



**OMG Systems Modeling Language TM Version 2**

**(SysML® v2)**

**Requirements Support Document**

OMG Document: SysEng/2017-11-01

|  |  |  |
| --- | --- | --- |
| **Release Name** | **Date** | **Description** |
| Dec Meeting Update | 01 Nov 2017 | Update to support SysML v2 RFP and SysML v2 API and Services RFP to be voted on at December 2017 OMG meeting |
| OMG Review | 23 Sept 2017 | Initial OMG Review |
| Review Draft | 3 July 2017 | For review by industry |
| Initial Draft | 4 June 2017 | For review within RFP Working Group |

Table of Contents

[1 Introduction 5](#_Toc497318668)

[1.1 Document Purpose 5](#_Toc497318669)

[1.2 Background 5](#_Toc497318670)

[1.3 Organization of this Document 8](#_Toc497318671)

[1.4 Summary Changes from SysML v1 9](#_Toc497318672)

[2 System Modeling Environment (SME) Overview 10](#_Toc497318673)

[2.1 Role of System Modeling Environment in a Model-Based Engineering Environment 10](#_Toc497318674)

[2.2 SME Capabilities and Effectiveness Measures in Support of MBSE 11](#_Toc497318675)

[2.3 SME Architecture 12](#_Toc497318676)

[3 Specific Requirements on Proposals 15](#_Toc497318677)

[3.1 Mandatory Language Requirements 15](#_Toc497318678)

[3.1.1 Language Architecture and Formalism 15](#_Toc497318679)

[3.1.2 Data Model 26](#_Toc497318680)

[3.1.3 Reference Model and Model Libraries 96](#_Toc497318681)

[3.1.4 Language Conformance 98](#_Toc497318682)

[3.1.5 Other 99](#_Toc497318683)

[3.2 Mandatory API and Services Requirements 102](#_Toc497318684)

[3.2.1 API 102](#_Toc497318685)

[3.2.2 Services 105](#_Toc497318686)

[3.2.3 API and Services Conformance 107](#_Toc497318687)

[3.3 Non-mandatory API and Services Features 108](#_Toc497318688)

[3.3.1 Model Construction Services 108](#_Toc497318689)

[3.3.2 Model Visualization Services 110](#_Toc497318690)

[3.3.3 Model Analysis Services 114](#_Toc497318691)

[3.3.4 Model Management Services 116](#_Toc497318692)

[3.3.5 Workflow & Collaboration Services 121](#_Toc497318693)

[3.3.6 Interoperability Services 122](#_Toc497318694)

[3.3.7 Extension Services 123](#_Toc497318695)

[Appendix A References & Glossary Specific to this RFP 124](#_Toc497318696)

[A.1 References Specific to this RFP 124](#_Toc497318697)

[A.2.1 Bibliographic Citation List 124](#_Toc497318698)

[A.2.2 OMG Standards List 127](#_Toc497318699)

[A.2.3 Other Standards List 129](#_Toc497318700)

[A.2 Glossary Specific to this RFP 130](#_Toc497318701)

[Appendix B General Reference and Glossary 160](#_Toc497318702)

[B.1 General References 160](#_Toc497318703)

[B.2 General Glossary 163](#_Toc497318704)

Table of Figures

[Figure 1.1. SysML v2 Approach Diagram 7](#_Toc497318705)

[Figure 2.1. The SME is part of the broader MBE environment that enables systems engineers to perform MBSE 10](#_Toc497318706)

[Figure 2.2. The System Model (Source: A Practical Guide to SysML, 3rd Edition, Friedenthal, Moore, and Steiner 2014) 11](#_Toc497318707)

[Figure 2.3. System Modeling Environment-Logical Architecture 13](#_Toc497318708)

[Figure 2.4. System Modeling Environment Layered Architecture 15](#_Toc497318709)

[Figure 3.1. SysML v2 Models and Model Libraries 16](#_Toc497318710)

[Figure 3.2. SysML v2 Metamodel and Profile 17](#_Toc497318711)

[Figure 3.3. SysML v2 Language Metamodel Specification 18](#_Toc497318712)

[Figure 3.4. SysML v2 Model Interchange 19](#_Toc497318713)

[Figure 3.5. Language Formalism and Uniform Interpretation 20](#_Toc497318714)

[Figure 3.6. Core SEBoK Concepts (Extract from draft SECM-2015 Industry Reference. Used with permission) 27](#_Toc497318715)

[Figure 3.7. Organization of SysML v2 Modeling Concepts 28](#_Toc497318716)

[Figure 3.8. Root Concepts 29](#_Toc497318717)

[Figure 3.9. Value Type and Definition Element 30](#_Toc497318718)

[Figure 3.10. Component Definition and Item Definition 31](#_Toc497318719)

[Figure 3.11. Component Definition and Item Definition-Elaborated 32](#_Toc497318720)

[Figure 3.12. Function Definition and Constraint Definition 33](#_Toc497318721)

[Figure 3.13. State Machine 34](#_Toc497318722)

[Figure 3.14. Interface Definition 35](#_Toc497318723)

[Figure 3.15. Configuration Element and Individual Element 36](#_Toc497318724)

[Figure 3.16. State and Time History 37](#_Toc497318725)

[Figure 3.17. Requirements 38](#_Toc497318726)

[Figure 3.18. Analysis and Verification 39](#_Toc497318727)

[Figure 3.19. Decision and Variant 40](#_Toc497318728)

[Figure 3.20. View and Viewpoint 41](#_Toc497318729)

[Figure 3.21. Cross-cutting Model Element Concepts 42](#_Toc497318730)

[Figure 3.22. Cross-cutting Relationship Concepts 43](#_Toc497318731)

[Figure 3.23. Variant Modeling Concepts 44](#_Toc497318732)

[Figure 3.24. Specifying an unambiguous system design configuration 58](#_Toc497318733)

[Figure 3.25. Sample of Structure Modeling Concepts 59](#_Toc497318734)

[Figure 3.26. SysML v2 Interface Concept 65](#_Toc497318735)

[Figure 3.27. Interface Concepts 65](#_Toc497318736)

[Figure 3.28. Top Level Behavior - Take Picture 72](#_Toc497318737)

[Figure 3.29. Camera Context with Take Picture Functions 73](#_Toc497318738)

[Figure 3.30. Camera Context with State Machine 74](#_Toc497318739)

[Figure 3.31. Requirement Groups 81](#_Toc497318740)

[Figure 3.32. Example Weight Requirement 82](#_Toc497318741)

[Figure 3.33. Requirement Concepts 83](#_Toc497318742)

[Figure 3.34. Verification Concepts 89](#_Toc497318743)

[Figure 3.35. SysML v2 Analysis Concepts 92](#_Toc497318744)

[Figure 3.36. Vehicle Weight & Acceleration Analysis 93](#_Toc497318745)

[Figure 3.37. System Model Interoperability Concepts (Source: Alex Reichwein) 100](#_Toc497318746)

[Figure 3.38. SysML v2 API Specification Approach 103](#_Toc497318747)

[Figure 3.39. Model Visualization Concepts 111](#_Toc497318748)

[Figure 3.40.  Model Visualization Concepts (Source: C. Schreiber, J. Feingold, and M. Sarrel) 112](#_Toc497318749)

[Figure 3.41. Integrated System Model (ISM) Lifecycle Management Concepts (Source: SysML v2 RFP Model Management WG) 117](#_Toc497318750)

[Figure 3.42. Workflow & Collaboration Concepts 121](#_Toc497318751)

# Introduction

## Document Purpose

The content of this document was developed by the Object Management Group (OMG) SysML v2 RFP Working Group. This document provides the draft requirements and supporting information for the SysML v2 Request for Proposals (RFP) that will be issued by the OMG that include the requirements for the SysML v2 RFP and the SysML v2 API and Services RFP. The RFPs define the requirements that the specification for the next generation System Modeling Language (SysML®) called SysML v2 must satisfy. The SysML v2 Specification and SysML v2 API and Services Specification, in turn, will be implemented by tool vendors in their modeling tools to enable practitioners to perform their model-based systems engineering (MBSE) activities. The requirements are included in tables at the end of the subsections in Section 3 of this document, and are also captured in a separate excel table with additional supporting information.

Reviewers of this document are encouraged to read Sections 1 and 2 to understand the context for these requirements, and refer to the Glossary in Section 4 for clarification of terms.

## Background

The following extract is from the two previous INCOSE INSIGHT articles that provide background on this effort.

* INCOSE INSIGHT Article in August 2015 entitled "**Evolving SysML and the System Modeling Environment to Support MBSE** (Friedenthal and Burkhart 2015)"
* INCOSE INSIGHT article in December 2016 entitled "**Evolving SysML and the System Modeling Environment to Support MBSE-Part 2** (Friedenthal 2016)"

**The need for a standard systems modeling language.** The transition to a MBSE approach is essential for systems engineering to meet the demands of increasing system complexity, productivity and quality, and shorter design cycles. Many other engineering disciplines, such as mechanical, electrical, and controls engineering, utilize models as an integral part of their practice. Models have long been important for systems engineering as well to support systems analysis and design, but MBSE emphasizes the need to create a coherent model of the system architecture that helps integrate other aspects of the design, including electrical, mechanical, and software.

The system model provides a shared view of the system that can enhance communication and coordination across the system development lifecycle. This model represents an authoritative source of information that is maintained to ensure consistency and traceability between requirements, design, analysis, and verification. The model-based approach contrasts with the traditional document-based approach in which information is captured independently in many different documents using common applications such as Word, Visio, Excel, and PowerPoint. To take full advantage of a model-based approach, the system model must be maintained as part of the technical baseline, and integrated with other engineering models and tools.

The capability to express system concepts in the form of models can result in quality improvements by reducing downstream design errors, and in productivity improvements through reuse of models throughout the lifecycle and across projects. Systems engineers realize other benefits, such as the ability to automate tasks like change impact analysis, and auto-generation of reports and documentation with increased confidence that the information is valid, complete, and consistent.

A systems modeling language enables systems engineers to express fundamental concepts about the system such as system composition, interconnection and interfaces, functional and state-based behavior, parametric aspects, and traceability relationships between requirements, design, analysis, and verification. The modeling language is an essential capability to specify and architect increasingly complex systems. A standard systems modeling language can help overcome the informational "Tower of Babel" by providing a means to express these concepts in a standard and precise way that enables communications between engineers and tools.

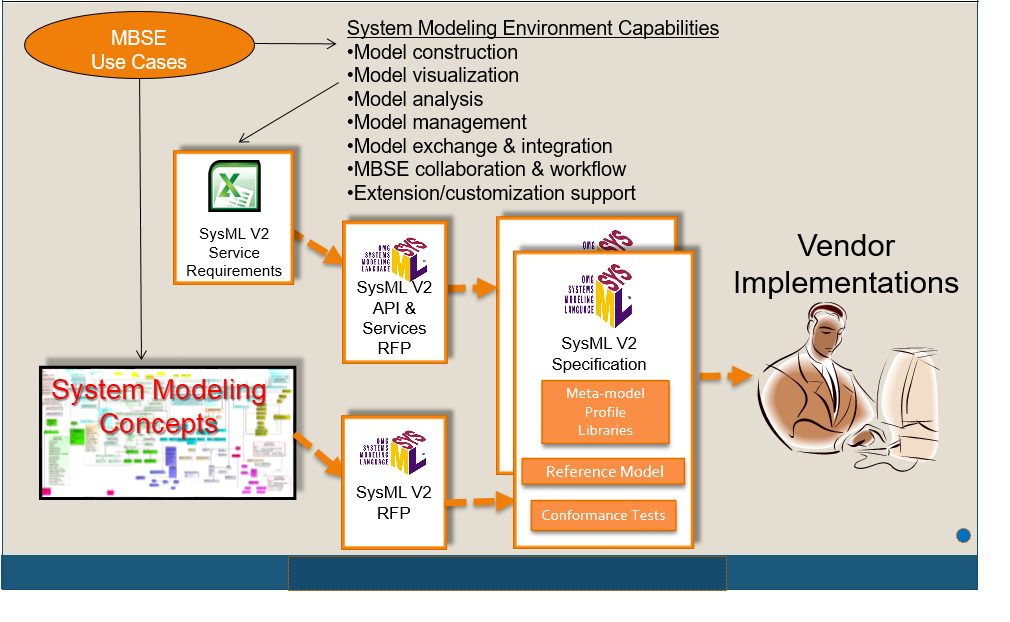
**SysML background.** The Object Management Group (OMG) adopted the OMG Systems Modeling Language TM ([www.omgsysml.org](http://www.omgsysml.org)) specification in 2006 and version 1 was available in 2007. The SysML specification resulted from a collaborative effort between INCOSE, the OMG, and the ISO STEP AP-233 Working Group (WG) to develop the requirements for the language, and then develop the system modeling language solution in response to those requirements.

SysML is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities. The language provides graphical representations with a semantic foundation for modeling system requirements, behavior, structure, and constraints.

Since its adoption, SysML enabled broad recognition and increased adoption of model-based systems engineering practices across industry. Systems engineers, tool vendors, and academia have learned much from this experience, including both the strengths and weaknesses of SysML as a language, and the benefits and challenges of adopting and applying MBSE with SysML.

Based on these experiences, the OMG SysML v2 RFP Working Group was initiated on July 23, 2016 at the OMG meeting in Orlando, Florida and began working on the requirements for SysML v2. This followed an approximately year-long effort to establish a baseline concept for a System Modeling Environment (SME). The SME is the environment that systems engineers interact with to perform model-based systems engineering activities, and the MBSE use cases and the SME concept are used to help derive requirements for SysML v2 as shown in the figure below.

Figure 1.1. SysML v2 Approach Diagram



The initial high-level requirements for the SME are documented in the August 2015 edition of the INCOSE INSIGHT. The article is entitled 'Evolving SysML and the System Modeling Environment to Support MBSE' and defines 7 capabilities, 8 measures of effectiveness (moe's), and 11 driving requirements for the SME to support the specification, design, analysis, and verification of systems. A second article was published in the December, 2016 edition of the INCOSE INSIGHT entitled 'Evolving SysML and the System Modeling Environment to Support MBSE - Part 2'. This article summarizes the baseline SME Concept in response to the requirements in the earlier article. Both articles are available in final draft form under the Articles section of the SysML v2 RFP Working Group Wiki at:

<http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:sysml_assessment_and_roadmap_working_group>

The requirements contained in this document are incorporated into two separate SysML v2 RFPs, and are used to define the requirements for two SysML v2 specifications, which will then be implemented by tool vendors. The SysML v2 RFP specifies the requirements on the language, and the SysML v2 API and Services RFP specifies the requirements for the standard API and services.

The overarching objectives for SysML v2 are to facilitate increased adoption and effectiveness of MBSE over SysML v1. This is accomplished by enhancing support for MBSE with improved precision, expressiveness, consistency, and integration among the language concepts, and improved interoperability with other engineering models and tools. SysML is also intended to facilitate improved usability by model developers and consumers.

The requirements in the SysML v2 RFP reflects lessons learned from experiences with SysML v1. The SysML v2 RFP includes refined requirements to represent the SysML v1 concepts (e.g., structure, behavior, parametrics, requirements), and requirements for additional system modeling concepts. The SysML v2 API and Services RFP also includes requirements for a standard API and services that support model construction, model visualization, model analysis, model management, and workflow and collaboration. Service requirements were not included in the RFP for SysML v1.

Submission teams will develop the SysML v2 Specification and the SysML v2 API and Services Specification in response to the requirements in both RFPs. Tool vendors will then implement these specifications.

The modeling concepts and associated requirements will be satisfied in the SysML v2 Specification, which will specify both a SysML metamodel and a profile of UML. A vendor can choose to implement the metamodel or profile or both. The combination of a metamodel and a profile enable a broader range of vendor implementations. The metamodel supports implementation of the system concepts without some of the constraints imposed by UML, while the profile supports implementation of the system concepts that is more closely aligned with SysML v1 implementations.  The SysML v2 Specification is also intended to provide foundational concepts and extension mechanisms to facilitate integrations with other domain specific languages such as safety, reliability, security, and testing, and to maintain continuity with other related OMG modeling standards such as UAF and UML.

The API and service requirements will be satisfied in the SysML v2 API and Services Specification. The standard API facilitates interoperability by enabling external tools, plug-ins, and user interfaces to access the system model using standard service requests. The SysML v2 API and Services Specification is intended to be implemented by multiple types of tools that are part of a System Modeling Environment.

## Organization of this Document

This document is organized into the following sections.

1. **Introduction.** This section includes the purpose, background, and organization of this document.
2. **System Modeling Environment (SME) Overview.** This section provides an overview of the System Modeling Environment.
3. **Specific Requirements on Proposal.** This section is organized into subsections for each logical grouping of requirements. Each subsection includes relevant concepts that reflect systems modeling needs to motivate the requirements, and a summary of key issues with SysML v1 in terms of its support for these concepts. In some cases, motivating examples are provided to further illustrate the concept and/or highlight SysML v1 issues. The requirements are then included in tables at the end of each subsection.
4. **References & Glossary Specific to Document.** This section includes a glossary of terms that are used to represent the concepts and requirements in this document, related standards, and a list of references used in this document.
5. **General Reference and Glossary.** This section includes a glossary of more general terms that apply to OMG standards, and references to other more general OMG standards that may have been used or are intended to be used in support of these requirements.

## Summary Changes from SysML v1

The API and Services requirements are additional scope for SysML v2 over SysML v1. The requirements in this RFP will include mandatory requirements for a small number of mandatory service requirements that include a query service and bidirectional service. The remaining service requirements are optional requirements, which of the submission team can choose which of these service requirements to specify.

The language requirements include the scope of the original UML for SE RFP that was used to specify SysML v1. However, many of the requirements are refined based on lessons learned. A summary of SysML v2 concepts are contained in the Data Model section of this document.

The original requirement for SysML v1 was for a profile only, whereas in SysML v2, we are requiring the specification of both a profile and a metamodel. In addition, there is emphasis on establishing a more precise semantics. There is also a requirement to ensure more effective tool-neutral model interchange beyond what was provided with the XMI.

The requirements are also captured in separate spreadsheets on the [SysML v2 Requirements Review Page](http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:sysml_v2_requirements_review_page), which includes columns with additional information that is not included in this document. In particular, the spreadsheet includes columns that contain the following information for each requirement.

* The extent to which a SysML v1 construct satisfies the requirement (none, partial, full)
* The corresponding SysML v1 construct that addresses the requirement (partial or full only)

This spreadsheet can be used to further evaluate how SysML v2 compares with SysML v1. It is also expected that a similar spreadsheet will be used to show traceability between the SysML v2 constructs and the SysML v2 requirements.

# System Modeling Environment (SME) Overview

## Role of System Modeling Environment in a Model-Based Engineering Environment

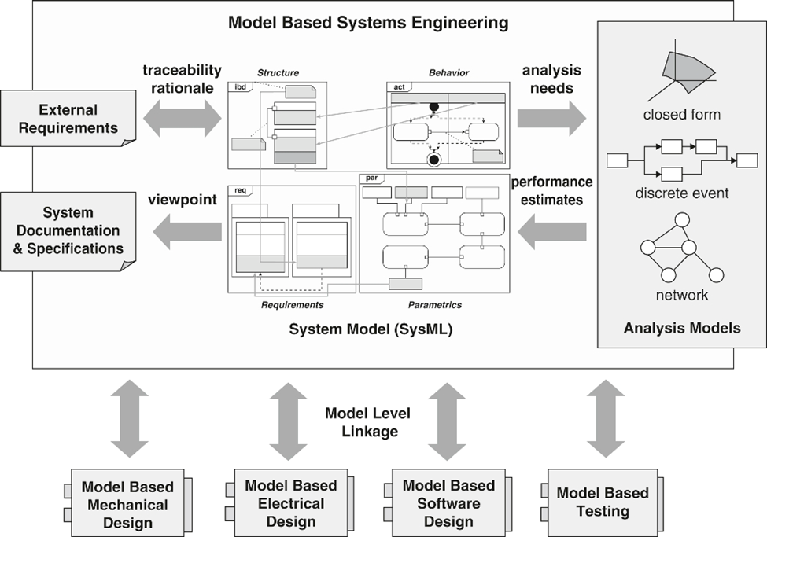
In the figure below, each of the disciplines contributes to the development of the technical baseline of the system as part of a Model-Based Engineering (MBE) approach. Each member of the development team use their discipline-specific models to capture and analyze different aspects of the design. The MBE environment is the overall set of tools that all disciplines use to implement a MBE approach. The System Modeling Environment (SME) is the part of the overall MBE environment that systems engineers use to perform MBSE and interact with other members of the development team. (Note: The focus for the SME is limited to SysML v2 modeling, but it is recognized that the SME can include many other kinds of system models and data.)

Figure 2.1. The SME is part of the broader MBE environment that enables systems engineers to perform MBSE



The system model provides an overall description of the system that facilitates integration with the other engineering models and tools as shown in the figure below.

Figure 2.2. The System Model (Source: A Practical Guide to SysML, 3rd Edition, Friedenthal, Moore, and Steiner 2014)



The systems engineer and others use the SME System Modeling Environment to perform model-based systems engineering as part of an overall development process to flow requirements from the mission/enterprise level to systems, subsystems, and components, and verify the components, subsystems, system, and mission requirements are satisfied. This process continues throughout the development lifecycle, with the aim of delivering systems and products that meet the stakeholder needs. Typical MBSE use cases were defined as part of the requirements development, including a detailed change management scenario that are available at <http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:use_case_working_group>.

## SME Capabilities and Effectiveness Measures in Support of MBSE

The definition of the initial SME capabilities and driving requirements are identified in the August 2015 INCOSE INSIGHT article (Friedenthal and Burkhart). This article defines 7 capabilities, 8 measures of effectiveness (moe's), and 11 driving requirements the SME should support. These capabilities enable the systems engineer to perform MBSE as part of a broader model-based engineering effort, and include:

* Model construction
* Model visualization
* Model analysis
* Model management
* Model interoperability
* MBSE workflow and collaboration
* Extension/customization support

The effectiveness measures for the SME are used to evaluate how effectively the SME supports the above capabilities. Some of these measures are difficult to quantify, and will require further refinement. The set of effectiveness measures include:

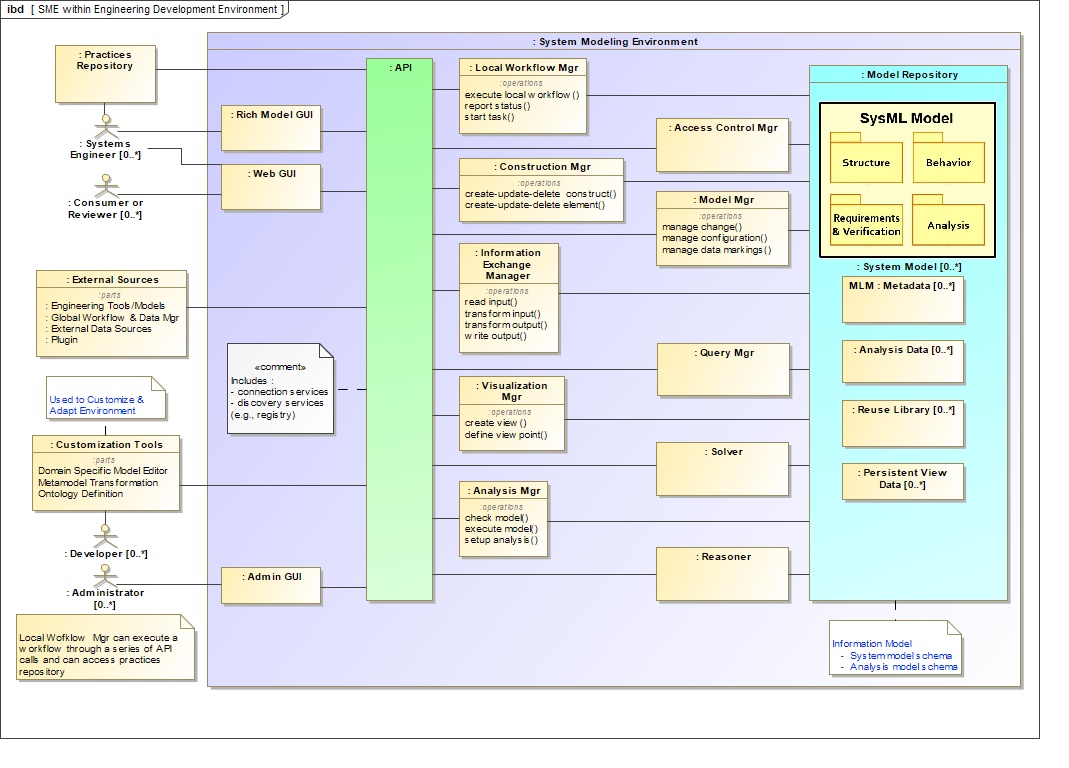
* **Expressiveness:** Ability to express the system concepts needed to describe systems
* **Precision:** Ability to represent the concepts in a concise way that enables unambiguous human and computer interpretation
* **Consistency/integrity**: Level of integration of concepts to ensure consistency and integrity of the language
* **Presentation/communication**: Ability to effectively support communications with diverse stakeholders that includes presentation and generation of technical baseline information related to specifications, design, analysis and verification of the system and their relationships
* **Model construction:** Ability to efficiently and intuitively construct models
* **Interoperability:** Ability to exchange data with other SysML models, other engineering models and tools, and structured data sources.
* **Manageable:** Ability to efficiently manage change to models and provide data protection services
* **Secure:** Ability to protect all relevant data from threats
* **Usability:** Ability for stakeholders to efficiently and intuitively create, maintain, interpret, and use the model
* **Adaptability/Customizability:** Ability to extend models to support domain-specific needs
* **Scalability:** Ability to scale from small to medium to large models

## SME Architecture

The SME must provide the functionality needed to enable systems engineers and others to evolve the system model throughout the life cycle. The diagram below is a view of the logical architecture of the SME. The model repository contains the data about the system, including the system model, analysis data, metadata, and reuse libraries. This repository is shown as a single logical repository, but may be federated across multiple physical repositories.

The diagram shows the systems engineer using a rich model graphical user interface that provides the full functionality of the SME to create, maintain, and use the system model and other data in the model repository. The systems engineer and other disciplines can also interact with the SME using a web interface that provides the functionality needed to use and/or review the system model and other data. The user interface can present different views of the model to address different stakeholder needs and concerns. For example, a power subsystem engineer may view the power interfaces, and a mechanical engineer may view the system breakdown and mass allocation.

Figure 2.3. System Modeling Environment-Logical Architecture



The SME also enables other engineering tools and models and plug-ins to access the repository. The graphical user interfaces and the external tools access the model repository by requesting standard services through an application program interface (API). The API provides the interface to the model repository and supports the SME modeling capabilities described above that include model construction, model visualization, model analysis, model management, and workflow and collaboration services.

Applications external to the SME that provide global workflow and data management, can interact with the SME and other discipline-specific environments to control the configuration and manage change across the overall MBE environment, and across the system life cycle. This includes synchronization tasks via notifications to and from users of the SME as the engineering work products change state. The combination of the PLM capability and the integrated system model facilitate a collaborative engineering environment.

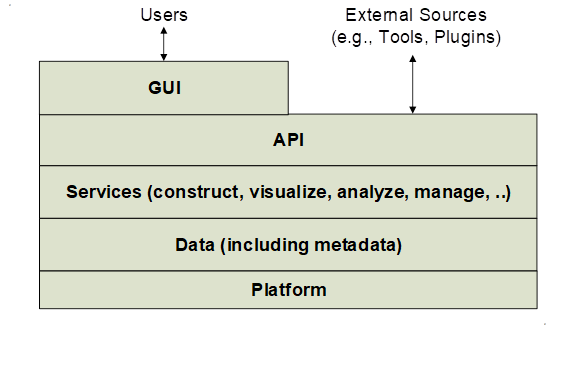
In addition, the logical architecture includes Customization Tools to further extend and customize the SME to support different domain and program needs, and ensure the customized environment continues to be interoperable with the rest of the MBE environment.

Finally, there is a practices repository that stores the systems engineering and modeling practices that are implemented by the users of the SME. The local workflow manager is intended to facilitate the systems engineer and others to perform these practices.

The SME can be implemented by multiple vendors. A systems engineer should be able to request a service of the SME through the standard API to access the model repository without regard for where the model data resides or what SME implementation provides the access to the model. The API also enables requests to be made from the SME to access data in other engineering tools that conform to a standard interface.

The logical components from the SME Logical Architecture are allocated to different layers of the SME architecture as shown in the SME Layered Architecture in the diagram below. The platform layer provides the basic computing infrastructure. The data layer stores the model data in the repository. The services layer contains a set of applications that implement the services needed to support the SME capabilities (for example: model construction, visualization, analysis, management). An application program interface (API) provides the interface to request these services. The graphical user interface (GUI) provides the interface for the users. The GUI and other tools request the services through the API. The API also enables the Customization Tools to modify and extend the data model and other aspects of the environment.

Figure 2.4. System Modeling Environment Layered Architecture



# Specific Requirements on Proposals

## Mandatory Language Requirements

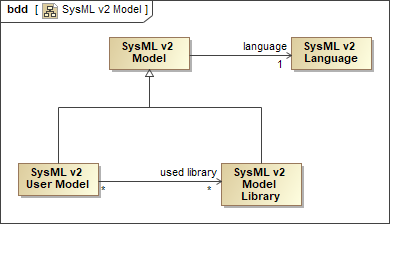
The mandatory requirements for SysML v2 are intended to support the MBSE use cases and SME concept as described above. This section is organized into logical groupings of requirements. Each subsection contains key concepts that reflect the system modeling needs and issues with SysML v1, and some motivating examples to highlight the concepts and/or issues. The set of requirements applicable to each section is included at the end of each subsection.

### Language Architecture and Formalism

#### Language Architecture

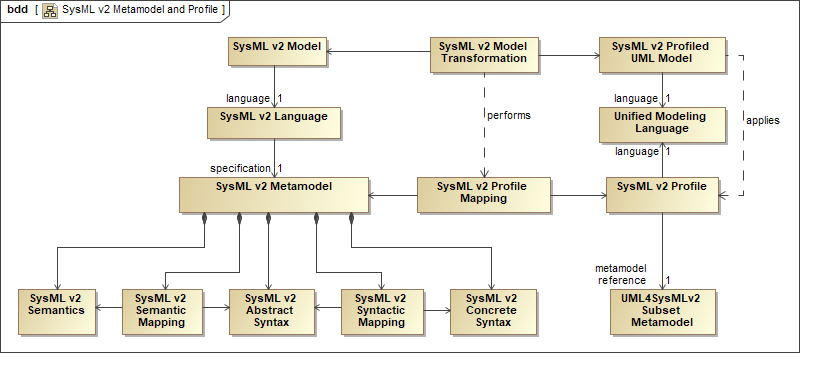
SysML v2 is a modeling language used to represent SysML v2 models. As shown in the figure below, SysML v2 models include both models created by SysML v2 end users and model libraries containing reusable modeling components that may be used in the creation of user models. In particular, some of the SysML v2 language requirements may be implemented in the SysML v2 specification as user-level model libraries.

Figure 3.1. SysML v2 Models and Model Libraries



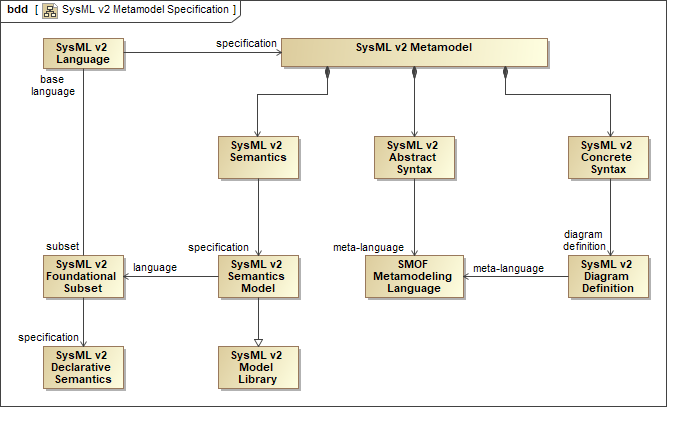
The SysML v2 language is specified using a SysML v2 metamodel that defines the language's semantics, abstract syntax and concrete syntax, and the relationships between them, as shown in the figure below. The language also provides mechanisms to support further customization to reflect domain specific concepts. In addition, the SysML v2 metamodel is mapped to a SysML v2 profile. This allows a SysML v2 model to also be represented as an extension of a UML model, using the profile to adapt UML syntax and semantics to those of SysML v2. The combination of a metamodel and a profile enable a broader range of vendor implementations. The metamodel supports implementation of the system concepts without some of the constraints imposed by UML, while the profile supports implementation of the system concepts in a way that is more closely aligned with SysML v1 implementations.

Figure 3.2. SysML v2 Metamodel and Profile



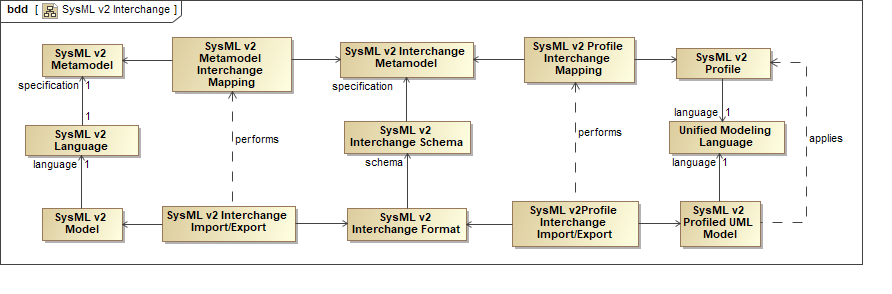
As shown in the Figure below, the SysML v2 abstract syntax may be specified using the OMG-standard Meta-Object Facility (MOF) or its extension, Semantic MOF (SMOF), which provides the basis for the proposed Metamodel Extension Facility (MEF). The SysML v2 concrete syntax may be specified using common Backus-Naur Form (BNF) productions for textual notations and OMG-standard MOF-based Diagram Definition for graphical notations. These MOF-based standards, however, only provide support for structural modeling of the (concrete and abstract) syntactic constructs of the language. The SysML v2 semantics can then be specified using an approach similar to those used in the OMG precise semantics standards for UML, in which the SysML v2 language itself is used to model its own semantics. This circularity of specifying SysML v2 semantics in SysML v2 itself stops at the root level of a foundational subset of the language for which the semantics are specified using a separate declarative formalism (as described further in the section on Formalism below).

Figure 3.3. SysML v2 Language Metamodel Specification



Having the SysML v2 metamodel aligned with the SysML v2 profile also allows for the specification of a SysML v2 format for interchanging models between tools, usable by both metamodel and profile-based tools. As shown in the figure below, this is done by defining mappings from both the SysML v2 metamodel and the SysML v2 profile to a common SysML v2 interchange metamodel, consistent with the mapping between the SysML v2 metamodel and the SysML v2 profile. That is, a SysML v2 model based on the metamodel and profile are related by the SysML v2 metamodel-to-profile mapping and represented in the same way for the purposes of model interchange. This allows a SysML v2 model exported from a metamodel-based tool to be imported into a profile-based tool, and vice versa. Furthermore, the SysML v2 interchange format enables interchange of both the abstract syntax representation of a SysML v2 model and the concrete syntax representation of views of that model.

