Contents

[1 Introduction 3](#_Toc461452853)

[1.1 Primary Goals for Requirement and Verification Needs Effort 3](#_Toc461452854)

[1.2 Limitations of current SysML requirements modeling 3](#_Toc461452855)

[2 Definitions 4](#_Toc461452856)

[2.1 Component 4](#_Toc461452857)

[2.2 Verification 4](#_Toc461452858)

[2.3 Validation 4](#_Toc461452859)

[2.4 Requirement Context 4](#_Toc461452860)

[3 Requirements from UML 4SE RFP 4](#_Toc461452861)

[4 SE Requirement and Verification Needs 4](#_Toc461452862)

[4.1 Basic Formal Requirement Statement Concept 4](#_Toc461452863)

[4.2 Allocation of System Level Requirements to Sub-system level requirements 5](#_Toc461452864)

[4.3 Requirement Analysis 6](#_Toc461452865)

[4.4 Assign Requirement Property Needs for: 6](#_Toc461452866)

[4.5 Requirement Management 7](#_Toc461452867)

[4.6 Requirement Reuse 7](#_Toc461452868)

[5 Model Examples 8](#_Toc461452869)

[5.1 Textual Requirement Transformations 8](#_Toc461452870)

[5.2 Requirement Transformation Example – Stopping Distance with Design Margin 8](#_Toc461452871)

[5.2.1 Original Textual Requirement Statement 8](#_Toc461452872)

[5.2.2 Transformed Formal Requirement Statement 8](#_Toc461452873)

[5.3 Requirement Transformation Example – Curb Weight 9](#_Toc461452874)

[5.3.1 Original Textual Requirement Statement 9](#_Toc461452875)

[5.3.2 Transformed Formal Requirement Statement 9](#_Toc461452876)

[5.4 Requirement Transformation Example – Initialization Time 9](#_Toc461452877)

[5.4.1 Original Textual Requirement Statement 9](#_Toc461452878)

[5.4.2 Transformed Formal Requirement Statement 9](#_Toc461452879)

[5.5 Requirement Transformation Example – Produce Diagnostic Report 10](#_Toc461452880)

[5.5.1 Original Textual Requirement Statement 10](#_Toc461452881)

[5.5.2 Transformed Formal Requirement Statement 10](#_Toc461452882)

[5.6 Requirement Transformation Example – Diagnostic Report Content 10](#_Toc461452883)

[5.6.1 Original Textual Requirement Statement 10](#_Toc461452884)

[5.6.2 Transformed Formal Requirement Statement 10](#_Toc461452885)

[5.7 Requirement Transformation Example – Requirement Name 11](#_Toc461452886)

[5.7.1 Original Textual Requirement Statement 11](#_Toc461452887)

[5.7.2 Transformed Formal Requirement Statement 11](#_Toc461452888)

[5.8 Tractor/Implement Example 11](#_Toc461452889)

[5.9 PC Mouse Example 11](#_Toc461452890)

[5.10 Power Plug Example 12](#_Toc461452891)

[5.11 Simple Two Plate Example 12](#_Toc461452892)

[5.12 Door Hinge Example 12](#_Toc461452893)

[5.13 Layered Interface Example 12](#_Toc461452894)

[5.14 Hybrid SUV 12](#_Toc461452895)

[6 Resources 12](#_Toc461452896)

[7 Open Questions 14](#_Toc461452897)

[8 Addendum A 15](#_Toc461452898)

[8.1 UML 4SE RFP Requirements 15](#_Toc461452899)

[8.1.1 Requirement 15](#_Toc461452900)

[8.1.2 Verification 16](#_Toc461452901)

# Introduction

This document includes some considerations to be addressed when developing the SysML v2 requirements for modeling requirements. The intent is not to provide a solution but to express the needs of the Systems Engineers and others users of SysML v2.

The solution may be provided completely by SysML or partially, in fact many of these capabilities already exist in SysML. Some of these capabilities may require the integration between SysML tools and other domain specific tools. However, SysML is intended to provide an integrated solution within the Systems development Environment.

