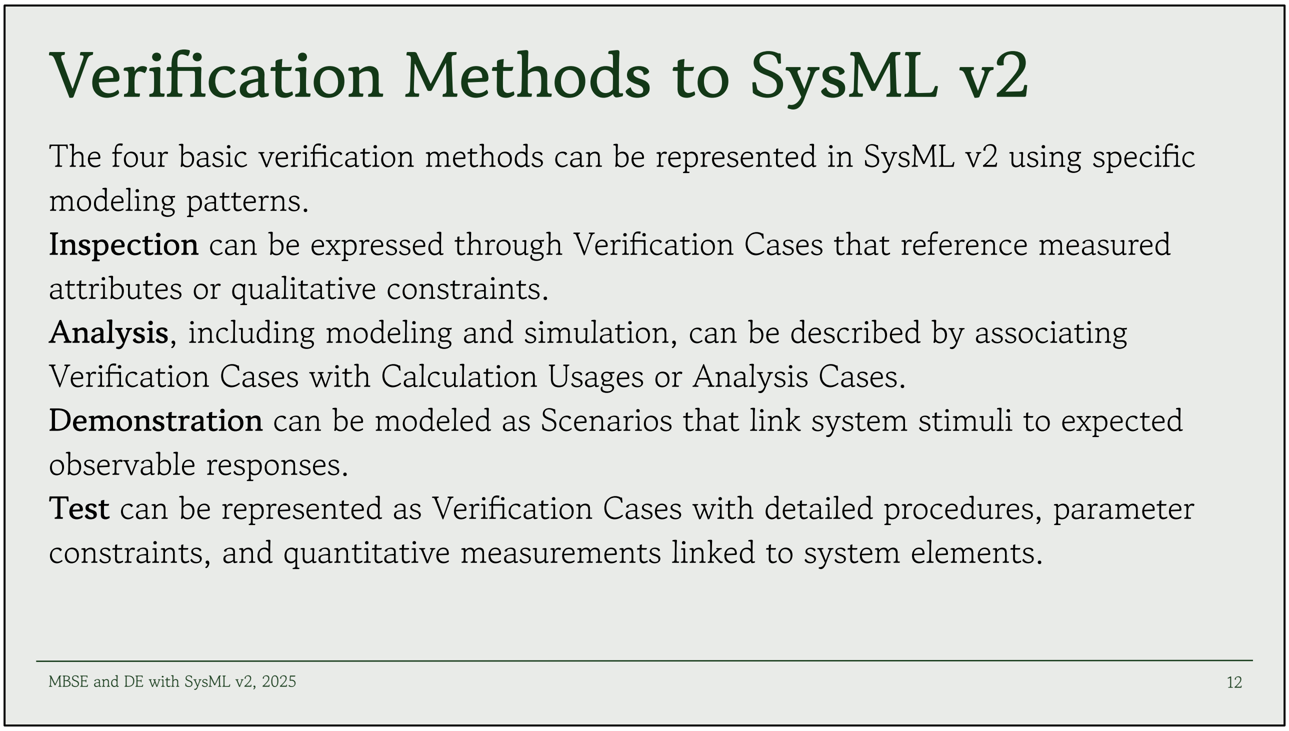
```
verification def
VerificationDef1 {
   subject s1 :
Subject1;
   objective {
      doc /* '...' */
      verify
requirement1;
   }
}
```



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

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

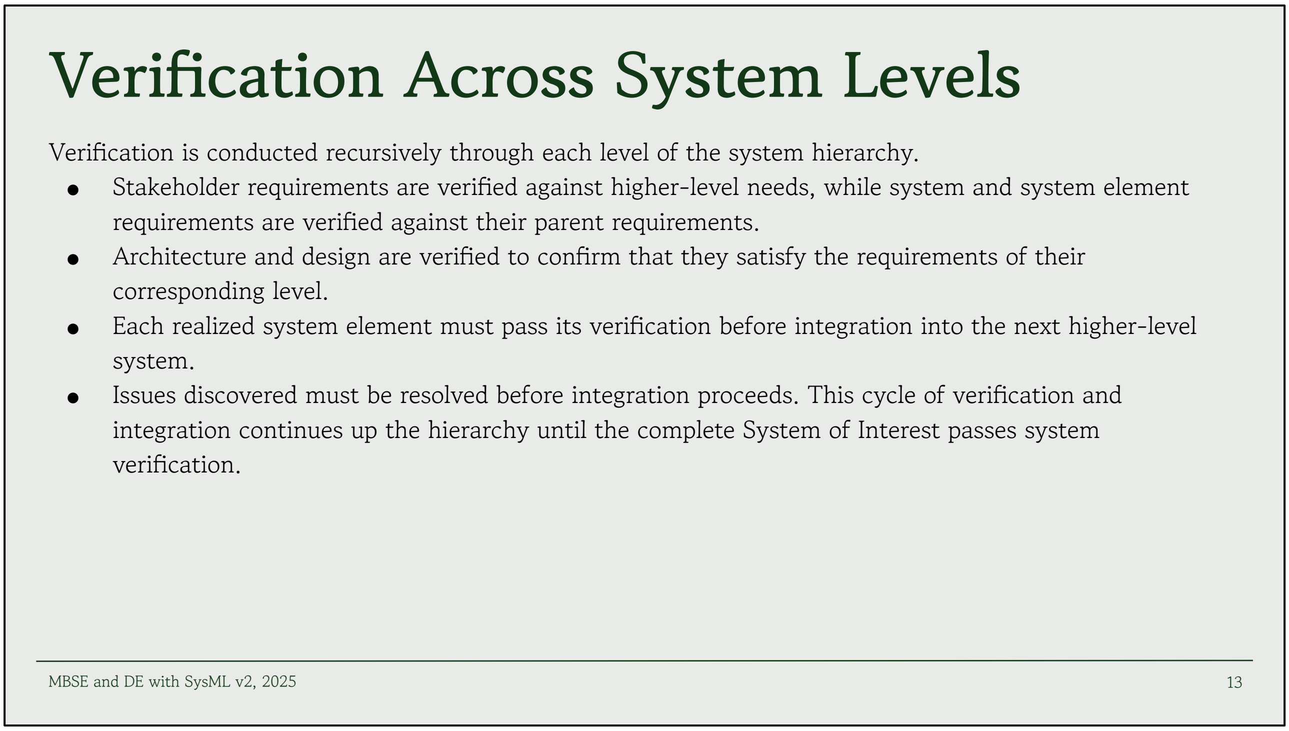
Verification Cases are the central mechanism for capturing verification in SysML v2.
A **verification definition** describes a general test method or procedure.
A **verification usage** applies that procedure to a specific system instance.
This separation mirrors other SysML constructs and promotes reuse. For example, a general vibration test can be defined once but applied to each subsystem. Because these cases are modeled entities, they can be queried, traced to requirements, or reported like any other system element. The verify relationship connects them to requirements, and constraints or calculations can be added to define measurable pass/fail criteria. This keeps verification fully integrated in the model, not external.



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.

SysML v2 provides constructs to model each verification method.
Inspection cases capture observable attributes like color or weight.
Analysis uses Calculation Usages or Analysis Cases to represent simulations or similarity arguments.
Demonstration focuses on scenarios with stimulus-response mappings.
Test cases include explicit input conditions, expected outputs, and parameter constraints, giving structure to quantitative evaluation.
Each method is modeled as a Verification Case, with specific SysML constructs chosen to represent the method's nature.