Figure 3.4. SysML v2 Model Interchange



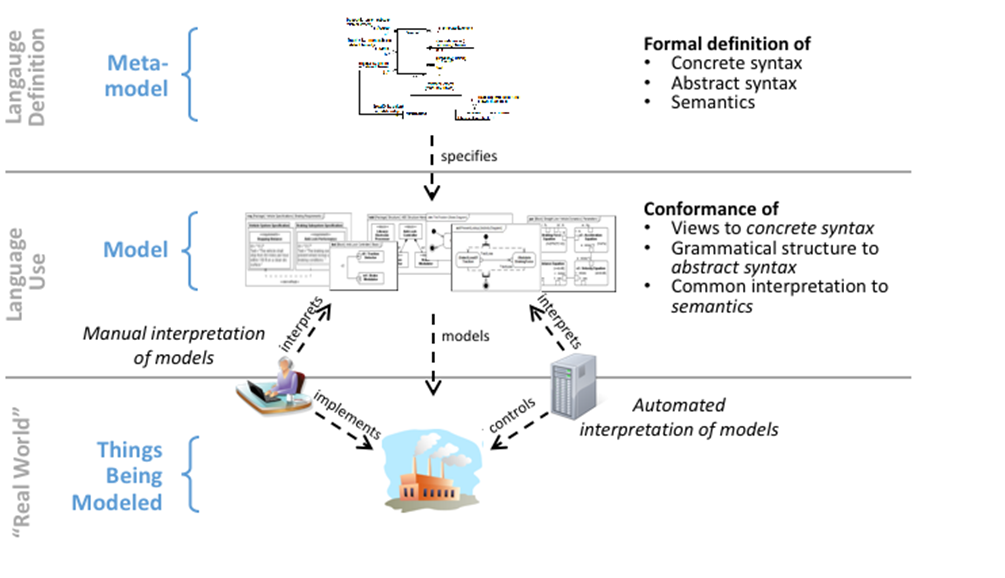
#### Formalism

A formalism is used to specify the SysML v2 language. In mathematics and logic, formalism has to do with how something is structured and expressed, as opposed to the actual content of what is being expressed. Formalism aims to express content in a well-defined form, such that this expression can be given a uniform interpretation. A formalism may also extend to rules for the consistent manipulation of the form of expression, such as the ability to construct formal proofs using deduction rules based on given axioms and to reason about the system being represented.

For SysML, the formalism defines how the language itself is specified in terms of its syntax and semantics, as opposed to what is in the language. This includes: the abstract syntax that specifies the grammar of the language, including the basic constructs of the language analogous to verbs and nouns, and the rules for constructing legal sentences (i.e., statements); the concrete syntax that specifies the symbols (textual, graphical and diagrammatic) that define how grammatical constructs in the language can be presented; and the semantics that specify the meaning of the constructs so that they can be interpreted in the domain that the model is intended to represent. The specification of how models are interchanged can also be considered part of the formalism. The rigorous specification of the abstract syntax, concrete syntax, semantics, and interchange format is intended to ensure the precision and integrity of the language.

The figure below shows the relationship between the language formalism and the things being modeled. The goal of a formal specification for SysML is to provide a uniform syntactic and semantic interpretation for the language. That is, a SysML model should be interpreted in a consistent way and subject to an objective evaluation as to whether it conforms to the SysML v2 Specification, whether this interpretation is done by a human that interprets a view of the model, or a machine that interprets the model.

Figure 3.5. Language Formalism and Uniform Interpretation



SysML v1 is specified using the formalism of UML 2 profiles. A UML profile is an extended subset of the UML metamodel, which is itself specified using the formalism provided by the Meta Object Facility (MOF). The SysML v1 specification defines the abstract syntax of SysML as a profile of UML that extends the subset of UML abstract syntax using stereotypes, extends the concrete syntax of UML diagrams, and adopts and adapts UML semantics as appropriate.

There are some significant limitations of the formalism used for SysML v1 that result in ambiguities of interpretation. For example, SysML v1 does not include a complete formal mapping between the concrete syntax and the abstract syntax, which can result in ambiguity in how a SysML diagram conforms to the rules of the grammar. In addition, the semantics of SysML v1 are often defined in English rather than a more precise formal representation, which can result in ambiguity of meaning.

In contrast, SysML v2 will have a more formal specification of its abstract syntax, concrete syntax and semantics, and the mappings between them. To maximize the flexibility of this specification, the required approach is to specify a small set of foundational concepts and their base semantics using a mathematical declarative semantics. Then, model libraries written in SysML itself, grounded in the base semantics, are used to further extend the concepts of the language and their associated semantics. These extensions will represent the core domain concepts for the SysML v2 language. SysML v2 is also intended to include additional user-level model libraries that extend these core concepts, and provide a mechanism to further customize the language.

The advantage of grounding SysML semantics in a declarative approach is that well-known techniques of mathematical logic can then be used to make formal deductions based on the assertions made in a model, in order to prove things that are true or not about the system or domain that is being modeled. Declarative semantics contrast with the operational semantics which specify how a model executes, such that the execution results are evaluated to determine how the system will behave. It is expected that the full semantics for SysML v2 will include both declarative and operational components.

As an example of how the semantics of SysML v2 could be built up from a declarative base, consider the case of the semantics of control nodes used in activity diagrams. Currently (in UML and, so, SysML v1), each type of control node such as a fork node, join node, decision node, or merge node is defined with its own unique semantics. In SysML v2, the general concept of a control node might be specified along with its base semantics. The specific semantics for fork, join, decision and merge nodes could then be specified in the core model library, specializing the base control node semantics. The language formalism would include rules for how this could be done in an unambiguous, rigorous way. A formal mathematically-based language does not have to be difficult to use. The usability of the language will be emphasized using graphical, textual and tabular notations appropriate for practicing system engineers.

#### Language Architecture and Formalism Requirements

The language architecture and formalism requirements are included in the table below. These requirements are about how the abstract syntax, concrete syntax, and semantics are to be specified for SysML v2, and include requirements related to extending the language and interchanging and transforming models.

**Table 3.1. Language Architecture and Formalism Requirements**

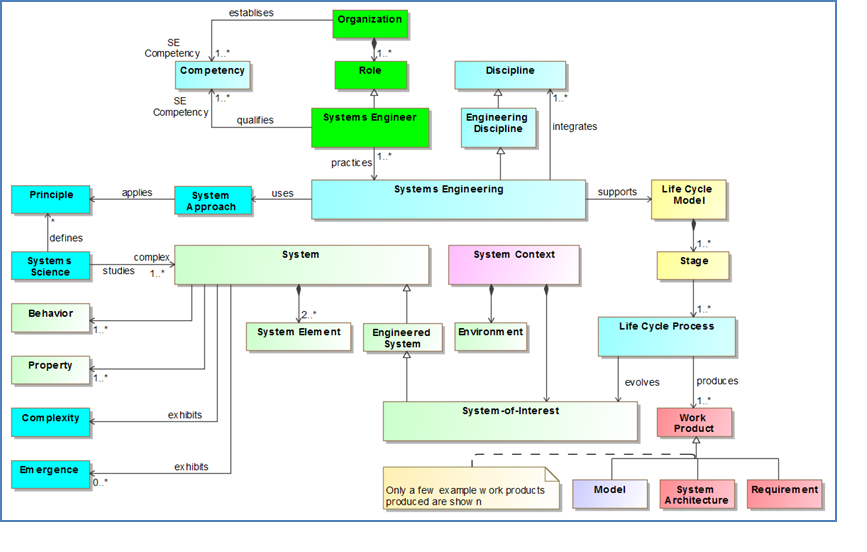
| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| LNG 1 | Language Architecture and Formalism Requirements Group | This group specifies how the language is structured and defined.  **Supporting information:** Some concepts may be implemented as user-level model libraries. |  |
| LNG 1.1 | Metamodel and Profile Group |  |  |
| LNG 1.1.1 | SysML Metamodel | SysML v2 shall be specified using a metamodel that includes abstract syntax, concrete syntax, semantics, and the relationships between them. |  |
| LNG 1.1.2 | Metamodel Specification | The SysML v2 metamodel shall be specified in MOF or SMOF.  **Supporting Information:** MOF is a subset of SMOF. SMOF provides support for the Metamodel Extension Facility (MEF). |  |
| LNG 1.1.3 | SysML Profile | SysML v2 shall be specified as a SysML v2 profile of UML that includes, as a minimum, the functional capabilities of the SysML v1.x profile, and a mapping to the SysML v2 metamodel.  **Supporting information:** Equivalent functional capability can be demonstrated by mapping the UML metaclasses and SysML stereotypes between SysML v2 and SysML v1. | SysML v1.x Profile |
| LNG 1.2 | Semantics Group |  |  |
| LNG 1.2.1 | Semantic Model Libraries | SysML v2 semantics shall be modeled with SysML v2 model libraries.  **Supporting Information:**   1. Simplifies the language when model libraries are used to extend the base declarative semantics without additional abstract syntax. 2. Enables SysML to be improved and extended more easily by changes and additions to model libraries, rather than always through abstract syntax. |  |
| LNG 1.2.2 | Declarative Semantics | SysML v2 models shall be grounded in a declarative semantics expressed using mathematical logic.  **Supporting Information:**  Semantics are defined formally to reduce ambiguity. Declarative semantics enable reasoning with mathematical proofs. This contrasts with operational semantics that requires execution in order to determine correctness.  The semantics provide the meaning to the concepts defined in the language, and enable the ability to reason about the entity being represented by the models. | Semantics of UML and SysML |
| LNG 1.2.3 | Reasoning Capability | SysML v2 shall provide a subset of its semantics that is complete and decidable.  **Supporting Information:** This enables the ability to reason about the entity being modeled by querying the model, and returning results that satisfy the specified set of constraints.  As an example, a query could return valid vehicle configurations that have a vehicle mass<2000kg AND vehicle has a sunroof. |  |
| LNG 1.3 | Abstract Syntax Group |  |  |
| LNG 1.3.1 | Syntax Specification | SysML v2 abstract and concrete syntax shall be specified using a subset of SMOF (including constraints on syntactic structure).  **Supporting Information:**  Expressing the syntax formally using a single consistent language which is more understandable to the user. |  |
| LNG 1.3.2 | View Independent Abstract Syntax | The SysML v2 abstract syntax representation of SysML v2 models shall be independent of all views of the models.  **Supporting Information:** Rationale  This is intended to define the concept independent of how it is presented. This enables a consistent representation of concepts with common semantics across a diverse range of views, including graphical, tabular, and other textual representations.  It also allows more consistent implementation of services on models (e.g., model checking, execution, and analysis) independent of specific views of those models. |  |
| LNG 1.4 | Concrete Syntax Group |  |  |
| LNG 1.4.1 | Concrete Syntax to Abstract Syntax Mapping | The SysML v2 concrete syntax representation of all views of a SysML model shall be separate from, and mapped to the abstract syntax representation of that model (including the ability to map to one or more images or snippets of images).  **Supporting Information:**  Enables views to provide unambiguous concrete representation of the abstract syntax of the model.  Enables views to be rendered in a consistent way across tools. | Diagram Definition |
| LNG 1.4.2 | Graphical and Textual Concrete Syntax | SysML v2 shall provide a standard graphical and human readable textual concrete syntax.  **Supporting information:** Graphical and textual concrete syntax representations can be used in combination to more efficiently and effectively present the model. Refer to Alf as an example of a textual notation. | Graphical syntax only |
| LNG 1.4.3 | Syntax Examples | All examples of model views in the SysML v2 specification shall include the concrete syntax of the view, and the mapping to the abstract syntax representation of the parts of the models being viewed.  **Supporting Information:**  Experience has shown that the mapping of examples to the concrete and abstract syntax is not always obvious. Making these mappings explicit helps clarify their formal specification. |  |
| LNG 1.5 | Extensibility Group |  |  |
| LNG 1.5.1 | Extension Mechanisms | SysML v2 syntax and semantics shall include mechanisms to subset and extend the language.  **Supporting Information:** This is essential to enable further customization of the language. SysML v1 includes a stereotype and profile mechanism to extend the language.  It should also enable extension of metadata, e.g. data owner, model protection data, model precision | Stereotype, Profile |
| LNG 1.5.2 | Extensibility Consistency | All SysML v2 extension mechanisms shall be applicable to SysML v2 syntax (concrete and abstract) and semantics, and be consistent with how these are specified in SysML v2.  **Supporting Information:**  The SysML v2 Specification includes syntax, semantics, and vocabulary, so extending the language requires all of these to be extensible. |  |
| LNG 1.6 | Model Interchange, Mapping, and Transformations Group |  |  |
| LNG 1.6.1 | Model Interchange | SysML v2 shall provide a tool-neutral format for unambiguously interchanging the abstract syntax representation of a model and the concrete syntax representation of views of the model.  **Supporting Information:** The interchange should facilitate long term retention, file exchange, and version upgrades.  Consider consistency with related interchange standards, such as AP233. For the concrete syntax, consider consistency with Diagram Definition and Diagram Interchange. | XMI |
| LNG 1.6.2 | Model Mappings and Transformations | SysML v2 shall provide a capability to specify model mappings and transformations.  **Supporting Information:** SysML may be used to represent the metamodel of other languages and data sources to enable transformation between SysML models, other data sources, and models in other languages. These languages include languages for queries, validation rules, expressions, viewpoint methods, and transformations.  A common need is to map elements between SysML and Excel that supports import of Excel data into a SysML model, and export of SysML model elements to Excel. Another example is a mapping between SysML models and Simulink models. | QVT |
| LNG 1.6.3 | UML Interoperability | SysML v2 shall provide the capability to map shared concepts between SysML and UML. | SysML Profile of UML |

### Data Model

**Systems Engineering Concept Model (SECM).** SysML v2 is intended to provide the capability to model systems with a precisely defined vocabulary. A Systems Engineering Concept Model (SECM) is used to capture the key concepts to represent systems, and is a primary input to help specify the requirements for the SysML v2 metamodel, profile, and model libraries. The SECM is used as part of the analysis to derive and integrate the SysML v2 requirements. It should be considered by submission teams as an important input, but is not part of the mandatory requirements in the SysML v2 RFP.

The high-level concepts in the SECM are intended to be consistent with industry standards for systems engineering that include the Systems Engineering Body of Knowledge (SEBoK), the ISO standard for Systems and Software Engineering -- System lifecycle processes (ISO/IEC/IEEE 15288:2015), and the INCOSE Systems Engineering Handbook v4. These sources and others provide high-level concepts that are inputs to the requirements for SysML v2. The figure below is an extract from the SECM-2015 Industry Reference showing some of the core concepts in the SEBoK. There are many other concepts in the industry reference model beyond what is shown in this Figure.

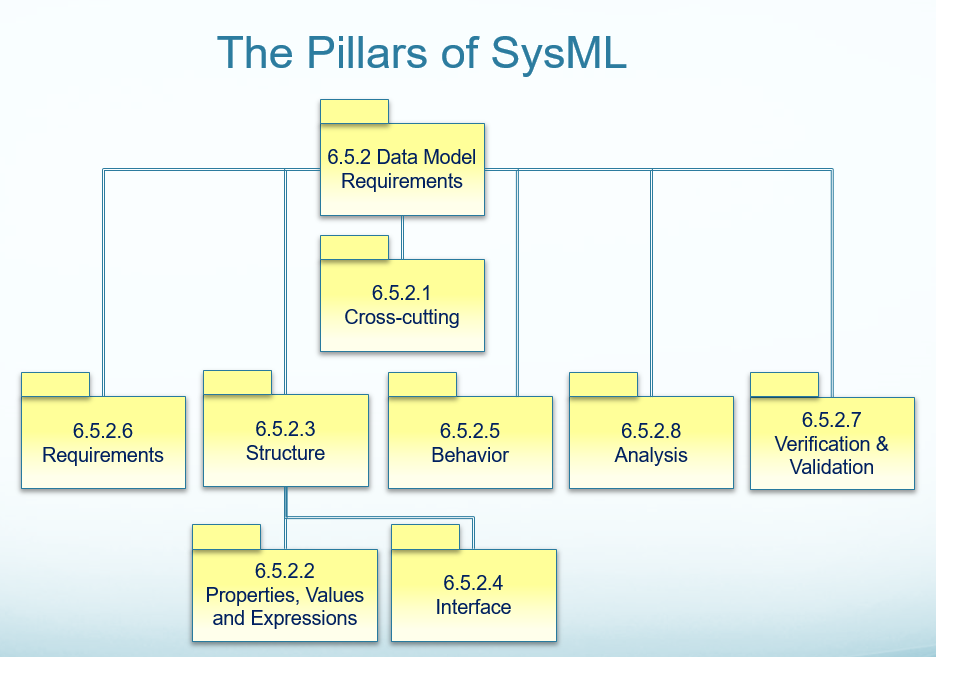
Figure 3.6. Core SEBoK Concepts (Extract from draft SECM-2015 Industry Reference. Used with permission)



SysML v2 includes concepts directly related to the specification, design, analysis, and verification of systems. The SysML v2 concepts are intended to align with the industry standards, but the scope of SysML v2 is not intended to address the full scope of the industry reference model. At the same time, SysML v2 may include additional concepts that are not explicitly referred to in the industry reference model.

**Data model requirements.** The scope of SysML v2 system modeling concepts encompasses the scope of SysML v1, which includes support for modeling structure, behavior, parametric, and requirements, often referred to as the 4 pillars of SysML. The SysML v2 concepts also include additional concepts related to verification, analysis, and other concepts beyond what is in SysML v1. The organization of the system modeling concepts are indicated in the figure below.

Figure 3.7. Organization of SysML v2 Modeling Concepts



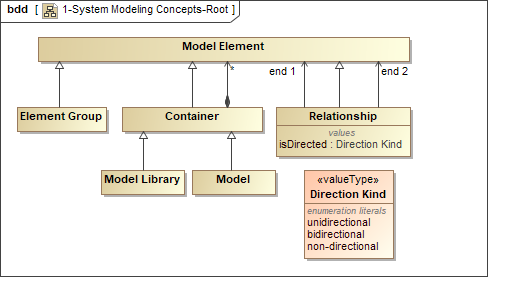
In addition to extending the SysML v1 concepts, a major emphasis for SysML v2 is to ensure integration and consistency of these concepts across the language. This is in part accomplished by defining a core set of concepts and patterns, and then applying them consistently to define other concepts. For example, the concept of decomposition can be applied consistently to structure and behavior, and the concept of precedence can be applied consistently in different behavior representations, such as activities and state machines. Logical expressions such as AND, OR, XOR, and NOT can also be applied consistently throughout the language.

The following concepts and their description provide an introduction that reflect the intent of many of the SysML v2 requirements. This summary is an abstraction of more detailed models that were developed for this effort, which in some cases reflect proposed implementations of the requirements. Any inconsistencies between the more abstract model in this section and the detailed models can be addressed as part of the submission. This summary model and the more detailed model are part of the SECM that will be provided as an input to the Submission Teams to provide context for the requirements.

Although many of the concepts may be similar to those in UML and SysML, the terms are often different to avoid the implication of a particular solution. An effort is also made to apply consistent patterns in the names of the concepts. An example is the consistent naming of terms that reflect the definition and usage pattern, such as Component Definition and Component Usage, and Port Definition and Port Usage.

**Root concepts.** The root concepts that are reflected in the cross cutting requirements are included in Figure 3.9. A Model Element is the root element. A Container contains other model elements, and is analogous to a package in SysML. An Element Group is a grouping of Model Elements that establishes criteria to be a member of a group. Unlike a Container, the Element Group does not impose constraints such as deletion semantics on its members. Model and Model Library are kinds of Containers, where a Model is a top-level Container and a Model Library contains elements that are designated to be reused. Finally, the Relationship relates 2 model elements, and can be directed, non-directed, or both. All other relationships are specialized from this more general relationship.

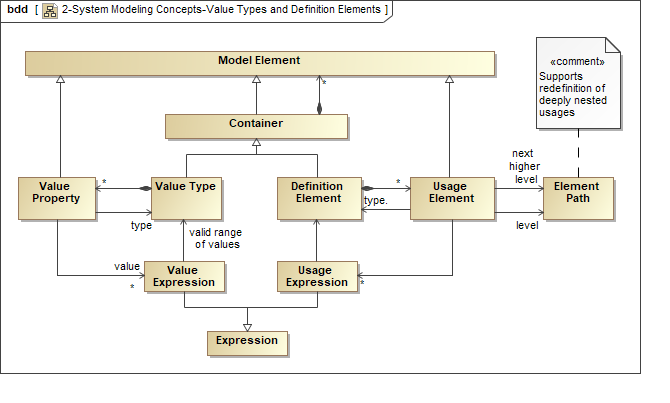
Figure 3.8. Root Concepts



**Value Type and Definition Element.** The two important kinds of Containers shown in Figure 3.10 are Value Types and Definition Elements. A Value Type is used to represent data structures with units and quantity kinds. A Value Property is typed by a Value Type to represent quantitative properties. The Value Type defines the valid range of values that a Value Property can have. A Value Expression establishes the specific value for a Value Property. A Value Type can contain other Value Properties. The kinds of Value Types have been significantly expanded beyond the primitive Value Types in SysML v1 to include vectors, collections, and other more complex data structures.

The concepts of definition and usage, such as block and part, are core concepts in SysML v1 that also apply to many of the SysML v2 language concepts. The Definition Element and Usage Element provide the ability to define a concept one time, and then reuse it in many different contexts. Usage Elements represent many concepts that are referred to as structural and behavioral features in UML and SysML. A Usage Element is typed by a Definition Element, and a Definition Element can contain other Usage Elements. An Element Path can unambiguously refer to a deeply nested usage element, and over-ride the definition for a particular localized usage (Note: analogous to SysML redefinition). This concept is further elaborated in the SECM. The Usage Expression allows the representation of specific usages and their logical expressions such as {(Usage Element A AND Usage Element B) OR (Usage Element C AND Usage Element D)}.

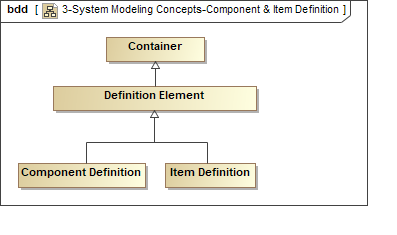
Figure 3.9. Value Type and Definition Element



**Component Definition and Item Definition.** Two particular types of Definition Elements are Component Definition and Item Definition as shown in Figure 3.11. A Component Definition typically represents a system, subsystem, or other element that compose a system or other external entity. An Item Definition represent the kinds of things that flow through a system or between a system and other external entities. A simple example of a Component Definition is a Pump, and an example of an Item Definition is Water which can flow in and out of the Pump.

Both of these concepts can be represented in SysML v1 as a Block, which is defined as a modular unit of Structure. SysML v1 also includes more specific concepts to represent items that flow, such as flow properties and item properties. These would be referred to as item usages using the SysML v2 vocabulary.

Figure 3.10. Component Definition and Item Definition

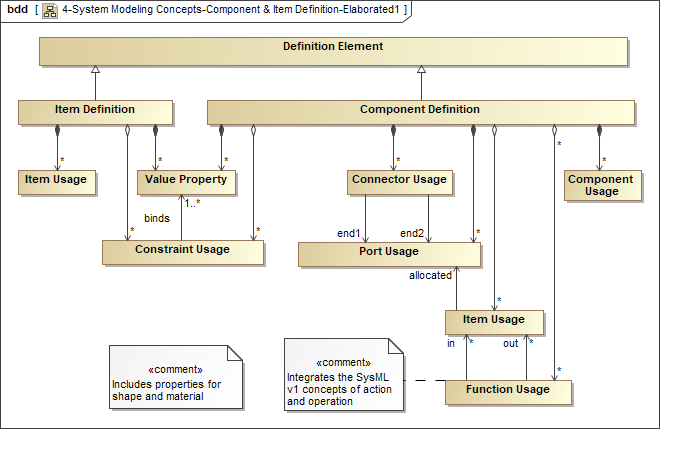


**Component Definition and Item Definition-Elaborated.** The Component Definition and Item Definition are further elaborated in Figure 3.12 to show the kinds of usage elements and value properties that they contain.

The Item Definition includes Value Properties and Constraint Usages to constrain its Value Properties, and includes Item Usages to create nested item structures. An example of a nested item structure may correspond to a message structure that includes a header and a body, which is further decomposed into specific fields that capture application data.

The Component Definition contains Value Properties and Constraint Usages, and Component Usages to define nested component structures. It also contains Port Usages and Connector Usages to connect Component Usages. Component Definition also contains Function Usages that are analogous to an Operation of a Block, but is also intended to represents an action that the Block performs to transform inputs to outputs, or to change the state of the owning Component. The input and output Item Usages are allocated to Port Usages. A Component Definition can also contain Connector Usages that connect Port Usages on it nested Component Usages. A Component Definition can also contain a shape property that specifies the simplified geometry and size of a Component in a reference coordinate system, which is intended to facilitate specification of physical envelopes.

Figure 3.11. Component Definition and Item Definition-Elaborated

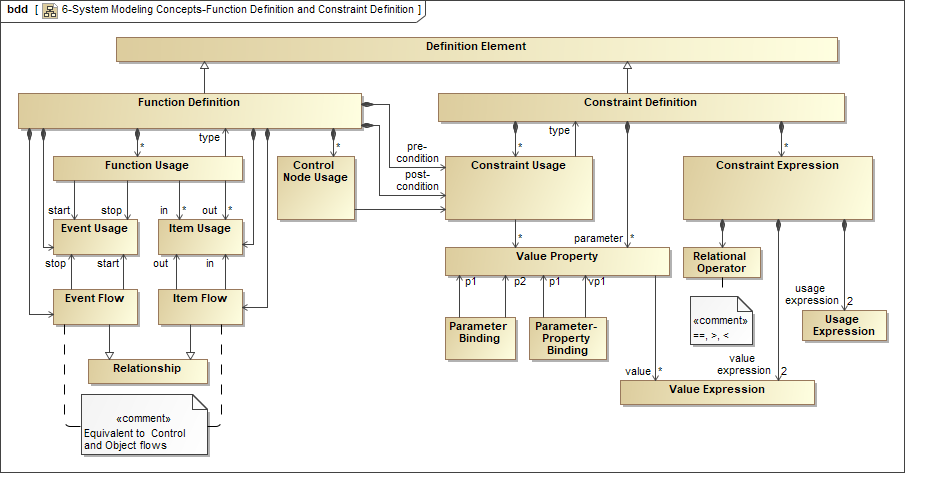


**Function Definition and Constraint Definition.** As noted in the previous figure, both a Component Definition and Item Definition contain Constraint Usages, and a Component Definition can also contain Function Usages. The Constraint Definition and Function Definition are also Definition Elements as shown in Figure 3.13. They both use the standard pattern that enable the Definition Element to decompose into Usage Elements, and the Usage Elements are typed by a Definition Element, enabling a nested tree of usages.

The Function Definition contains Function Usages and Control Node Usages which are analogous to actions and control nodes in SysML v1. Function Usages can include both inputs and outputs (i.e., Item Usages), and start and stop events (i.e., Event Usages). An Output from one Function Usage is connected to the input of another Function Usage by an Item Flow, and the stop event of one Function Usage can be connected to the start event of another Function Usage by an Event Flow. Item Flow and Event Flow are analogous to Object Flow and Control Flow in SysML v1. Although not shown, these flows can also connect Control Node Usages that constrains the sequence of flow similar to a join specification in SysML v1. Finally, Function Definitions can include preconditions and post-conditions that must be satisfied prior to initiating and completing a function.

A Constraint Definition contains Constraint Usages and Constraint Expressions that constrain the parameters of expressions, similar to SysML v1 Constraint Blocks.

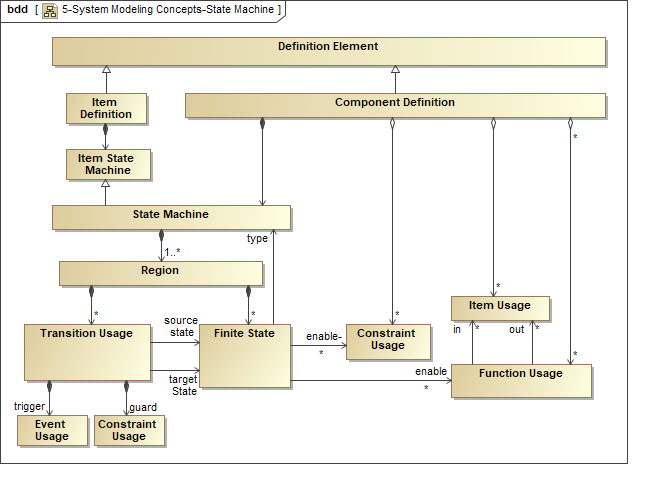
Figure 3.12. Function Definition and Constraint Definition



**State Machine.** The State Machine of a Component Definition and Item Definition specify its finite (i.e., discrete) states and the transitions between them as shown in the figure below. It also contains Regions that enable each Region to have a single active finite state at any point in time.

The State Machine is a Definition Element which enables a Finite State to be typed by a State Machine. A Finite State can enable Constraint Usages and Function Usages in response to an event and guard condition. An Item State Machine for an Item Definition is a more generalized state machine that can define its discrete states and transitions, such as the transition between the solid, liquid, and gas state of H2O. A State Machine for an Item Definition can enable Constraint Usages, but does not enable Function Usages.

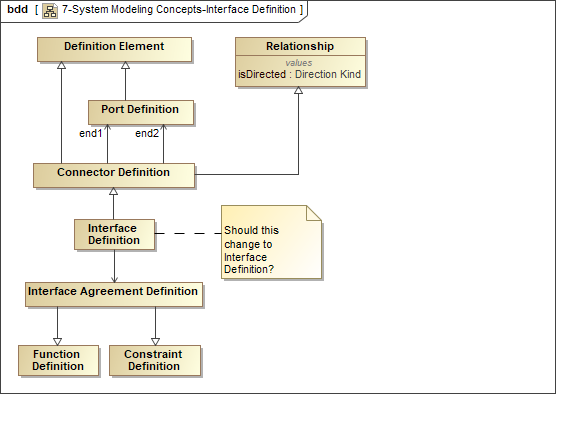
Figure 3.13. State Machine



**Interface Definition.** An Interface definition in SysML v2 constrain the physical and functional interaction between structural elements. The Interface includes two ends, the connection between them, and the constraints on the connection. As shown in the figure below, the Interface Definition is a subclass of a Connector Definition, which corresponds to a SysML v1 association block that can be used to type connectors.

The Connector Definition includes the definition of its ends as Port Definitions (aka Interface End Definitions), and includes an Interface Agreement which constrains the interaction across the connection. The two types of Interface Agreements include both a Function Definition and Constraint Definition. Function Definitions are generally used to constrain the exchange of Items, such as with a communication protocol, and Constraint Definitions are generally used to constrain physical interactions such as voltage and current (i.e., Across and Through Variables). Although not shown in the Figure, a special type of component called an Interface Medium enables connection between other components, such as a pipe, network, **or** cable. Interfaces also support nested ports and layered interfaces.

Figure 3.14. Interface Definition

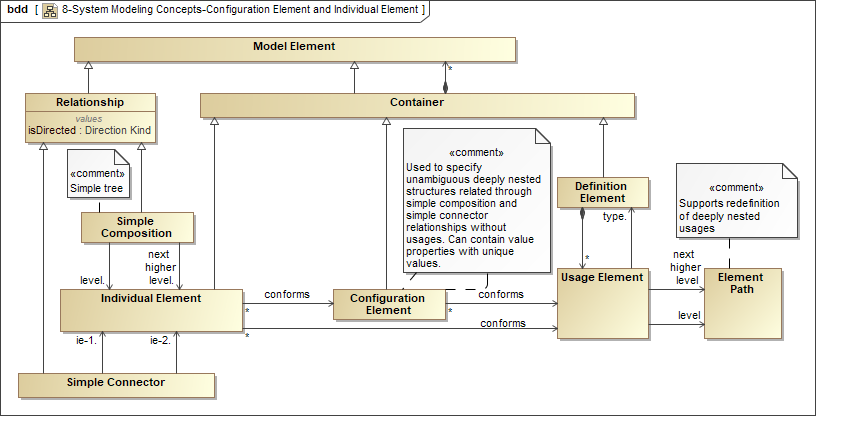


**Configuration Element and Individual Element.** A Definition Element can be decomposed into a tree of Usage Elements as noted before. However, SysML v2 requires a mechanism to define an unambiguous deeply nested structure using Configuration Elements. This is intended to provide a straight forward way to specify a design configuration. A simple example is a vehicle that has 4 wheels, and each wheel has several lug bolts. The design configuration would enable the definition of an unambiguous product structure where each lug bolt on each wheel is clearly identified, and the torque value for each lug bolt can also be uniquely defined by its localized usage.

An Individual Element represents a model of a particular element that is uniquely identified, such as a model of a particular Vehicle on the factory floor with a Vehicle Identification Number (VIN). The structure of an Individual Element can be modified, such as replacing its wheels with a new kind of wheel and tire. A Simple Composition and Simple Connector is used to define a tree of Individual Elements and connect Individual Elements. The same Simple Composition and Simple Connector can be used to compose and connect Configuration Elements.

A Configuration Element can conform to a Definition Element such as a Component Definition, and an Individual Element can conform to a Configuration Element. However, they are not required to conform to any particular element, which enables one to create models of Individual Elements and/or Configuration Elements independently. For example, one can model an Individual Element of the specific Vehicle on a factory floor without requiring a corresponding Configuration Element or a Definition Element.

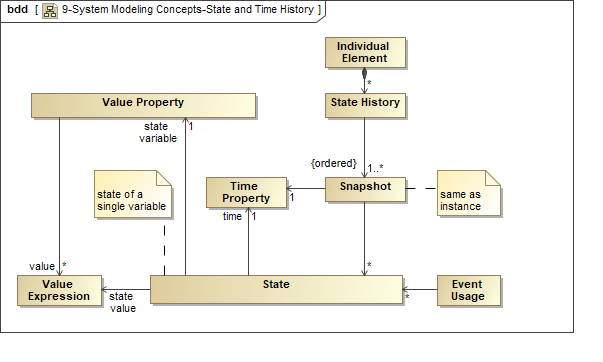
Figure 3.15. Configuration Element and Individual Element



**State and Time History.** An Individual Element can have a state and time history. The state history is defined as a series of ordered Snapshots of an Individual Element, where each Snapshot represents the state of the Individual Element at a point in time. The Snapshot represents the values of each of its value properties at a particular point in time. As an example, the Snapshot of an Engine may include the values of its temperature and torque at a point in time. The value properties whose value can change over time are sometimes referred to as state variables.

Each Individual Element can contain multiple State Histories, where each State History can represent a particular estimate of its change in state over time. A State History for a Component Definition implies that each conforming Individual Element will have this State History.