## Primary Goals for Requirement and Verification Needs Effort

1. To capture **the** **concepts of a requirement specification, and the relationships to realization and verification**
2. To support the **capture of model based specifications that are more precise, analyzable, and verifiable**.
3. To enable the ability to capture a specification model that contains **information** that **is** generally **equivalent to that contained in a requirement specification document**
4. To **reduce specification ambiguity**
5. To enable the **ability to automate the validation of requirements and verification information within the model**
6. To **maximize the ability to reuse requirements** model elements
7. To maximize the **ability to automate tedious tasks and reduce overall modeling effort**, e.g. establishing and maintaining traceability.
8. The requirement concepts are consistent with the behavior, structure, and other concepts of the language
9. Ensure the ability to readily model different requirement and verification viewpoints that address a broad range of concerns

## Limitations of current SysML requirements modeling

1. Integration with other modeling standards, such as UTP, is difficult.
2. Requirements are not easy to reuse.
3. Requirements are only text based limiting the ability to automatically verify and use in analysis
4. Maintenance of traceability relationships is an issue since it is all manual and therefore easily and quickly degrades from human error
5. Tabular selected views of the requirements
	1. Tables are often a simple and intuitive means for entering and viewing requirement related information
	2. The content of the cells may be distributed from across different parts of a model, i.e. not from a single package

# Definitions

## Component

(1) An entity with discrete structure, such as an assembly or software module, within a system considered at a particular level of analysis. (ISO/IEC 1998)

(2) One of the parts that make up a system. (IEEE 2008)

 [3, SEBoK Glossary]

A component is the generic term for the level of decomposition at which system elements are fully specified, and for which design disciplines can implement them. [3, similar to SEBoK Glossary Discussion] Note that a component can correspond to any level of the system hierarchy.

## Verification

The purpose of the Verification process is to provide objective evidence that a system or system element fulfils its specified requirements and characteristics.

The Verification process identifies the anomalies (errors, defects, or faults) in any information item (e.g., system requirements or architecture description), implemented system elements, or life cycle processes using appropriate methods, techniques, standards or rules. This process provides the necessary information to determine resolution of identified anomalies. [15288]

## Validation

The purpose of the Validation process is to provide objective evidence that the system, when in use, fulfills its business or mission objectives and stakeholder requirements, achieving its intended use in its intended operational environment. [15288]

## Requirement Context

The requirement context are a set of items that can provide supporting information and define the requirement. Context items can include:

1. Supporting text that can provide further insight to the environment in which the requirement belongs, for example this can be of the form of an example, a definition or clarifying statement.
2. Properties can be referenced in Result or Precondition expressions.
3. Other requirements within the requirement context and their relationship to this requirements. These relationships can include derive, trace and copy.
4. Relationships to other elements including satisfy and verify

# Requirements from UML 4SE RFP

See Addendum A.

# SE Requirement and Verification Needs

## Basic Formal Requirement Statement Concept

* A requirement imposes a Constraint on one or more Realized element(s), by specifying the limit for the required value(s) for specific features of the realized element (i.e. Constraint) and the valid values for the realized element (i.e. Constraint Evaluation).  For example:
	+ Constraint: Required Weight = 3000 lbs. and Constraint Evaluation: the RealizedWeight <= Required Weight,
	+ Constraint: the ReqdPwr =  k \* ReqdWt and Constraint Evaluation: the RealizedPwr > ReqdPwr
* This Verification Method evaluates the realized element using the Realization Evaluation, and using the Constraint Evaluation to compare the result(s) from the Realization Evaluation with the required value(s).
	+ The Verification Method evaluates the Unit Under Verification (UUV) to determine its value that can be compared to the required value.
		- If the Verification Method is an analysis, then the value is an estimated value(s), i.e. weight = sum of the weights
		- If the Verification Method is a test, then the value is a measured value(s).
		- There other Verification Methods such as inspection, demonstration, verification by similarity (analogy), or sampling
		- The Verification Method generally involves the use of verification components to determine the actual values, i.e. put vehicle on scale and weigh
		- A Verification Method may involve a combination of the above methods, such as hardware, software, and people in the loop, where there can be some analysis and some test.
	+ The Verification Method then compares the value derived from the Unit Under Verification (UUV) with the required value.
	+ From this comparison the Verification Method determines the Verdict (None, Pass, Inconclusive, or Fail)