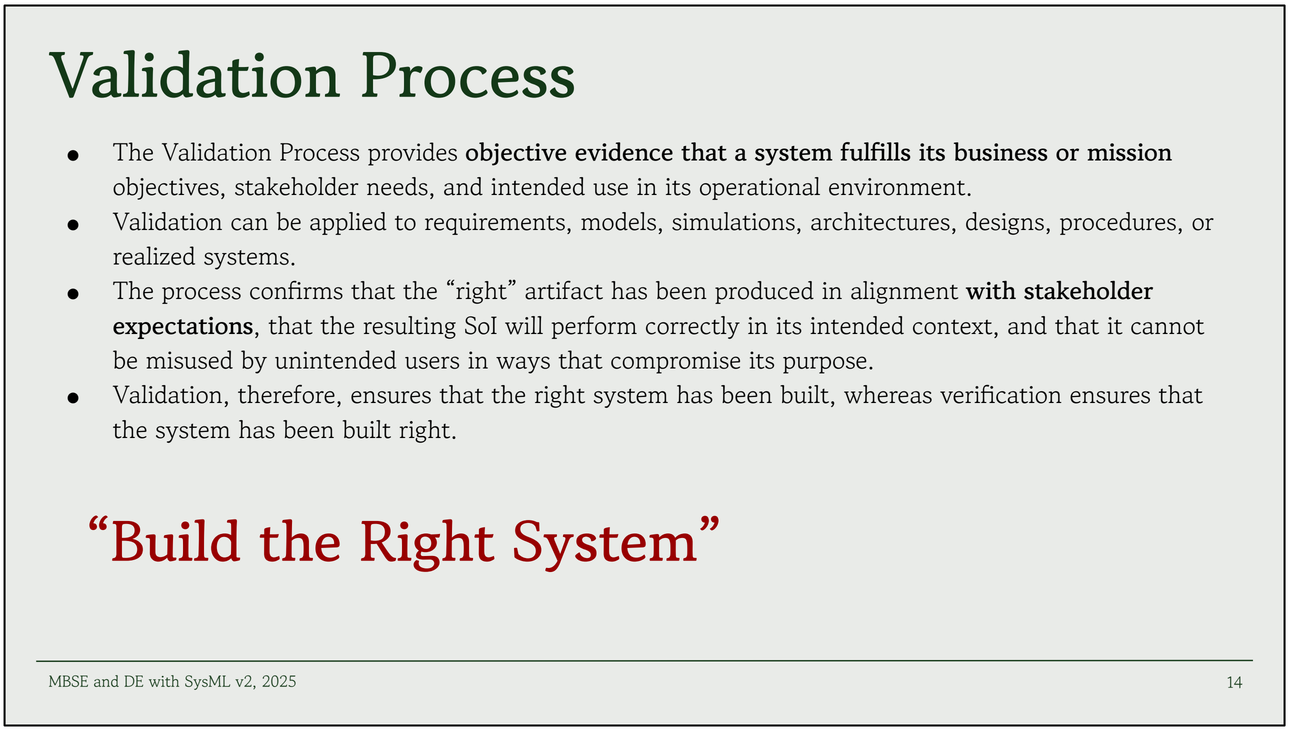


Verification Across System Levels

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.

Verification is not limited to the top-level system; it is performed at every level of decomposition and integration. The process starts by verifying stakeholder requirements against the needs they were derived from. System and system element requirements are then verified against parent requirements. Architecture and design verification ensures they will deliver systems capable of passing higher-level verification. Realized system elements are tested before being integrated upward, ensuring only verified components enter the next integration stage. Any discrepancies must be corrected early. This recursive approach continues until the fully integrated SoI has been verified.

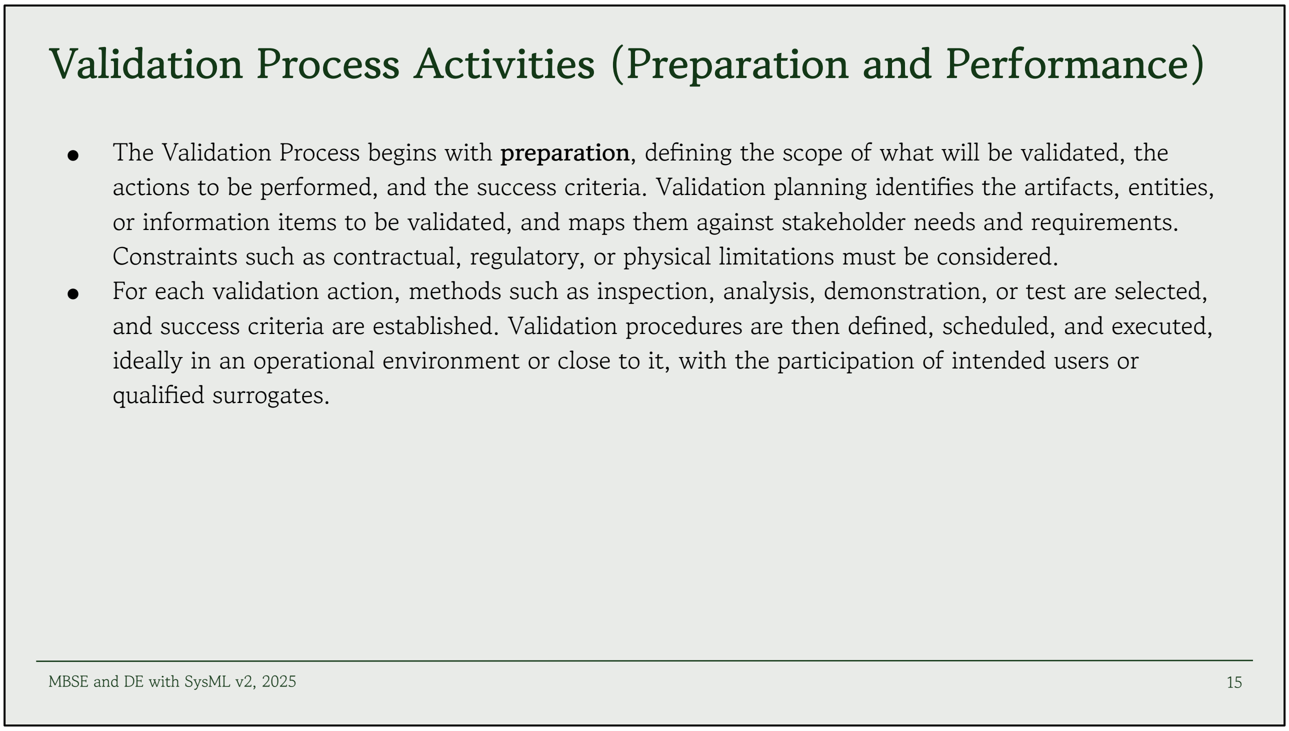


Validation Process

- 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”