Figure 3.16. State and Time History

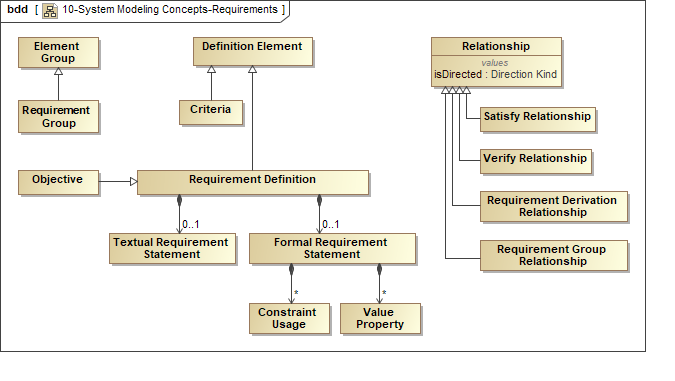


**Requirements.** A Requirement in SysML v2 will extend the SysML v1.5 Requirement which includes the ability to more precisely specify a requirement with a Formal Requirement Statement, in addition to a Text Requirement Statement. The Formal Requirement Statement can be specified by constraints.

Requirements are grouped into Requirement Groups to provide context for the requirements. Requirement Groups can also contain other nested Requirements Groups that enable creation of a nested specification of requirements. Each requirement in a requirement group can be related to other elements using requirements relationships such as Satisfy, Verify, Derive, and others similar to SysML v1.

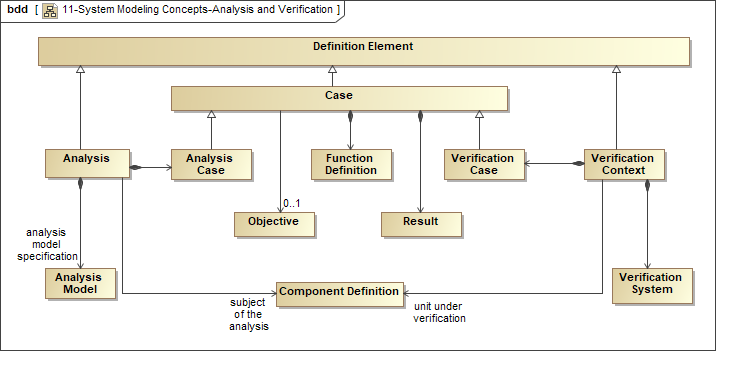
An Objective is considered a specialized requirement that reflects a desired or required end state. The Criteria define an expression that specifies the characteristics of interest and their relative weighting which can be used as a basis for an evaluation.

Figure 3.17. Requirements



**Analysis and Verification.** SysML v2 includes additional concepts to support Analysis and Verification. Both Analysis and Verification can apply similar patterns to represent an Analysis or Verification Context that include the Component Definition, Configuration, or Individual being analyzed or verified, the analysis models or verification system used to perform the verification or analysis, and the Analysis Case or Verification Case used to define how the analysis or verification is performed. The concept of Case is a common concept that is specialized to define an Analysis Case and Verification Case.

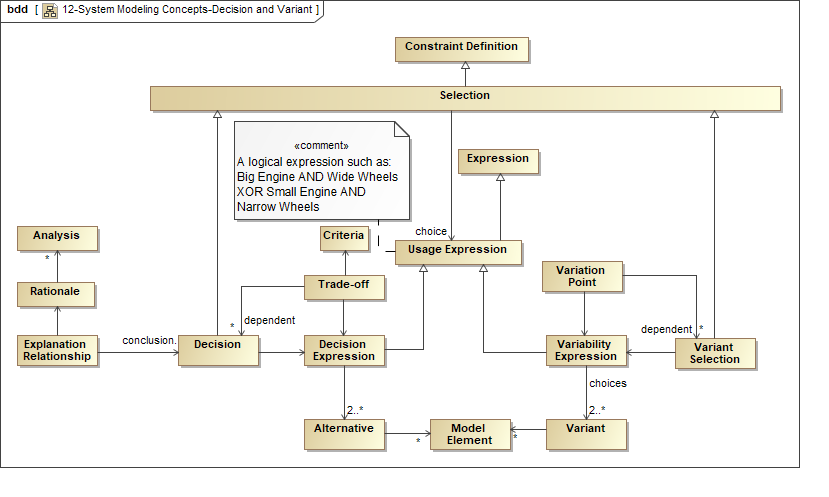
Figure 3.18. Analysis and Verification



**Decision and Variant.** SysML v2 requires additional concepts to support decisions analysis, such as trade studies, and variant modeling. Some common patterns for these concepts are noted in Figure 3.20. In particular, both a Decision and Variant Selection involve a set of choices, called Alternative and Variant, respectively. An Expression can be used to define the choices such as A or B or C. The name for the set of choices is called a Trade-off and a Variation Point. A Selection is made among choices and called a Decision and a Variant Selection respectively. The available choices may be dependent on other Selections.

The Explanation Relationship relates the Decision to the Rationale, which in turn refers to the Supporting Analysis. The Rationale can be applied more generally to refer to the basis for any conclusion.

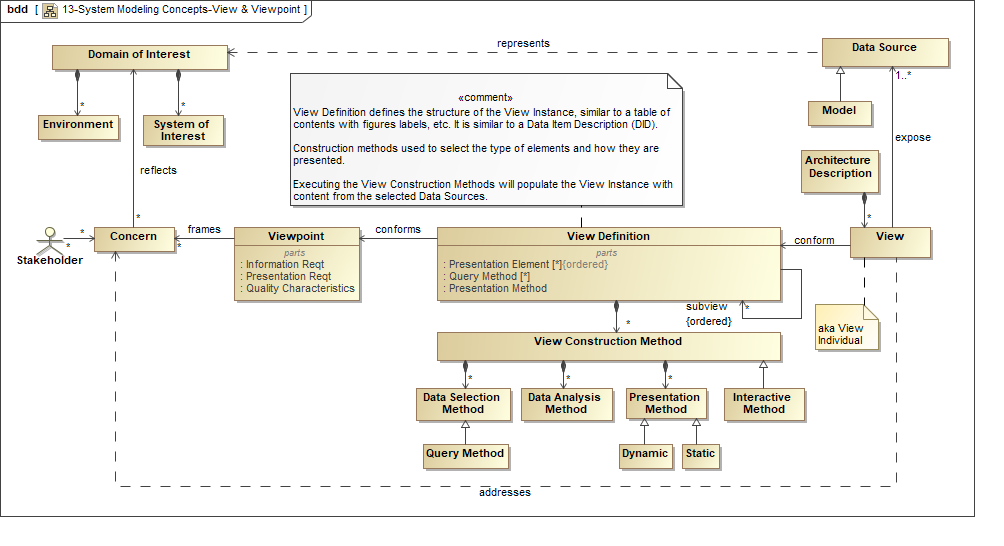
Figure 3.19. Decision and Variant



**View and Viewpoint.** SysML v2 includes concepts to enable the generation of Views of the system or Domain of Interest that address diverse Stakeholder Concerns. A View can represent a particular diagram, table, or complete document that is presented to Stakeholders to address their concerns. The Model is treated as a Data Source that is used to create the View.

A Viewpoint specifies the type of information and the format of the presentation that a View must provide. The View Definition defines the structure of the View in terms of its Sub-View Definitions, such as a Table of Contents for a document. The View Definition also includes methods to construct the View. The View is generated by applying the Construction Methods to query a particular model and present the results in a specific artifact. The concepts of View and Viewpoint are intended to generally align with the proposed concepts from ISO 42010, Systems and Software Engineering, Architecture description [ArchDes].

Figure 3.20. View and Viewpoint



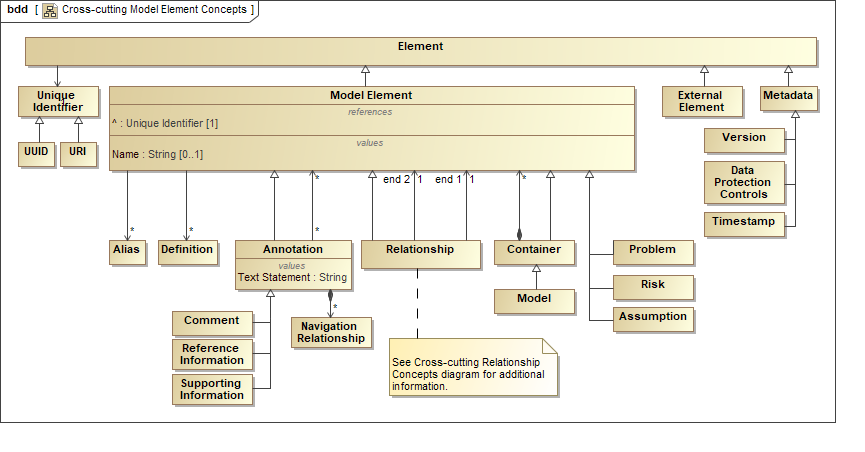
#### Cross-cutting

##### Cross-cutting Introduction

The Cross-cutting concepts and associated requirements apply to all model elements. A model element is the most general element in the model, and includes features that are common to all other kinds of model elements that are specified in the language.

As shown in the figure below, a model element contains certain properties such as unique id, name, alias, and definition. An annotation is a sub-class of model element that can refer to other model elements.

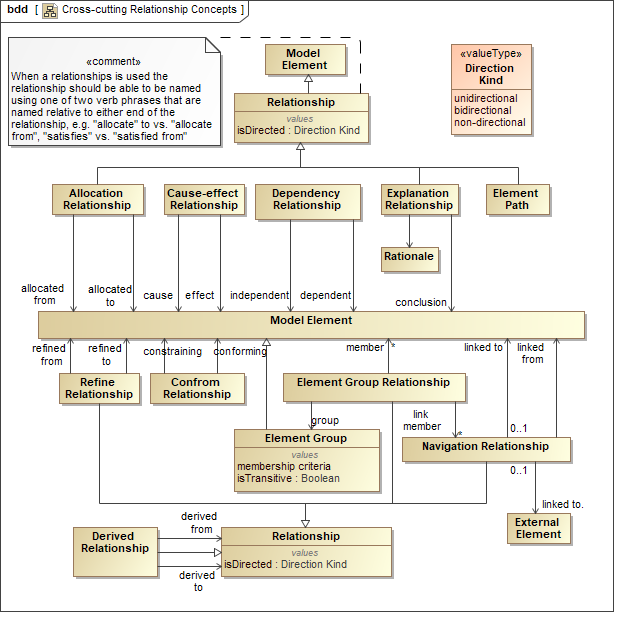
Figure 3.21. Cross-cutting Model Element Concepts



A container is a kind of model element that contains other model elements and applies scoping rules such as namespace rules to the contained elements. A Model is top-level container, and a Model Library is a kind of container designated to contain reusable model elements.

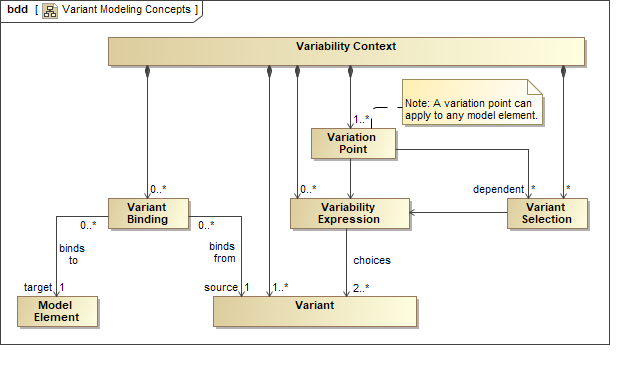
In addition, a model element can have a relationship with other model elements. The Relationship can relate any kind of model element. This relationship be used directly or further specialized into more specific relationships. This includes the dependency, allocation, cause-effect, explanation, and element group relationships that can have any model element on one end of the relationship but may constrain the other end to specific kinds of model elements.

Figure 3.22. Cross-cutting Relationship Concepts



The following figure shows the concepts of variability that can be applied to any model element. These include the concept of a variation point that identifies some part of the model that can vary, and a variant that identifies the specific choices. For example, wheel size can be a variation point, and narrow wheel and wide wheel are variants. Variability constraints can be defined to constrain the valid choices (i.e., variants) for one or more variation points. A variant binding concept is also included to enable a separate variant model to represent the variant concepts and refer to the base model elements in the SysML model.

Figure 3.23. Variant Modeling Concepts



The concepts of View and Viewpoint are also included as Cross-cutting Concepts and are illustrated at the end of the introductory section to the Data Model. These concepts are used by the required Visualization services described in the Services section below to generate Views that conform to Viewpoints.

##### Cross-cutting Requirements

**Table 3.2. Cross-cutting Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| CRC 1 | Cross-cutting Requirements Group | The following specify the requirements that apply to all model elements. |  |
| CRC 1.1 | Model and Model Library Group |  |  |
| CRC 1.1.1 | Model | SysML v2 shall include a capability to represent a model (aka system model) that contains a set of uniquely identifiable model elements.  **Supporting Information:** This is intended to be a kind of Container or Namespace. | Model |
| CRC 1.1.2 | Model Library | SysML v2 shall include a capability to represent a Model Library that contains a set of model elements that are intended to support reuse.  **Supporting Information:** This is intended to be a kind of Container or Namespace. | Model Library |
| CRC 1.1.3 | Container | SysML v2 shall include the capability to represent a Container that is a model element that contains other model elements.  **Supporting Information:** This provides a way to organize the model and should include considerations for rules to uniquely identify the content of a container. Containers can contain other containers. | Package |
| CRC 1.2 | Model Element Group |  |  |
| CRC 1.2.1 | Model Element | SysML v2 shall include a root element that contains features that apply to all other kinds of elements in the model. | Model Element |
| CRC 1.2.2 | Unique Identifier | SysML v2 shall include a capability to represent a single unique identifier for each model element that cannot be changed.  **Supporting Information:** The unique identifier should enable assignment of URIs. | UUID is part of the XMI specification |
| CRC 1.2.3 | Name and Aliases | SysML v2 shall include a capability to represent a name and one or more aliases for any named model element.  **Supporting Information:** Aliases enable users to assign more than one name for the same element, such as a shortened name.  Selected kinds of model elements may not require a name (e.g. dependency), or the name may be optional, but still should be distinguishable within a namespace.  A common use of aliases is the use of an abbreviated or shortened name. | Named Element |
| CRC 1.2.4 | Definition / Description | SysML v2 shall include a capability to represent one or more definitions and/or descriptions for each model element and select those that apply. | Owned Comment |
| CRC 1.2.5 | Annotation | SysML v2 shall include a capability to represent an annotation of one or more model elements that includes a text string that can include a link that refers to a Navigation relationship, and a classification field that can be used to identify the comment as a question, a response, an issue, a problem, or a proposed change.  **Supporting Information:** Annotations should be able to be related to other elements. | Comment |
| CRC 1.2.6 | Element Group | SysML v2 shall include a capability to represent a group of model elements that can be ordered and can satisfy user-defined criteria for membership in the group.  **Supporting Information:**   1. A query can be used to dynamically update the members of the group. 2. A relationship between an element group and another element applies to each member of the element group. 3. A member of an element group is not intended to impose ownership constraints on the members. 4. Element group is expected to be specialized for different kinds of members, such as contain requirements, functions, and structural elements, which may impose additional constraints on its members. | Element Group |
| CRC 1.2.7 | Additional Cross-Cutting Concepts Group | The requirements in this group include additional concepts that can be associated with any model element. |  |
| CRC 1.2.8.1 | Problem | SysML v2 shall include a capability to represent a problem that causes an undesired affect from a particular stakeholder.  **Supporting Information:** A problem is often represented as a cause in a cause-effect relationship. | Problem |
| CRC 1.2.8.2 | Risk | SysML v2 shall include a capability to represent a Risk that identifies the kind of risk (e.g., cost, schedule, technical), and the likelihood of occurrence, and the potential impact. |  |
| CRC 1.3 | Model Element Relationships Requirements Group |  |  |
| CRC 1.3.01 | Relationship | SysML v2 shall include a capability to represent a Relationship between any two model elements, which may have a name and direction. | Relationship |
| CRC 1.3.02 | Derived Relationship | SysML v2 shall include a capability to represent a relationship that is derived from other relationships.  **Supporting Information:**  An example is a derived relationship from a transitive relationship where B relates to A and C relates to B, then C relates to A.  Another example is a connector between two composite parts that is derived from a connector between their nested parts. |  |
| CRC 1.3.03 | Dependency Relationship | SysML v2 shall include a capability to represent a Dependency Relationship where one side of the relationship refers to the independent element and the other side of the relationship refers to the dependent element. | Dependency |
| CRC 1.3.04 | Cause-Effect Relationship | SysML v2 shall include a capability to represent a Cause-Effect Relationship where one side of the relationship refers to the cause and the other side of the relationship refers to the effect. |  |
| CRC 1.3.05 | Explanation Relationship | SysML v2 shall include a capability to represent an Explanation Relationship where one side of the relationship refers to the rationale and the other side of the relationship refers to the element being explained (i.e. what is concluded). | Anchor on a rationale |
| CRC 1.3.06 | Conform Relationship | SysML v2 shall include a capability to represent a Conform Relationship where the conforming element is constrained by the element on the other side of the relationship. |  |
| CRC 1.3.07 | Refine Relationship | SysML v2 shall include a capability to represent a Refine Relationship where the refined side of the relationships refers to the more precisely specified element. | RefineReqt |
| CRC 1.3.08 | Allocation Relationship | SysML v2 shall include a capability to represent an Allocation Relationship where one side of the relationship refers to the allocated from, and the other side of the relationship refers to the allocated to. | Allocate |
| CRC 1.3.09 | Element Group Relationship | SysML v2 shall include a capability to represent an Element Group Relationship where one side of the relationship refers to the member, and the other side of the relationship refers to the Element Group. | Anchor |
| CRC 1.3.10 | Navigation Relationship | SysML v2 shall include a capability to represent a Navigation Relationship between a model element and another model element or an external element, similar to a hyperlink, where one side of the relationship refers to the linked to, and the other side of the relationship refers to the linked from. The external element can be a data element, a file, and/or an element of an external model.  **Supporting information:**  This is a navigation aid that standardizes what many tools already do.  The navigation can specify the ability to navigate from either end of the relationship. | Some tools support navigation links, but not in a standard way. |
| CRC 1.3.11 | Copy Relationship | SysML v2 shall include a capability to represent a Copy Relationship where one side of the relationship refers to the element (or elements) being copied and the other side of the relationship refers to the copy (or copies).  **Supporting Information:**  The primary goals for this relationship are to establish provenance to support traceability, and to enable reuse of catalog items. This relationship provides the ability to copy elements such as a Container (e.g., package) and its contents, within a model and from one model to another. Additional constraints can be defined to specify the rules for what part of the element being copied can be modified in the copy. It is assumed that updates to the copied element are not propagated, unless there is a rule to support this. |  |
| CRC 1.4 | Variability Modeling Group | The requirements in this group should accommodate approaches to model variants as choices among design options. The modeling approaches may include a separate variability model to identify the design choices.  **Supporting information:** refer to ISO/IEC 26550:2015 |  |
| CRC 1.4.1 | Variation Point | SysML v2 shall include a capability to model variation points that identify features that can vary across a set of variants (e.g., vehicles with manual or automatic transmission, variable number of axles, or variable wheel size). A variation point may be dependent on another variant selection. (e.g., number of axles and wheel size is dependent on selection of load size). |  |
| CRC 1.4.2 | Variant | SysML v2 shall include a capability to model variants that correspond to particular selections that are associated with a variation point. |  |
| CRC 1.4.3 | Variability Expression and Constraints | SysML v2 shall include a capability to model variability expressions that constrain possible variant choices (e.g., 3 axles plus large wheel size or 2 axles plus small wheel size). |  |
| CRC 1.4.4 | Variant Binding | SysML v2 shall include a capability to model the binding between a variant and the model elements that vary.  **Supporting Information:** The binding is intended to enable the use of a separate variability model that defines variation that may span multiple kinds of models such as a SysML model, simulation model, and a CAD model. |  |
| CRC 1.5 | View and Viewpoint Group | The following specify the requirements associated with View and Viewpoint. These concepts are used by the Visualization Services to generate Views of a Model. |  |
| CRC 1.5.1 | View Definition | SysML v2 shall include a capability to define the structure of a class of artifacts that can be presented to a stakeholder.  **Supporting Information:** An individual View is intended to be a specific artifact, such as a document, diagram, or table that is presented to a stakeholder. The individual View conforms to a View Definition that defines a set of construction methods that include query methods and presentation methods for creating an individual View. The View is generated when query methods are executed to query a particular model (or more generally one or more data sources) to select the kinds of model elements, and then the presentation methods are executed to present the information in a specified format to generate the individual View. The View Definition for a document can be thought of as its table of contents along with the list of figures and tables. The View Definition can be specialized, and decomposed into sub-views that can be ordered. | View |
| CRC 1.5.2 | Viewpoint | SysML v2 shall include a capability to represent a Viewpoint that frames a set of stakeholders and their concerns. It specifies the requirements a View must satisfy.  **Supporting Information:**  The stakeholder and their concerns should be represented in the model.  The intent is to align the view and viewpoint concepts with the update to ISO 42010. | Viewpoint |
| CRC 1.6 | Metadata Group | The requirements in this group identify metadata is a kind of model element that can apply to other model elements or to other elements that refers to a model element (e.g., a model configuration item). Also, refer to the requirement for Analysis Metadata in the Analysis requirements section.. |  |
| CRC 1.6.1 | Version | SysML v2 shall include a capability to represent the version of one or more model elements, or of another element that refers to one or more model elements. |  |
| CRC 1.6.2 | Time Stamp | SysML v2 shall include a capability to represent a model management time stamp for one or more elements, or of another element that refers to one or more model elements. |  |
| CRC 1.6.3 | Data Protection Controls | SysML v2 shall include a capability to represent Data Protection Controls for one or more model elements, or of another element that refers to one or more elements.  **Supporting Information:** This can include markings such as ITAR, proprietary or security classifications |  |

#### Properties, Values & Expressions

##### Properties, Values & Expressions Introduction

The foundation concepts in SysML v1 for specifying quantitative and qualitative characteristics and supporting engineering analysis are value properties and value types, and the usage of constraint blocks that capture reusable equations and parameters in parametric diagrams, and a rich non-normative model for representing quantities and units. SysML v2 will extend these concepts to include a more comprehensive set of value types such as arrays, vectors, and a discretely sampled function that captures discrete functions that are often specified as tabular data. SysML v2 also includes concepts related to coordinate transformations and geometric shapes to enable representation of basic geometry. SysML v2 provides improved support for probability concepts, quantities, units and scales, as well as a selection of a default expression language that can be applied to any feature or property.

##### Properties, Values & Expressions Requirements

**Table 3.3. Properties, Values & Expressions Requirements**

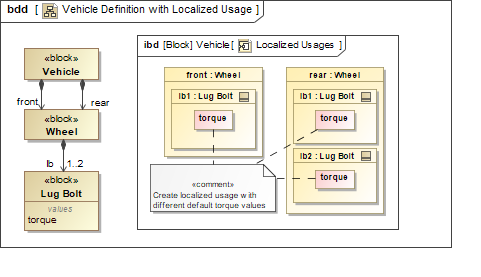
| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| PRP 1 | Properties, Values and Expressions Requirements Group | The requirements in this group provide a unified representation of the type of properties, variables, constants, operation parameters and return types as well as literal values and value expressions. This includes types to represent variable size collections, compound value types, and measurement units and scales. |  |
| PRP 1.01 | Unified Representation of Values | SysML v2 shall include a capability to represent any value-based characteristic in a unified way, called a value property, which shall include representation of a constant, a variable in an expression or a constraint, state variable, as well as any formal parameter and the return type of an operation.  **Supporting Information:**  An "invariant" can be attached to a value property to assert that is does not vary over time. A constant is an invariant value property of some higher-level context (ultimately the "universe" in case of fundamental physics constants).  Provisions should be made to distinguish between a fundamental physical or mathematical constant (i.e., Pi) from a constant value within the context of a particular model or model execution (i.e., amplifier gain). | Value Property, Formal Parameter of an Operation, Default Value, Static Value, Initial Value |
| PRP 1.02 | Value Type | SysML v2 shall include a capability to represent a Value Type as a named definition of the essential semantics and structure of the set of allowable values of a value-based characteristic. | Value Type |
| PRP 1.03 | Value Expression | SysML v2 shall include a capability to represent a value as a literal or through a reusable Value Expression that is stated in an expression language. A Value Expression shall include the capability to represent opaque expressions. | Opaque and OCL expressions, Value Specification |
| PRP 1.04 | Logical Expressions | SysML v2 shall include a capability to represent, as part of the Expression language, logical expressions that support as a minimum the standard boolean operators AND, OR, XOR, NOT, and conditional expressions like IF-THEN-ELSE and IF-AND-ONLY-IF, in which symbols bound to any characteristics (e.g. value properties or usage features) may be used. |  |
| PRP 1.05 | Unification of Expression and Constraint Definition | SysML v2 shall include a capability to represent a reusable constraint definition in the form of an equality or inequality which can be evaluated to true or false, and where the left and right-hand sides of the constraint definition are Value Expressions. | Constraint Block |
| PRP 1.06 | System of Quantities | SysML v2 shall include a capability to represent a named system of quantities that support definition of numerical Value Types in accordance with the ISO/IEC 80000 standard.  **Supporting Information:** The typical Systems of Quantities is the ISO/IEC 80000 International System of Quantities (ISQ) with seven base quantities: length, mass, time, electric current, thermodynamic temperature, amount of substance and luminous intensity. | SystemOfQuantities in Annex E.5 QUDV |
| PRP 1.07 | System of Units and Scales | SysML v2 shall include a capability to represent a named system of measurement units and scales to define the precise semantics of numerical Value Types in accordance with the [ISO/IEC 80000] standard.  **Supporting Information:** Similar to SysML v1 QUDV, SysML v2 should include model libraries representing the [ISO/IEC 80000] units, as well as the conversion to US Customary Units defined in [NIST SP 811] Appendix B. | SystemOfUnits in Annex E.5 QUDV |
| PRP 1.08 | Range Restriction for Numerical Values | SysML v2 shall include a capability to represent a value range restriction for any numerical Value Type.  **Supporting Information:** This requirement allows further restriction of the range of values beyond what is specified by its type. A simple example is a planar angle typed by a real number Value Type and a degree measurement scale. However, the value range may be further restricted from 0 to 360 degrees for positioning a rotational knob. This can also include the definition of optional lower and upper bounds on an associated measurement scale. |  |
| PRP 1.09 | Automated Quantity Value Conversion | SysML v2 shall include a capability to represent all information necessary to perform automated conversion of the value of a quantity (typed by a numerical Value Type) expressed in one measurement scale to the value expressed in another compatible measurement scale with the same quantity kind.  **Supporting Information:** This capability is needed to rebase a set of (smaller) system models coming from various contributors on a single coherent set of measurement scales, so that an integrated (larger) system model can be consistently constructed and analyzed. | Most concepts are defined in Annex E.5 QUDV, but measurement scales are lacking detail to fully automate value conversions. |
| PRP 1.10 | Primitive Data Types | SysML v2 shall include a capability to represent the following primitive data types as a minimum: signed and unsigned integer, signed and unsigned real, string, boolean, enumeration type, ISO 8601 date and time, and complex.  **Supporting Information:** These are intended to be represented in a Value Type Library as they are in SysML v1. | Primitive ValueType Library |
| PRP 1.11 | Variable Length Collection Value Types | SysML v2 shall include a capability to represent variable length value collections where each member of the collection is typed by a particular Value Type and is referable by index, and where the collection may be one of the established collection types: sequence (ordered, non-unique), set (unordered, unique), ordered set (ordered, unique) or bag (unordered, non-unique). |  |
| PRP 1.12 | Compound Value Type | SysML v2 shall include a capability to represent both scalar and compound Value Types, where a scalar Value Type represents elements with a single value, and compound Value Type represents elements with a fixed number of component values, where each component value is typed in turn by a scalar Value Type or another compound Value Type.  **Supporting Information:** Such compound Value Types are needed to support the representation of vector, matrix, higher order tensor, computer data record, complex number, quaternion, and other richer Value Types. | ValueType |
| PRP 1.13 | Discretely Sampled Function Value Type | SysML v2 shall include a capability to represent variable length sets of values that constitute discrete time series data, frequency spectra, temperature dependent material properties, and any other datasets that can be represented through a discretely sampled mathematical function.  **Supporting Information:** Such a discretely sampled function can be defined by a tuple of one or more Value Types that prescribe the type of the domain (independent) variables, and a tuple of one or more Value Types that prescribe the range (dependent) variables, as well as a variable length sequence of tuples that represent the actual set of sampled values. |  |
| PRP 1.14 | Discretely Sampled Function Interpolation | SysML v2 shall include a capability to represent an interpolation scheme for a Discretely Sampled Function Value Type for derivation of the function's range values for domain values that are in-between sampled values. |  |
| PRP 1.15 | Probabilistic Value Distributions | SysML v2 shall include a capability to represent the value of a quantity with a probabilistic value distribution, including an extensible mechanism to detail the kind of distribution, i.e. the probability density function for continuous random variables, or the probability mass function for discrete random variables. | Annex E.7 Distribution Extensions |
| PRP 1.16 | System Simulation Models | SysML v2 shall include a capability to represent signal flow graph models and lumped parameter models as well as combinations thereof.  **Supporting Information:** See [SysPISF] for details.  This requirement is augmented by the analysis requirements. |  |
| PRP 1.17 | Across and Through Value Properties | SysML v2 shall include a capability to define across and through properties of flows on Interface Ends that participate in representing physical interactions in lumped parameter models.  **Supporting Information:** Typically, the across and through properties are defined together as a pair, where the across property does not conserve energy and the through property does. For example, in a lumped parameter model of an electric circuit, the across and through properties are voltage and current respectively. See [SysPISF] for details. |  |
| PRP 1.18 | Basic Geometry | SysML v2 shall include a capability to represent basic two- and three-dimensional geometry of a structural element, including a base coordinate frame as well as relative orientation and placement of shapes through nested coordinate frame transformations, where the basic shape definitions are provided in a model library.  **Supporting Information:** These capabilities are intended to provide basic geometry and coordinate frame representations to support specification of physical envelopes. The intent is that each block or equivalent will have its own reference coordinate system, and transformations can be applied between coordinate systems of different blocks. The shape of a block is defined as a property (e.g., 3-dimensional rectangular shape with length, height, and depth) whose values can be defined in its reference coordinate system. Consider references to standard formats (e.g., ISO 10303 (STEP), IGES) |  |
| PRP 1.19 | Materials with Properties | SysML v2 shall include a capability to represent named materials with their material properties in a model library and assignment of such materials to structural elements.  **Supporting information:** This requirement is intended to specify a model library with a generic material kind that has generic material properties that can be further specialized. |  |

#### Structure

##### Structure Introduction

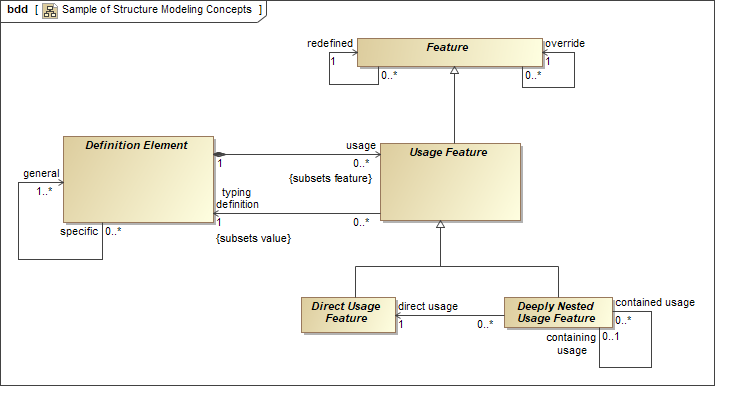
The structure modeling concepts are intended to facilitate modeling of deeply nested elements of a system or other composite structure and their interconnection. A limitation of SysML v1 is highlighted in the figure below showing a nested structure of Lug Bolts that are part of Wheels that are part of a Vehicle. The challenge in SysML v1 is the ability to easily represent the structures corresponding to a specific design configuration that have localized values. For example, in the internal block diagram in the figure below, the torque on the 2nd lug bolt on the rear wheel should be clearly distinguishable from the torque on the 1st lug bolt. This can be done in SysML v1 but may require the use of advanced features such as property specific types, redefinition, subsetting, and bound references.

Figure 3.24. Specifying an unambiguous system design configuration



Modeling structure in SysML v2 builds on SysML v1 concepts of definition and usage (e.g., blocks and parts), but adds the concept of a deeply nested part as shown in the figure below to facilitate modeling of deeply nested structures.

Figure 3.25. Sample of Structure Modeling Concepts



Like SysML v1, SysML v2 can also model structure with variabilities that may be represented with part multiplicity and component sub-classes. In SysML v2, additional concepts are provided to model system design configurations with the variation in the system structure removed, which means the system structure is defined as an unambiguous tree of usage elements. The system design configuration can include property values that over-ride the values in the more general definition, such as the torque values on the lug bolts in the example above.

Another limitation of SysML v1 is the ability to model an as-built system that has measured values that over-ride the as-designed values associated with the system design configuration. This can be done in SysML v1 by creating an instance. However, once this is done, there can be no further changes to the structure of the system, which limits the ability to model a system across its lifecycle. For example, the replacement of a component in the as-built system is not easily represented by an instance.

SysML v2 includes the concept of a model of an individual system, such as an as-built system on the factory floor with a serial number. The individual system can include property values corresponding to its measured values that further over-ride the property values of the system design configuration. The individual model has its own lifetime where its structure can change over time, and its property values can change over time.

The structure concepts are intended to be applied to other parts of the language such as behavior to establish a consistent approach to define how elements are decomposed.