## Allocation of System Level Requirements to Sub-system level requirements

1. The Requirement Allocation relationship is used to assign higher architecture level (e.g. system level) requirements to one or multiple lower level architecture subcomponents (e.g. subsystem level).
2. The allocation initiates the analysis process for deriving requirements for the lower level subcomponents.
3. An allocation of a requirement to sub-components may also include a value distribution assignment to the sub-components and include a design margin. For example if the vehicle required weight <=3000 lbs., when this weight is allocated to sub-components the weight may be distributed as follows; 1000 lbs. to the chassis and drivetrain; 500 lbs. to the power source, 300 lbs. to the body; 800 lbs. to the passenger compartment; 200 lbs. to the transmission; and 200 lbs. reserved as a design margin.
4. The subcomponent teams examine the allocated requirements to first determine if the allocated requirements can be fully realized by the subcomponent, partially realized or if one or more of the allocations cannot be realized by the subcomponent requiring the allocated requirement to be re-allocated to one or more of the other sub-components.
5. If an allocated requirement was allocated to multiple sub-components, the sub-component teams need to determine what part of the allocated requirement each of the sub-components will realize and to ensure complete coverage across the sub-components.
6. The effort completes when new sub-component requirements (sub-component shall…) are derived and a requirement derive relationship is established from the sub-component requirements to the allocated component level requirements to provide traceability.
7. At this point the requirement allocation relationship may be removed or left for historic or archival needs.

## Requirement Analysis

1. Create a view that isolates a set of requirements for review
	1. This action addresses the grouping and filtering of requirements for a particular task.
2. Each requirement is typically assigned a set of attributes that help categorize the requirement. The specific categories are determine by the methodology. Some typical attributes include Verification Method Type, safety related, requirement type (e.g. functional, performance, safety, etc.) See the INCOSE Requirements Writing Guide for a more complete list of attributes typically used by organizations.
	1. Each attribute is assigned a value.
	2. A set of requirements is often identified by the values in one or more of the attributes
3. Adjust Requirements as necessary to ensure they are clear, concise and complete
	1. Add, remove or change a requirement statement, attribute or a relationship to or from this requirement
4. Decompose a compound requirement into 2 or more singular requirements. When this is done a decompose relationship is established from the individual requirements to the compound requirement.
5. Refine the requirement if it is not a “good” requirement (see INCOSE Requirement Writing Guide).
	1. A requirement refinement relationship is used between any two requirements where a second requirement is used to re-express the first element more precisely and/or with less ambiguity.
	2. A refinement also occurs when transitioning a text based requirement statement to a formal requirement statement and may also occur when a version of the requirement is performed.
6. Quickly asses how each requirement change impacts other elements
7. Conduct reviews on changed requirement sets
8. Merge the set of updated requirements into a new version of the requirement specification.

## Assign Requirement Property Needs for:

* 1. All properties
		1. See Properties and Expression Needs artifacts for property definitions
		2. All can be simple types, such as string, or complex as in an array of values
	2. Requirement ID –
		1. Is this really necessary since each model element has a unique ID?
		2. Smart Numbering can help organize requirements into similar groups and can be used to sort
	3. Requirement Name – A short text phrase providing insight to the requirement content
	4. Requirement Subject –
		1. The element that that the requirement is imposed upon.
		2. To avoid maintenance issue this should be a reference to the name of a defined model element
	5. Define property based pre-condition constraints
		1. Define one or more properties of the requirement that establish a pre-constraint that must be true before the requirement can be verified.
			1. Each property can be a single value, a set of values, an array of values
			2. Each value or set of values have units
			3. Each value or set of values can have a tolerance/probability
			4. From Stopping distance Example –
				1. Values assigned to wet/dry conditions property and the initial speed property as displayed in the table
				2. Values assigned to alpha for wet and dry conditions
			5. Define one or more external references
				1. References can be to files that contain values for other initial conditions or documents that express textual pre-conditions

Federal Motor Vehicle Safety Standards And Regulations, U.S. Department Of Transportation, Standard 105

* 1. Requirement Statement
		1. Textual statement – Statement must be clear, concise and unambiguous.
			1. If the sentence conforms to a standard structure than a content checker could detect poor phrases and inforce the structure.
		2. Formal requirement statement - Logical or mathematical expressions
			1. Must conform to a pre-define syntax.
	2. Requirement Context – a model element within the context of the requirement that provides properties to support a requirement result expression or precondition expression

## Requirement Management

1. Baseline a set of requirements and supporting information as a new revision
2. Isolate a set of requirements for analysis
3. Measure the impact of a set of proposed changes
4. Merge reviewed changes into a new baseline
5. Compare two baseline revisions and easily identify what was changed, what was added and what was been deleted between to two revisions.