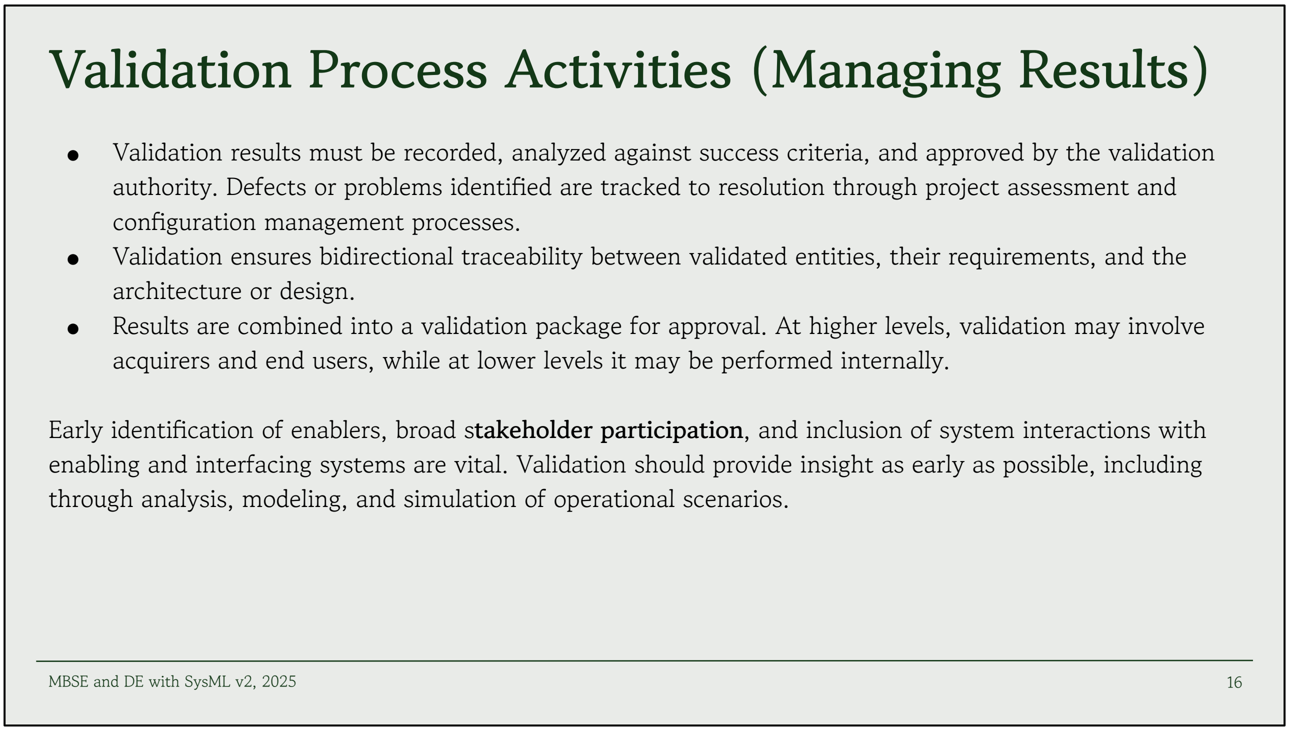
Validation focuses on fitness for purpose rather than compliance with specifications. Its purpose is to confirm that the system, when operated by its intended users in its operational environment, achieves its mission objectives and stakeholder needs.
Validation applies not only to the realized system but also to requirements, models, and design artifacts that influence the final product. It provides evidence that the system or element is the “right” one to deliver stakeholder value.
It also considers unintended use, ensuring the system cannot be exploited in ways that undermine its purpose. This distinguishes validation from verification: one ensures the right system is built, the other that it is built right.



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.

Preparation is essential for validation because it establishes both scope and feasibility. The entities to be validated are mapped against stakeholder needs, and constraints like safety or cost are factored into the plan. Methods are chosen to yield objective evidence with sufficient confidence. Validation procedures are defined and scheduled on the project plan, requiring resources, facilities, and user involvement. Execution should occur in environments close to operational conditions, ensuring the system is tested in realistic scenarios and by real or representative users.

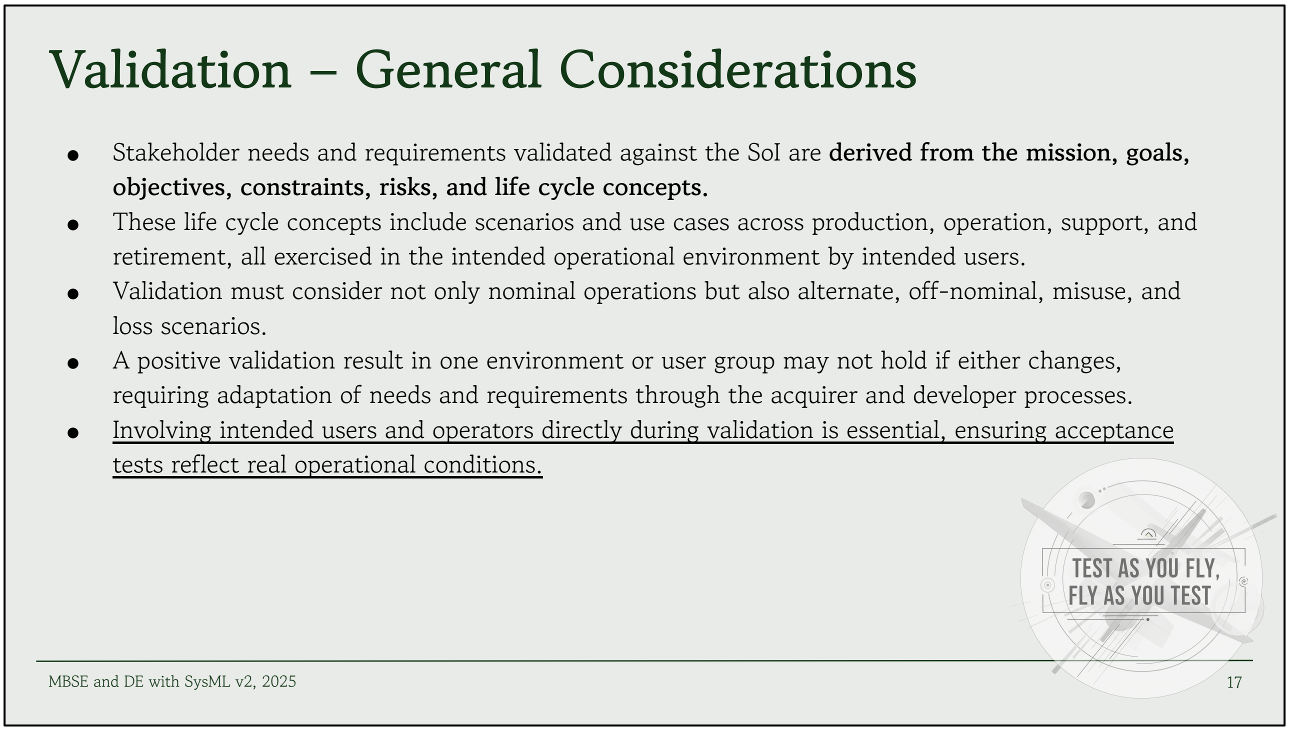


Validation Process Activities (Managing Results)

- 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.

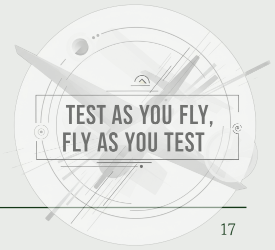
Early identification of enablers, broad s**takeholder participation**, and inclusion of system interactions with enabling and interfacing systems are vital. Validation should provide insight as early as possible, including through analysis, modeling, and simulation of operational scenarios.

Managing results is about ensuring validation evidence is credible, traceable, and accepted. Results are recorded, analyzed against criteria, and formal approval is obtained. Problems are logged and resolved systematically, feeding back into requirements or design if needed. Traceability is maintained between validation artifacts, stakeholder needs, and system elements. In practice, high-level validation often requires acquirer or user involvement, while lower-level validations may be supplier-led. Best practices emphasize starting validation planning early, involving broad stakeholders, and using simulation and operational scenarios to reveal insights long before the final system is fielded.