##### Structure Requirements

**Table 3.4. Structure Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| STC 1 | Structure Requirements Group | This group of requirements is intended to represent composable, deeply nested, connectible structure that supports definition of a family of configurations, specific configurations, and individual elements that are uniquely identified.  **Supporting Information:**  These requirements refer to definition elements and usage elements analogous to structured classifiers and classifier features in UML. A particular specialization of these concepts in SysML v1 is used to represent blocks and parts,  The requirements also refer to configuration elements and individual elements. Configuration elements are used to unambiguously represent deeply nested structures as a tree of configuration elements. Individual elements are used to represent a particular element that can be uniquely identified, which is not to be interpreted as a UML or SysML instance. A particular system, such as a system with a serial number on the manufacturing floor, can be represented by an individual element which in turn can be represented as a tree of individual elements.  The terms Component Definition and Component Usage refer to a particular kind of Definition Element and Usage Element that are analogous to a Block and Part in SysML v1. The terms Item Definition and Item Usage are also used to refer to a particular kind of Definition Element and Usage Element that correspond to something that flows through a system, such as Water. Component and Item are introduced in the Interface requirements section. |  |
| STC 1.01 | Modular Unit of Structure | SysML v2 shall include a capability to represent a modular unit of structure - called a Definition Element - that defines its characteristics through value properties, interface ends (ports), constraints, and other structural and behavioral features.  **Supporting Information:** Such modular units of structure can be regarded as the fundamental named building blocks from which system representations, i.e. architectures, can be constructed. The capability enables modeling multiple levels of a hierarchy (e.g., system-of systems, system, subsystem and components) that can include logical and physical representations of hardware, software, information, people, facilities, and natural objects.  The concept model refers many specializations of Definition Element. One example is the Component Definition which is intended to represent any level of a product structure. The concept model refers to an Item Definition as a specialized Definition Element to represent an element that flows through a system, such as water or a message. As noted above, the decomposition of Definition Elements may include variability that may be represented by multiplicity, subclasses, and/or a range of property values, which is removed when selecting a specific configuration. | Block |
| STC 1.02 | Usage Element | SysML v2 shall include a capability to represent the usage of a Definition Element, called a Usage Element, in order to support reuse in different contexts.  **Supporting Information:**  The Concept Model includes many specializations of Usage Element that include Component Usage and Item Usage. | Structural Feature, Behavioral Feature, ElementPropertyPath, NestedConnectorEnd |
| STC 1.03 | Generic Hierarchical Structure | SysML v2 shall include a capability to represent hierarchical composition structure of Definition Elements. | Composite Association, Part Property |
| STC 1.04 | Reference Element | SysML v2 shall include a capability to represent a reference from one element to any other element within a shared scope. | Reference Property, Reference Association |
| STC 1.05 | Multiplicity of Usage | SysML v2 shall include a capability to define the multiplicity of any particular Usage Element or Reference Element as an integer range (i.e., lower bound and upper bound).  **Supporting Information:**  Multiplicity refers to the number of Individual Elements. | Multiplicity on properties. |
| STC 1.06 | Definition Element Specialization | SysML v2 shall include a capability to represent a specialization from a more general Definition Element into a more specific Definition Element, where the more specific element inherits all features of the more general element. | Generalization/Specialization |
| STC 1.07 | Unambiguous Deeply Nested Structure | SysML v2 shall support a capability to unambiguously represent Usage Elements at any level of nesting.  **Supporting Information:** Deeply nested Usage Elements (more than one level) must be able to over-ride their definition with a localized definition. | ElementPropertyPath, NestedConnectorEnd, Redefinition, Subsetting |
| STC 1.08 | Structure With Variability | SysML v2 shall include a capability to represent multiple possible variant configurations of a system-of-interest with a single collection of Definition Elements and Usage Elements, where at each usage level in the (de)composition, a variant from different possible variant choices can be selected.  **Supporting Information:** A Structure With Variability enables the definition of a product line architecture, see e.g. ISO 26550. Some common variant choices are defined by multiplicity range. sub-classes, and different values of a value property. | Multiplicity of property, specialization of classifiers Redefined property, Subsetted property |
| STC 1.09 | Structure Resolved to a Single Variant | SysML v2 shall include a capability to represent a single variant of a system-of-interest as a tree of Configuration Elements that establishes a fully expanded hierarchical (de)composition that can conform to an associated Structure with Variability where a single selection is made for each variability choice (aka variation point).  **Supporting Information:** A SysML v2 implementation should support auto-generation of a tree of configuration elements from a decomposition of definition elements with variability based on a set of rules. A SysML v2 implementation should ideally also provide a capability to semi-automatically generate the reverse transformation from a tree of configuration elements to a decomposition of definition elements. | Sub-class with Redefinition, Subsetting, Property Specific Type |
| STC 1.10 | Structure of an Individual | SysML v2 shall include a capability to represent a (de)composition of an Individual Element that is uniquely identifiable, and that can conform to an associated Structure resolved to a Single Variant and/or a Structure with Variability.  **Supporting Information:** Such a digital representation of a real-world system is sometimes called a 'digital twin'. The elements in a Structure of an Individual are typically designated by a unique serial number, a batch number or an effectivity code. | Instance Specification |
| STC 1.11 | Usage Specific Localized Type | SysML v2 shall include a capability to represent local override, redefinition, or addition of features with respect to the features defined by its more general type at any level of nesting.  **Supporting Information:** The more-general to more-specific type chain is: Definition Element - direct Usage Feature - deeply nested Usage Feature - Configuration Element - Individual Element.  The localized usage should support capabilities equivalent to redefinition and sub-setting for usage elements at any level of nesting. | PropertySpecificType Redefinition, Subsetting |

#### Interfaces

##### Interface Introduction

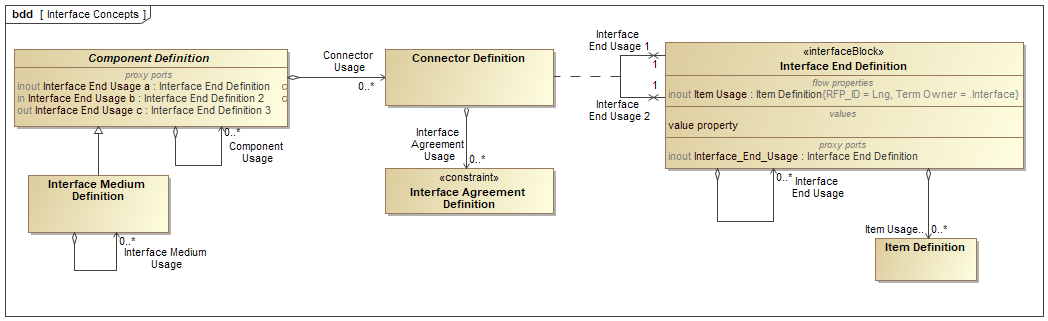
The goals for interface modeling include the ability to model a diverse range of interfaces (e.g., electrical, mechanical, software, user). The core concept of an interface includes 2 interface ends, an interface connection and any constraints related to connecting the ends as shown below.

Figure 3.26. SysML v2 Interface Concept



The concepts of definition and usage apply to the interface ends and interface connection consistent with SysML v1 (e.g., proxy port typed by interface block and full port typed by block). These concepts are being aligned with the structure modeling concepts described above to allow multiple levels of nested interfaces. SysML v2 also requires support for layered interfaces such as when modeling an Open Systems Interconnection (OSI) stack with application layers down to physical layers. Another important concept is an interface agreement, which can be used to specify and constrain the connection, interface ends, and any items that are exchanged. The interface agreement can specify behavioral constraints on the exchange of messages consistent with a defined protocol, and can specify physical constraints associated with a physical layer, such as the torque and angular rate constraints for a motor to gear interface. The interface concepts are highlighted in the figure below. An interface medium is a kind of component that provides interconnections that enable other components to interact. Examples include an electrical harness, a communication network, or a pipe to connect fluid flows.

Figure 3.27. Interface Concepts



One of the key issues with SysML v1 is the inability to readily support views of different abstraction levels of complex interfaces. For example, a desired view of a detailed multi-layer interface may show one layer of a stack and hide others. Another view may show highly detailed pinouts between electrical connectors. This ability to view different abstraction levels is being addressed by the visualization service requirements to facilitate the generation of views that support different stakeholder viewpoints, such as a software view of the application layer or a hardware view of the physical layer.

Similarly, it is often desirable to create an interface in the model, and then add more detail over time. Early in the design, the model may only represent a simple connector between two parts. Later, ports may be added, then data flows, interface agreements and intermediate systems such as networks. The requirements needed to address this issue are addressed by the model construction service requirements.

##### Interface Requirements

**Table 3.5. Interface Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| INF 1 | Interface Requirements Group | SysML v2 is intended to provide a robust capability to model interfaces that constrain the physical and functional interaction between structural elements. An interface in SysML v2 includes two (2) interface ends, the connection between them, and any constraints on the interaction. An interface should support the following:   1. Different levels of abstraction that include logical, functional, and physical interfaces, nested interfaces, and interface layers; 2. Diverse domains that include a combination of electrical, mechanical, software, and user interfaces; 3. Reuse of interfaces in different contexts; 4. Generation of interface control documents and interface specifications   **Supporting Information:**  The ability to construct and visualize different views of interfaces, including different abstraction levels, are addressed by the visualization and construction services.  A Port is also used to refer to an Interface End. |  |
| INF 1.01 | Interface Definition and Reuse | SysML v2 shall provide the capability to define an interface that can be used in different contexts that includes the definition of the interface ends, the interface connections, and the constraints on the interaction.  **Supporting Information:**  Interfaces must conform to the structural concepts of definition and usage. The constraints can constraint properties, such as conservation laws that can apply to a physical interface, and/or constraints on exchanged items such as protocol constraints that can apply to message exchange, and/or geometric constraints that can apply to a physical interface such as between a plug and socket. | Port Definitions including Interface Blocks and Blocks, Association and Association Blocks used to type Connectors, Item Flows, Constraints |
| INF 1.02 | Interface Usage | SysML v2 shall provide the capability to represent a usage of an interface that constrains the interaction between any two (2) structural elements. | Ports, Connectors, Parts |
| INF 1.03 | Interface Decomposition | SysML v2 shall provide the capability to represent nested interfaces, such as when modeling two electrical connectors with pin to pin connections. | Nested ports |
| INF 1.04 | Interface End Definitions | SysML v2 shall provide the capability to represent the definition of an Interface End whose features constrain the interaction of the end, including items that can be exchanged and their direction, behavioral features, and constraints on properties.  **Supporting Information:**  Interface End Definitions are also referred to as Port Definitions and Interface End Usages are referred to as Port Usages or Ports for short. | Interface Blocks with flow properties, value properties, and behavioral features |
| INF 1.05 | Conjugate Interface Ends | SysML v2 shall provide the capability to reverse the direction of the items that are exchanged in an Interface End. | Conjugate Ports |
| INF 1.06 | Item Definition | SysML v2 shall provide the capability to represent the kind of items that can be exchanged between Interface Ends.  **Supporting Information:** The items represent the type of things that are exchanged, such as water or electrical signals. The items may have physical characteristics such as mass, energy, charge, and force, and logical characteristics such as information that is encoded in the physical exchange.  In addition to being exchanged, these items may also be stored.  An item that is input to a component should become a stored item usage that can be transformed by function usages. An item, such as an engine that is an input and output of an assembly process, may also have the role as a component, when it is assembled into a vehicle.  Item Definitions must conform to the structural concepts of definition and usage. The rate at which a usage of an Item Definition is updated may be marked with an update rate that is continuous or discrete valued. (Refer to Behavior Requirement called "Discrete and Continuous Time Behavior") | Blocks, Signal |
| INF 1.07 | Interface Agreement Group |  |  |
| INF 1.07.1 | Item Exchange Constraints | SysML v2 shall provide the capability to constrain the interaction between the interface ends that includes constraints on the items to be exchanged, the allowable sequences and directions of those items, timing of the exchange and other characteristics. The items exchanged shall be consistent with the type and direction of the items specified in the connected Interface Ends. | Activities, state machines and sequence diagrams in an association block with participant properties |
| INF 1.07.2 | Property Constraints | SysML v2 shall provide the capability to constrain the interaction between the interface ends that include mathematical constraints on the properties exposed by the Interface Ends.  **Supporting Information:** The value properties may further be marked as Across or Through variables consistent with standard usage of the terms (e.g. across and through variables, for specifying properties that are constrained by conservation laws).  Refer to Properties, Values and Expression requirement called "Across and Through Value Properties" | Parametric diagrams that specify constraints on the ends of an association block with participant properties. |
| INF 1.07.3 | Geometric Constraints | SysML v2 shall provide the capability to constrain the interaction between the interface ends that include geometrical constraints on either Interface End.  **Supporting Information:** An example are the geometric constraints associated with connecting a plug and socket. |  |
| INF 1.08 | Interface Medium | SysML v2 shall include a capability to represent an Interface Medium that enable 2 or more components to interact.  **Supporting Information:** The Interface Medium may represent either an abstract or physical element that connects elements to enable interactions. Examples of an interface medium included an electrical harness, a communications network, a fluid pipe, the atmosphere, or even empty space. The interface medium may connect one to many components, which include support for peer-to-peer, multi-cast, and broadcast communications.  Consider replacing the term Interface Medium with Transport Medium. | A block with a user-defined stereotype indicating its special function.  Also, a connector typed by an Association Block that has part properties, e.g. hotwater:Pipe and coldwater:Pipe. |
| INF 1.09 | Interface Layers | SysML v2 shall provide the capability to represent interfaces between layers of an interface stack.  **Supporting Information:**  A layer of a stack can be represented as a component. A layer in a stack transforms the data to match the input to the adjacent layer. For example, an application layer may correspond to a component that transforms packets to match the TCP layer, and the TCP layer may correspond to a component that transforms the data to match the IP layer. | Complex combination of the ports and flow concepts. |
| INF 1.10 | Allocating Functional Exchange to Interfaces | SysML v2 shall provide the capability to allocate or bind the outputs and inputs of a function to interface ends (or nested interface ends), and validate the consistency between the inputs and outputs of a function and the interface ends.  **Supporting Information:**  This allocate or binding should be inherited by the Component subclasses. | On Port metaproperty. |

#### Behavior

##### Behavior Introduction

Behavior modeling concepts describe the interaction between individual structural elements and their change of state over time.

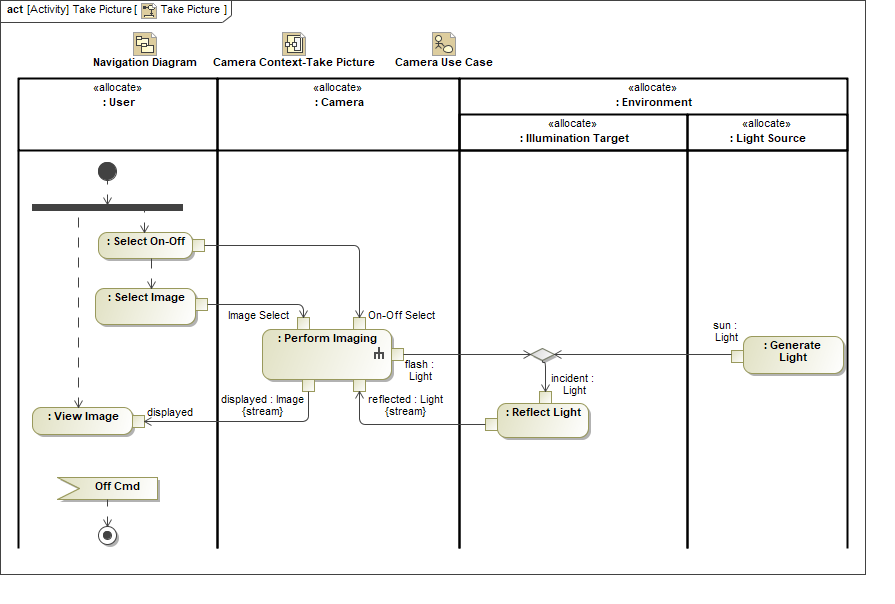
Some of the SysML v1 issues that should be addressed by SysML v2 behavior models include:

* Lack of integration between structure and behavior (e.g., action inputs and outputs, flow properties, item flows, input/output of do/behavior)
* Lack of integration between different kinds of behavior (seq, act, stm, uc, timing)
* Improved support for continuous flow semantics
* Inconsistent decomposition semantics between activity decomposition on a bdd and activity decomposition using call behavior actions
* Inability to decompose inputs and outputs I/O and how they are input to or output from different actions and/or activities
* Lack of representation of timelines
* Inability to easily have actions that create and destroy relationships such as connectors
* More flexible and expressive pre/post conditions
* Support for software architecture concepts such as threads
* Complexity of the relationship between operations and actions/activities
* Inability to represent a reference activity that spans activity partitions in an activity diagram similar to reference interactions in a sequence diagram

As noted above, one of the critical issues in SysML v1 is the limitations associated with integrating behavior and structure with respect to input and outputs of activities, item flows on connectors, and flow properties on ports. The key requirement for SysML v2 is to seamlessly integrate behavior and structure by integrating the behavior inputs and outputs with component interfaces.

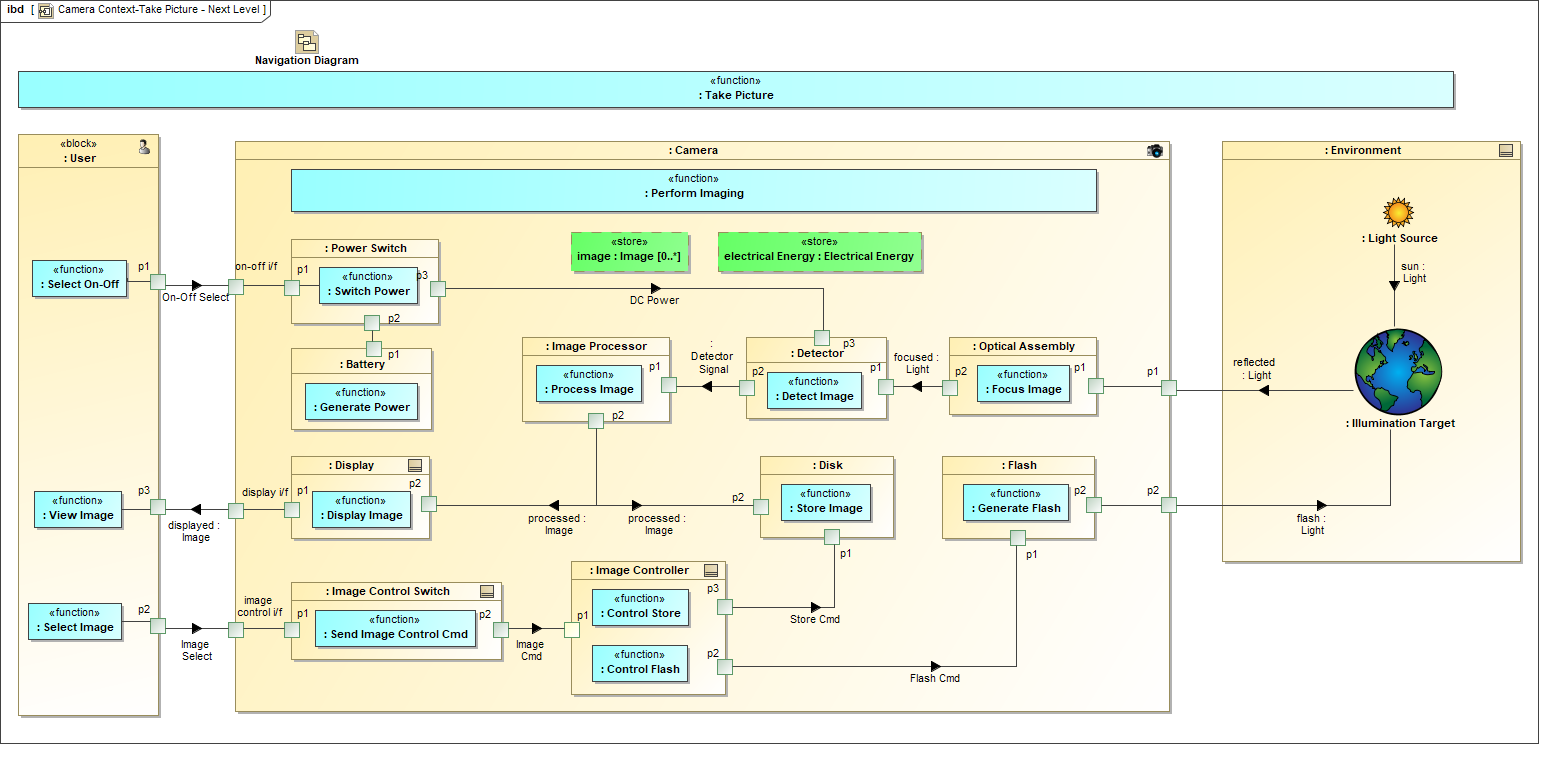
The Camera model example below illustrates the need for this integration. In the first figure, the Take Picture activity integrates structure and behavior by representing the actions that the User, Camera, and Environment perform to take a picture. The Camera accepts Light as an input from the Environment and the User inputs, and executes the Perform Imaging action to produce an Image to present to the User and store in the Camera (not shown).

Figure 3.28. Top Level Behavior - Take Picture



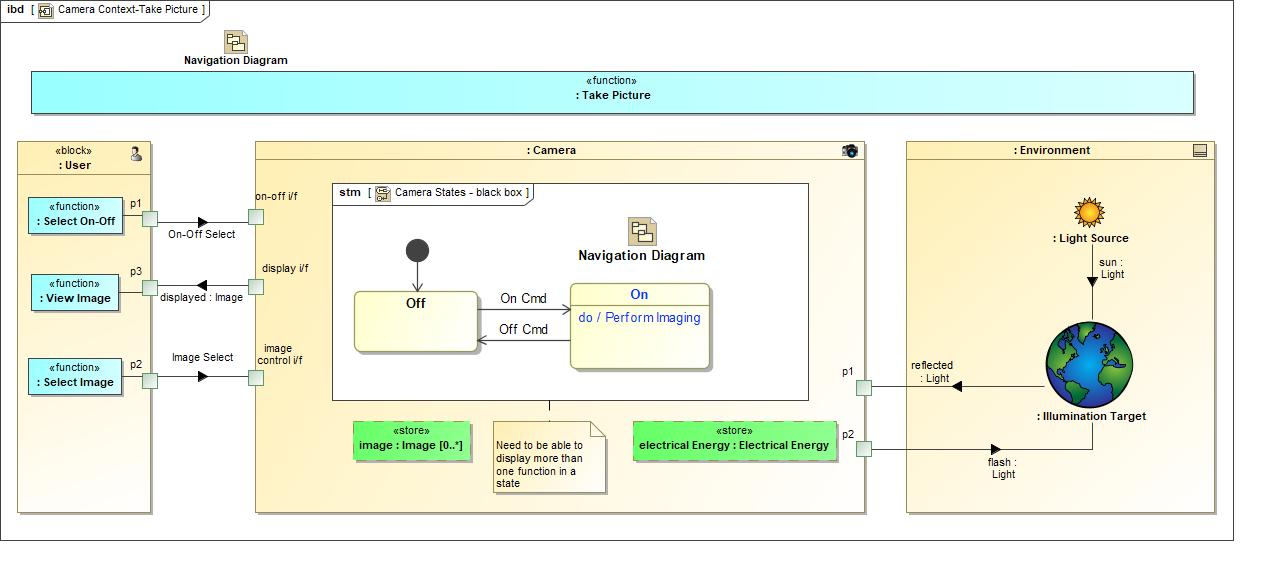
In the figure below, the internal block diagram highlights the interconnection of the parts of the camera with their allocated actions (i.e., functions). SysML v2 must enable the straightforward construction and presentation of different views of the model while maintaining consistency between the activities, their inputs and outputs, the flow properties on the ports (not shown), and the item flows on the connectors.

Figure 3.29. Camera Context with Take Picture Functions



Another important view is the decomposition of the activities without representing the flow of control and flow of inputs and outputs. SysML v1 supports this capability, but does require an additional concept called an adjunct property to establish correspondence between the activity decomposition and the actions in an activity diagram. In SysML v2, behavior decomposition should apply the same modeling concepts that apply to structure decomposition to simplify and integrate the language. Another limitation in SysML v1 is that a do-behavior of a state machine, such as the Perform Imaging activity in the figure below, can only accept inputs and produce outputs by sending and receiving signals and by performing read/write actions to properties of the owning block. SysML v2 must ensure that activities are not constrained as to how they use state machines to integrate with activity inputs and outputs.

Figure 3.30. Camera Context with State Machine



##### Behavior Requirements

**Table 3.6. Behavior Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| BHV 1 | Behavior Requirements Group |  |  |
| BHV 1.01 | Behavior | SysML v2 shall include the capability to model a Behavior that represents the interaction between individual structural elements and their change of state over time. | Activity, State Machine, Interaction, Simple Time |
| BHV 1.02 | Behavior Decomposition | SysML v2 shall include the capability to decompose a behavior to any level of decomposition, and to define localized usages of behavior at nested levels of decomposition.  **Supporting Information:**  The decomposition of behavior should conform to a similar pattern as the decomposition of structure, and include capabilities for specialization, redefinition, and sub-setting.  The decomposition should also include the equivalent capability to decompose a SysML v1 activity on a BDD, and the ability to decompose actions using a structured activity node. | Composited Association of Behavior Classifiers with Adjunct Properties |
| BHV 1.03 | Function-based Behavior Group |  |  |
| BHV 1.03.1 | Function-based Behavior | SysML v2 shall include the capability to represent a controlled sequence of actions (or functions) that can transform a set of input items to a set of output items.  **Supporting Information:**  SysML v2 should provide an integrated approach to specify behavior that reflects similar capabilities to SysML v1 activities and sequence diagrams, which are expected to be different views of the same underlying model.  The input items and output items correspond to item usages and their associated value properties whose values can vary over time. Item flows connect an output item usage to an input item usage.  The start and stop events should be represented explicitly (e.g., control pins). Event flows connect a stop event to a start event.  Supporting Information:  The specific features of activities and sequence diagrams to be included in SysML v2 beyond what is specified in this section should be defined in the proposal. | Activity, Interaction, Object Flow, Control Flow |
| BHV 1.03.2 | Composite Input and Output | SysML v2 shall include the capability to model composite inputs and outputs of function-based behavior with separate flows defined for the constituent inputs and outputs.  **Supporting Information:**  Refer to a Simulink Bus Object and a Modelica Expandable Connector |  |
| BHV 1.03.3 | Function-based Behavior Constraints | SysML v2 shall include the capability to model constraints on a function-based behavior that includes the ability to represent a declarative specification in terms of its pre-conditions and post-conditions, and any constraints that apply throughout execution of the behavior. | Pre and post conditions |
| BHV 1.03.4 | Opaque Behavior | SysML v2 shall include the capability to represent a behavior that embeds the definition in a language such as a programming language. | Opaque Behavior |
| BHV 1.03.5 | Behavior Library | SysML v2 shall include a library that can be populated with commonly used behaviors to support execution that includes functions to store items, such as data and energy. | FUML actions library |
| BHV 1.03.6 | Structure Modification Behavior | SysML v2 shall include the capability to represent behaviors that can modify the structure of an element over time, such as the creation and destruction of interconnections and composition.  **Supporting Information:**  An example is the behavior associated with the separation of a first stage rocket, or the assembly or disassembly of a product. | Primitive Actions |
| BHV 1.04 | State-based Behavior Group |  |  |
| BHV 1.04.1 | Regions, States, and Transitions | SysML v2 shall include the capability to represent the state behavior of a structural element in terms of its concurrent regions with mutually exclusive finite states, and transitions between finite states.  **Supporting Information:** A state change can result from a change in structure. | State Machine |
| BHV 1.04.2 | Integration of Function-based Behavior with Finite State Behavior | SysML v2 shall include the capability to model function-based behavior both on transitions between finite states, and upon entry, exit, and while in a finite state. | Entry, Exit, Do Behavior and Transition effect |
| BHV 1.04.3 | Integration of Constraints with Finite State Behavior | SysML v2 shall include the capability to model constraints both on transitions between finite states, and upon entry, exit, and while in a finite state. | State invariants |
| BHV 1.05 | Discrete and Continuous Time Behavior | SysML v2 shall include the capability to model behaviors whose inputs and outputs vary continuously as a function of time, or discretely as a function of time. | Continuous, streaming |
| BHV 1.06 | Events | SysML v2 shall include the capability to model signal events, time events, and change events and their ordering.  **Supporting Information:** The ordering of actions (i.e., functions) is accomplished through ordering of their start and completion events. Events can trigger a change from one finite-state to another. Events should be able to be explicitly represented in both function-based behavior and finite-state behavior. Events can be defined and used in different contexts. | Triggering events on state machines, accept event actions, send signal actions |
| BHV 1.07 | Control Nodes | SysML v2 shall include the capability to model control nodes that specify a logical expression of conditions and events to enable a flow.  **Supporting Information:** For Example: {Inputs A < a1 AND B>=b2 OR C AND NOT D} must be true). | Join, Fork, Merge, Decision, Join Specification |
| BHV 1.08 | Time Constraints | SysML v2 shall include the capability to specify the absolute or relative time associated with an event that includes start events, stop events, and duration constraints between events to represent the time-line associated with a behavior.  **Supporting Information:** Time is a property typed by a Value Type whose quantity kind and units are specified as part of QUDV. | Simple Time |
| BHV 1.09 | State History | SysML v2 shall provide the capability to represent a state history of an individual element as a sequence of snapshots to describe how the individual element changes over time. . The state history shall contain a reference time scale consistent with QUDV, and can include a start time, end time, and time step.  **Supporting Information:**  A snapshot represents the state of an individual element at a point in time by capturing the values of each of its value properties. An example is a snapshot of a vehicle that may include the value of its position, velocity, and acceleration at a point in time, and the snapshot of its engine that may include the value of its power-out and temperature at the same point in time. The value properties that vary with time are also called state variables.  The state history of a configuration element represents the default state history for each of its conforming individual elements. |  |
| BHV 1.10 | Behavior Execution | SysML v2 shall include the capability to execute function-based and state-based behavior to specify the state history of individual elements and their interactions with other individual elements.  **Supporting Information:** The behavior of a Definition Element or Configuration Element represent the default behavior of the conforming Individual Elements. | FUML |
| BHV 1.11 | Integration between Structure and Behavior |  |  |
| BHV 1.11.1 | Allocation of Behavior to Structure | SysML v2 shall include the capability to represent the behavior of one or more structural elements.  **Supporting Information:**  This should support the ability to define a state machine of a structural element, with finite states that enable actions (i.e., functions) and constraints. In addition, this should support the ability to specify the functions performed by a component, and the applicable constraints, without specifying the finite state that enables them. The representation should allow more than one structural element to perform a single function, such as when two people carry a load. This is analogous to a reference interaction in a SysML v1 sequence diagram that spans multiple lifelines and displays the participating lifelines. The reference interaction refers to another sequence diagram. | Allocate, Allocated Activity Partition, Structured Activity Node, Reference Interaction |
| BHV 1.11.2 | Integration of Control Flow and Input/Output Flow | SysML v2 shall ensure that inputs, outputs, and events can be represented consistently across behavior and structure.  **Supporting Information:**  In SysML v1, it is often difficult to ensure consistent representation of control flow and input/output flow. Examples include potential inconsistencies between:   * Flows on activity diagrams and messages on sequence diagrams. * Flows on activity diagrams and item flows on ibd * Inputs and outputs on activity diagram and corresponding inputs and outputs on activity decomposition on a bdd * Inability to represent input/output of activities on do behaviors of state machines | Adjunct properties,  On Port |
| BHV 1.12 | Case | SysML v2 shall include the capability to represent a case that can be specialized into a use case, verification case, analysis case, and domain specific cases, such as safety case and assurance case.  **Supporting Information:** A case is a series of steps with an associated objective that produce a result or conclusion. An analysis case and assurance case correspond to a set of steps to implement a study or investigation. |  |

#### Requirements

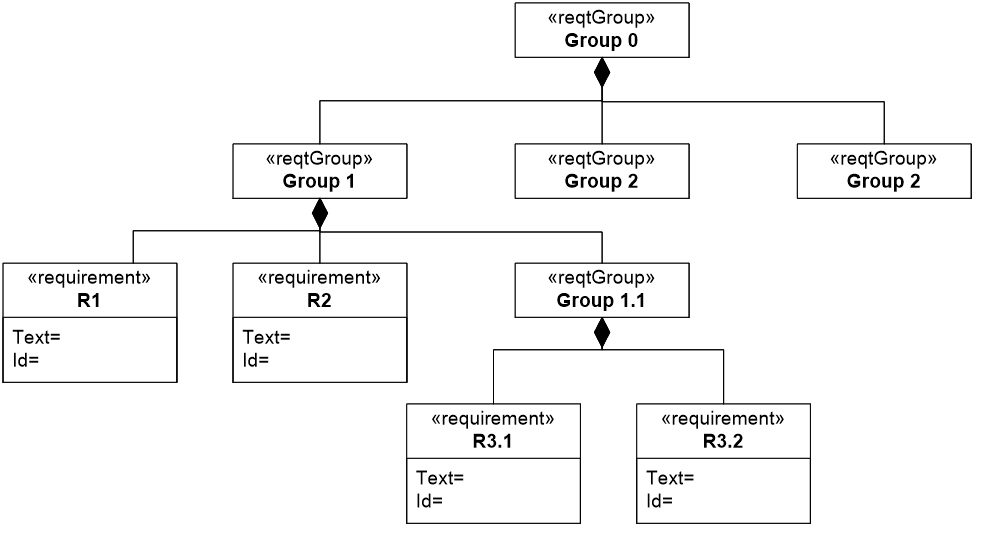
##### Requirement Introduction

SysML v1 includes text-based requirements and requirements relationships with other requirements, design, analysis, and verification elements. It also provides a means to construct requirements hierarchies that correspond to the requirements in a specification. Some of the limitations associated with SysML v1 requirements include the inability to automatically verify a text requirement by analysis (although this is being done in a non-standard way), the ability to group requirements, and the ability to easily reuse a requirement. In addition, the standard attributes of requirements are limited to text and id, and do not include many commonly used requirements attributes and categories.

SysML v2 will include a Requirement Group as a container for requirements that can include additional context information for the requirements contained in the group. The Requirement Group does not contain a "shall statement". In the figure below, Group 1 contains requirements R1 and R2 and Group 1.1 contains requirements R3.1 and R3.2. A logical expression can be applied to the members of a Group. For example, Group 1 can contain R1 AND R2 AND (R3.1 OR R3.2). A requirement can also be contained in more than one Requirement Group to enable reuse of the requirement.

In SysML v1, a requirement can be decomposed into other requirements. In SysML v2, the requirement group is used to decompose requirements.  A compound text requirement is decomposed into multiple individual requirements by defining a requirement group that contains the individual requirements. The original compound requirement is refined by the requirement group that contains the individual requirements.

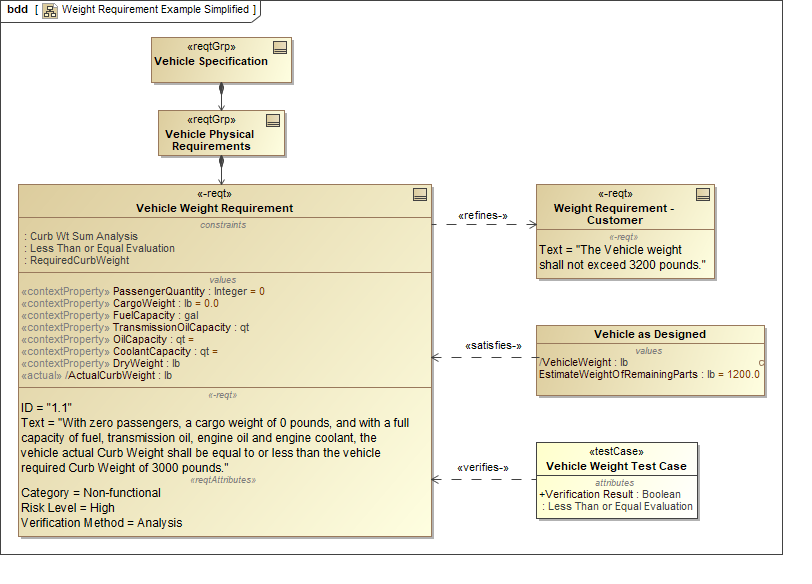
Figure 3.31. Requirement Groups



Text requirements can easily be misinterpreted. In SysML v1.5, the SysML requirement was refactored to enable both text based requirements and more precise statements of requirements, commonly referred to as property based requirements. SysML v2, is intended to represent property-based requirements by enabling requirements to contain formal expressions that constrain property values, such as weightActual<1000 kilograms. More complex expressions can also be specified to impose constraints on design solutions such as the required vehicle stopping distance as a function of speed and road conditions.