## Requirement Reuse

1. Select and download specification elements from a library that contain a set of supporting context properties, requirements, and element properties for the element
2. Minimize the use of informal textual only based requirements and transition requirements to a formal requirement statements
	1. A formal requirement statement is a requirement that maximize automation of tasks
		1. Can be used to verify the element requirements,
		2. Can be used to automate the validation of requirements, i.e. complete, clear, concise, etc.
		3. Can be used in analysis of the element
3. Easily change the value or reference to defined elements, for example
	1. Change reference to the “subject” to a different define element
	2. Change the references to context and element properties
	3. Change the ID to the next number in a pre-defined number sequence.
	4. Select if the requirement uses a shall, will or should as the statement verb

# Model Examples

The following examples are captured in the SECM. All the examples will be used to demonstrate some modeling difficulties with the existing SysML spec.

## Textual Requirement Transformations

The purpose of this section is to provide some examples of transforming a textual requirement statement to a formal requirement statement. The various types of examples could include;

* Performance
	+ Simple property e.g. weight (physical)
	+ More complex such as stopping distances with dependencies on other requirement statement values, and probability example such as an alignment accuracy or equivalent
* Interface
* Behavior
* State dependent behavior
* Informal
* Supply Chain and Manufacturing, e.g. COTS
* Failure example
* Two verification by analysis and test

## Requirement Transformation Example – Stopping Distance with Design Margin

**Type** – Performance, more complex with dependencies on other requirement

### Original Textual Requirement Statement

When traveling at a speed of 60 mph, the vehicle shall stop within 120 feet.

### Transformed Formal Requirement Statement

**Subject** = a Property of Vehicle = Stopping Distance

**Pre conditions**

1. Start Event – When the brake pedal is initially pressed.

2. Context Property, State of Surface = Dry (measure dry? measure moisture content on surface, or inspection)

2. Context Property, Concrete Surface Roughness <= 125 µm)

3. Context Property, Surface incline = +/- 0.5 degree

4. Vehicle Stopping Speed = 60 mph +/- 0.1 mph

**Constraint** = Boundary Expression:

Vehicle Property: Required Stopping Distance = 120 feet

**Constraint Evaluation:** Example including a design margin

Actual Stopping Distance <= .9 \* Required Stopping Distance

**Generated text statement** - When the brake pedal is initially presses, the state of the surface is dry, the concrete surface roughness is less than 125 µm, surface incline is +/- 0.5 degrees and the vehicle stopping speed is 60 mph +/-0.1 mph, the vehicle stopping distance shall be less than or equal to 108 feet.

## Requirement Transformation Example – Curb Weight

**Type** – Performance – simple property requirement

### Original Textual Requirement Statement

The Vehicle weight shall not exceed 3200 pounds.

### Transformed Formal Requirement Statement

**Subject** = Vehicle Property = Vehicle::Required Curb Weight

**Pre conditions**

1. Context Property, Number of Passengers = 0

2. Context Property, Cargo Weight = 0 lbs.

3. Vehicle Property, Fuel Capacity = 14 gallons (Full)

4. Vehicle Property, Transmission Oil Capacity = 2 quarts (Full)

5. Vehicle Property, Engine Oil Capacity = 5 quarts (Full)

6. Vehicle Property, Coolant Capacity = 5 gallons (Full)

**Constraint** = Boundary Expression:

Vehicle::Required Curb Weight = 3000 lbs.

**Constraint Evaluation:**

Actual Curb Weight <= Required Curb Weight

**Generated Text Statement:** With zero passengers, cargo weight of 0 pounds, a fuel capacity of 14 gallons, transmission oil capacity of 2 quarts, engine oil capacity of 5 quarts, and coolant capacity of 5 gallons, the Curb Weight of the vehicle shall be equal to or less than 3000 pounds.

**Requirement Supporting Text** –

Curb weight is the total weight of a vehicle with standard equipment, all necessary operating consumables such as motor oil, transmission oil, coolant, air conditioning refrigerant, and a full tank of fuel, while not loaded with either passengers or cargo.

## Requirement Transformation Example – Initialization Time

**Type** - State Based Performance Requirement

### Original Textual Requirement Statement

After applying power the system shall be in the operational state within 30 seconds.

### Transformed Formal Requirement Statement

**Subject** = System Property = System::Initialization Time

**Pre conditions**

1. Start Event – When system power is applied.
2. System State = Idle

**Constraint** = Expression:

Stop Event: When State transitions to Operational State

Required Time from Idle to Operational State = 30 seconds

**Constraint Evaluation:**

Initialization Time <= Required Initialization Time

**Generated Text Statement:** When system power is applied and the system is in the idle state the system shall transition to the operational state in less than or equal to 30 seconds.