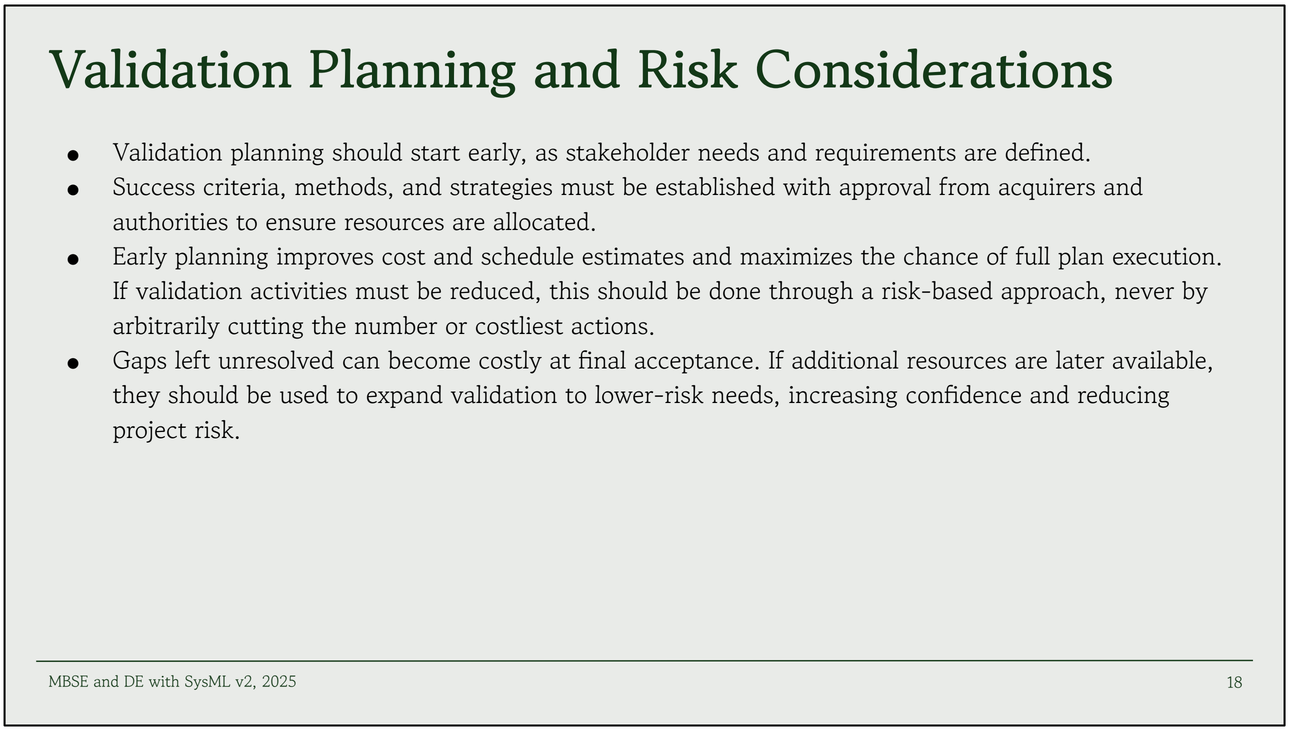


Validation – General Considerations

- 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.
- Involving intended users and operators directly during validation is essential, ensuring acceptance tests reflect real operational conditions.



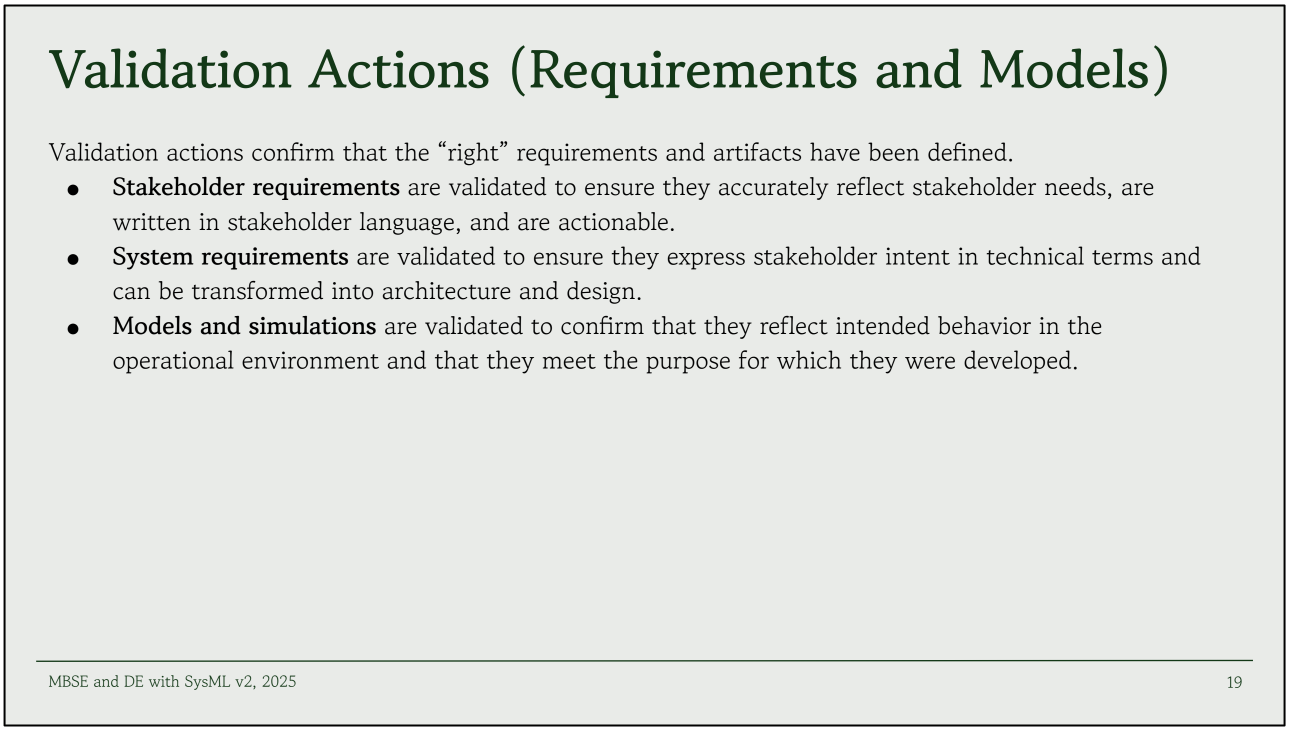
Validation is grounded in stakeholder needs and mission context. Life cycle concepts provide the operational scenarios against which the system is validated. **The principle "test as you fly, fly as you test" captures this approach.** Validation cannot focus only on nominal use; misuse, alternate, and failure scenarios must also be included. Because environments and user groups can change, validation must be responsive to evolving stakeholder requirements. Direct participation by intended users in their operational environment ensures results are credible and acceptable. Even if validation occurs at the supplier's facility, acquirers often repeat or extend validation in their own environment to confirm suitability.



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.

Planning for validation cannot wait until late in development. As needs and requirements are defined, validation strategies and criteria must be agreed upon with stakeholders. This ensures budgets and schedules account for validation activities. A disciplined, risk-based approach is required if resources are constrained. Blind cuts lead to gaps that surface at final acceptance, causing delays and cost overruns. If extra resources appear later, they should be directed to validate non-critical requirements as well, to reduce residual risk and increase stakeholder confidence.