The weight requirement in the figure below illustrates some of the requirements concepts including the requirement group, property based requirements, and requirement attributes and categories. The SysML v2 requirement contains the formal expression that are specified as constraints, the properties that are being constrained, along with the id and text statement. The SysML v2 requirement also includes the ability to specify assumptions, such as whether the weight of oil, gas, and other fluids are included in the vehicle weight requirement. The assumptions are represented by context properties that can have values assigned or constrained.

Figure 3.32. Example Weight Requirement

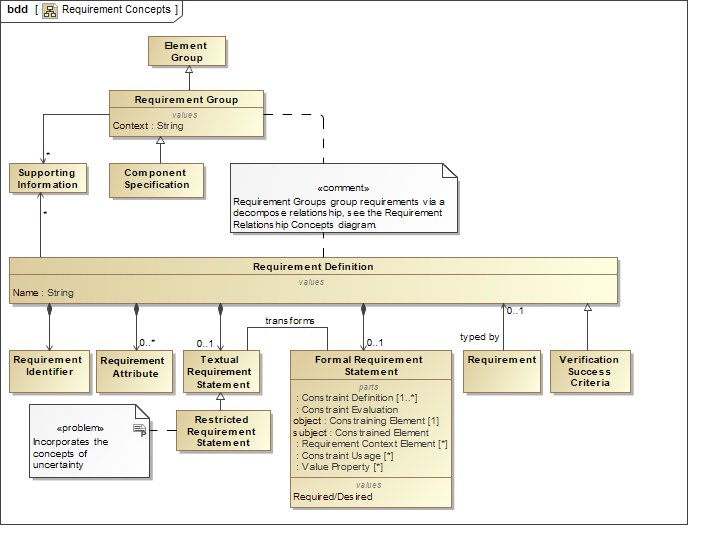


SysML v2 also supports the concept for restricted natural language text to specify requirements more precisely.

SysML v2 applies the structural concepts to both reuse requirements, and unambiguously define a tree of requirements. A generic structure of requirements can be defined using requirements groups that contain requirement definitions. An unambiguous hierarchy of requirements can be defined for a specific design configuration.  A tree of individual requirements can specify an individual element.

The SysML v2 requirement will also include standard requirement attributes and requirement categories based on the INCOSE Handbook. Some of the SysML v1.2 requirements concepts are highlighted in the figure below.

Figure 3.33. Requirement Concepts



##### Requirements for Requirements

**Table 3.7. Requirements for Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| RQT 1 | Requirement Group | The requirements in this group are used to represent requirements and their relationships. |  |
| RQT 1.1 | Requirement Definition Group |  |  |
| RQT 1.1.1 | Requirement Definition Name | SysML v2 shall include a capability to represent a requirement definition that can be used to constrain a solution. | Requirement Name |
| RQT 1.1.2 | Requirement Identifier | SysML v2 shall include a capability to represent an identifier for each requirement that is adaptable to a user defined numbering scheme, and can be set to not change. | Requirement ID |
| RQT 1.1.3 | Requirement Attributes | SysML v2 shall include a capability to represent the following optional requirement attributes for a requirement definition.   * Requirement Identifier (conforms to a user specified naming and identifier production scheme) * Requirement Status * Priority * Risk * Originator/Author * Owner * User-defined Attributes (e.g., confidence level, uncertainty status, etc.)   **Supporting Information:** These attributes are derived from commonly used attributes as defined in the INCOSE Handbook and ReqIF, and should be reconciled with other model element metadata and model element attributes that apply more generally. | Non Normative extensions |
| RQT 1.1.4 | Textual Requirement Statement | SysML v2 shall include a capability to represent a requirement definition that contains an optional textual requirement statement. | Requirement text statement |
| RQT 1.1.5 | Restricted Requirement Statement Group |  |  |
| RQT 1.1.5.1 | Restricted Requirement Statement | SysML v2 shall include a capability to represent a requirement definition that contains an optional restricted requirement statement which may include predefined key words and sentence structures. |  |
| RQT 1.1.5.2 | Restricted Requirement Statement Extensibility | SysML v2 shall include a capability to extend a restricted requirement statement with additional key words and sentence structures. |  |
| RQT 1.1.5.3 | Restricted Requirement Statement Transformation | SysML v2 will include a capability to maintain consistency between the restricted requirement statement and the textual requirement statement and/or the formal requirement statement. |  |
| RQT 1.1.6 | Formal Requirement Statement Group |  |  |
| RQT 1.1.6.1 | Formal Requirement Statement | SysML v2 shall include a capability to represent a requirement definition that contains an optional formal requirement statement that includes one or more constraints that an acceptable solution must satisfy.  **Supporting Information:** It is desired to also enable the element that is intended to satisfy the requirement to contain the formal requirement statement. This can provide a more lightweight modeling style. | Non-normative extension for a property based requirement |
| RQT 1.1.6.2 | Assumptions | SysML v2 shall include a capability to represent a formal requirement statement that includes one or more expressions to specify the assumptions and conditions for acceptable solutions (e.g., the weight of a car includes the fuel weight)  **Supporting Information:** This should be consistent with the concept of Assumption that is applied in other parts of the model. |  |
| RQT 1.2 | Groups of Requirements |  |  |
| RQT 1.2.1 | Requirement Group | SysML v2 shall provide the capability to model a group of requirements that are used to constrain a solution.  **Supporting Information:** This is intended to be a sub-class of Element Group. | Requirement |
| RQT 1.2.2 | Requirement Usage (localized) | SysML v2 shall include a capability to represent localized values of a requirement usage that can over-ride the values of its requirement definition.  **Supporting Information:** The structural concepts of definition, usage, configuration, and individuals are intended to support reuse of requirement definitions, and unambiguously define a tree of requirements that specify a design configuration or an individual element. |  |
| RQT 1.2.3 | Requirement Usage Identifier | SysML v2 shall include a capability to represent each requirement in a requirement group with an identifier that is adaptable to a user defined numbering scheme, and that the user can specify whether the identifier can change or not. | Requirement ID |
| RQT 1.2.4 | Requirement Usage Ordering | SysML v2 shall include a capability to represent the order of each requirement in a requirement group that is not constrained by its requirement identifier.  **Supporting Information:** This primarily allows the user to further organize the requirements, but it does not impact the meaning of the requirements. For example, there may be a requirement group with one requirement to open a value and another requirement to close a valve. The user may want to order the open requirement as the first requirement in the group. |  |
| RQT 1.3 | Requirement Relationships Group |  |  |
| RQT 1.3.1 | Requirement Specialization | SysML v2 shall include a capability to represent a generalization relationship that relates a specialized requirement definition to a more general requirement definition. |  |
| RQT 1.3.2 | Requirement Satisfaction | SysML v2 shall include a capability to represent a satisfy relationship that relates a requirement to a model element that is asserted to satisfy it.  **Supporting Information:** This is intended to be a specialization of the Conform Relationship. | Satisfy |
| RQT 1.3.3 | Requirement Verification | SysML v2 shall include a capability to represent a verify relationship that relates a verification case to the requirement it is intended to verify. | Verify |
| RQT 1.3.4 | Requirement Derivation | SysML v2 shall include a capability to represent a derive relationship that relates a derived requirement to a source requirement. | DerivedRequirement |
| RQT 1.3.5 | Requirement Group Relationship | SysML v2 shall include a capability to represent a relationship between a requirement group and the members of the group that can include either a requirement or another requirement group.  **Supporting Information:** This relationship groups requirements into a shared context. |  |
| RQT 1.3.6 | Relationships to a Requirement Group | SysML v2 shall include a capability to apply requirement relationships that relate to a requirement group to each member of the requirement group.  **Supporting Information:** This applies more generally to element groups. |  |
| RQT 1.3.7 | Relationship Logical Constraint | SysML v2 shall include a capability to represent a logical expression (e.g. AND, OR, XOR, NOT, and conditional expressions like IF-THEN-ELSE and IF-AND-ONLY-IF) to one or more requirement relationships of the same kind, with an associated completeness property (e.g., complete satisfaction or partial satisfaction) and with a default expression of "And" for the logical expression.  **Supporting Information:** As an example, two blocks that have a satisfy relationship with the same requirement are asserted to completely satisfy the requirement by default |  |
| RQT 1.4 | Requirement Supporting Information | SysML v2 shall include a capability to represent supporting information for a requirement, requirement definition, and a requirement group.  **Supporting Information:** This is a kind of annotation that applies more generally to any model element. |  |
| RQT 1.5 | Goals, Objectives, and Evaluation Criteria | SysML v2 shall include a capability to represent goals, objectives, and evaluation criteria.  **Supporting Information:**  Criteria can be viewed as a superclass of a requirement that is used as a basis for evaluation, but does not specify specific values. For example, a cost requirement may be to require the cost to be less than a particular value, where-as a cost criterion may be to select a design with the lowest cost. Goals can be a type of criteria. For example, a goal of the system is to minimize the cost. An objective represents a desired end state. For example, the mission objective is to land a person on the moon and safely return them to earth. An objective can be thought of as a kind of requirement. |  |

#### Verification & Validation

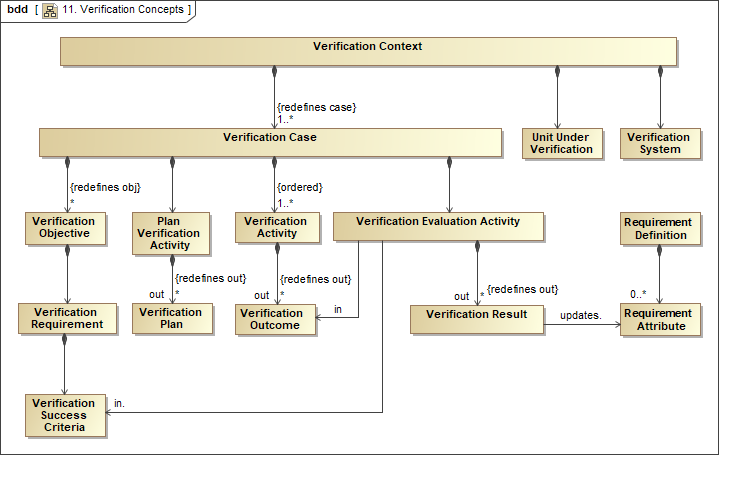
##### Verification & Validation Introduction

SysML v2 can be used to support verification planning and execution by modeling the verification system, and how the verification system is used to verify that the system and component requirements are satisfied. The verification concepts shown in the figure below are intended to integrate with the requirements concepts, and include verification objectives and criteria, a verification case, the verification system, the unit under verification, and the verification result.

Validation concepts are also intended to be similarly supported. Whereas verification is used to determine the extent to which a requirement is satisfied by the design, validation is used to determine the extent to which a requirement supports the need or a higher level requirement. Similar concepts can be applied to both. The intent is for the concepts to be adapted to support both verification and validation. For example, verification methods and validation methods can be defined.

A Verification Case (and Validation Case) generalizes the Test Case from SysML v1, and is used to specify how a requirement is satisfied (or validated). This Verification Case includes the Verification Activity that produces the Verification Outcome (e.g., test data). The Verification Evaluation Activity evaluates how the Verification Outcome satisfies the Verification Success Criteria and associated requirements to produce a Verification Result (e.g., pass/fail).  These concepts are generally specialized from other foundational concepts.

Figure 3.34. Verification Concepts



##### Verification & Validation Requirements

**Table 3.8. Verification & Validation Requirements**

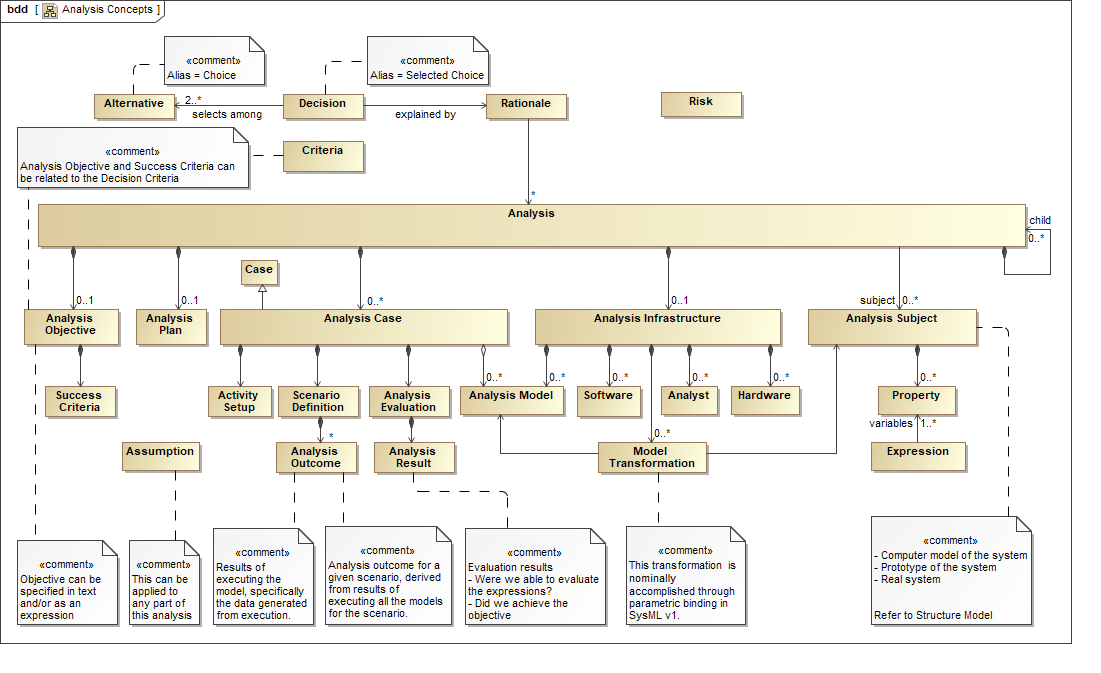
| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| VRF 1 | Verification and Validation Requirements Group | The requirements in this group represent how to evaluate whether systems satisfy their requirements using verification methods.  **Supporting Information:** The requirements for validation are not called out explicitly, but are intended to be supported in a similar way as the requirements for verification. |  |
| VRF 1.1 | Verification Context | SysML v2 shall include the capability to model a Verification Context that includes the unit-under-verification, the verification case, and the verification system and associated environment that performs the verification. |  |
| VRF 1.2 | Verification Case Group |  |  |
| VRF 1.2.1 | Verification Case | SysML v2 shall include the capability to model a verification case to evaluate whether one or more requirements are satisfied by a unit under verification.  **Supporting Information:** This is intended to be a specialization of Case. | Test Case |
| VRF 1.2.2 | Verification Objectives | The verification case shall include verification objectives to be implemented by the verification activities. |  |
| VRF 1.2.3 | Verification Success Criteria | The verification case shall include the criteria used to evaluate whether the verification objectives are met and the requirements are satisfied. |  |
| VRF 1.2.4 | Verification Methods | The verification case shall include the methods used to verify the requirements. The methods, including inspection, analysis, demonstration, test, external verification, engineering reviews, and similarity, shall be included in a library. More than one method can be applied to verify a requirement.  **Supporting information:**  A verification method may include additional classification such as qualification test and acceptance test.  An external verification is a method used in some industries, such as an Underwriters Labs. |  |
| VRF 1.2.5 | Verification Activity | The verification method shall include activities to collect the verification data, and include the ability to reference this data.  **Supporting Information:** The data may be extensive and not captured directly in the model. |  |
| VRF 1.2.6 | Verification Evaluation Activity | The verification method shall include activities to evaluate the verification data and the verification success criteria and generate a verification result of how well the requirements are satisfied (e.g., pass/fail/unverified). |  |
| VRF 1.3 | Verification System | SysML v2 shall include the capability to model the system and associated environment that is used to verify the unit under verification. (Note: the verification system may include verification elements that are combinations of operational and simulated hardware, software, people, and facilities.) |  |
| VRF 1.4 | Verification Relationships Group |  |  |
| VRF 1.4.1 | Verification Objectives to Verification Cases | SysML v2 shall include the capability to model relationship between the verification cases and their verification objectives. |  |
| VRF 1.4.2 | Validate Relationship | SysML v2 shall include the capability to model the relationship between the validation case and the model element being validated.  **Supporting Information:** An element being validated may represent a requirement, design, as-built system, model, etc.  The Verify Relationship is included in the requirements section. |  |

#### Analysis

##### Analysis Introduction

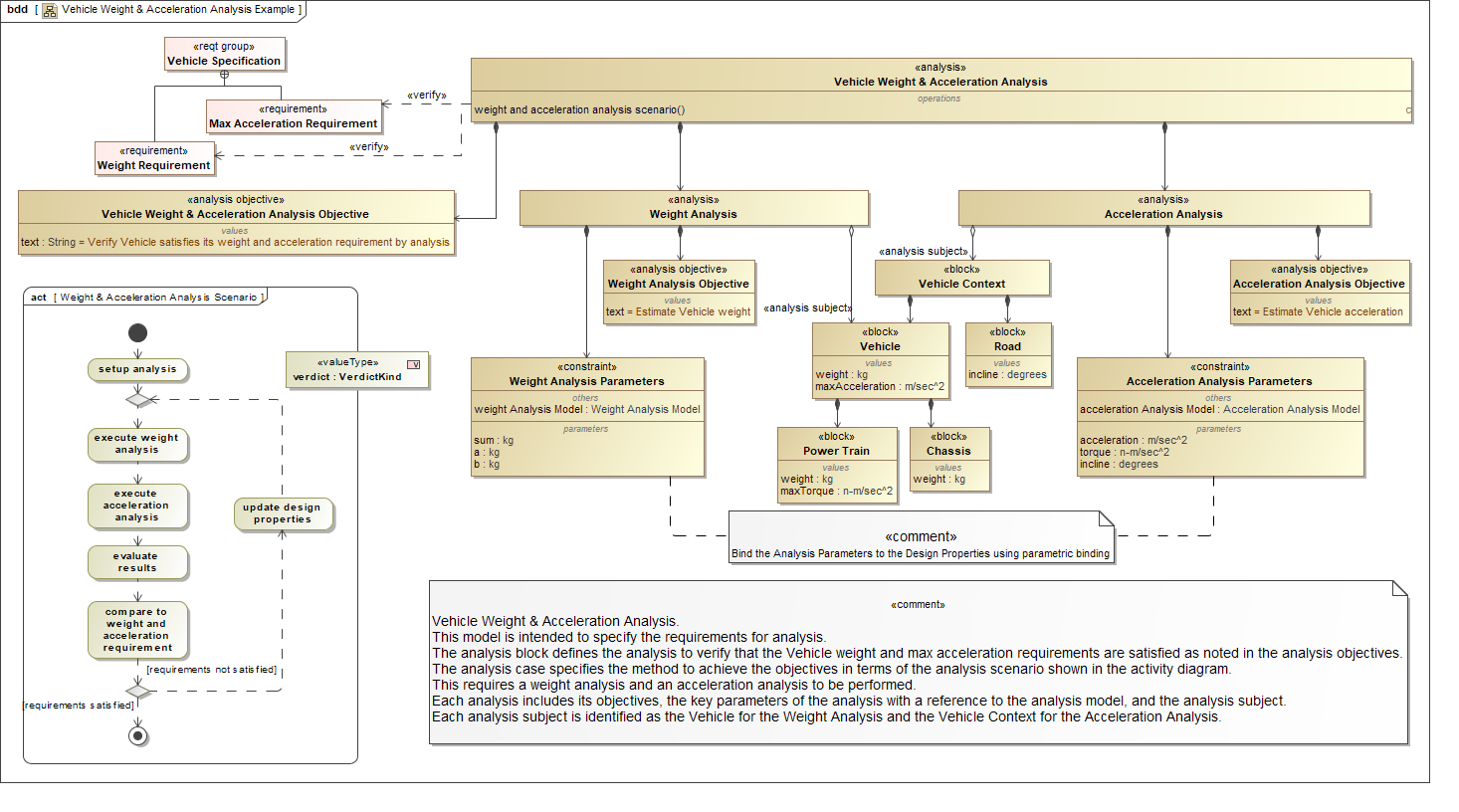
Although SysML v1 includes parametrics, it lacks some essential concepts to facilitate integration with analysis models. These concepts include concepts such as analysis, analysis case, analysis scenarios, subject of the analysis, analysis model, and analysis results. The analysis concepts for SysML v2 are shown in the figure below, and are similar to the verification concepts described in the previous section. Similar to the verification concepts, these concepts are generally specialized from other foundational concepts.

Figure 3.35. SysML v2 Analysis Concepts



The example in the figure below illustrates how these concepts can be applied to specify the requirements for the Vehicle weight and acceleration analysis. The analysis block contains the analysis objectives to verify that the Vehicle weight and acceleration satisfy its requirements. The analysis case specifies the analysis scenario in an activity diagram that can require the execution of multiple models to achieve the analysis objectives. The scenario in the figure requires a weight analysis and an acceleration analysis to be performed. Each of these analyses is part of the Vehicle Weight & Acceleration Analysis block, which include their objectives, the key parameters of the analysis, and the subject of the analysis. The subject of the analysis is identified as the Vehicle for the Weight Analysis, and the Vehicle Context for the Acceleration Analysis. The constraints contain the key parameters, and also include a reference to the analysis models that execute the analysis. These concepts are used to specify the analysis and integrate with the appropriate analysis models and tools to perform the analysis.

Figure 3.36. Vehicle Weight & Acceleration Analysis



SysML v2 also includes a set of analysis services that leverage the analysis concepts described in this section to further support integrating SysML with analysis. The analysis service requirements are described in the Services section below.

##### Analysis Requirements

**Table 3.9. Analysis Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| ANL 1 | Analysis Requirements Group | The requirements in this group are used to specify an analysis, along with other requirements such as Properties, Values, and Expressions. |  |
| ANL 1.01 | Analysis | SysML v2 shall include the capability to specify an Analysis, including the subject of analysis (e.g., system), the analysis case, and the analysis models and related infrastructure to perform the analysis. |  |
| ANL 1.02 | Subject of the Analysis | SysML v2 shall include the capability to model the relationship between the analysis and the subject of the analysis (system being analyzed). |  |
| ANL 1.03 | Parameters of Interest | SysML v2 shall include the capability to identify the key parameters of interest including measures-of-effectiveness (MoE) and other key measures of performance (MoP). | Value Property, MoE |
| ANL 1.04 | Analysis Case | SysML v2 shall include the capability to model the analysis case to specify the analysis scenarios and associated analysis methods needed to produce an analysis result that achieves the analysis objectives.  **Supporting Information:** This is intended to be a specialization of Case. |  |
| ANL 1.05 | Analysis Objectives | SysML v2 shall include the capability to model the objective of the analysis being performed in text or as a mathematical formalism, e.g. math expression, so that it can be evaluated. |  |
| ANL 1.06 | Analysis Scenarios | SysML v2 shall include the capability to model the scenarios that identify the analysis models to be executed, the conditions and assumptions, and the configurations of the subject of the analysis and the related infrastructure to perform the analysis. |  |
| ANL 1.07 | Analysis Assumption | SysML v2 shall include the capability to model the assumptions of the analyses in a text or mathematical form, e.g. constraints and boundary conditions. |  |
| ANL 1.08 | Analysis Decomposition | SysML v2 shall include the capability to decompose an analysis into constituent analyses. |  |
| ANL 1.09 | Analysis Model | SysML v2 shall include the capability to specify an analysis model.  **Supporting Information:** Analysis models can be defined natively in SysML (e.g. parametric model or behavior model) or externally (e.g. equation-based math models, finite element analysis models, or computational fluid dynamics models). The level of fidelity of the specification of the analysis model can vary from an abstract specification that defines the intent of the analysis including its input and output parameters, to a detailed specification that a particular solver can execute. |  |
| ANL 1.10 | Analysis Model - System Model Transformation | SysML v2 shall include the capability to represent the transformation and the mapping between the analysis model and the system model.  **Supporting Information:**  **T**his transformation will represent the algorithm or derivation process, if used, for generating analysis models from system model (or vice versa), and the mapping will provide a mechanism to verify and synchronize analysis models when the system model changes (or vice versa). Refer to the requirement for Model Mappings and Transformations under the Language Architecture and Formalism Requirements. |  |
| ANL 1.11 | Analysis Result | SysML v2 shall include the capability to relate the results of executing analysis models to the analysis.  **Supporting Information:** The results may be stored in the SysML v2 model itself or an in an external store (e.g. CSV file or database). The results will be queried to evaluate how well the analysis objectives are satisfied, and to obtain the supporting rationale for decisions taken based on the analysis. |  |
| ANL 1.12 | Analysis Infrastructure | SysML v2 shall include the capability to represent the hardware, software, and the personnel (analysis experts) required for performing the analysis. |  |
| ANL 1.13 | Analysis Metadata | SysML v2 shall include the capability to represent the metadata relevant to specifying the analysis. |  |
| ANL 1.14 | Decision Group |  |  |
| ANL 1.14.1 | Trade-off | SysML v2 shall include a capability to represent an evaluation among a set of alternatives that can result in a decision based on a set of criteria. A trade-off may be dependent on other decisions. |  |
| ANL 1.14.2 | Alternative | SysML v2 shall include a capability to represent a set of alternatives. |  |
| ANL 1.14.3 | Decision Expression | SysML v2 shall include a capability to model decision expressions that constrain the possible decisions (e.g., alternative A OR (alternative B and alternative C)). |  |
| ANL 1.14.4 | Decision | SysML v2 shall include a capability to represent a decision as one or more selections among alternatives.  **Supporting Information:** This Decision and Rationale can be related through an Explanation relationship. The Rationale can refer to the supporting analysis. |  |
| ANL 1.14.5 | Criteria | SysML v2 shall include a capability to represent criteria that is used as a basis for a decision or evaluation. |  |
| ANL 1.14.6 | Rationale | SysML v2 shall include a capability to represent rationale for a decision or other conclusion. | Rationale |

### Reference Model and Model Libraries

#### Reference Model and Model Libraries Introduction

Over the many years of use of SysML v1, there has been strong demand to provide reusable assets that are broadly available across industry and within an organization. The language is quite flexible to enable users to apply the language using their own methods and modeling practices. This flexibility can result in models that are difficult to reuse across industry and organizations. The intent for SysML v2 will be to further encourage shared models through reuse libraries and application of common modelling patterns.

The SysML v2 specification will include a reference model that provides a robust example of the application of SysML to a broadly relevant domain such as an automobile that can serve to capture some of the common modelling patterns. The reference model can also be used to support conformance evaluation of vendor implementations, and can also be used to demonstrate SysML concepts and support various training needs. In addition, the generic elements for potential commonly used model libraries is also being requested to encourage the population of these libraries in a standard way.

#### Reference Model and Model Libraries Requirements

**Table 3.10. Reference Model and Model Libraries Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| RML 1 | Reference Model and Model Libraries Group |  |  |
| RML 1.1 | Reference Model | SysML v2 shall include a reference model that demonstrates the application of the SysML v2 language concepts to a commonly understood domain. |  |
| RML 1.2 | Model Libraries | SysML v2 shall include Model Libraries that contain generic elements that can be further specialized to define domain specific libraries in the following domain areas:   * Primitive Value Types * Units and Quantity Kinds * Components * Natural environments * Interfaces * Behaviors * Requirements * Verification methods * Analyses * Basic geometric shapes * Basic material kinds * Viewpoint methods * View definitions (i.e. different kinds of documents and other artifacts) * Domain-specific symbols   **Supporting information:** The generic elements provide a common starting point for development of domain specific model libraries that can be elicited in future RFPs and/or the open source community. |  |

### Language Conformance

#### Language Conformance Introduction

The SysML v2 API and Services Specification will specify conformance tests to enable vendors and end users to evaluate their level of conformance with the SysML v2 Specification. This may include metamodel and profile conformance, concrete syntax conformance, API conformance, service conformance, and model interoperability conformance.

#### Language Conformance Requirements

**Table 3.11. Language Conformance Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| CNF 1 | Conformance Requirements Group | These requirements specify that the proposals provide a suite of test cases that a conformant SysML v2 implementation must satisfy. The test cases can more generally be verification cases.  The SysML v2 specification will specify the conformance levels for each conformance area below. Vendors are expected to identify specific levels of conformance their implementation is intended to support. |  |
| CNF 1.1 | Formalism Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation with the SysML v2 specification formalism requirements. |  |
| CNF 1.2 | Metamodel & Profile Conformance Group |  |  |
| CNF 1.2.1 | Metamodel Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation with the SysML v2 specification metamodel and profile requirements. |  |
| CNF 1.2.2 | Profile Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation with the SysML v2 specification profile requirements. |  |
| CNF 1.3 | Concrete Syntax Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation with the SysML v2 specification concrete syntax requirements. |  |
| CNF 1.4 | Model Interoperability Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation with the SysML v2 specification model interoperability requirements. | XMI |
| CNF 1.5 | Traceability Matrix | SysML v2 shall include a traceability matrix (include reference) that demonstrates how each language feature is verified by the conformance test suite. |  |

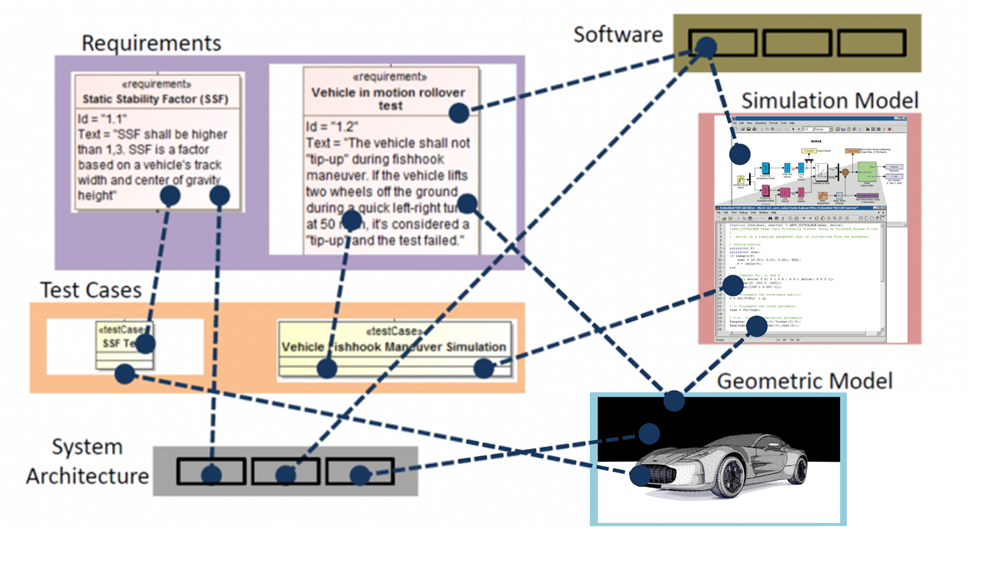
### Other

#### Interoperability

##### Interoperability Introduction

A major emphasis for SysML v2 is to improve interoperability with other engineering models and tools, and other external data sources. In the figure below, interoperability implies relationships between model elements that span different models. In many cases, an element in the SysML model has a direct correspondence to an element in another model, such as when a component in the SysML model corresponds to a component in another model. However, the component in the SysML model may be an abstraction of the corresponding component in the other model. The corresponding element in the other model may add further detail such as the case when a component in the SysML model is represented by a component in a CAD model with detailed geometry. In this way, the SysML model is a specification of the detailed design or analysis model. The relationships between the SysML model and the other model must be defined and managed to ensure effective interoperability.

Figure 3.37. System Model Interoperability Concepts (Source: Alex Reichwein)



SysML v1 focused on interchange between SysML models in different vendor tools using the XMI standard. SysML v2 is intended to provide much broader support for interoperability that include requirements for:

* A standard API to enhance integration between SysML models and other models and tools
* A standard format to interchange SysML models that is not constrained to be XMI
* Services to import and export structured data that is defined in standard formats such as comma delimited data formats
* A capability to transform SysML models
* Migration of previous versions of SysML to current versions of SysML
* A mapping between the shared concepts between SysML and UML to facilitate semantic interoperability between the system and software model.

Some other standards that SysML v2 can leverage for interoperability include Open Services for Lifecycle Collaboration (OSCL), Functional Mockup Interface (FMI), and the series of STEP Application Protocols.

##### Interoperability Requirements

**Table 3.12. Interoperability Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| OTR 1 | Interoperability Requirements Group | Other requirements from other topic areas that also relate to interoperability include API 01, LNG 1.6, and the Interoperability Services Group, SVC 6. |  |

#### Usability

##### Usability Introduction

Usability is a critical driver to the ultimate acceptance and use of SysML v2, and is fundamental to enabling more cost-effective application of MBSE. In particular, the usability goals are to reduce the learning curve for new users, improve productivity for constructing models for new and experienced users, and improve the visualization capability and interaction with the model for a diverse set of stakeholders that consume the model data.

The SysML v2 RFP includes a requirement to specifically address these usability concerns. This includes the requirement for the SysML v2 specification to demonstrate how it addresses the usability criteria that are specified in this RFP for different classes of users.