## Requirement Transformation Example – Produce Diagnostic Report

**Type** - Behavior

### Original Textual Requirement Statement

The vehicle shall send a diagnostic report to the diagnostic tester.

### Transformed Formal Requirement Statement

**Subject** = Vehicle (System)

**Pre conditions**

1. Start Event – The Diagnostic Tester cord is inserted to the diagnostic tester Plug
2. Subject Property – Vehicle State = Idle, ignition on, or running

**Constraint** = Examination:

Send a diagnostic report to the diagnostic tester

**Constraint Evaluation:**

Constraint Evaluation = True

**Generated Text Statement:** When the vehicle is in an idle state and the Diagnostic Tester is plugged in, the vehicle shall send a diagnostic report to the diagnostic tester.

## Requirement Transformation Example – Diagnostic Report Content

### Original Textual Requirement Statement

The Diagnostic Report shall contain the following data: VIN, report execution date/time, list of vehicle errors and vehicle state when the report was run.

### Transformed Formal Requirement Statement

**Subject** = Vehicle (System)

**Pre conditions**

1. Start Event – A known error is produced with an Error ID
2. Context Property – Date and time value the error was produced +/- .05 second
3. Subject Property - Vehicle State = Idle, ignition-on, or running
4. Subject Property = Known VIN number value
5. Starting Events
	1. Starting Event 1 – Vehicle state is idle
	2. Starting Event 2 – Vehicle state is Ignition-on
	3. Starting Event 3 – Vehicle state is running

**Constraint** = Examination:

A diagnostic report is produced

The report contains the known VIN number, the known date/time the report was executed, the correct vehicle state, and at least the known produced error? (True/False)

**Constraint Evaluation:** Examination

Produced VIN data = Actual VIN data, and

Report Execution Date/Time = +/- 0.5 sec, and

Produced Vehicle State = Vehicle State, and

Produced Error is reported correct = True

## Requirement Transformation Example – Requirement Name

**Type -**

### Original Textual Requirement Statement

After applying power the system shall be in the operational state within 30 seconds.

### Transformed Formal Requirement Statement

**Subject** = System Property =

**Pre conditions**

1. Event –
2. Subject Property -

**Constraint** = Expression:

**Constraint Evaluation:**

# Resources and References

1. SEBoK
2. 15288
3. INCOSE SE Handbook
4. INCOSE Guide for Writing Requirements – 2015
5. PBR Working Group Material
	1. Fundamental Requirement Concepts\_yb.pptx 8/21/15 - Rick
	2. Needs Hierarchy 04.docx commented by Rick 5/14/2016
	3. Relaxing\_Constraints\_Rationale-rs-20150430 - Original by Manas
	4. Requirements management within a full model based engineering approach, paper from Yves
	5. SysML Issue-precise expression of requirements-sf-a-2.doc
6. SE Use Case Workflows – Requirements analysis activity, definitions, etc.
	1. The material that is available includes material from the SEBoK, 15288 and INCOSE Handbook, but it may not include all the material in these references.
7. UML 4SE RFP – Requirements, see Addendum A

# Open Questions

7/21/2016 – Final merging Sandy’s and John’s diagrams

1. Change multiplicity from composition relationship to Verification Component from 1 to 1..\*
2. Reduced number of notes
	1. Incorporated some of the notes into the concept definitions. See Notes on bottom right.

7/15/2016

1. Pre-condition – Can also be an event. “The stopping distance starts at the moment the brake pedal is touched and depressed.
2. Is there a smaller set of elements types that the subject can be
	1. A reference to a Property of a block.

7/12/2016 Discuss differences between Sandy’s and John’s diagram

1. Sandy’s doesn’t decompose a formal requirement and identify what model element each references
2. In Sandy’s is the traditional text requirement still shown?
	1. Is it Text Expression? If so then it would include the Constraining Element
3. In Sandy’s diagram preconditions are not explicitly called out, they are included in the Constraint Boundary.
4. Distinction between – One, probably the comparison, is missing in Sandy’s drawing
	1. Requirement Statement (Constraint Boundary + Constraint Expression)
	2. The comparison used to verify requirement, which may include a design safety margin between the measured and specification value
	3. The method used to evaluate the design (Design Expression). This could be taking a measurement, a live test, or a calculation (demonstration)

# Addendum A

## UML 4SE RFP Requirements

### Requirement

#### Requirement specification

UML for SE shall provide the capability to model requirements associated with the desired capabilities, properties, behavior, and/or structure of a system, including the following types of requirements:

1. Operational
2. Functional
3. Interface (inputs and outputs, ports, etc.)
4. Performance
5. Activation/deactivation
6. Storage
7. Physical
8. Design constraint or resource constraint
9. Specialized (i.e. safety, reliability, maintainability, usability, security, cost, other life cycle requirements, etc.)
10. Measure of effectiveness (MOE)

Note 1: Requirements should include values and associated tolerances, where applicable.