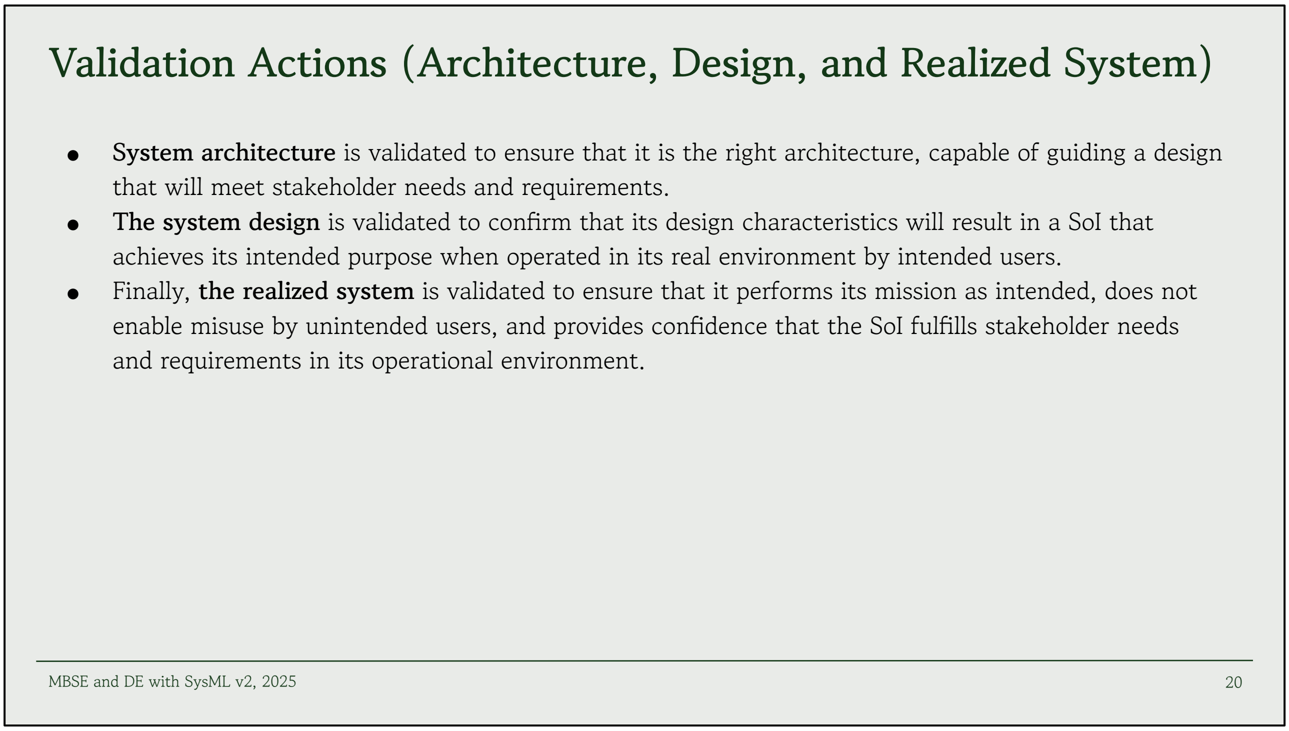


Validation Actions (Requirements and Models)

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 starts at the requirements and modeling level.
Stakeholder requirements must be clear, accurate, and actionable, ensuring that if the system were built to them, it would satisfy the original need.
System requirements must accurately translate these needs into technical form, supporting architecture and design.
Models and simulations must be validated both for accuracy in representing real-world behavior and for fitness to their intended purpose.
These activities provide early assurance that the system is being defined correctly, not just that it is internally consistent.



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.

At higher levels, validation shifts to architecture, design, and the realized system.
Architecture validation ensures the system structure and principles lead to a realizable solution aligned with stakeholder intent.
Design validation demonstrates that the characteristics of the design will deliver the intended performance in the real environment.
The final stage, system validation, confirms that the delivered product, service, or enterprise fulfills its mission for intended users, while also safeguarding against misuse or unintended operation. These levels together provide evidence that the “right system” has been built.

Validation Actions

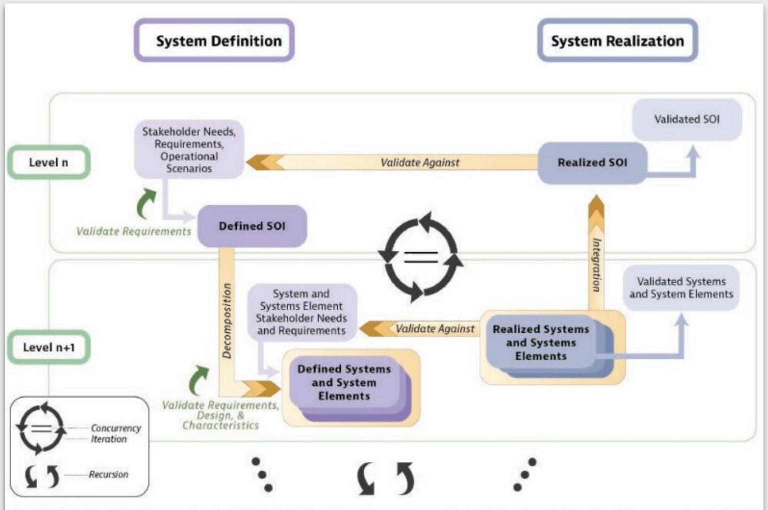
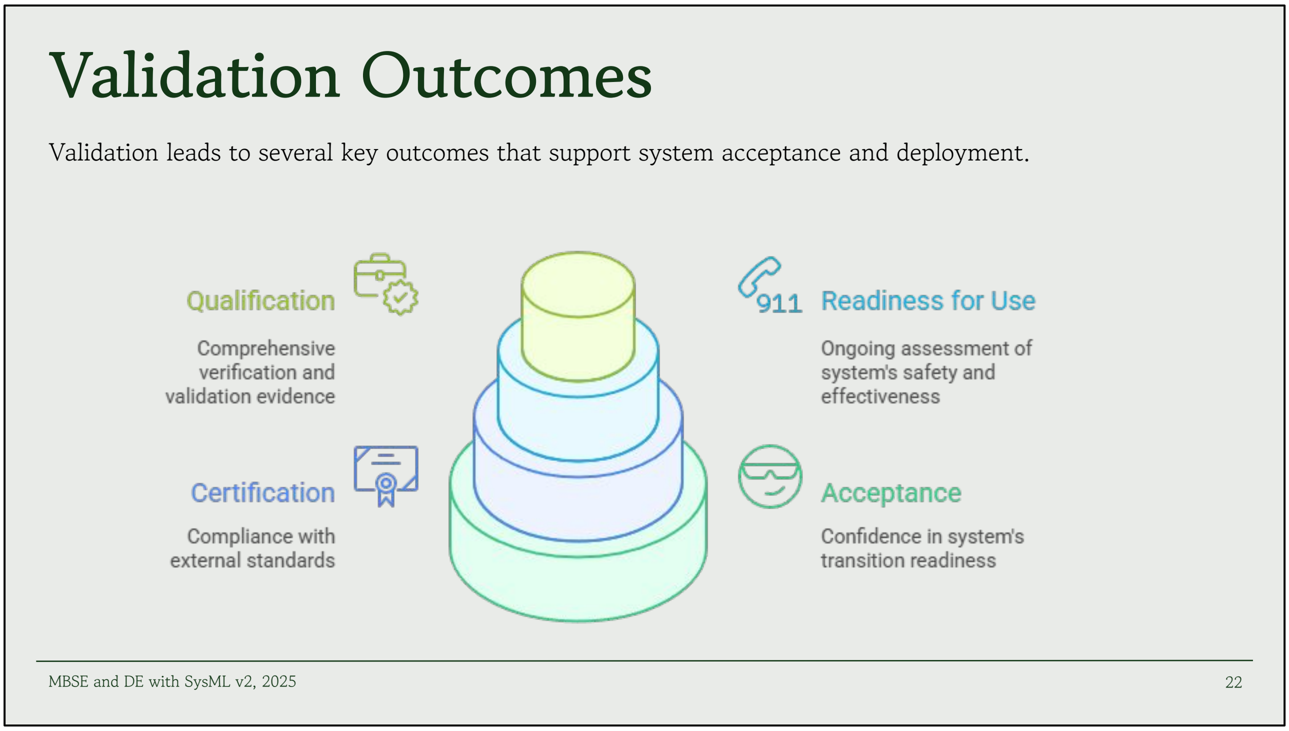

System Definition
System Realization
Level n
Stakeholder Needs, Requirements, Operational Scenarios
Validate Against
Realized SOI
Validated SOI
Validate Requirements
Defined SOI
Integration
Decomposition
System and Systems Element Stakeholder Needs and Requirements
Validate Against
Realized Systems and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

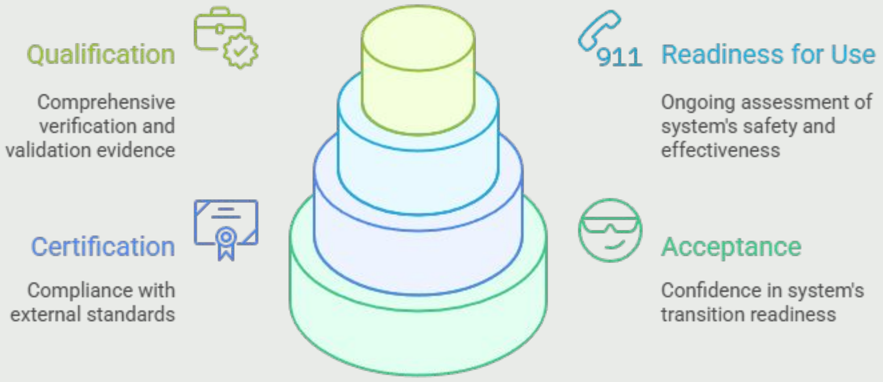
FIGURE 2.53 Validation per level. INCOSE SEH original figure created by Walden from Faisandier. Usage per the INCOSE Notices page. All other rights reserved.