Some ways in which SysML v2 implementations are expected to address usability concerns include:

* Model construction services that include the use of patterns and workflow support
* Model visualization services that support a highly flexible view and viewpoint
* Model construction, visualization, and model management services that facilitate uses across the development lifecycle from sketching capability to high fidelity models with rigorous model checking
* Customization and extensibility capabilities such as use of aliases, different native languages, domain specific symbol libraries concepts
* Consistency across language such as consistent terminology, symbols, shared concepts
* Consistent workflows, modeling patterns, and user interfaces
* Ability to quickly find relevant information within the current view
* Ability to quickly navigate to related information that is not in the current view
* Setting defaults to support common uses (e.g., default multiplicity of 1)
* Effective user documentation and support

##### Usability Requirements

**Table 3.13. Usability Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| OTR 2 | Usability Group | An objective for SysML v2 is to address SysML v1 usability issues, and enable systems engineers and others to perform MBSE more effectively. The following usability goals apply to a diverse class of SysML v2 users:   1. User understanding when creating or interpreting models 2. User engagement when creating or interpreting models (this particularly applies to consumers of the model data) 3. User productivity when creating models across the lifecycle |  |
| OTR 2.1 | Usability Evaluation | The SysML v2 submission shall demonstrate how the SysML v2 specification satisfies the following usability criteria to meet the usability goals for the different classes of users and goals.  To be provided |  |

## Mandatory API and Services Requirements

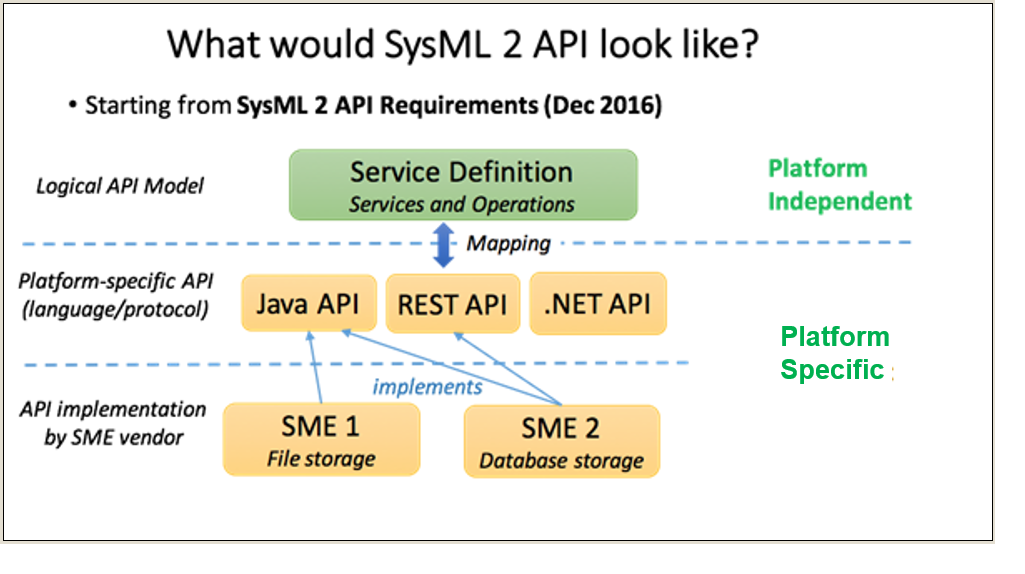
### API

#### API Introduction

The SysML v2 RFP requires a standard way to request services through an API to support interoperability. SysML v2 will allow any client such as an external tool, plug-in, or user interface to request services to access the system model repository in a standard way. As highlighted in the figure below, the API requirement is to produce a platform-independent model (i.e., logical API model) with selected platform-specific bindings that are then implemented by modeling tools to enable access to the system model repository. The platform-independent model provides a service definition that is consistent with the information model. The platform-specific binding defines the services using a particular technology (e.g., Java, web services). A formal mapping is maintained between the platform-independent model and platform-specific bindings.

The API must also provide various infrastructure services to support connection to the model repository, service registries, and capabilities that enable a client of the API to request services that are implemented by a federated multi-vendor implementation.

Figure 3.38. SysML v2 API Specification Approach



#### API Requirements

**Table 3.14. API Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| API 1 | API Requirement Group | SysML v2 will include an API specification that will be implemented by SysML v2 vendors. Software application code written using this API specification will work with each conformant vendor implementation of this API specification without requiring any change in the code. |  |
| API 1.1 | API Scope | SysML v2 API shall support the SysML v2 mandatory service requirements. |  |
| API 1.2 | API Architecture | SysML v2 API shall provide:   1. A **platform-independent model** that defines the services and the operations provided by the API. Services are collections of operations. Inputs and outputs of each operation shall be defined. The model shall be defined using SysML. 2. **Mapping approach** to map the platform-independent model to platform-specific bindings. The mapping shall be formally represented using SysML. 3. **Platform-specific bindings** to two or more commonly used platforms (such as Java, .NET, HTTP or GraphQL). The platform-specific bindings shall also include API documentation for each of the services and their operations.   **Supporting Information:** The platform-independent model and the mapping approach should be complete and defined formally to allow for specifying other platform-specific bindings.  See the SysML 2 API Prototype presentation [34, **[Error! Hyperlink reference not valid.](http://OMG Technical Meeting, Reston VA, May 21, 2017 - SysML2_API_Prototype_OMG_Reston_2017-03-21.pdf   " \t "_top)**] for an example of analysis services. |  |
| API 1.3 | API Conformance | SysML v2 API shall specify the conformance rules for an implementation of the platform-specific binding(s) of the API. SysML v2 API shall also provide a formal mechanism to evaluate and score the conformance level of a SME's implementation of the API specification. |  |
| API 1.4 | API Infrastructure Services | SysML v2 API shall provide the following infrastructure-level service specifications in addition to the systems engineering domain service specifications listed in API Scope requirement. SMEs that claim conformance to these services shall provide an implementation for the following.   * Connection to model repository - Connect to SysML v2 data model repository managed by the SME * Service registry - Check if a SME provides a service registry, i.e. a collection of all SysML v2 API services offered by the SME. * Service discovery/lookup - Query of the service registry provided by the SME to lookup specific services. * Federated implementations - support a federated multi-vendor SME implementation where a client of the API can request services that are implemented by multiple vendors with their own model repositories.   **Supporting Information:** Refer to DDS [DDS] and CORBA [CORBA] as examples |  |
| API 1.5 | Design Constraints | The SysML v2 API shall meet the following design constraints:   * Allow extensions of the platform-independent model and platform-specific bindings   **Supporting Information:** Refer to DDS [DDS] and CORBA [CORBA] as examples |  |

### Services

This section contains the mandatory service requirements for the SysML v2 specification service definitions.

#### Query Services

##### Query Services Introduction

The requirements in this group specify services related to accessing the information in a SysML model.

##### Query Services Requirements

**Table 3.15. Query Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVM 1 | Query Services Group | The requirements in this group specify services to create, update, delete, and execute queries of the SysML model, persist the results, and dynamically update the query results as the model is updated.  **Supporting Information:**  The query services are used by other services including visualization, analysis, applying patterns, and selecting members of element groups. |  |
| SVM 1.1 | CRUD for Queries | SysML v2 shall specify services to provide the ability to create, read, update and delete queries. |  |
| SVM 1.2 | Query Model | SysML v2 shall specify services to provide the ability to query the information content in a SysML model. |  |
| SVM 1.3 | Persist Query Results | SysML v2 shall specify services to provide the ability to persist the results of a query. |  |

#### External Relationships Management Services

##### External Relationships Management Services Introduction

These services enable a user of the system model to create elements in the system model and manage the elements in models that are in other tools.

##### External Relationships Management Services Requirements

**Table 3.16. External Relationships Management Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVM 2 | External Relationships Management Services Group | The requirements in this group are intended to support workflows that enable requirements, test cases, and other model elements to be related to elements in external tools, and potentially managed in the external tool. |  |
| SVM 2.1 | Named Partitions | SysML 2 shall specify services to create and update named partitions that identify a set of element types and their properties and relationships that are managed by a specific kind of tool. |  |
| SVM 2.2 | Relationship to External Elements | SysML v2 shall specify services that can relate model elements that conform to a named partition to the corresponding elements in an external tool. The element in the external tool is called a related external element. |  |
| SVM 2.3 | Viewing Related Elements | SysML v2 shall specify services to view related external elements that are exposed by an external tool. |  |
| SVM 2.4 | Bidirectional Service Interface | SysML v2 shall specify bidirectional service interfaces that enable elements, properties, and relationships that conform to a named partition to be shared between tools where the master is the owner, but not necessarily the author of the elements.  **Supporting Information:** The master is often a domain specific tool that manages the element types specified by the named partition. |  |

### API and Services Conformance

#### API and Services Conformance Introduction

The SysML specification will specify conformance tests to enable vendors and end users to evaluate their level of conformance with the SysML v2 API and Services specification.

#### API and Services Conformance Requirements

**Table 3.17. API and Services Conformance Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| ACF 1 | API and Services Conformance Requirement Group | Place Holder Requirement Group |  |
| ACF 1.1 | API Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation with the SysML v2 specification API requirements. |  |
| ACF 1.2 | Service Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation with the SysML v2 specification service requirements. |  |
| ACF 1.3 | Functional Thread Conformance | SysML v2 shall provide test cases to assess conformance of a SysML v2 implementation that support the following functional threads using a combination of services.   * Change impact thread (requirements to design to analysis to verification) * Additional threads to show how the SysML v2 specification supports various MBSE use cases |  |

## Non-mandatory API and Services Features

This section contains the optional service requirements for the SysML v2 specification service definitions in the API model described in the previous section. Optional means that the Submission Team can determine which requirements to incorporate into the specification.

### Model Construction Services

#### Model Construction Services Introduction

The background that led to the model construction concept and requirements are described on the [Model Construction Wiki](http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:systems_engineering_model_construction_focus_area). The model construction concept is intended to provide a more efficient, intuitive, and adaptable means for model development by users with a diverse range of experience and needs. The requirements reflect the need to create, update, and delete any model element or group of model elements using graphical, tabular, or textual entry. In addition, the requirements for model construction apply to other elements that are not model elements that include metadata, modeling patterns, model queries, expressions, validation scripts, viewpoint methods, model transformations, and external links. Model construction is intended to be facilitated by mechanisms, such as patterns and wizards that enable a user to perform a specific systems engineering task or practice.

A user can construct models through a user interface that enables creating, updating, and deleting model elements, or by importing data from an external file, database or software application, and transforming the data to the corresponding SysML model elements. The model construction supports the definition and application of modeling patterns as a means for efficient and intuitive model construction.

Model updates can be completed successfully or rolled back to a prior consistent state. When deleting a model element, the services should ensure the deletion semantics are applied.

The requirements for model construction have interdependencies with other requirements related to model visualization, model management, workflow and collaboration, and model analysis.

#### Model Construction Services Requirements

**Table 3.18. Model Construction Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVC 1 | Model Construction Services Group | The requirements in this group specify services to construct model elements and other elements used to constructs models (i.e. model constructor). |  |
| SVC 1.1 | Read and Transform Input Group |  |  |
| SVC 1.1.1 | Read Input Data | SysML v2 shall specify a service to read data from an external data source that is available in the following formats.   1. Comma delimited data 2. Requirement interchange format 3. HTML 4. Open Office 5. Office Open XML (ECMA 376) |  |
| SVC 1.1.2 | Transform Input Data | SysML v2 shall specify a service to transform data from and an external data source to SysML v2 model elements in accordance with the mapping rules between the external data schema and the SysML v2 metamodel |  |
| SVC 1.1.3 | Maintain Source Identifier | SysML v2 shall maintain the element identifier of the source data if provided. |  |
| SVC 1.2 | Create Elements Group |  |  |
| SVC 1.2.1 | Create Model Elements and Model Constructors | SysML v2 shall specify a service to create model elements and other model constructors using textual, graphical, and/or tabular entry. |  |
| SVC 1.2.2 | Create Unique Identifier | SysML v2 shall create a unique identifier when creating a model element or other model constructors. |  |
| SVC 1.3 | Update Elements Group |  |  |
| SVC 1.3.1 | Update Model Elements and Model Constructors | SysML v2 shall specify a service to update model elements and other model constructors using textual, graphical, and/or tabular entry. |  |
| SVC 1.4 | Delete Elements Group |  |  |
| SVC 1.4.1 | Delete Model Elements and Model Constructors | SysML v2 shall specify a service to delete model elements and other model constructors in accordance with the deletion semantics. |  |
| SVC 1.4.2 | Preserve Unique Identifier | SysML v2 shall retain the unique identifier when deleting a model element or other model constructor for future reference to the deleted element, and to preclude reuse of this identifier by another model element or other model constructor. |  |
| SVC 1.5 | Patterns Group |  |  |
| SVC 1.5.1 | Create Pattern | SysML v2 shall specify a service to create a model pattern. |  |
| SVC 1.5.2 | Update Pattern | SysML v2 shall specify a service to update a model pattern. |  |
| SVC 1.5.3 | Delete Pattern | SysML v2 shall specify a service to delete a model pattern. |  |
| SVC 1.5.4 | Apply Pattern | SysML v2 shall specify a service to create, update, and delete model elements that conform to a defined model pattern. |  |
| SVC 1.6 | Model Metadata Services Group | This set of services supports the use of model element metadata associated with the model and model elements. |  |
| SVC 1.6.1 | Create Metadata | SysML v2 shall specify a service to create metadata using textual, graphical, and/or tabular entry. |  |
| SVC 1.6.2 | Update Metadata | SysML v2 shall specify a service to update metadata using textual, graphical, and/or tabular entry. |  |
| SVC 1.6.3 | Delete Metadata | SysML v2 shall specify a service to delete metadata using textual, graphical, and/or tabular entry. |  |
| SVC 1.7 | Crosscutting Group |  |  |
| SVC 1.7.1 | Abstraction Level Construction | SysML v2 shall consider how to construct models through elaboration and refinement to transition from one level of abstraction to another, while preserving the earlier abstraction. (Note: this may be considered a transformation of one abstraction level to another that can be viewed in different viewpoints.) |  |

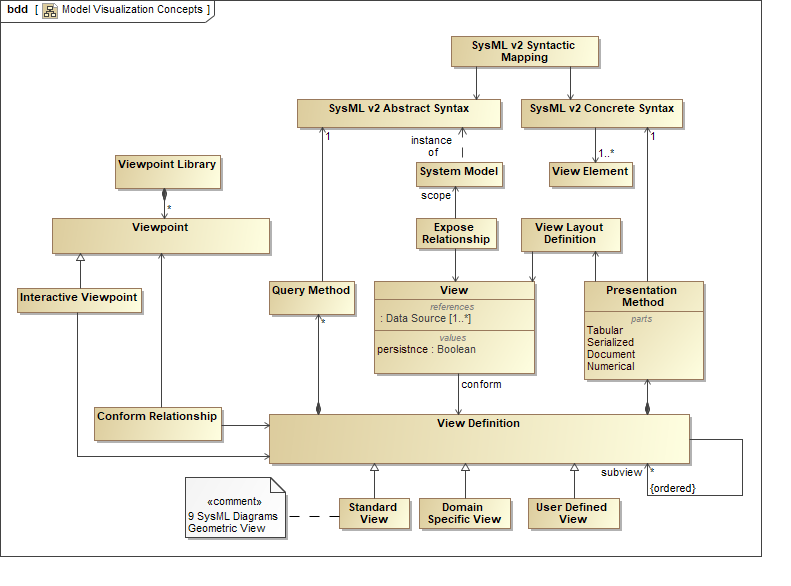
### Model Visualization Services

#### Model Visualization Services Introduction

The background that led to the model visualization concept and requirements are described on the [Model Visualization Working Group](http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:model_visualization_working_group).

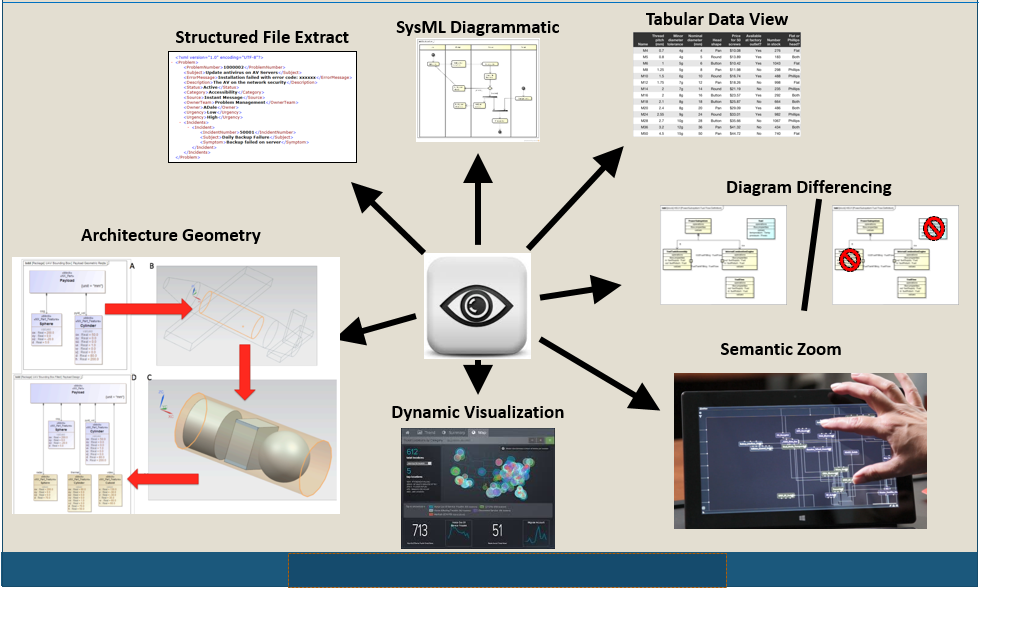
The concrete syntax requirements are included as part of the formalism requirements and the visualization requirements. These include requirements to define the concrete syntax as part of the SysML v2 metamodel to formally define the symbols used in the language, and to establish a formal mapping between the concrete syntax and the abstract syntax. In addition, there is a requirement to specify a standard symbol set to support the standard SysML views, and for the concrete syntax to be customizable to support domain specific extensions. The visualization concept reflects a model view controller paradigm where the controller is the view construction method. A primary driving service requirement is to create, read, update, delete, and execute view construction methods that conform to the viewpoint. Executing the view construction methods includes querying the model and presenting the query results in a view. This concept supports dynamic visualization, semantic zoom and pan, diagram differencing, and the ability to provide diverse renderings including graphical, tabular, text, and geometric views of the model. The visualization requirements also include the requirement to map the model elements to the view elements and provide support for domain specific symbol libraries, including user defined images to represent any model element. Interactive viewpoints can be specified to enable the user to modify the view such as zoom, pan, and filter, and in some cases, provide updates to the model. These concepts are illustrated in the figure below.

Figure 3.39. Model Visualization Concepts



SysML v2 is required to support the standard SysML v1 diagrams and a geometric view. In addition, the viewpoint method is required to be highly flexible to address a diverse range of systems engineering visualization needs. The views must also be capable of being stored and retrieved.

Figure 3.40.  Model Visualization Concepts (Source: C. Schreiber, J. Feingold, and M. Sarrel)



#### Model Visualization Services Requirements

**Table 3.19. Model Visualization Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVC 2 | Visualization Services Group | The requirements in this group specify services to enable visualization of the SysML v2 model. |  |
| SVC 2.1 | View Group | The following specify the requirements to render views that contain view elements. |  |
| SVC 2.1.1 | View Rendering Format | SysML v2 shall specify a service to render a view in diverse formats that include graphical, geometric, tabular, textual, and numerical formats. |  |
| SVC 2.1.2 | Manual and Automatic View Rendering | SysML v2 shall specify a service to render the layout of the view elements manually and automatically. |  |
| SVC 2.1.3 | View Filter, Zoom, and Layering | SysML v2 shall specify a service to filter, zoom, and layer the view elements. |  |
| SVC 2.1.4 | View Navigation | SysML v2 shall specify a service to navigate between views. |  |
| SVC 2.1.5 | Standard Views | SysML v2 shall specify a service to render the predefined SysML v1 diagrams and the following standard views.   * Geometric view * Timing diagram |  |
| SVC 2.1.6 | Domain-Specific Symbol Libraries | SysML v2 shall specify a service to create, read, update, and delete domain-specific symbol libraries, including user-defined images that can be mapped to the abstract syntax. |  |
| SVC 2.1.7 | Persistent Views | SysML v2 shall specify a service to persist views and its mapping to the abstract syntax in a standard format and re-rendered in a view that is the same as the original view. |  |
| SVC 2.1.8 | View Interchange | SysML v2 shall specify a service to interchange view information so that the view can be presented by a SysML v2 conformant tool using the standard SysML v2 tool independent model interchange format. |  |
| SVC 2.1.9 | View Management Service | SysML v2 shall specify a service to manage changes to persistent views that includes the ability to provide view differencing, and to control markings on views. |  |
| SVC 2.2 | Viewpoint Group | The following specify the requirements to specify viewpoints. |  |
| SVC 2.2.1 | Viewpoint Method | SysML v2 shall specify a service to create, read, update, delete, and execute viewpoint methods that query the model, and render the query results in a view based on the mapping between the abstract syntax and concrete syntax.  **Supporting Information:** Include the ability to return derived relationships so that they can be rendered in a view. Derived relationships are defined in the SysML v2 RFP. |  |
| SVC 2.2.2 | Interactive Viewpoint Method | SysML v2 shall specify a service to create, read, update, delete, and execute viewpoint methods that support interactive behavior between the user and the view, and the ability to update the model.  **Supporting Information:** The ability to update the model is also a requirement on model construction. |  |
| SVC 2.2.3 | Interactive Viewpoint Method for Document Generation | SysML v2 shall specify a service to enable the generation and viewing of documents in a mode of what you see is what you get (WYSIWYG), and relate document text to model elements, insert selected views (e.g., diagrams, tables) into the document, and save text to the model, and optionally synchronize the text with the values in the model (e.g., name, number).  **Supporting Information:** This is also a requirement on model construction. |  |
| SVC 2.2.4 | Composite Viewpoints | SysML v2 shall specify a service to compose viewpoints from other viewpoints that have conforming views that can be used to support document generation and other more complex views, such as panes with nested views. |  |
| SVC 2.2.5 | Viewpoint Library | SysML v2 shall specify a service to create, read, update, and delete viewpoint libraries that include predefined viewpoint methods (Note: this includes the methods to generate the SysML v2 standard views and support for standard role-based viewpoints to address different classes of users such as novice modelers, experience modelers, and model consumers.). |  |

### Model Analysis Services

#### Model Analysis Services Introduction

The background that led to the model analysis concept and requirements are described on the [Model Analysis Wiki](http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:system_analysis_workgroup). The goal for these services are to facilitate multi-domain and multi-disciplinary analysis of the system model. The services are intended to support full lifecycle analysis that includes both quantitative and qualitative (logical) analysis. This includes consistency checks, impact and coverage analysis, constraint solving, and simulation execution. Model checking is performed to assess model well-formedness as an important first step prior to analysis to ensure the model results are meaningful. Quantitative analysis includes typical engineering analysis of performance and physical properties to support trade studies, design optimization, and verification. Logical analysis is used to reason about the system by performing queries of the model to get answers to questions such as what is the impact of a requirements change.

Some analysis can be performed by the SysML modeling tools that provide the analysis infrastructure, and other analysis require external solvers, simulation engines, and reasoners. When the analysis inputs are provided by the SysML model to the external analysis tool, the analysis results must be integrated back into the SysML model.

The primary analysis service requirements are to setup and execute the analysis, Store the analysis results, and Query the analysis. The analysis services must integrate with the analysis concepts, and the concepts for properties, values, and expressions, as well as the other service requirements related to model construction, model visualization, and model management. This includes the need to perform model transformations to support analysis.

#### Model Analysis Services Requirements

**Table 3.20. Model Analysis Services Requirements**

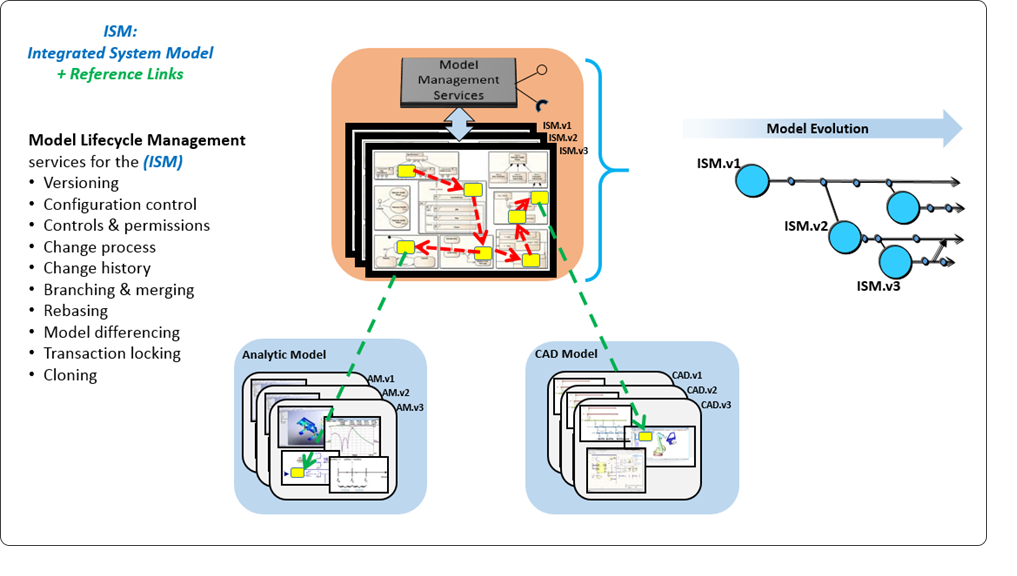
| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVC 3 | Analysis Services Group | The requirements in this group specify services to setup, execute, store, and query analysis models and artifacts. |  |
| SVC 3.1 | Analysis Setup | SysML v2 shall specify services to setup an analysis that includes the following:   1. Objectives, type, and fidelity of the analysis to be performed 2. Key metrics (MoEs, KPPs, PoIs) 3. The representation of the system which is being analyzed 4. The analysis case that defines the scenarios being considered 5. Analysis models for the analysis scenarios, including generation/derivation of analysis models from system model 6. Analysis infrastructure required to formulate and execute analysis models |  |
| SVC 3.2 | Execute Analysis | SysML v2 shall specify services to execute analysis models for each analysis scenario using the analysis infrastructure. |  |
| SVC 3.3 | Validation Service | SysML v2 shall include the capability to define and execute validation rules to ensure the model is well formed. |  |
| SVC 3.4 | Store Analysis | SysML v2 shall specify services to version and store all the information related to an analysis, including analysis models and their execution results that may be stored in the SysML v2 model repository or in an external repository. |  |
| SVC 3.5 | Query Analyses and Results | SysML v2 shall specify services to query all the information related to an analysis, including querying and filtering analysis models and execution results that may be stored in the SysML v2 model repository or in an external repository. |  |

### Model Management Services

#### Model Management Services Introduction

The background that led to the model management concept and requirements are described on the [Model Management Wiki](http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:model_lifecycle_management_working_group). The goals of the model management services are to manage the configuration of the system model, and manage the change process to update the system model. In addition to the system model, the scope of model management includes the links to other models and external data sources. This broader scope that includes the system model and the links is referred to as the integrated system model. Model management also encompasses management of other model constructs that are used to develop and maintain the system model such as those identified by the model construction services above. The model management concept is highlighted in the figure below.

Figure 3.41. Integrated System Model (ISM) Lifecycle Management Concepts (Source: SysML v2 RFP Model Management WG)



A key model management requirement is for each model element is to have a unique identifier, and for model elements to be capable of being versioned. The Universal Unique Identifier (UUID) was proposed as a potential standard for a globally unique id, along with the required metadata such as timestamp and its context (e.g., when created, changed, approved).

A model configuration item (MCI) refers to one or more model element ids, and is used to determine the level of the model that can be versioned. When any element referred to by an MCI is changed, the MCI is assigned a new version. An MCI can also refer to other MCI's. This general concept contrasts with many of the existing file-based versioning concepts that assign versions to entire models, and not to individual elements within a model. Providing more granular versioning capability will enable change histories to be defined at a more granular level, which provides more control over the configuration.

For SysML v2, there is a requirement for the metadata to be extensible. The intent is to identify what metadata is required to manage the system model, without over constraining where this metadata is stored, how it is stored, and whether the metadata is part of the SysML model. There is also a requirement to interchange the metadata in a standard way.

The primary model management services are to create versions, create baseline configurations, log changes, compare differences, merge capability that presents conflicting changes to the user to resolve, generate version histories, manage data protection controls such as data markings, and manage user authorizations to the data. Services are also required to extend the metadata.

#### Model Management Services Requirements

**Table 3.21. Model Management Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVC 4 | Model Management Services Requirement Group |  |  |
| SVC 4.1 | Model Management Services Group | The requirements in this group specify services to manage the model configuration and changes to model elements. |  |
| SVC 4.1.01 | Identify Scope | SysML v2 shall specify services to identify the model scope to be reviewed and/or updated by a task (i.e., a task change set). |  |
| SVC 4.1.02 | Provide Access | SysML v2 shall specify services to provide access permissions for identified users of a task to the model scope. |  |
| SVC 4.1.03 | Define MCI Definition Default Rules | SysML v2 shall specify services to define the rules to determine what level of model granularity will be versioned, which is referred to as a Model Configuration Item (MCI).  **Supporting Information:** Examples include Container (i.e. Package), Block, Model Element, and View Definition.  The rules shall also specify whether a model element can be referred to by more than one MCI. |  |
| SVC 4.1.04 | Authorize Model Version Update | SysML v2 shall specify services to authorize MCI version updates. |  |
| SVC 4.1.05 | Create or Update MCI versions | SysML v2 shall specify services to create or update MCI versions. |  |
| SVC 4.1.06 | Log Change | SysML v2 shall specify services to log changes to model elements for a version. |  |
| SVC 4.1.07 | Compare and Identify Differences | SysML v2 shall specify services to identify differences between any two or more versions of the same MCI. |  |
| SVC 4.1.08 | Create Baseline Configuration | SysML v2 shall specify services to create a baseline configuration with MCIs. |  |
| SVC 4.1.09 | Create Branch Configuration | SysML v2 shall specify services to create a branch of a baseline configuration with MCIs. |  |
| SVC 4.1.10 | Update Branch with MCIs | SysML v2 shall specify services to update a branch of a baseline. |  |
| SVC 4.1.11 | Merge Branch | SysML v2 shall specify services to merge a branch to a trunk of a baseline or to another branch, and present conflicting changes to the user to resolve. |  |
| SVC 4.1.12 | Re-base Branch from Trunk | SysML v2 shall specify services to update a branch from a trunk baseline. |  |
| SVC 4.1.13 | Create Model Control | SysML v2 shall specify services to create the initial control of a model including the creation of a MCI and versioning. |  |
| SVC 4.1.14 | Add to Parent MCI | SysML v2 shall specify services to add additional model elements or MCIs to a parent MCI. |  |
| SVC 4.1.15 | Provide History | SysML v2 shall specify services to provide log change history that was created by the log change service. |  |
| SVC 4.1.16 | Model Integrity | SysML v2 in the model management services group shall NOT modify any system model content.  **Supporting Information:** The model management metadata is separate from the user model. |  |
| SVC 4.1.17 | Timestamp generation | SysML v2 shall specify a service to provide a standard formatted timestamp with a context.  **Supporting Information:** Example: timestamp= [time=2009-06-15T13:45:30; context=last-change]  Refer to ISO 86010 |  |
| SVC 4.2 | Manage Data Protection Control Services Group | The requirements in this group specify services to control markings of the model information (e.g., security classification, proprietary, ITAR, and others). |  |
| SVC 4.2.1 | Create Data Protection Controls | SysML v2 shall specify services to create data protection controls that include data classifications and data marking rules. |  |
| SVC 4.2.2 | Read Data Protection Controls | SysML v2 shall specify services to read data protection controls. |  |
| SVC 4.2.3 | Update Data Protection Controls | SysML v2 shall specify services to update data protection controls. |  |
| SVC 4.2.4 | Delete Data Protection Controls | SysML v2 shall specify services to delete data protection controls. |  |
| SVC 4.3 | Model Link Service | SysML v2 shall specify services to create, read, update, delete, and execute external links between SysML based system models and other structured data. |  |
| SVC 4.4 | Metadata Services Group | The intent of this set of services is to provide the capability to define new metadata terms that are needed by any of the SME service groups.  Scope:  Extending the required metadata elements that can be applied to model elements. |  |
| SVC 4.4.1 | Create Metadata Element | SysML v2 shall specify services to create new metadata elements to support capability extensions by any SME service group.  **Supporting Information:** This would be similar to extending a metaclass with additional properties. |  |
| SVC 4.4.2 | Read Metadata Element | SysML v2 shall specify services to read metadata elements. |  |
| SVC 4.4.3 | Update Metadata Element | SysML v2 shall specify services to update metadata elements. |  |
| SVC 4.4.4 | Delete Metadata Element | SysML v2 shall specify services to delete metadata elements. |  |
| SVC 4.4.5 | Exchange Metadata Elements | SysML v2 shall specify services to exchange metadata elements.  **Supporting Information**: Model exchange should include mechanism for preserving user defined metadata within the context of model exchange. |  |
| SVC 4.4.6 | MLM Metadata Persistence | SysML v2 shall specify services to provide persistent model management metadata either in the data model of authoring tool or in a separate repository owned by the service provider (applies to services that authoring tool utilizes). |  |

### Workflow & Collaboration Services

#### Workflow & Collaboration Services Introduction

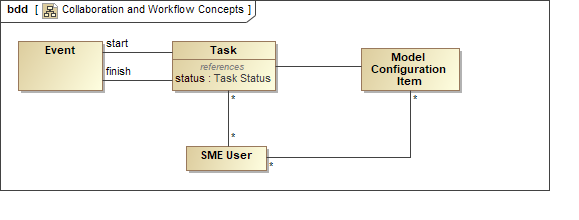
The background that led to the workflow and collaboration concept and requirements are described on the [MBSE Workflow and Collaboration Wiki](http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:mbse_workflow_and_collaboration_working_group). The Workflow and Collaboration concept is defined in the context of an of an external workflow management capability that manages the overall engineering workflow.

The overall concept assumes that the MBSE practices are defined using a process modeling tool that can be a SysML tool or other specialized tool, and captured in a master practices repository. The practices are tailored to a particular project, and used as an input for project planning, execution, and monitoring. As part of the planning process, the tasks are assigned to roles, and roles are assigned to individuals to perform the modeling tasks.

The minimum requirements for SysML v2 are to provide services that support integration of the modeling tasks with external process definition and workflow management capabilities. In particular, SysML v2 should enable an external workflow environment to assign roles to specific users, accept notification to begin a task, and provide status/metrics on the performance and completion of the task.

The assignment of a user to a task should also include the scope of the model that the user must access to perform the task (i.e. the change set). This is indicated in the figure below by the relationship between the Task and User to the Model Configuration Item.

Figure 3.42. Workflow & Collaboration Concepts



#### Workflow & Collaboration Services Requirements

**Table 3.22. Workflow & Collaboration Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVC 5 | Workflow and Collaboration Services Group | The requirements in this group specify services to integrate with external workflow management capabilities and facilitate the execution of modeling tasks. |  |
| SVC 5.1 | View Users | SysML v2 shall specify services to view available users. |  |
| SVC 5.2 | Start Task | SysML v2 shall specify services to create a specific task for a User. |  |
| SVC 5.3 | Task Status | SysML v2 shall specify services to provide user task state (e.g. completion) and metrics. |  |

### Interoperability Services

#### Interoperability Services Introduction

The Interoperability Services support the interoperability requirements as defined in the interoperability section below.

