«DigitalEngineering»

DEKonsult

SysML Version 2 Approaching Industrial Use

Hans Peter de Koning (DEKonsult)

ESA MBSE2021 Workshop, 29-30 September 2021, Virtual Event

Note: Material in this presentation is based on publicly released information from the SysML Version 2 Submission Team, of which the author is a member.

What is SysML?

- Systems Modeling Language – by Object Management Group (OMG)
  - a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities [from https://www.omgsysml.org]
  - A language supporting MBSE – Not an MBSE methodology
  - SysML v1 is a profile & extension of UML2 – Unified (SW) Modeling Language
  - OMG standard (officially "adopted specification")
    - Version 1.0 released 2007
    - In real industrial use since 2010 – v1.2
    - Many tool implementations – COTS and open source
    - Latest release is v1.6 (December 2019)
    - v1.7 under development – will be the final version 1
  - Also used for system modelling / concepts in OMG UAF (Unified Architecture Framework)
    - UAF is for system-of-systems and enterprise architecture
    - UAF is the unification of DoDAF, MoDAF and NAF

Strength: Enabled implementation on top of mature UML tools & good support for software intensive systems

Weakness: "Software engineering flavoured" tools caused steep learning curve for many systems engineers

OMG SysML v2 Development Timeline

Standards Development Organization

2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022

SysML v2 RFP WG

Request for Proposal

- Including SE Concept Model
- Started 2015
- Language RFP released Dec 2017
- API RFP released May 2018

Submission Team

- Initially two submission teams – since end 2018 merged into one
- Started Jan 2018 on language (KerML & SysML)
- Started June 2018 on API & Services
- Agile workflow with monthly sprints
- Initial submission of language & API Spec's delivered to OMG on 17 Aug 2020
  (3 parts: KerML, SysML language, API & Services)
- Includes working open-source pilot implementations and training material with examples
  - Monthly releases on Github

- Major stakeholders' review held in Feb 2021
- Revised submission delivered to OMG on 30 Aug 2021 for review
- Monthly releases of pilot implementations
- Final submission to OMG scheduled for Feb 2022

Finalization Task Force

- Finalization of the specifications
- Monthly releases of pilot implementations and training material
- Production tool developments (both COTS and open-source)

SysML v2 Requirements and Constraints

- Extensive RFP (Request for Proposal)
  - Based on thorough analysis addressing the shortcomings of SysML v1
  - Broad participation from many industry sectors
  - Part 1: Systems Modeling Language (SysML®) v2 RFP
    - 141 mandatory and 31 non-mandatory requirements
    - See https://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-2
  - Part 2: Systems Modeling Language (SysML®) v2 API and Services RFP
    - 19 mandatory and 25 non-mandatory requirements
    - See https://www.omg.org/cgi-bin/doc.cgi?ad/2018-6-3
- SysML v2 shall be based on SMOF (Semantic Meta Object Facility)
  - Provides support for temporal aspects and multiple classifications
  - Information modelling founded on strong formal, semantic framework
  - Allows for mapping to other semantic frameworks like RDF/OWL2 DL
- Must provide migration path from SysML v1 – that can be automated
  - For both tool and model/data transition

Snippet from Language RFP

6.5.2.5 Behavior Requirements

BHV 1: Behavior Requirements Group

BHV 1.01: Behavior

Proposals for 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.

SysML v1.X Constructs: Activity, State Machine, Interaction, Simple Time

BHV 1.02: Behavior Decomposition

Proposals for 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.

SysML v1.X Constructs: Composited Association of Behavior Classifiers with Adjunct Properties

BHV 1.03: Function-based Behavior Group

BHV 1.03.1: Function-based Behavior

OMG RFP 9 December 2017 49

SysML v2 Submission Team (SST)

- SST formed December 2017
  - Leads:
    Sandy Friedenthal (SAF Consulting)
    Ed Seidewitz (Model Driven Solutions)
- Broad team of end-users, vendors, academia, and government liaisons
  - Currently around 193 members from 80 organizations
  - Large aerospace representation, but many other industry sectors as well
  - Majority of SysML tool vendors on board
- Develops integrated submission for both Language and API & Services