Validation occurs at multiple levels of the system and its life cycle.
At the **requirement level**, stakeholder requirements are validated to ensure they reflect true needs and are actionable, while system requirements are validated to confirm they communicate intent in technical terms.
At the **model and simulation level**, validation checks that models represent intended behavior in the operational environment and fulfill their purpose.
At the **architecture level**, validation ensures the architecture is the right one to lead to a system meeting stakeholder needs.
At the **design level**, validation confirms the design will achieve its intended purpose when operated in real conditions.
Finally, at the **system level**, validation ensures the realized SoI performs as intended for its users and prevents misuse.



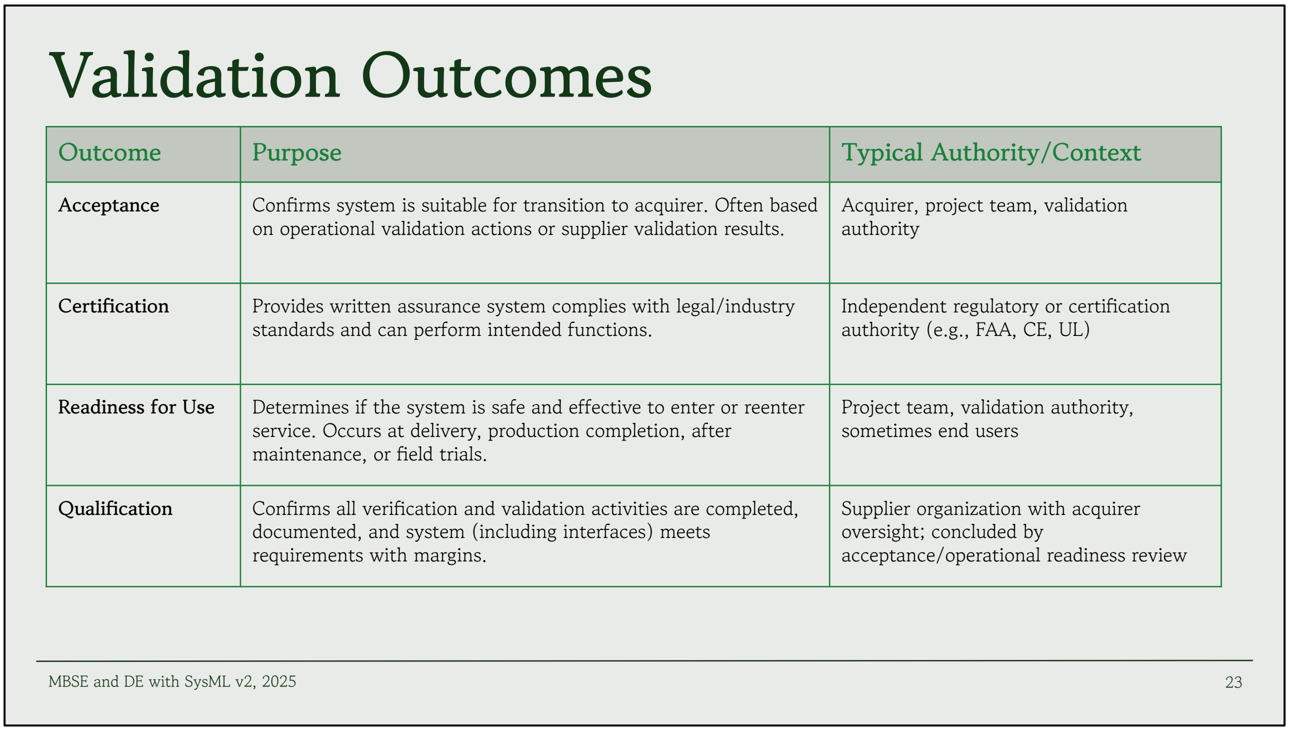
Validation Outcomes

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


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

The outcomes of validation extend beyond test results to formal decisions and authorizations.

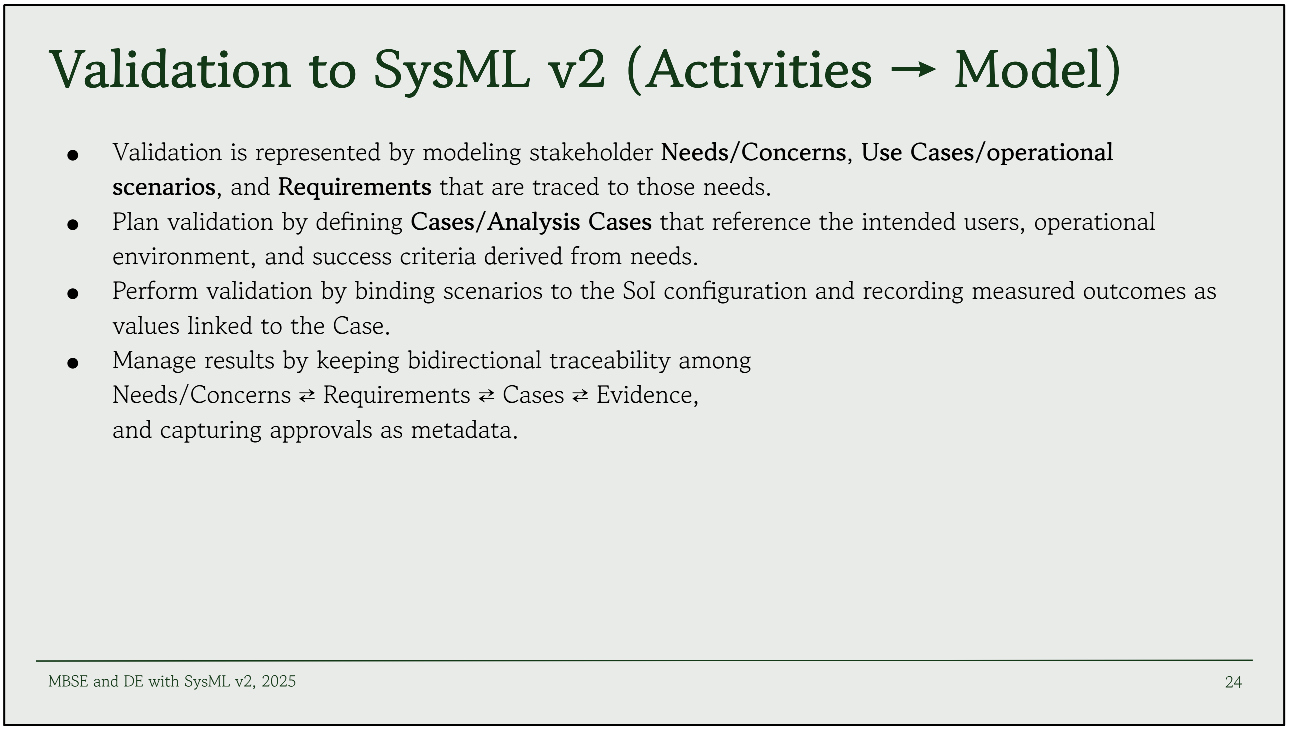
- **Acceptance** is performed prior to transition, allowing the acquirer to decide if the system is suitable, often through operational validation actions or reviews of validation results.
- **Certification** provides written assurance by an independent authority that the system meets defined procedures and complies with legal or industry standards, such as airworthiness or CE/UL approvals.
- **Readiness for Use** assessments determine whether the system can safely and effectively enter or reenter service, occurring at milestones like first delivery, production completion, maintenance, or field trials.
- **Qualification** confirms that all verification and validation actions have been successfully completed, documented, and that the SoI, including its interfaces, meets applicable requirements. Qualification concludes with acceptance and readiness reviews, declaring the system fit for intended use.