#### Interoperability Services Requirements

**Table 3.23. Interoperability Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVC 6 | Interoperability Services Group | The requirements in this group specify services to support exchange of information using a SysML model. |  |
| SVC 6.1 | Model Transformation Service | SysML v2 shall specify services to create, read, update, delete, and execute model transformations specified as SysML models. |  |
| SVC 6.2 | SysML Version to Version Transformation | SysML v2 shall provide services to transform a model in a previous version of SysML to a model in the next version of SysML, beginning with the transformation from the current version of SysML v1 to SysML v2.  **Supporting Information:** This includes the ability to transform the abstract syntax, concrete syntax and semantics. Some of the SysML v2 execution semantics are specified in other specifications including fUML, PSCS, and PSSM. |  |
| SVC 6.3 | Standard Read/Write Format | SysML v2 shall specify services to read and write a SysML model, its metadata, and its external links in the standard SysML v2 tool-independent interchange format. |  |
| SVC 6.4 | Structured Data Service | SysML v2 shall specify services to export and import structured data that includes, as a minimum, the comma delimited data format, requirements interchange format, HTML, Open Office, Office Open XML (ECMA 376). |  |

### Extension Services

#### Extension Services Introduction

The Extension Services support the extension of the standard language to support domain specific customization. (Note: The submission team can choose an alternative mandatory service from the list of Non-Mandatory Services to specify)

#### Extension Services Requirements

**Table 3.24. Extension Services Requirements**

| **ID** | **Name** | **Requirement Text** | **SysML v1.x Construct** |
| --- | --- | --- | --- |
| SVC 7 | Extension Services Group | The requirements in this group specify services to extend the language to support domain-specific customizations consistent with the extension mechanisms in LNG 1.5.  **Supporting Information:** This is expected to provide services analogous to defining a profile and applying a profile. |  |
| SVC 7.1 | Create Extensions | SysML v2 shall specify services to create consistent extensions of the abstract syntax, concrete syntax, semantics and the mappings between them. |  |
| SVC 7.2 | Apply Extensions | SysML v2 shall specify services to apply the relevant extensions to a user model. |  |

# Appendix A References & Glossary Specific to this RFP

## A.1 References Specific to this RFP

### A.2.1 Bibliographic Citation List

The following documents are referenced in this document:

[1] Created for SECM - This citation indicates that this text was created specifically for the SECM and no other reference is known. Enter [1, created for SECM] at the end of the text field.

[2] INCOSE. 2011. INCOSE Systems Engineering Handbook, Version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

[3] BKCASE Editorial Board. 2015. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.5. R.D. Adcock (EIC). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed DATE. [www.sebokwiki.org](http://www.sebokwiki.org). BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.

[4] ISO/IEC 2008. Systems and Software Engineering -- System Life Cycle Processes. Geneva, Switzerland: International Organization for Standardization / International Electromechanical Commissions. ISO/IEC/IEEE 15288:2008 (E).

[5] ISO/IEC 2015. Systems and Software Engineering -- System Life Cycle Processes. Geneva, Switzerland: International Organization for Standardization / International Electromechanical Commissions. ISO/IEC/IEEE 15288:2015 (E).

[6] Wikipedia: Safety: Mar 31, 2015: <http://en.wikipedia.org/wiki/Safety#Safety_measures>

[7] Douglas, Bruce: Safety Analysis of UML Models

[8] Wikipedia. Main Page. Mar 31, 2015. <http://en.wikipedia.org>

[9] Roedler, G.J. and Jones, C. December 27, 2005. Technical Measurement, Version 1.0, Practical Software and Systems Measurement (PSM) and International Council on Systems Engineering (INCOSE). INCOSE-TP-2003-020-01

[10] INCOSE (2015). Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities (4th ed.) D. D. Walden, G. J. Roedler. K. J. Forsberg, R.D. Hamelin, and, T. M. Shortell (Eds.). San Diego, CA: International Council on Systems Engineering. Published by John Wiley & Sons, Inc.

[11] Merriam Webster on-line dictionary

[12] UML 4SE RFP. SE Definitions List, April 01 2003: <http://syseng.omg.org/UML%20for%20SE%20Definitions%20030401.xls>

[13] Business Dictionary.com - <http://www.businessdictionary.com/>

[14] INCOSE. 2015. Guide for Writing Requirements. Version 2, San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2010-006-02.

[15] OMG Unified Modeling Language (OMG UML), Version 2.5, March 2015, OMG Document Number - formal/2015-03-01

[16] OMG Systems Modeling Language (OMG SysML), Version 1.4, September 2014, OMG Document Number: formal/2015-06-03

[17] ISO Online Browsing Platform (OBP), Terms and Definitions, <https://www.iso.org/obp/ui/#home>

[18] Weilkiens, Tim: Variant Modeling with SysML, MBSE4U - Tim Weilkiens, Apr 12 2016, ISBN 978-3-9817875-4-2

[19] Hilbert, D., & Ackerman, W. (1950) Mathematical Logic (L. Hammond, G. Leckie, & F. Steinhardt, Trans.). New York, NY: Chelsea Publishing Company

[20] Friedenthal, Sanford, Moore, Alan, Steiner, Rick. A Practical Guide to SysML: the systems modeling language. New York, NY: Elsevier, 2015. Third Edition

[21] Friedenthal, S. 2016. "Evolving SysML and the System Modeling Environment to Support MBSE, Part 2" INSIGHT (December Volume 19 Issue 4, Pg. 76-80)

[22] Torroni, P., Yolum, P., Singh, M., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., & Mello, P., (2009). Modeling Interactions via Commitments and Expectations. In Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models (p. 263-284) Hershey, PA: IGI Global.

[23] Winskel, D., & Pitts, A. (2005) Lecture Notes on Denotational Semantics for Part II of the Computer Science Tripos. Retrieved at: <http://www.cl.cam.ac.uk/~gw104/dens.pdf>

[24] (2017) Oxford Living Dictionaries. Retrieved at [https://en.oxforddictionaries.com](https://en.oxforddictionaries.com/definition/formalism)

[25] ISO 9241-210:2010. Ergonomics of human-system interaction - Part 210: Human-centered design for interactive systems. Geneva, Switzerland: International Organization for Standardization.

[26] OMG Systems Modeling Language (OMG SysML), Version 1.5, May 2017, OMG Document Number: formal/2017-05-0

[27] Satellites to Supply Chains, Energy to Finance - SLIM for Model-Based Systems Engineering, Part 1: Motivation and Concept of SLIM. Manas Bajaj, Dirk Zwemer, Russell Peak, Alex Phung, Andy Scott, Miyako Wilson (2011). Presented at the 21st Annual INCOSE International Symposium, Denver, CO, June 20-23, 2011. PDF available at <http://www.omgsysml.org/SLIM_for_MBSE_Bajaj_Part1.pdf>

[28] Satellites to Supply Chains, Energy to Finance - SLIM for Model-Based Systems Engineering, Part 2: Applications of SLIM. Manas Bajaj, Dirk Zwemer, Russell Peak, Alex Phung, Andy Scott, Miyako Wilson (2011). Presented at the 21st Annual INCOSE International Symposium, Denver, CO, June 20-23, 2011. PDF available at <http://www.omgsysml.org/SLIM_for_MBSE_Bajaj_Part2.pdf>

[29] Introduction to SLIM - [www.intercax.com/slim](http://www.intercax.com/slim)

[30] MOF Support for Semantic Structures (SMOF), Version 1.0, April 2013, OMG Document Number - formal/2013-04-02

[31] Meta Object Facility (MOF), Version 2.5.1, November 2016, OMG Document Number - formal/2016-11-01

[32] Matthews, P.H. (2014). The Concise Oxford Dictionary of Linguistics: Oxford University Press. Retrieved at: <http://www.oxfordreference.com/>

[33] Object Management Group. RFP Template, June 2015, OMG Document Number ab/15-06-01

[34] Bajaj, Manas. SysML v2 API Prototype. OMG Technical Meeting, Reston VA, March 21, 2017. [SysML2\_API\_Prototype\_OMG\_Reston\_2017-03-21.pdf](http://www.omgwiki.org/OMGSysML/lib/exe/fetch.php?media=sysml-roadmap:sysml2_api_prototype_omg_reston_2017-03-21.pdf)

[35] Friedenthal, S., Burkhart, R.  2015. "Evolving SysML and the System Modeling Environment to Support MBSE" INSIGHT (August 2015 Volume 18 Issue 2, Pg 39-42)

### A.2.2 OMG Standards List

The following documents are referenced in this document:

[ALF] Action Language for Foundational UMLTM (ALF TM)

<http://www.omg.org/spec/ALF>

[API2KB] Application Programming Interfaces (API) to Knowledge Bases (KB) RFP

<http://www.omg.org/cgi-bin/doc.cgi?ad/2010-06-09>

[BMM] Business Motivation Metamodel TM (BMMTM)

<http://www.omg.org/spec/BMM>

[BPMN] Business Process Model and Notation TM (BPMNTM)

[http://www.omg.org/spec/BPMN](http://www.omg.org/spec/BPMN/index.htm%C2%A0)

[DDS] Data Distribution Service TM (DDSTM)

<http://www.omg.org/spec/DDS>

[DMN] Decision Model and Notation TM (DMNTM)

<http://www.omg.org/spec/DMN>

[DD] Diagram Definition TM (DDTM)

<http://www.omg.org/spec/DD>

[FUML] Semantics of a Foundational Subset for Executable UML Models (FUMLTM)

<http://www.omg.org/spec/FUML>

[MARTE] UML Profile for MARTE

<http://www.omg.org/spec/MARTE/>

[MOF] Meta Object Facility TM (MOFTM) Core

<http://www.omg.org/spec/MOF/>

[MOFVD] Versioning and Development Lifecycle TM (MOFVDTM)

<http://www.omg.org/spec/MOFVD>

[OCL] Object Constraint Language TM (OCLTM)

<http://www.omg.org/spec/OCL>

[ODM] Ontology Definition Metamodel TM (ODMTM)

<http://www.omg.org/spec/ODM/>

[PSCS] Precise Semantics of UML Composite Structures TM (PSCSTM)

<http://www.omg.org/spec/PSCS>

[PSSM] Precise Semantics of UML State Machines (PSSM)

<http://www.omg.org/spec/PSSM>

[PST] Precise Semantics of Time

In process

[QVT] Query View Transformation TM (QVTTM)

<http://www.omg.org/spec/QVT>

[ReqIF] Requirements Interchange Format (ReqIF TM)

<http://www.omg.org/spec/ReqIF>

[RAS] Reusable Asset Specification (RAS)

<http://www.omg.org/spec/RAS>

[S&R] Profile for Safety and Reliability

In process

[SBVR] Semantics of Business Vocabulary and Business Rules TM (SBVRTM)

<http://www.omg.org/spec/SBVR>

[SMOF] MOF Support for Semantic Structures TM (SMOFTM)

<http://www.omg.org/spec/SMOF/>

[SPEM] Software and Systems Process Engineering Metamodel TM (SPEMTM)

<http://www.omg.org/spec/SPEM>

[SysPISF] SysML Extension for Physical Interaction and Signal Flow Simulation (SysPISF)

<http://www.omg.org/spec/SysPISF/1.0/Beta1/>

[SysML] OMG Systems Modeling Language Version TM (SysML®)

<http://www.omg.org/spec/SysML/>

[UAF] Unified Architecture Framework (UAF) previously UPDM

<http://www.omg.org/spec/UAF>

[UML] Unified Modeling Language TM (UML®)

<http://www.omg.org/spec/UML>

[UTP] UML Testing Profile TM (UTPTM)

<http://www.omg.org/spec/UTP>

[XMI] XML Metadata Interchange TM (XMI®)

<http://www.omg.org/spec/XMI>

### A.2.3 Other Standards List

The following documents are referenced in this document:

[FMI] Functional Mock-Up Interface (FMI)

<http://fmi-standard.org/>

[STEP] ISO 10303-233:2012 (STEP)

<https://www.iso.org/standard/55257.html>

[ArchDes] ISO 42010 - Systems and software engineering - Architecture description

<http://cabibbo.dia.uniroma3.it/asw/altrui/iso-iec-ieee-42010-2011.pdf>

[SQuaRE] ISO/IEC 25062:2006(en) - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Common Industry Format (CIF) for usability test reports

<https://www.iso.org/obp/ui/#iso:std:iso-iec:25062:ed-1:v2:en>

[ISO 15288] ISO/IEC 15288:2015 - Systems and software engineering - System lifecycle processes

<https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:15288:ed-1:v1:en>

[ISO 15704] Industrial automation systems - Requirements for enterprise-reference architectures and methodologies

<https://www.iso.org/obp/ui/#iso:std:iso:15704:ed-1:v1:en>

[ISO 26550] ISO/IEC 26550:2015 - Software and Systems Engineering - Reference model for product line engineering and management

<https://www.iso.org/obp/ui/#iso:std:iso-iec:26550:ed-2:v1:en>

[ISO 80000] Quantities and units -- Part 1: General: ISO 80000-1:2009

<https://www.iso.org/standard/30669.htm>l

[ISO-TC184] Interoperability, integration, and architectures for enterprise systems and automation applications

<https://www.iso.org/committee/54192.html>

[HCD] ISO/DIS 9241-220.2(en) Ergonomics of human-system interaction - Part 220: Processes for enabling, executing and assessing human-centered design within organizations

<https://www.iso.org/obp/ui/#iso:std:iso:9241:-220:dis:ed-1:v2:en>

[SE Handbook] INCOSE Systems Engineering Handbook

<http://www.incose.org/ProductsPublications/sehandbook>

[OSLC] Open Services for Lifecycle Collaboration (OSLC)

<http://open-services.net/>

[SEBoK] Systems Engineering Body of Knowledge (SEBoK)

[www.sebokwiki.org](http://www.sebokwiki.org)

## A.2 Glossary Specific to this RFP

***Abstract Relationship -*** A relationship between a more abstract element and a more concrete element. [1, created for SECM]

***Abstract Syntax -*** The set of modeling concepts, their attributes and their relationships, as well as the rules for combining these concepts to construct partial or complete models. Based on [15, UML spec]

***Activity Setup -*** The activity that takes place before an analysis can be executed. This includes:

1. Model the types of analyses that need to be performed on the system representation
2. Model the analysis objectives mathematically
3. Define the key parameters (KPPs/MoEs) being computed or patterns/anti-patterns to be matched.
4. Define mapping/transformation from system model to analysis model
5. Execute the model transformation - Create or generate analysis model based on mapping/transformation (tool-neutral or tool-dependent)

[1, created for SECM]

***Alias -*** An assumed or additional name. [11, Merriam Webster on-line dictionary]

An additional name that can include an acronym, an abbreviated name, a less formal name, or any other name intended to convey the same meaning. [1, created for SECM]

***Allocation Relationship -*** A mapping relationship between model elements that is often intended to assign responsibility of one element to another. An example is the allocation of a function to a component which means the component is assigned the responsibility to perform the function. [1, created for SECM]

Allocate relationship provides a mechanism for associating elements of different types, or in different hierarchies, at an abstract level. Allocate is used for assessing user model consistency and directing future design activity. It is expected that an allocate relationship between model elements is a precursor to a more concrete relationship between the elements, their properties, operations, attributes, or sub-classes. [16, derived from SysML spec]

***Alternative -*** A choice that is available to a decision maker. [1, created for SECM]

***Analysis -*** A systematic investigation of a real or planned system to determine the information requirements and processes of the system and how these relate to each other and to any other system. (ISO/IEC/IEEE 2009) [3, SEBoK Glossary]

An investigation of some domain of interest that is intended to further understanding. The scope of the analysis includes the domain of interest that is the subject of the analysis, the models and supporting infrastructure used to perform the analysis, and the analysis case that defines how the analysis will be performed. [1, created for SECM]

The systematic investigation of a real or planned system to compare, evaluate, and select candidate system architectures, and/or determine causes & resolutions of failures and exceptions. [SEBoK, NASA SE Handbook 2007].

***Analysis Case -*** A process or set of steps intended to achieve an analysis objective. [1, created for SECM]

***Analysis Evaluation -*** The activity that occurs for an analysis case to evaluate the outcome data against the success criteria and determines an outcome. [1, created for SECM]

***Analysis Infrastructure -*** All items needed to conduct the analysis effort, including the analysis models, modeling tools, measurement devices, people, procedures, documentation and information. [1, created for SECM]

***Analysis Model -*** The computation model used to calculate the system properties (relevant to the analysis) to meet the analysis objectives. An Analysis model could be computer-based executable model (e.g. Mathematica/MATLAB code or FEA/CFD model), or a model representing physical measurement on a prototype or actual system. An analysis model includes the following characteristics:

1. Language in which the model is formulated
2. Software used to formulate the model
3. Type of model
4. Result data from executing the model
5. Relationship to the system representation, e.g. design model. This relationship embodies the model transformations required to generate or update the analysis model from the system representation

[1, created for SECM]

***Analysis Objective -*** Represents the objective of the analysis, and can often be specified in terms of questions to be answered. The objective can be specified in the form of textual description and/or as an expression. When the objective is modeled using a set of math expressions their formal evaluation can be automated. The objective of an analysis is met if the expressions can be successfully evaluated by the information generated during the analysis. [1, created for SECM]

***Analysis Outcome -*** The output created from executing a scenario of an analysis case. [1, created for SECM]

***Analysis Plan -*** Information item that presents a systematic course of action for achieving a declared purpose, including when, how, and by whom specific activities are to be performed. (ISO/IEC/IEEE 15289:2011) [3, SEBoK Glossary]

***Analysis Result -*** Represents the evaluations of all the expressions in the Analysis Objective. An analysis is successful if its objective has been met. [1, created for SECM]

***Analysis Subject -*** Also Known as "Subject of the Analysis" - The entities being analyzed in order to satisfy the analysis objectives. Since the scope of system analysis spans across the lifecycle, the subject of the analysis could be either of the following:

* Design representation of the system, such as a digital mock-up (computer model) of a spacecraft being developed
* Prototype of the system, such as a scaled or real prototype of the spacecraft
* Deployed system, such as the actual spacecraft deployed in orbit.

[1, created for SECM]

***Analyst -*** 1. A member of the technical community (such as a systems engineer or business analyst) who is skilled and trained to define problems and to analyze, develop, and express algorithms. IEEE Std. 1233-1998 (R2002) IEEE Guide for Developing System Requirements Specifications.3.1 (ISO/IEC/IEEE 24765:2010(en), 3.104) [17, ISO OBP Definitions]

***Annotation -*** A note added by way of comment or explanation. [11, Merriam Webster on-line dictionary]

A text statement that can also contain one or more navigational links to reference information that provides additional explanatory information about the annotated model element(s). [1, created for SECM]

***API -*** In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building application software. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer. [8, Wiki]

***Assumption -*** A statement that is asserted to be true. [1, created for SECM]

***Baseline -*** An immutable configuration with its associated versions and variants. Baselines are often used to freeze the configuration at critical points in the development lifecycle. [1, created for SECM]

***Basic 2D/3D Library -*** A digital library containing a collection of predefined model elements representing a set of reusable basic two and three dimensional geometric shapes that can be copied or referenced while constructing a model. [1, created for SECM]

***Batch Mode -*** User initiated operations for creating model constructs using an external collection of model element properties, operations and/or relationships. [1, created for SECM]

***Behavior -*** The interaction between individual elements and their change of state over time. Note: the state of an element refers to the values of its state variables at a point in time. [1, created for SECM]

***Case -*** A process or set of steps intended to achieve an objective. The process often includes collecting and examining data or evidence about some subject in a context to produce a result or conclusion. [1, created for SECM] [1, created for SECM]

***Cause-effect Relationship -*** Relates a cause to an effect. The cause and effect may be represented by any model element, such as a state. [1, Created for SECM]

***Change Log -*** A chronological listing of changes to an MCI. Each change log is associated to a version of a MCI. [1, created for the SECM]

***Comment -*** A comment can also contain navigational links from this element or text within this element to other model elements or external elements. [1, created for SECM]

A Comment is a textual annotation that can be attached to a set of (Model) Elements. [15, UML Spec]

***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) A set of functional services in the software, which, when implemented, represents a well-defined set of functions and is distinguishable by a unique name. (ISO/IEC 2008)

[3, SEBoK Glossary]

In systems terms, we use component as the generic term for the level of decomposition at which system elements are no longer considered complex, and for which specialist design disciplines can be used. [3, SEBoK Glossary Discussion]

***Component Definition -*** A type of definition element that can perform functions and present interface ends (i.e., ports) that connect to other components... Components can also contain sub-components that can connect with each other. [1, created for SECM]

***Component Library -*** A digital library containing a collection of predefined component definitions that can be copied or referenced construct a model. [1, created for SECM]

***Component Realization -*** A component that provides a solution that is intended to conform to the component specification. [1, created for SECM]

***Component Specification -*** A Component Specification provides the physical, behavioral, and interface specification for a component to be designed and built. [1, created for SECM]

A specification is defined as:

a: a detailed precise presentation of something or of a plan or proposal for something, usually used in plural [11, Merriam-Webster on-line]

***Concept -*** An abstraction; a general idea inferred or derived from specific instances. (Oxford Dictionaries Online 2012) [3, SEBoK Glossary]

***Concrete Syntax -*** The definition of the textual, graphical or other notations by which a model may be directly represented to users. [1, created for SECM]

***Configuration Element -*** A relationship where one element (conforming element) is asserted to meet the constraints imposed by another element (i.e., the constraining element). An example is a view definition that conforms to a viewpoint where the viewpoint specifies the requirements that a view definition must satisfy.

Note: Other relationships may be specializations of the conform relationship. For example, a satisfy relationship is a conform relationship when a requirement is the constraining element. Other kinds of relationships such as realization, elaboration, and abstraction may be considered subclasses of this relationship.

[1, created for SECM]

***Configuration Management -*** The process to manage and control system elements and configurations over the lifecycle as well as managing consistency between a product and its associated configuration definition. [5, ISO/IEC/IEEE 15288]

***Configuration Model -*** A model that represents a fully expanded hierarchical composition structure for a particular configuration of a system and its interfaces.

The Configuration Model can be used in two ways:

1. As an explicit representation that is generated from a Definition Model with a set of configuration parameters for any Variability Choice, e.g. chosen multiplicity or usage expression, as well as possible filtering or pruning.
2. Direct use of the Configuration Model as a simple 'loosely' typed hierarchical composition structure.

For the case (2) it is in principle possible to generate a (best effort) modular / typed Definition Model with some heuristic algorithm that detects elements of the same type.

Note: It is possible that a Configuration Model is over specified or out of sync w.r.t. to its associated Definition Model, and vice versa. Tool implementations will need to handle this situation with rules, refactoring and synchronization functionality.

[1, created for SECM]

***Conform Relationship -*** A relationship where the element on one side of the relationship imposes constraints on the element on the other side of the relationship. [1, created for SECM]

***Connector Definition -*** Relates two component definitions so the component usages can be connected and interact. [1, created for SECM]

***Constrained Element -*** This element is constrained by an element on the other side of a Conform Relationship (or a subclass of a Conform Relationship). The constrained element must be evaluated to determine whether it conforms to the constraints imposed by the other end.

[1, created for SECM]

***Constraining Element -*** The element that imposes constraints on a Constrained Element. [1, created for SECM]

***Constraint Definition -*** A specification of a constraint expression and the parameters of the expression that can be used in different contexts. [1, created for SECM]

***Constraint Evaluation -*** The result of evaluating a constraint between the constraining element and the constrained element. [1, created for SECM]

***Constraint Expression -*** An expression that can be evaluated to true or false. [1, created for SECM]

***Container -*** A kind of model element that contains other model elements and applies scoping rules such as namespace rules to the contained elements.

Containers can contain other containers providing a mechanism to organize the model. [1, created for SECM]

***Context -*** A model element that establishes a scope for representing usages (or roles) of defined concepts in a particular situation. An example is a vehicle context where the same type of tire can be a front or rear tire, depending on how it is used in its vehicle context, or the same type of tire can be a swing when it is used in the context of a swing-set. [1, created for SECM]

***Control Node -*** A kind of function used to control sequencing of item flows and event flows by specifying a logical expression of output flows in terms of its input flows. Note: SysML v1 examples include join specification, join, fork, decision, and merge nodes in activities, and junction and choice pseudo-states in state machines, and alt, par, and opt interaction operators in sequence diagrams. [1, created for SECM]

***Criteria -*** An expression that specifies the characteristics of interest and their relative weighting and or objective function that can be used as a basis for an evaluation. [1, Created for SECM]

***Data Model -*** A Data Model is a model that organizes elements of data and standardizes how they relate to one another. [1, created fro SECM]

A data model explicitly determines the structure of data. Data models are specified in a data modeling notation, which is often graphical in form. [2] [8, Wiki]

***Data Protection Controls -*** Metadata that is used to protect data that includes access permissions, roles, data rights, and security markings. [1, created for SECM]

***Decision -*** A selection among alternatives based on some criteria. [1, created for SECM]

b: a determination arrived at after consideration: conclusion made the decision to attend graduate school [11, Merriam Webster on-line]

***Declarative Semantics -*** Association of meaning that specifies what rather than how. Communication with declarative semantics specifies what actions should be brought out in an interaction, rather than how they are brought out. [22, Modeling Interactions via Commitments and Expectations]

***Definition -*** a: a statement expressing the essential nature of something

b: a statement of the meaning of a word or word group or a sign or symbol dictionary definitions

c: a product of defining

[11, Merriam Webster on-line dictionary]

***Definition Element -*** An element that defines a class of individual elements with shared features. [1, created for SECM]

***Definition Model -*** A Definition Model represents a strongly typed, modular, hierarchical composite structure. It allows for inclusion of variation points so that it can represent a set of possible system variants. Variation points can include multiplicity ranges, subtrees, alternative composition with or without constraints, etc.

The Definition Model contains a bag of building blocks represented by Definition Elements that may directly use (i.e. one level deep) zero or more Direct Usage Features of other Definition Elements.

In many cases there is a need to unambiguously identify and reference more deeply nested usage elements, e.g. to introduce local override of values for a particular usage two or more levels down from a Definition Element or to define an interface connector between nested interface ends. For this purpose, an element path enables unambiguous traversal from a top element to a deeply nested usage feature.

If no features need to be overridden, redefined nor added at a usage more than one level deep, then a Definition Model constitutes a complete implicit definition of the decomposition structure provided that a single top element (i.e. context element) is identified. With that, it is possible to automatically generate a corresponding Configuration Model that represents the full and explicit (deeply nested) expansion of the decomposition structure.

It is important to note that Deeply Nested Usage Features need only be created (and persisted) if there is a need to override feature values, (re)define features or reference usages at a deeply nested level. Otherwise their representation can be automatically derived 'on the fly' as their existence is fully implied by Definition Elements and Direct Usage Features only.

In case there are Variation Points present in the Definition Model, choices must be made within the range of possible variabilities in order to transform the Definition Model into a Configuration Model that represents a single actual variant that complies with the implicit definition represented by the Definition Model.

Using some heuristics, and perhaps with some human assistance, it is in principle also possible to devise an algorithm that can automatically derive a Definition Model from a given Configuration Model. This is attractive since it would allow a beginning system modeler to start with the simpler to understand Configuration Model and transform it to the more powerful and generalized Definition Model. Such a capability would also support use cases for reverse engineering of existing system architectures.

[1, created for SECM]

***Dependency Relationship -*** A relationship between two elements where a change in the independent element may impact the dependent element. [1, created for SECM]

A Dependency is a Relationship that signifies that a single model Element or a set of model Elements requires other model Elements for their specification or implementation. This means that the complete semantics of the client Element(s) are either semantically or structurally dependent on the definition of the supplier Element(s). [15, UML Spec]

***Derived Relationship -*** A relationship that is derived from other relationships. Note: This is not to be confused with a Requirement Derivation Relationship.

An example of a derived relationship is a transitive relationship between C and A, where B relates to A and C relates to B.

Another example of a derived relationship is a connector between two composite parts that is derived from a connector between their nested parts. [1, created for SECM]

***Design Constraint -*** A the requirement that further constrains the selection of design choices beyond the constraints imposed by the black box specification... [1, created for SECM]

***Domain Specific View -*** A domain view is one or more views that are defined using presentations, shapes and icons that are specific for that domain, such as electrical views, software views, and mechanical views. [1, created for SECM]

***Element -*** An entity that can be assigned an identifier and may be contained in or external to the subject model. [1, created for SECM]

***Element Group -*** A mechanism for grouping various model elements without imposing ownership constraints or impacting the model element that are members of the group. Criteria is defined to determine which members belong to the group. Examples of Element Groups include model elements that are associated with a particular release of the model, have a certain risk level, or model elements from a legacy design that are part of a new design. [1, created for SECM]

***Element Group Relationship -*** Relates an element group to a member of the group. In addition to membership criteria, logical expressions can be applied to membership of the group, such as AND, OR, XOR, NOT, and conditional expressions like IF-THEN-ELSE and IF-AND-ONLY-IF. [1, created for SECM]

***Environment -*** Any entity that is external to a system of interest that affects the system of interest or is affected by the system of interest system through direct or indirect interactions between the system of interest and the external entities. [1, created for SECM]

(1) Anything affecting a subject system or affected by a subject system through interactions with it, or anything sharing an interpretation of interactions with a subject system. (IEEE 1175.1-2002 (R2007), 3.6) (2) The surroundings (natural or man-made) in which the system-of-interest is utilized and supported; or in which the system is being developed, produced or retired. (INCOSE 2010) [3, SEBoK Glossary]

***Event -*** Occurrence defined by a change in conditions that may trigger a response (e.g., a change in state or start of a function). [1, created for SECM]

1. Occurrence of a particular set of circumstances. ISO/IEC 16085:2006 (IEEE Std. 16085-2006), Systems and software engineering - Lifecycle processes - Risk management.3.2.

2. An external or internal stimulus used for synchronization purposes

[17, ISO OBP Definitions]

***Explanation Relationship -*** A relationship between an element being rationalized, i.e. the conclusion, and the element justifying the conclusion, i.e. the rationale.

A conclusion that is explained can be represented by any type of element including elements such as blocks, requirements, or relationships, such as a satisfy relationship between a requirement and design element.

The rationale can refer to the supporting information, such as reference to one or more analysis.

[1, created for SECM]

***Expression -*** In mathematics, an expression or mathematical expression is a finite combination of symbols that is well-formed according to rules that depend on the context. Mathematical symbols can designate numbers (constants), variables, operations, functions, brackets, punctuation, and grouping to help determine order of operations, and other aspects of logical syntax. [8, Wiki]

***External (Resource) Collection -*** A file based or database or link based mechanism to persist descriptions of model elements. [1, created for SECM]

***External Element -*** An entity external to the subject model. In the context of model management, this often refers to items such as a file, web page, or a model element in another model [1, created for SECM]

***Finite State -*** The condition of an individual element for a period of time that constrains its structure and behavior. [1, created for SECM]

***Formal Requirement Statement -*** A formal requirement captures all aspects of a requirement in a machine-readable form, vs. text in a Textual requirement. This enables requirements to be stated more precisely, and to be evaluated in an automated way in support of verification and validation.

[1, created for SECM]

***Formalism -*** A description of something in formal mathematical or logical terms. [24, Oxford Living Dictionaries]

***Function Definition -*** A transformation from inputs to outputs through a controlled sequence of actions (e.g., function usages). [1, created for SECM]

(4) A function is defined by the transformation of input flows to output flows, with defined performance. [3, SEBoK Glossary]

***Functional Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be a functional requirement. [1, created for SECM]

***Generalization Relationship -*** A taxonomic relationship between a more general Classifier and a more specific Classifier. Each instance of the specific Classifier is also an instance of the general Classifier. The specific Classifier inherits the features of the more general Classifier. A Generalization is owned by the specific Classifier. [15, UML 2.5 Spec]

***Geometric View -*** Representation of the shape and size of a physical object in terms of its geometric relationships. A SysML geometric view is intended to specify physical envelopes and includes simple shapes defined in a reference coordinate system. [1, created for SECM]

***Hardware -*** 1. Physical equipment used to process, store, or transmit computer programs or data. 2. All or part of the physical components of an information system. ISO/IEC 2382-1:1993, Information technology - Vocabulary - Part 1: Fundamental terms.01.010.07 cf. software [17, ISO OBP Definitions]

***Hyperlink -*** A hyperlink is a reference to data that the reader can directly follow either by clicking, tapping, or hovering. A hyperlink points to a whole document or to a specific element within a document. Hypertext is text with hyperlinks. [8, Wiki]

***Individual Element -*** A representation of an identifiable object that may or may not exist in the physical world. An example is an individual element that represents a specific automobile with a vehicle identification number, which in turn is composed of individual elements that represent each of its components with their respective serial numbers. An individual element may conform to a configuration element that corresponds to the design configuration that may be realized by multiple individual elements. The individual element can be thought of as a "digital twin". [1, created for SECM]

***Individual Model -*** A model that is a digital representation of an individual system or product that actually or potentially exists in the real world.