Note 2: A stakeholder need, which represents a desired capability, is typically expressed as a high-level requirement, which may be further defined in terms of operational, functional, performance and other requirement types. The representation of high-level capabilities may include use cases, or other abstract models, augmented with text.

Note 3: The representation of system requirements is intended to address the complete life cycle process, from concept through disposal of a system, and as such may include requirements for the enabling systems (i.e. the production system, support system, etc), as well as the operational system.

#### Requirement properties

UML for SE shall provide the capability to associate properties to a requirement.

Note: The properties may include a reference to requirement criticality or weighting, level of uncertainty, risk, verification status, and/or other defined model elements. These may include elements of related models, such as the risk from a risk model.

#### Requirement relationships

UML for SE shall provide the capability to associate a requirement to one or more model elements, which include associations between:

1. Derived requirements and their source requirements (trace)
2. Requirements and the model elements that realize and/or implement the requirements

Note: This includes the allocation of requirements to components.

1. Requirements and goals of a system by hierarchical decomposition into lower level requirements and sub-goals

Note: This form of analysis is often used to identify high-level requirements before any system-level modeling takes place.

#### Problem

UML for SE shall provide the capability to model a deficiency, limitation, or failure of one or more model elements to satisfy a requirement or need, or other undesired outcome.

#### Problem association

UML for SE shall provide the capability to associate a problem with one or more model elements.

Note 1: A problem can be associated with the behavior, structure, and/or properties of a system or element at any level of the hierarchy.

Note 2: A problem can be associated either with the as-is system, which has the problem, or the to-be system, which is intended to correct the problem.

#### Problem cause

UML for SE shall provide the capability to model a relationship between a problem and its source problems (i.e. cause).

Note: This can be used to represent cause-effect relationships that are often depicted in fish-bone diagrams, failure modes and effects analysis, or fault tree analysis.

### Verification

#### Verification Process

UML for SE shall provide the capability to model the verification of a system, which is a process used to demonstrate the following:

1. The system requirements have been properly allocated to the system components, such that the system requirements are satisfied if the components satisfy their requirements.
2. The implemented/realized system satisfies its requirements.
3. The requirements have been specified correctly to satisfy the higher-level needs (i.e. validation).

Note 1: Verification methods include inspection, analysis, demonstration, test, or similarity.

Note 2:Validation methods may include focus groups, market testing, market surveys, prototyping, field demonstrations, and other elicitation methods.

#### Test case

UML for SE shall provide the capability to model the input stimulus, expected output, and associated test criteria that verify that the system satisfies its requirements or needs.

Note 1: The test case can be a test scenario, which replicates the behavior of the external environment interacting with the system, to demonstrate that the system satisfies its functional, interface, and performance requirements. Alternatively, the test case can be a measurement of a physical characteristic, or an analysis that demonstrates that the system satisfies its requirements.

Note 2: Test criteria may include non-functional aspects, such as performance, and other requirement types indicated in 6.5.4.1.

Note 3: Test cases may be grouped into test runs to accomplish a specific test objective. A single test case may appear in more than one test run. Test cases are sometimes sequenced in a test run, reflecting dependencies between test cases. Failure of a test case in a sequence may result in the remainder of the test run not being executed.

#### Verification result

UML for SE shall provide the capability to specify the outcome from executing one or more test cases or test runs.

#### Requirement verification

UML for SE shall provide the capability to model the comparison between a requirement and the verification results.

Note: The comparison may yield a result of pass, fail, or not executed.

#### Verification procedure

UML for SE shall provide the capability to model the functions needed to support execution of a test case or test run.

Note: This can include the functions to generate an input stimulus and monitor an output response.

#### Verification system

UML for SE shall provide the capability to model the system that implements the verification procedures.

Note: The verification system can include test hardware and software, such as simulators and measuring devices, test facilities, and test operators (users).