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

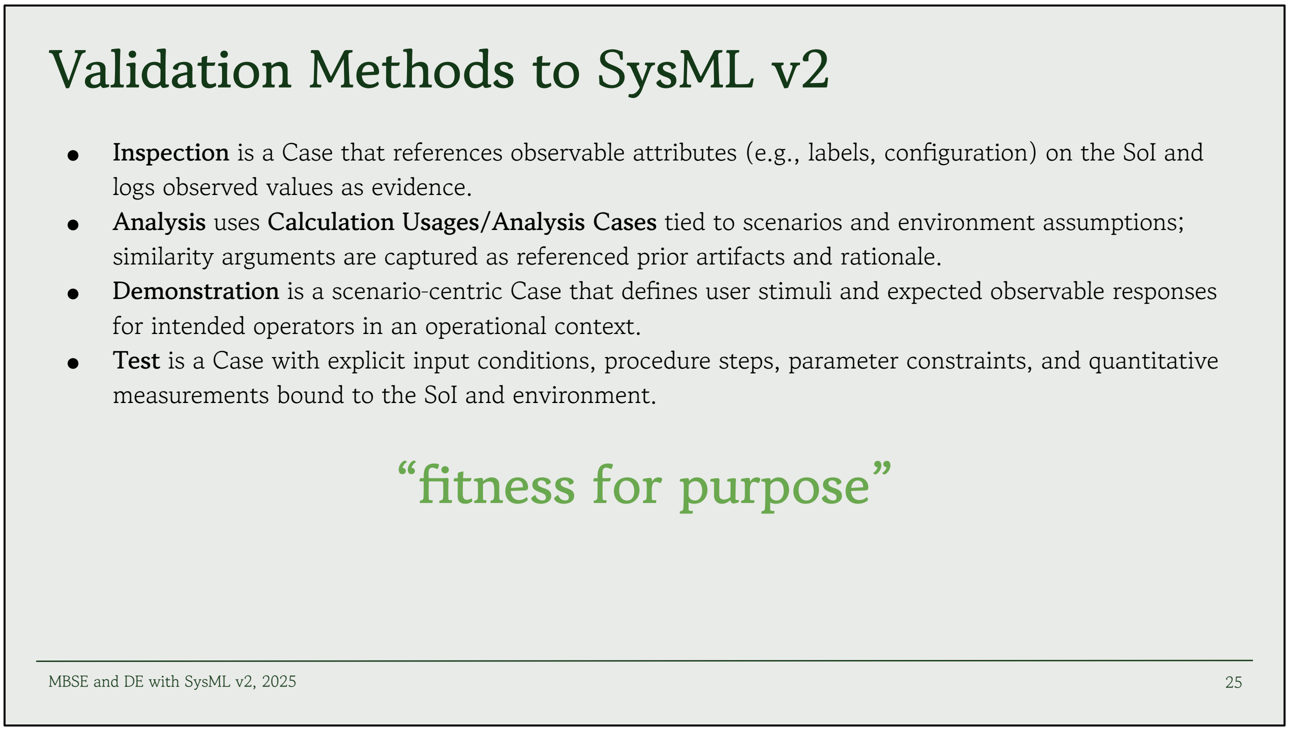
This table summarizes the four main validation outcomes.
Acceptance is an acquirer decision, based on evidence that the system is ready to transition.
Certification is external and independent, ensuring compliance with legal or industry standards.
Readiness for Use can occur multiple times during the life cycle, assessing whether the system can safely enter or reenter service.
Qualification is the most comprehensive, requiring all verification and validation evidence and confirming system performance with margins, including interface validation. It concludes with reviews that formally declare the system fit for use.



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.

SysML v2 has no validate link; use **Cases/Analysis Cases** plus **Use Cases** tied to stakeholder **Concerns/Needs**.
Evidence is values and calculations attached to the Case; acceptance/certification can be captured as metadata and views.
Keep traces from needs to requirements, from requirements to scenarios/cases, and from cases to recorded results.



Validation Methods to SysML v2

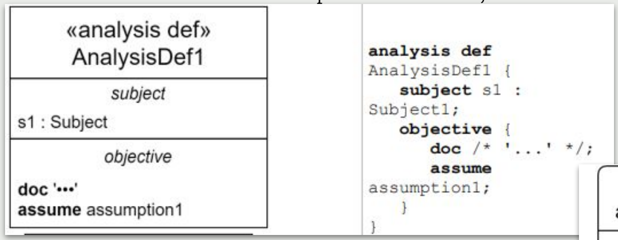
- **Inspection** is a Case that references observable attributes (e.g., labels, configuration) on the SoI and logs observed values as evidence.
- **Analysis** uses **Calculation Usages/Analysis Cases** tied to scenarios and environment assumptions; similarity arguments are captured as referenced prior 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”

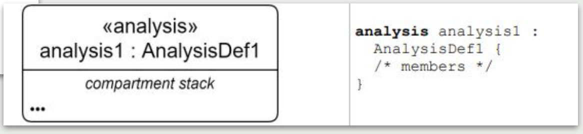
Model each method as a **Case** specialized by how evidence is produced: attributes for inspection, calculations/simulations for analysis, stimulus-response scenarios for demonstration, and instrumented procedures for tests.
Always reference the intended users and environment to keep validation about “fitness for purpose,” not just specification conformance.

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.



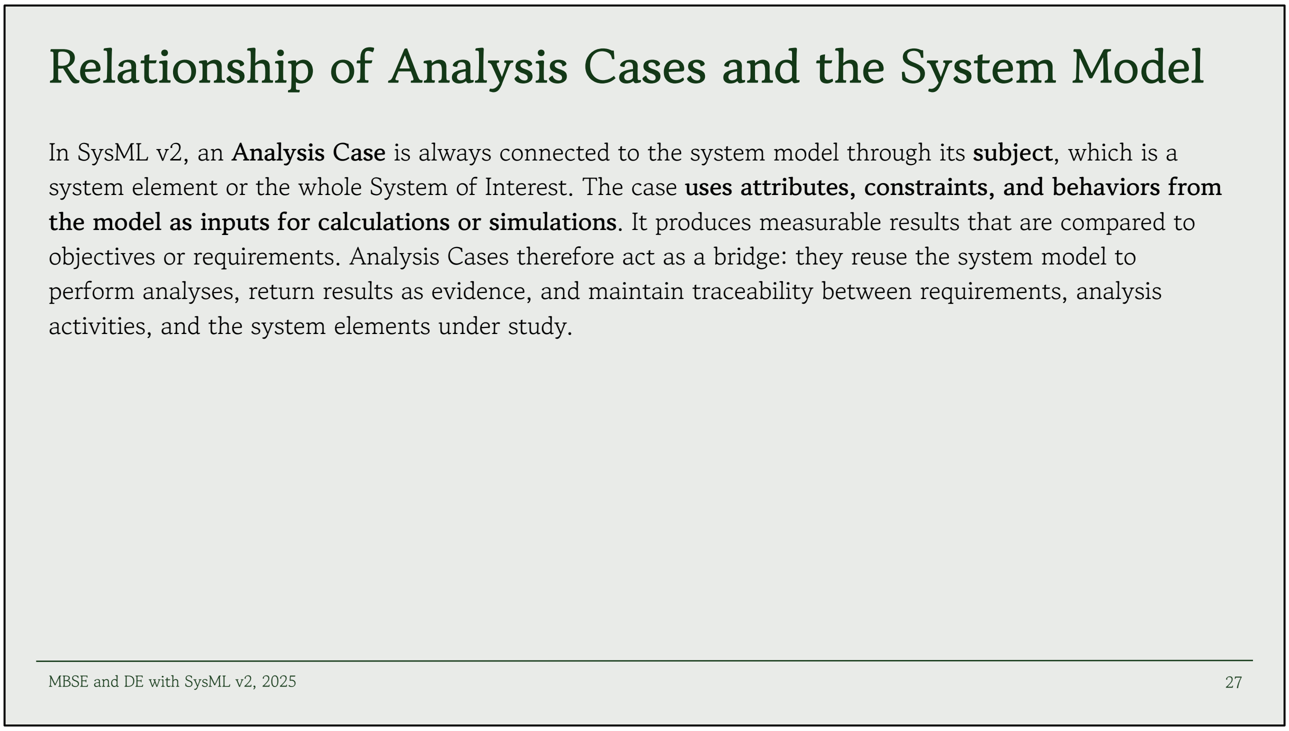
```
analysis def
AnalysisDef1 {
   subject s1 :
Subject1;
   objective {
      doc /* '...' */;
      assume
assumption1;
   }
}
```