An Individual Model can be regarded as a "digital record" of the state of an individual system or product that was built according to a particular Configuration Model for e.g. development testing, verification, shipping, deployment or operation. Sometimes such a model is also known as a "digital twin". [1, created for SECM]

***Integrated System Model -*** Contains the information about the system at any given stage during its lifecycle. It includes the system model (in SysML) and model-based connections between the system model and the various domain-specific models, such as CAD and CAE models that describe various aspects of the system and its sub-systems [27, SLIM Part 1] [28, SLIM Part 2]. The connections between the System Model (or its model elements) and domain-specific models (or its model elements) may have different behaviors [4, Intro to SLIM], such as (1) reference connections for basic traceability, (2) data map connections for exchange for parameter values, (3) function wrap connections for wrapping executable code in system model elements, and (4) model transform connections for generating and synchronizing model structures bi-directionally. [1, created for SECM]

***Interaction -*** A dynamic exchange between 2 or more individual elements. [1, created for SECM]

***Interactive Mode -*** User initiated operations using the graphical interface of the SysML user interface to interact with the modeling tool, for example to create, update, modify and delete model constructs and to maintain the model. [1, created for SECM]

***Interactive Viewpoint -*** An interactive viewpoint provides visualization services that allows the user to adjust the view to meet the specific user's needs at the time. This interactive behavior can include services to adjust the scope of model being viewed, filter information content of the view, zoom in and out of specific areas of the view and the ability to specify which diagram layers are included. A diagram layer consists of a group of diagram elements. Each layer can be included or removed, be assigned colors or offsets, etc.). [1, created for SECM]

***Interface Agreement Definition -*** Specifies the rules that constrain an interaction.

Two examples of interface agreements are the constraints on a physical interaction that are specified by conservation laws, and a communication protocol: [1, created for SECM]

In telecommunications, a communications protocol is a system of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. These are the rules or standard that defines the syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both. [8, Wiki, Communications Protocol]

***Interface Definition -*** 1. A relationship that enables the physical and functional interaction between elements that includes two (2) interface ends, the connection between them, and the constraints on the interaction. [1, created for SECM]

2. A shared boundary between two functional units, defined by various characteristics pertaining to the functions, physical signal exchanges, and other characteristics. (ISO/IEC 1993)

3. A hardware or software component that connects two or more other components for the purpose of passing information from one to the other. (ISO/IEC 1993)

4. To connect two or more components for the purpose of passing information from one to the other. (ISO/IEC/IEEE 200)

[3, SEBoK Glossary]

***Interface Medium Definition -*** A transmission medium between components. It presents interface ends that can connect to other components via interface agreements. Examples of interface medium include a wire, a cable harness, a computer bus, a network, a pipe that enables the flow of fluid, and the atmosphere. [1, created for SECM]

***Interface Requirement -*** One of the potential category requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be associated with an interface. [1, created for SECM]

***ISM -*** Acronym for Integrated System Model. See Integrated System Model for the definition. [1, created for SECM]

***Layered Interface -*** An interface that is decomposed into functional units, called layers that can accept inputs from an adjacent layer, transform the inputs, and provide the outputs to its other adjacent layer. Each layer has its own interface with its adjacent layers that are governed by interface agreements.. Each layer typically addresses a distinct set of concerns. A fundamental principle of an interface layer is that the layer below is independent of the layer above. A Protocol Stack is a set of layers that transforms items to enable their exchange, such as for purposes of communication. [1, created for SECM]

***Logical Expression -*** An expression that supports as a minimum the standard boolean operators AND, OR, XOR, NOT, and conditional expressions like IF-THEN-ELSE and IF-AND-ONLY-IF, in which symbols bound to any characteristics (e.g. value properties or usage features) may be used. [1, created for SECM]

***Machine-readable Data -*** Machine-readable data is data (or metadata) which is in a format that can be understood by a computer. [8, Wiki]

***Mapping -*** Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model or data source that conforms to another (possibly the same) metamodel. [33, OMG RFP Template]

***Mapping Rules -*** A set of rules that define how the elements of a model conforming to a particular metamodel are transformed into elements of another model or data source that conforms to another (possibly the same) metamodel. [1, created for SECM]

***Mathematical Logic -*** An extension of the formal method of mathematics to the field of logic. [19, Mathematical Logic, Hilbert & Ackerman]

***MCI -*** An acronym for Model Configuration Item. See Model Configuration Item for the definition. [1, created for SECM]

***MCI Configuration -*** A set of MCIs with their associated Versions and Variants. Within a specific Configuration, every MCI has a single and unique Version/Variant. A configuration is itself a MCI. [1, created for SECM]

***Meta Object Facility -*** Provides the basis for metamodel definition in OMG's family of MDA languages and is based on a simplification of UML2's class modeling capabilities. In addition to providing the means for metamodel definition it adds core capabilities for model management in general, including Identifiers, a simple generic Tag capability and Reflective operations that are defined generically and can be applied regardless of metamodel. [31, MOF Spec]

An OMG standard, closely related to UML, that enables metadata management and language definition. [33, OMG RFP Template]

***Metadata -*** Data that provides information about other data that is used to summarize basic information about data to make tracking and working with data easier. Some examples include:

* Means of creation of the data
* Purpose of the data
* Time and date of creation
* Creator or author of the data
* Location on a computer network where the data was created
* Standards used
* File size

[8, derived from Wiki, Metadata]

***Metamodel -*** A metamodel is a model of a modeling language. [1, created for SECM]

***MLM -*** Acronym for Model Lifecycle Management. See Model Lifecycle Management for the definition.  [1, created for SECM]

***Model -*** A representation of one or more concepts that may be realized in a physical world. [20, A Practical guide to SysML]

***Model Collection -*** A collection of model elements and/or modeling constructs used to construct the system model. [1, created for SECM]

***Model Configuration Item -*** A specific portion of the system model (content and granularity) that is maintained in a controlled fashion, i.e. has a unique ID and version history. MCI can be defined in different granularities, from an individual fine grained Model Element, a set of Model Elements, a set of Elements, to the entire Model. Any MCI can contain another MCI. [1, created for SECM]

***Model Constructor -*** A model element or other entity used to support the construction of a system model that includes model patterns, queries, rules and expressions, transformations, and links to external data elements. [1, created for SECM]

***Model Element -*** A constituent of a model. A model element in a system model typically is used to represent some aspect of the system or its environment that may include representations of structure, behavior, requirements, and their relationships at varying levels of granularity. [1, created for SECM]

***Model Library -*** A model element that contains other model elements that are designated for reuse.

The contained elements typically can be copied, sub-classed, or referenced for use in a model. [1, created for SECM]

***Model Lifecycle Management -*** Manages the Engineering Change Process to control the Model Configuration. [1, created for SECM]

***Model Pattern -*** A Model Pattern is a specification of a set of model elements including their relationships that can be used (i.e., applied) to create a conforming model fragment (e.g., an interface pattern). [1, created for SECM]

(1) An expression of an observed regularity. (Alexander 1979)

(2) A representation of similarities in a set or class of problems, solutions, or systems. (Alexander 1979)

(3) Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. (Alexander 1979)

[3, SEBoK Glossary]

***Model Repository -*** The model repository is located within the System Modeling Environment and contains the system models, analyses data, metadata, reuse libraries and persistent view data. [1, created for SECM]

***Model Transformation -*** A mapping between two modeling languages or other data sources that enables a model expressed in one modeling language to be expressed in whole or in part in the other modeling language or data source. (Created for SEBoK) [3, SEBoK Glossary]

***Model Validation -*** The process of ensuring the model correctly represents the domain or system-of-interest. (Friedenthal 2009) [3, SEBoK Glossary]

***Model-Theoretic Semantics -*** An account of meaning in which sentences are interpreted in terms of a model of, or abstract formal structure representing, an actual or possible state of the world: compare possible world. Usually, at least, an account of truth conditions; i.e. sentences are interpreted as true or false in such a model. [32, Oxford Dictionary of Linguistics]

***Modeling Language -*** A language used to represent models. [1, created for SECM]

***MOF -*** Acronym for Meta Object Facility. See Meta Object Facility for the definition.  [1, created for SECM]

***Navigation Relationship -*** A navigable connection from a model element or text within a model element to another model element within the same model or an external element. [1, created for SECM]

The connections may have different behaviors, such as (1) reference connections for basic traceability, (2) data map connections for exchange for parameter values, (3) function wrap connections for wrapping executable code in system model elements, and (4) model transform connections for generating and synchronizing model structures bi-directionally. [29, Intro to SLIM]

An identifier attached to an element (as an index term) in a system in order to indicate or permit connection with other similarly identified elements; especially: one (as a hyper link) in a computer file [11, Merriam-Webster]

***Objective -*** A desired or required state. An example is to achieve a certain level of performance within specified cost constraints. [1, created for SECM]

***Operational Semantics -*** Meanings for program phrases defined in terms of the steps of computation they can take during program execution. [23, Denotational Semantics Lecture]

***Performance Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. [1, created for SECM]

***Physical Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be a physical related requirement. [1, created for SECM]

***PIM -*** Acronym for Platform Independent Model. See Platform Independent Model for the definition. [1, created for SECM]

***Plan Verification Activity -*** An activity that creates a verification plan for the verification case. [1, created for SECM]

***Platform Independent Model -*** (PIM) A model of a system, subsystem, or component that contains no information specific to the technology that is used to realize it. [1, created for SECM]

***Platform Specific Binding -*** A general approach for mapping a Platform Independent Model to a corresponding Platform Specific Model in a specific technology platform, with an implementation that is directly usable in the target platform.

***Platform Specific Model -*** (PSM) - A model of a system, subsystem, or component that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform. [33, OMG RFP Template]

***Port Definition -*** Also known as "Interface End Definition". The specification of a connection point that enables one element to be connected to another. [1, created for SECM]

***Precondition Expression -*** A set of conditions that must exist prior to initiating a behavior. [1, created for SECM]

Something that must exist or happen before something else can exist or happen [11, Merriam-Webster on-line definition]

***Presentation Method -*** A method that provides the functionality to generate a view of the model data that includes graphical, tabular, and textual form. [1, created for SECM]

***Priority Attribute -*** This is how important the requirement is to the stakeholders. It may not be a critical requirement (that is, one the system must possess or it won't work at all), but simply something that the stakeholder(s) hold very dear. Priority may be characterized in terms of a level (1, 2, 3 or high, medium, low). Priority may be inherited from a parent requirement. High priority requirements must always be met for the project to be successful; lower priority requirements may be traded off when conflicts occur or when there are budget or schedule issues. [14, Guide Writing Requirements, 5.3.20]

***Problem -*** A Problem is a deficiency, limitation, or failure to satisfy a requirement or need or cause some other undesired outcome or effect from the perspective of a stakeholder. It may be used during any lifecycle phase including design, verification, manufacture, and operations. [16, derived from SysML spec]

***Process Requirement -*** One of the potential category requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be imposed on a process that is to be used during a development, manufacturing, support or maintenance process. [1, created for SECM]

***Property -*** Any named, measurable or observable attribute, quality or characteristic of a system or system element. (OMG 2003) [3 SEBoK Glossary]

***PSM -*** Acronym for Platform Specific Model. See Platform Specific Model for the definition. [1, created for SECM]

***Query -*** A precise request for information retrieval from a model, database, or other source of structure data. [8, Derived from Wiki]

***Query Language -*** Query languages are computer languages used to make queries in databases and information systems. [8, Wiki]

***Query Method -*** A method that provides the functionality to query a model and associated metadata. [1, created for SECM]

***Rationale -*** Argument that provides the justification for the selection of an engineering element. (Faisandier 2012) [3, SEBoK Glossary]

A model element that refers to other data that provides justification for a decision or other conclusion. [1, created for SECM]

***Realization Relationship -*** Realization is a specialized Abstraction relationship between two sets of model Elements, one representing a specification (the supplier) and the other represents an implementation of the latter (the client). Realization can be used to model stepwise refinement, optimizations, transformations, templates, model synthesis, framework composition, etc. [15, UML Spec]

***Realized Element -*** An element within the realization component that is constrained by the constraining element. [1, created for SECM]

***Reference Information -*** b: something (such as a sign or indication) that refers a reader or consulter to another source of information (such as a book or passage). [11, Merriam Webster on-line dictionary]

This text can contain navigational links from this element or text within this element to the actual referenced material. [1, created for SECM]

***Reference Model -*** A reference model in systems, enterprise, and software engineering is an abstract framework or domain-specific ontology consisting of an interlinked set of clearly defined concepts produced by an expert or body of experts in order to encourage clear communication. A reference model can represent the component parts of any consistent idea, from business functions to system components, as long as it represents a complete set. This frame of reference can then be used to communicate ideas clearly among members of the same community. [8, Wiki]

***Refine Relationship -*** A relationship between two elements where the element on one side provides a more precise representation than the other. An example is the relationships between two text statements, where one statement is verbose and ambiguous, and the other statement is concise and precise. [1, created for SECM]

***Relationship -*** The way in which two or more things are connected. [11, Merriam-Webster on-line]

A representation of an association between model elements. SysML constrains relationships to be have two ends (i.e., a binary relationship). A relationship can have a name, and have a direction between its source end and target end. [1, created for SECM]

***Reliability Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be a reliability related requirement. [1, created for SECM]

***Required/Desired -*** This is a property of the formal requirement statement that defines if the requirement is a mandatory (required) requirement or a desired requirement from a customer perspective. In textual requirement statements the verb "shall" or "will" are typically used, respectively, to do this. When a textual form of the formal requirement statement is automatically generated then this verb will be used in the textual statement. [1, created for SECM]

***Requirement -*** Also known as Requirement Usage. A usage of a Requirement Definition. [1, created for SECM]

***Requirement Attribute Library -*** A digital library containing a collection of predefined model elements representing a set of reusable requirement attributes that can be copied or reference while constructing a model.

This collection should include all attributes and types defined in the INCOSE Requirements Writing Guide (see reference14).

The intent is to make this library available to each organization and/or project to allow that organization or project to select which best fit their workflow needs.

[1, created for SECM]

***Requirement Attribute -*** Additional information included with a requirement definition, which is used to aid in the management of that requirement. [14, derived from Guide Writing Requirements, Definitions)

***Requirement Context Element -*** An element that is referenced in the formal requirement statement that is contained within the context of the constraining element. [1, created for SECM]

***Requirement Definition -*** A definition of a constraint that is used in the context of a specification that a valid solution must satisfy. [1, created for SECM]

Statement that identifies a product\* or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability. (ISO/IEC 2007)

\*includes product, service, or enterprise. [3, SEBoK Glossary]

***Requirement Derivation Relationship -*** A derived relationship imposes constraints that are needed to satisfy other constraints (i.e., source requirements). The derived requirement can impose requirements at the same level of the design as the source requirement, or at a lower-level of design. [1, created for SECM]

***Requirement Group -*** A model element that can group requirements to give them context. A specification is a top-level requirement group. [1, created for SECM]

***Requirement Group Relationship -*** A relationship between a requirement group and a requirement to establish context for the requirement definition. [1, created for SECM]

***Requirement Identifier -*** This is an identifier that uniquely identifies a requirement from other requirements. This identifier is not a paragraph number. It can be a separate identifier or automatically assigned by a Requirement Management Tool (RMT) the organization is using.

This identifier is not the same as the Unique Identifier) that every model element contains. This identifier should be unique across all requirements and can be tailored to meet a specific organization's needs. This identifier typically includes some encoding to help humans relate to its context (for example CR\_100 for a customer requirement, where CR\_ is a user-defined prefix unique to a requirement specification, and 100 is tool generated). [1, created for SECM]

***Requirement Status Attribute -*** This requirement attribute is intended to maintain the current status of the requirement. Typical values can include "draft", "ready for review", "accepted", "rejected", "implemented" and "verified".

A requirement can continue to change after being accepted, implemented and/or verified. This change control management is typically managed via the same change control process as other model elements. [1, created for SECM]

***Requirement Type/Category -*** Each organization will define types or categories to which a requirement fits, based on how they may wish to organize their requirements. The Type/Category field is most useful because it allows the requirements database to be viewed by a large number of designers and stakeholders for a wide range of uses. [14, Guide Writing Requirements, 5.3.25]

***Restricted Requirement Statement -*** A specific type of Textual Requirement Statement, specified by using a restricted/controlled natural language that puts restrictions on grammar (which can be realized by templates and patterns) and vocabulary (by using e.g., pre-defined keywords). Restricted Requirement Statements (RRS) strikes a balance between practicality and level of automation, bridges the gap from informal requirements specifications in natural language to formal, precise, and analyzable specifications. [1, created for SECM]

***Result Expression -*** This is the expression that determines the boundary used by the Constraint Evaluation. This boundary expression can be as simple as a text statement or define a volume in three-dimensional space. It can be presented in many forms including a table, equation, multidimensional graph or text. [1, created for SECM]

***Revision -*** A state associated with the lifetime of a MCI at a given point in time, as designated by the formal release change process. [1, created for SECM]

***Risk Attribute -*** A risk value assigned to each requirement based on risk factors. Requirements that are at risk include requirements that fail to have the set of characteristics that all well-formed requirements must have: necessary, singular, conforming, appropriate, correct, unambiguous, complete, feasible, and verifiable. Risk can also address feasibility/attainability in terms of technology, schedule, and cost. If the technology needed to meet the requirement is new with a low maturity, the risk is higher than if using a mature technology used in other similar projects. The requirement can be high risk if the cost and time to develop a technology is outside what has been planned for the project. Risk may be inherited from a parent requirement. [14, Guide Writing Requirements, 5.3.22]

***Safety Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be a safety related requirement. [1, created for SECM]

***Satisfy Relationship -*** A relationship between a requirement and the constrained element of the design solution, which asserts that the constraints imposed by the requirement on the design element are met.

Satisfy is intended to be a specialized kind of conform relationship where one side of the relationship must be a requirement.

[1, created for SECM]

***Scenario Definition -*** A sequence of actions that are performed by interacting components. [1, created for SECM]

***Security Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be a security related requirement. [1, created for SECM]

***Semantics -*** The rules by which syntactic expressions and model elements are assigned meaning. (ISO 13537:2010, 3.2.3.14) [17, ISO OBP Definitions]

***Service -*** A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement on-line.

A service has four properties according to one of many definitions of SOA:

1. It logically represents a business activity with a specified outcome.
2. It is self-contained.
3. It is a black box for its consumers.
4. It may consist of other underlying services.

[8, Wiki, Service-oriented architecture]

***SME -*** Acronym for System Modeling Environment. See System Modeling Environment for the definition.

***SME User -*** A specific, defined end user of the System Modeling Environment (SME). [1, Created for SECM]

***SMOF -*** MOF Support for Semantic Structures. This extension to MOF modifies MOF 2 to support dynamically mutable multiple classifications of elements and to declare the circumstances under which such multiple classifications are allowed, required, and prohibited. [30, OMG SMOF]

***Snapshot -*** The state of each state variable of an individual item at a point of time. For example, a snapshot of a vehicle may include the value of its position, velocity, and acceleration at a point in time, and the snapshot of its engine may include the value of its power-out and temperature at the same point in time. [1, created for SECM]

***Software -*** All or part of the programs, procedures, rules, and associated documentation of an information processing system. (ISO/IEC 2382-1:1993) [3, SEBoK Glossary]

A class of component which implements functionality that is specified by a computer program. [1, created for SECM]

***Specify Relationship -*** The Specify Relationship assumes that the realization has (or will) provided a satisfaction relationship from its constrained elements to each of the applicable requirements in the Component Specification. It can be thought of as a group of satisfied relationships.

The "conforms to" connection will be able to specify what part, or subset of requirements, of the specification are applicable. This can be done by identifying each requirement ID or by identifying one or more requirement groups IDs.

See the definition for the satisfy relationship to see the distinction in the terms satisfy, conforms, realize and specify, as defined for this effort.

[1, created for SECM]

***Stakeholder -*** (1) Individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations (ISO/IEC/IEEE 2015)

(2) Individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations; N.B. Stakeholders include, but are not limited to end users, end user organizations, supporters, developers, producers, trainers, maintainers, disposers, acquires, customers, operators, supplier organizations and regulatory bodies. (ISO/IEC June 2010)

(3) An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system or domain of interest. (Derived from ISO/IEC 2007)

(4) A stakeholder in an organization is (by definition) any group or individual who can affect or is affected by the achievement of the organization's objectives. (Freeman 1984)[3, SEBoK Glossary]

***Standard View -*** A standard view is a SysML defined diagram type. They may or may not be the same set that were defined for SysML 1.x. [1, created for SECM]

***State -*** The values of a state variable at a point in time. [1, created for SECM]

***State History -*** A ordered series of snapshots of an individual element that define how the value of its value properties change over the elements lifetime. Note that an analysis is often used to predict a state history associated with selected value properties, or in some cases, reconstruct a previous state history, such as when analyzing a failure that has occurred. Measurements are used to measure the state history associated with selected value properties. A predicted or measured state history has uncertainty associated with it. [1, created for SECM]

***State Machine -*** A representation of the finite states and the transitions between them for an individual element over its lifetime. In a particular finite state, or on transition between finite states, selected functions and constraints are enabled. A state machine may have multiple concurrent regions where only one finite state is active in each region at a given point in time.

Note: A lifetime of an individual element persists over the duration that it retains its identify. For example, a individual products lifetime may begin when it comes off the manufacturing line, and ends when it is dis-assembled for disposal. [1, created for SECM]

***State Variable -*** A value property whose value varies with time. [1, created for SECM]

***Structured Data -*** Structured data refers to information that has a high level of organization such as in a pre-defined data model, images, lists, spreadsheets, relational databases, etc. Structured data is data that has been organized into a formatted repository so that its elements can be made addressable for more effective processing and analysis. [Derived from WhatIs.com, 29 Nov 2016, http://whatis.techtarget.com/definition/structured-data].

***Supportability Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be a supportability related requirement. [1, created for SECM]

***Supporting Information -*** Supporting Information provides additional information to help better understand the intent of a model element and specifically for a requirement or requirement group. This information can include items such as an introduction to a set of requirements, one or more goals, a reference to further readings, justification, rationales, examples, diagrams, pictures, graphs, tables, etc. In addition it can include navigational links from this element or text within this element. [1, created for SECM]

***Syntax -*** Structure of expressions in a language, and the rules governing the structure of a language, independent of their meanings or the manner of their interpretation and use. (derived from ISO/PAS 16917:2002(en), 3.2.68) [17, ISO OBP Definitions]

***SysML -*** The OMG Systems Modeling Language (OMG SysML®) is a general-purpose language for representing systems.

SysML supports the specification, analysis, design, verification, and validation of a broad range of complex systems.

These systems may include hardware, software, information, processes, personnel, and facilities. [16, derived from SysML v1 specification]

***SysML v2 Declarative Semantics -*** A declarative specification of the semantics of the SysML v2 Foundational Subset using a mathematical logic formalism. [1, created for SECM]

***SysML v2 Diagram Definition -*** A specification of the standard diagrammatic views included in the SysML v2 Concrete Syntax, using a metamodel-based approach consistent with the OMG Diagram Definition standard. [1, created for SECM]

***SysML v2 Foundational Subset -*** The foundational subset of the SysML v2 Language in which the SysML v2 Semantic Model (or at least the core of it) and whose semantics is also separately specified by the SysML v2 Declarative Semantics. [1, created for SECM]

***SysML v2 Metamodel -*** A model of the SysML v2 modeling language that includes its abstract syntax, concrete syntax, semantics, and the mappings between them. [1, created for SECM]

***SysML v2 Model -*** A top-level container of model elements represented in the SysML v2 modeling language, which is specified by the SysML v2 metamodel. [1, created for SECM]

***SysML v2 Semantics Model -*** A SysML v2 Model Library that specifies the SysML v2 Semantics as a SysML v2 Model. [1, created for SECM]

***System -*** (1) A set of elements in interaction. (von Bertalanffy 1968)

(2) Combination of interacting elements organized to achieve one or more stated purposes (ISO/IEC/IEEE 15288:2015)

[3, SEBoK Glossary]

"A set of elements that can interact with one another, and can be viewed as a whole which can interact with other external elements." [1, created for SECM]

***System Context -*** (1) Describes the system relationships and environment, resolved around a selected system-of-interest. (Flood and Carson 1993) [3, SEBoK Glossary]

***System Element -*** A member of a set of elements that constitutes a system. A system element is a discrete part of a system that can be implemented to fulfill specified requirements. A system element can be hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities (e.g., water, organisms, minerals), or any combination. (ISO/IEC 15288:2015) [3, SEBoK Glossary]

***System Model -*** (3) A simplified representation of a system at some particular point in time or space intended to promote understanding of the real system. (Bellinger 2004)

(4) An abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system (Dori 2002)

(5) A selective representation of some system whose form and content are chosen based on a specific set of concerns. The model is related to the system by an explicit or implicit mapping. (Object Management Group 2010)

[3, SEBoK Glossary]

***System Model Management -*** Model Management Services for the Integrated System Model (ISM). It does not replace the linked model's native configuration management tool. [1, created for SECM]

***System Modeling Environment -*** (SME) A part of the overall Model-Based Engineering (MBE) environment that systems engineers use to perform model-based systems engineering (MBSE) and interact with other members of the development team. The SME must implement the SME services to provide the functionality needed to enable systems engineers and others to evolve the system model throughout the lifecycle. [21, Insight Article Part 2]

***Task -*** A piece of work to be undertaken. [1, Created for SECM]

***Task Status -*** The current situation (state) for a specific task (e.g. not started, stage of completeness, completed successfully, etc.) [1, Created for SECM]

***Textual Requirement Statement -*** The traditional "shall" textual statement used to state a requirement. [1, created for SECM]

***Time Property -*** A value property that represents the base quantity time. [1, created for SECM]

***Timestamp -*** A sequence of characters or encoded information identifying when a certain event occurred, including the date and time of day. The timestamp refers to digital date and time information attached to digital data. For example, computer files contain timestamps that tell when the file was last modified. [8, Wiki]

A timestamp should be represented using a common, time zone independent format that includes resolution and context such as UTC. Format example: time=2009-06-15T13:45:30; context=last change [1, created for SECM]

***Trade-off -*** A kind of analysis to evaluate a set of alternatives based on some criteria, and make a decision to select one or more of the alternatives. [1, created for SECM]

***UML Profile -*** A standardized set of extensions and constraints that tailors UML to particular use. [33, OMG RFP Template]

***Uncategorized Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies that categorizing requirements is part of the organization's process but the task has not been completed. [1, created for SECM]

***Unified Modeling Language -*** (UML) The objective of the Unified Modeling Language (UML) is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes. [15, UML Spec]

***Unique Identifier -*** This unique identifier is assigned to every element. This identifier must be unique universally, that is within the containing model, within the SME and external to the SME. [1, created for SECM]

***Unit Under Verification -*** A system or part of a system that is the subject of a verification procedure. [1, created for SECM]

***Units Library -*** A digital library containing a collection of predefined model elements representing a set of reusable units and quantity kinds that can be copied or referenced by a model. [1, created for SECM]

***URI -*** In information technology, a Uniform Resource Identifier (URI) is a string of characters used to identify a resource. Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. [8, Wiki]

***Usability -*** The extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. [25, ISO 9241-210:2010]

***Usability Requirement -*** One of the potential category of requirement selections available in the Requirement Type attribute. This selection identifies the requirement to be a user usability related requirement. [1, created for SECM]

***Usage Feature -*** Usage Feature - A generalization of a value property or a usage element that is contained by a Definition Element, and is typed by the applicable Value Type or Definition Element. [1, created for SECM]

***User Defined Requirement Attribute -*** This is a requirement attribute that is not available in the Requirement Attribute Library but can be can be created and defined by a specific organization or project to meet the needs of their workflow. See INCOSE Requirements Writing Guide [14] for additional attribute suggestions. [1, created for SECM]

***User Defined View -*** This type of view is one or more views that are defined specifically for a meet a user's or organization's needs. The presentations, shapes and icons may be unique for this specific use. [1, created for SECM]

***UUID -*** Universally Unique identifier (UUID) - A unique identifier assigned to every model element. This identifier must be unique both within the SME and external to the SME. This UUID conforms to IETF RFC 4122 <http://datatracker.ietf.org/doc/rfc4122/>. See also <http://en.wikipedia.org/wiki/Universally_unique_identifier> for a practical introduction on a UUID. [1, created for SECM]

***Validation Rule -*** A Validation rule is a criterion or constraint used in the process of data validation, carried out after the data has been encoded onto an input medium and involves a data vet or validation program. [8, Wiki]

***Value Expression -*** An expression that can be evaluated to yield a value typed by a Value Type. The expression is stated in an expression language that support all usual mathematical and logical operators. [1, created for SECM]

***Value Property -*** Any named, quantifiable characteristic of a class of elements whose range of values are constrained by a Value Type. [1, created for SECM]

***Value Type -*** Named definition of the set of possible values of a value-based characteristic. [1, created for SECM]

***Variability Context -*** A model that captures the desired variabilities and constraints for a set of configurations. [1, Created for SECM]

***Variability Expression -*** An expression that constrains the variant choices. [1, created for SECM]

***Variant -*** A variant (or option) represents a choice among one or more features. A variant can include additional variation points. [1, Created for SECM]

***Variant Binding -*** A variant binding binds a model element to an element in another model that specifies the variation. [1, Created for SECM]

***Variant Selection -*** Defines the selected variant based on the options identified by a variation point. [1, created for SECM]

***Variation Point -*** A definition of one or more choices of some characteristic feature of a definition element. The set of choices available may depend on other selections that have been made, such as the choice of entertainment systems that are available for a particular model of a vehicle. The addition of one or more choices allows for a compact and inherently consistent representation of options or alternatives at any level in the hierarchical composition that support modeling product lines. A configuration element reflects selection of a specific variant among the available choices. [1, created for SECM]

***Verification -*** (1a) Confirmation, through the provision of objective evidence, that specified (system) requirements have been fulfilled. (ISO/IEC 2008, section 4.38)

(1b) Verification is a set of activities that compares a system or system element against the required characteristics. This includes, but is not limited to, specified requirements, design description and the system itself. The system was built right. (ISO/IEC/IEEE 2015, 1, Section 6.4.6)

(2) The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation. (PMI 2013)

[3, SEBoK definition]

***Verification Outcome -*** Describes the data and any other results from performing the Verification Activity. [1, created for SECM]

***Verification Activity -*** An activity that specifies one or more steps of a verification case. [1, created for SECM]

***Verification Case -*** A set of steps required to achieve a verification objective. [1, created for SECM]

***Verification Context -*** A context for a unit under verification and the verification system that performs the verification as defined by the verification case. [1, created for SECM]

***Verification Evaluation Activity -*** An activity that compares the verification outcome data produced by the verification activity with the verification success criteria. [1, created for SECM]

***Verification Method -*** The verification method for each requirement simply states the planned method of verification (inspection, demonstration, test, analysis, and simulation). [10, INCOSE Handbook]

The Description property provides a textual description of the steps that will be taken in Verification Activity and Verification Evaluation Activity. [1, created for SECM]

The type of method may also include sampling and analogy. [2, SEBoK Verification]

***Verification Objective -*** The identification of the requirements that are to be verified. [1, created for SECM]

***Verification Plan -*** This plan identifies and includes a verification strategy, selected verification actions, verification procedures, verification tools, the verified element or system, verification reports, issue/trouble reports, and change requests on design. [3, SEBoK, System Verification]

***Verification Requirement -*** A requirement applied to the means of establishing compliance of an end item with its specification requirements. [1, created for SECM]

***Verification Result -*** The result of the Verification Evaluation. [1, created for SECM]

***Verification Success Criteria -*** The success criteria is a subset of the requirement being verified (e.g. selected points in a flight test envelope). [1, created for SECM]

***Verification System -*** An aggregation of enabling elements needed to perform verification activities. This includes the equipment, users and facilities used to perform the activity. [1, created for SECM]

***Verification System Element -*** A system element used to stimulate and interact with the unit under verification during the execution of the verification case. [1, created for SECM]

***Verify Relationship -*** A relationship between a requirement and a verification case that can be elaborated to specify how verification of the requirement is accomplished and to produce the verification result. [1, created for SECM]

***Version -*** A state associated with the lifetime of a MCI at a given point in time, as designated by the engineering change process. [1, created for SECM]

***View -*** Also known as "View Individual" - A representation of a specific artifact that is constructed to address one or more stakeholder concerns specified by a viewpoint. This may represent an electronic file or hard copy of a diagram, table, document, or even a physical model. [1, created for SECM]

***View Definition -*** The definition of the structure of a view in terms of its sub-views, and the methods needed to construct an individual view. The construction methods generally involve the querying of a model or other data source, and the presentation of the query results. A view definition is intended to conform to a viewpoint. [1, created for SECM]

***View Element -*** A constituent element of a Concrete Syntax Metamodel that defines how a model element is presented. [1, created for SECM]

***View Layout Definition -*** A layout definition defines how the view will be organized and presented. It applies to graphical symbols, tables, serialized data, etc. It can include non-model info such as a background. [1, created for SECM]

***Viewpoint -*** A digital library containing a collection of predefined model elements representing a set of reusable viewpoints that can be copied or reference while constructing a model. [1, created for SECM]

***Viewpoint Library -*** A digital library containing a collection of predefined model elements representing a set of reusable viewpoints that can be copied or reference while constructing a model. [1, created for SECM]

# Appendix B General Reference and Glossary

## B.1 General References

The following documents are referenced in this document:

[BCQ] OMG Board of Directors Business Committee Questionnaire

<http://doc.omg.org/bcq>

[CCM] CORBA Core Components Specification

<http://www.omg.org/spec/CCM/>

[CORBA] Common Object Request Broker Architecture (CORBA)

<http://www.omg.org/spec/CORBA/>

[CORP] UML Profile for CORBA

<http://www.omg.org/spec/CORP>

[CWM] Common Warehouse Metamodel Specification

<http://www.omg.org/spec/CWM>

[EDOC] UML Profile for EDOC Specification

<http://www.omg.org/spec/EDOC/>

[Guide] The OMG Hitchhiker's

<http://doc.omg.org/hh>

[IDL] Interface Definition Language Specification

<http://www.omg.org/spec/IDL35>

[INVENT] Inventory of Files for a Submission/Revision/Finalization

<http://doc.omg.org/inventory>

[IPR] IPR Policy

<http://doc.omg.org/ipr>

[ISO2] ISO/IEC Directives, Part 2 - Rules for the structure and drafting of International Standards

<http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456>

[LOI] OMG RFP Letter of Intent template

<http://doc.omg.org/loi>

[MDAa] OMG Architecture Board, "Model Driven Architecture - A Technical Perspective"

<http://www.omg.org/mda/papers.htm>

[MDAb] Developing in OMG's Model Driven Architecture (MDA)

<http://www.omg.org/mda/papers.htm>

[MDAc] MDA Guide

<http://www.omg.org/docs/omg/03-06-01.pdf>

[MDAd] MDA "The Architecture of Choice for a Changing World

<http://www.omg.org/mda>

[MOF] Meta Object Facility Specification

<http://www.omg.org/spec/MOF/>

[NS] Naming Service

<http://www.omg.org/spec/NAM>

[OMA] Object Management Architecture

<http://www.omg.org/oma/>

[OTS] Transaction Service

<http://www.omg.org/spec/OTS>

[P&P] Policies and Procedures of the OMG Technical Process

<http://doc.omg.org/pp>

[RAD] Resource Access Decision Facility

<http://www.omg.org/spec/RAD>

[ISO2] ISO/IEC Directives, Part 2 - Rules for the structure and drafting of International Standards

<http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456>

[RM-ODP]

ISO/IEC 10746

[SEC] CORBA Security Service

<http://www.omg.org/spec/SEC>

[TEMPL] Specification Template

<http://doc.omg.org/submission-template>

[TOS] Trading Object Service

<http://www.omg.org/spec/TRADE>

[UML] Unified Modeling Language Specification

<http://www.omg.org/spec/UML>

[XMI] XML Metadata Interchange Specification

<http://www.omg.org/spec/XMI>

## B.2 General Glossary

***Architecture Board (AB) -*** The OMG plenary that is responsible for ensuring the technical merit and MDA compliance of RFPs and their submissions. [33, OMG RFP Template]

***Board of Directors (BoD) -*** The OMG body that is responsible for adopting technology. [33, OMG RFP Template]

***Common Object Request Broker Architecture (CORBA) -*** An OMG distributed computing platform specification that is independent of implementation languages. [33, OMG RFP Template]

***Common Warehouse Metamodel (CWM) -*** An OMG specification for data repository integration. [33, OMG RFP Template]

***CORBA Component Model (CCM) -*** An OMG specification for an implementation language independent distributed component model. [33, OMG RFP Template]

***Interface Definition Language (IDL) -*** An OMG and ISO standard language for specifying interfaces and associated data structures. [33, OMG RFP Template]

***Letter of Intent (LOI) -*** A letter submitted to the OMG BoDs Business Committee signed by an officer of an organization signifying its intent to respond to the RFP and confirming the organizations willingness to comply with OMGs terms and conditions, and commercial availability requirements. [33, OMG RFP Template]

***Model Driven Architecture (MDA) -*** An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform. [33, OMG RFP Template]

***Normative Provisions -*** To which an implementation shall conform to in order to claim compliance with the standard (as opposed to non-normative or informative material, included only to assist in understanding the standard). [33, OMG RFP Template]

***Normative Reference References -*** To documents that contain provisions to which an implementation shall conform to in order to claim compliance with the standard. [33, OMG RFP Template]

***Platform -*** A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented. [33, OMG RFP Template]

***Platform Independent Model (PIM) -*** A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it. [33, OMG RFP Template]

***Request for Information (RFI) -*** A general request to industry, academia, and any other interested parties to submit information about a particular technology area to one of the OMG's Technology Committee subgroups. [33, OMG RFP Template]

***Request for Proposal (RFP) -*** A document requesting OMG members to submit proposals to an OMG Technology Committee. [33, OMG RFP Template]

***Task Force (TF) -*** The OMG Technology Committee subgroup responsible for issuing a RFP and evaluating submission(s). [33, OMG RFP Template]

***Technology Committee (TC) -*** The body responsible for recommending technologies for adoption to the BoD. There are two TCs in OMG the Platform TC (PTC) focuses on IT and modeling infrastructure related standards; while the Domain TC (DTC) focuses on domain specific standards. [33, OMG RFP Template]

***XML Metadata Interchange (XMI) -*** An OMG standard that facilitates interchange of models via XML documents. [33, OMG RFP Template]