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

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

Analysis cases are used to structure quantitative or qualitative analyses within the system model.
The **definition** captures the reusable pattern of the analysis, while the **usage** applies it to a specific system or subsystem.
The subject of the analysis is the entity under study, such as a UAV.
Results can be derived using SysML calculations, external solvers, or constraint equations.
In the drone example, a fuel economy analysis returns a measurable result which can be compared against a requirement. This makes analysis explicit and traceable within the model, supporting early validation and trade studies.

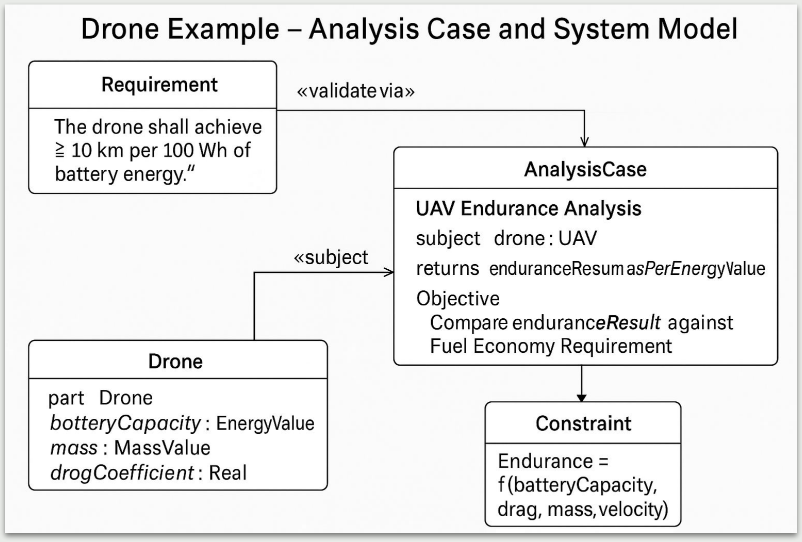


Relationship of Analysis Cases and the System Model

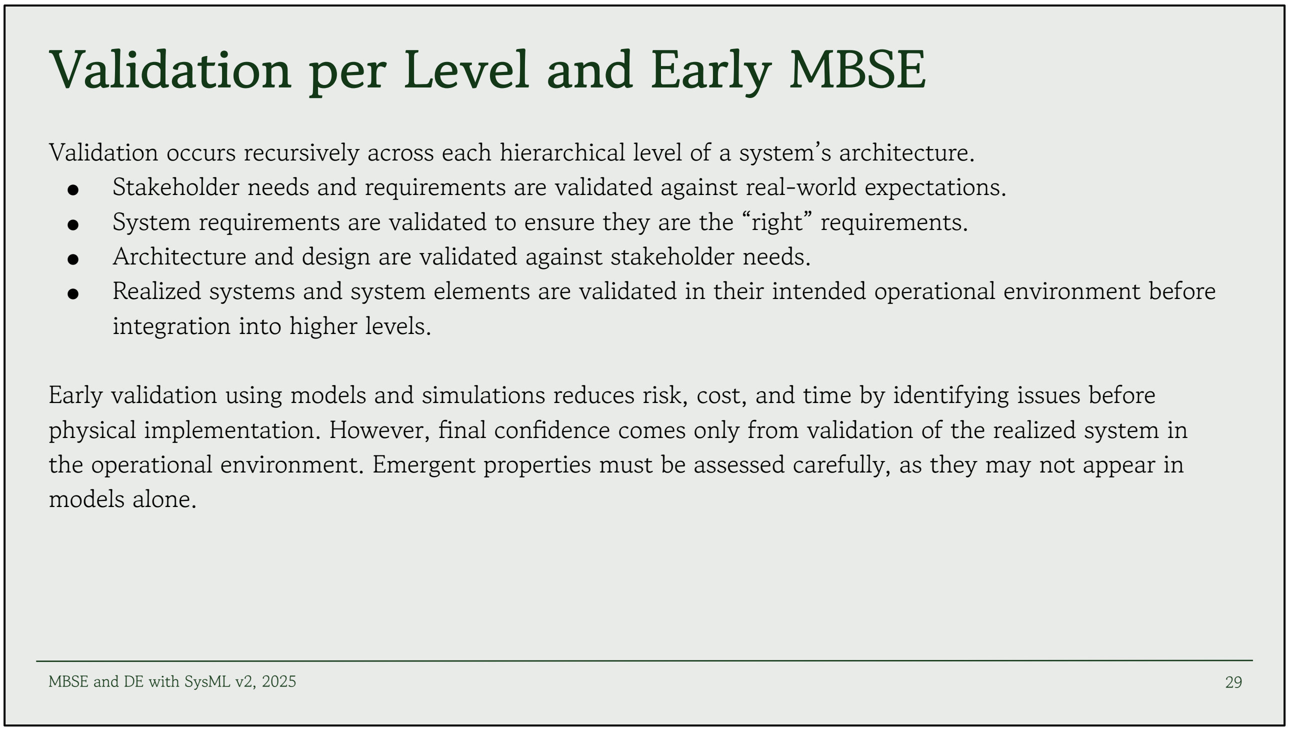
In SysML v2, an **Analysis Case** is always connected to the system model through its **subject**, which is a system element or the whole System of Interest. The case **uses attributes, constraints, and behaviors from the model as inputs for calculations or simulations**. It produces measurable results 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.

An Analysis Case is not isolated; it is bound to the **system model** through a subject, such as a drone or subsystem. It directly uses values like mass, thrust, or battery capacity, and applies system behaviors such as flight scenarios. Constraints and calculations defined in the model are executed within the case. The outputs become evidence to confirm whether requirements are satisfied. This ensures that analysis is integrated into the model, not external, providing **traceability** across requirements, analyses, and system design elements.

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 asPerEnergyValue
Objective
Compare enduranceResult against Fuel Economy Requirement
«subject
Drone
part Drone
botteryCapacity : EnergyValue
mass : MassValue
drogCoefficient : Real
Constraint
Endurance = f(batteryCapacity, drag, mass, velocity)

This diagram shows how an **Analysis Case** is bound to the system model. The UAV Endurance Analysis case takes the drone as its subject, uses parameters like mass, drag coefficient, and battery capacity from the model, and applies a constraint equation to calculate endurance. The result is compared to the Fuel Economy Requirement. This way, the **requirement, analysis, and system model are all linked in SysML v2**, creating a closed loop of traceability and measurable evidence.



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.

- Validation is recursive: each level of the system (needs → requirements → architecture → design → realized system) must be validated before integration upward.
- Early validation with modeling and simulation is critical in MBSE: it allows stakeholder feedback, expectation management, and design iteration before costly physical builds.
- Emergent properties, like cascading failures across interfaces, highlight why real system validation is essential.
- Models give theoretical assurance, but ultimate confidence requires operational validation with actual users in the real environment.
- Risk arises if validation relies only on simulations — always confirm with real system tests where possible.

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!