Participating Organizations

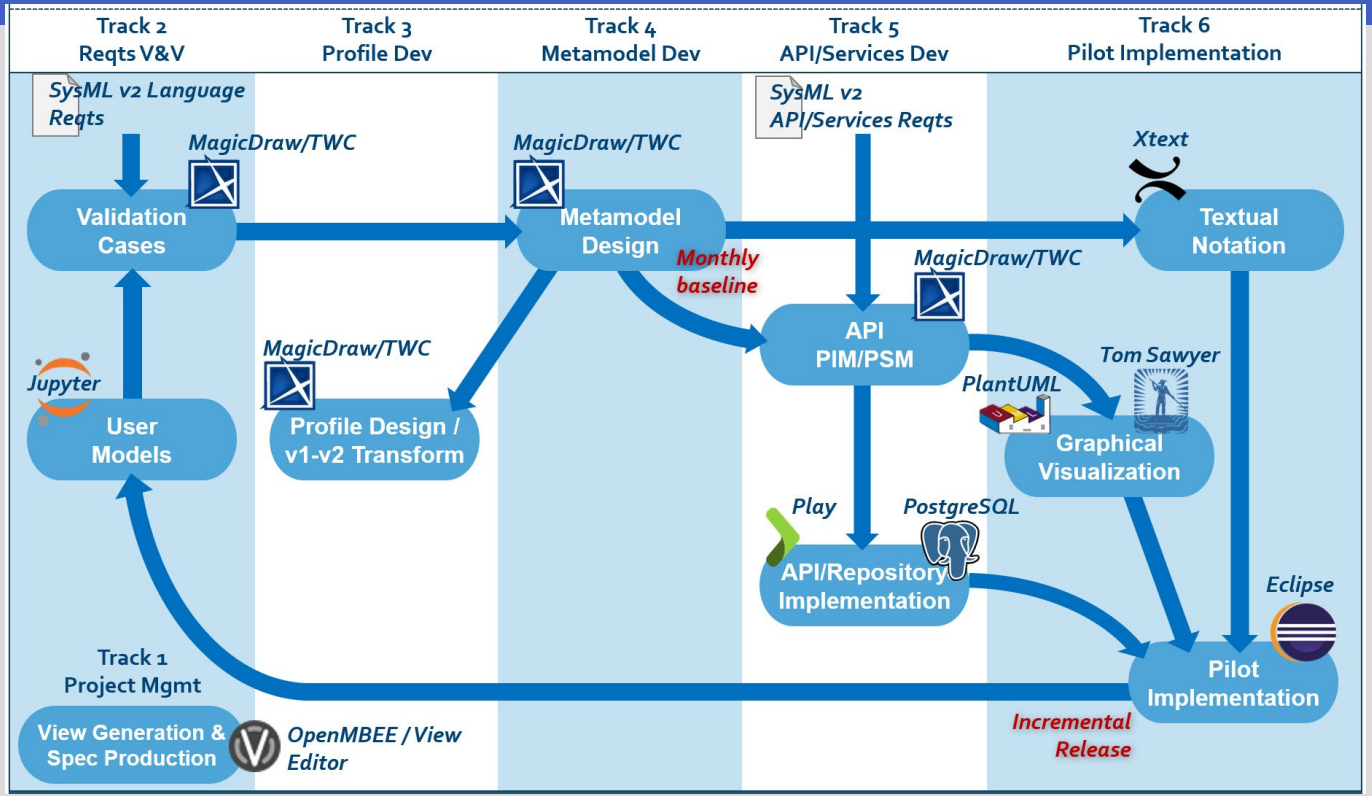
- Aerospace Corp
- Airbus
- ANSYS medini
- Aras
- Army Aviation & Missile Center
- Army CBRND
- BAE
- BigLever Software
- Boeing
- Army CCDC Armaments Center
- CalTech CTME
- CEA
- Contact Software
- Defence Science and Technology Group
- DEKonsult
- Delligatti Associates
- Draper Lab
- ESTACA
- Ford
- Fraunhofer FOKUS
- General Motors
- George Mason University
- GfSE
- Georgia Tech/GTRI
- IBM
- Idaho National Laboratory
- IncQuery Labs
- Intercax
- Itemis
- Jet Propulsion Lab
- John Deere
- Kenntnis
- KTH Royal Institute of Technology
- LieberLieber
- Lightstreet Consulting
- Lincoln Lab
- Lockheed Martin
- MathWorks
- Maplesoft
- Mercury Systems
- Mgnite Inc
- MID
- MITRE
- ModelAlchemy Consulting
- Model Driven Solutions
- Model Foundry
- NIST
- No Magic/Dassault Systemes
- OAR
- Obeo
- OOSE
- Ostfold University College
- Phoenix Integration/ANSYS
- PTC
- Qualtech Systems, Inc (QSI)
- Raytheon
- Rolls Royce
- Saab Aeronautics
- SAF Consulting *
- SAIC
- Siemens
- Sierra Nevada Corporation
- Simula
- Space Cooperative
- Sodius Willert
- System Strategy *
- Tata Consultancy Services
- Thales
- Thematix
- Tom Sawyer
- Twingineer
- UFRPE
- University of Western Switzerland (Rosas Center)
- University of Cantabria
- University of Alabama in Huntsville
- University of Detroit Mercy
- University of Kaiserslautern / VPE
- Vera C. Rubin Observatory
- Vitech
- 88solutions

Academia/Research
End User

Tool Vendors
Government Rep

INCOSE rep *

SST Agile / Incremental Development Workflow



Revised Submission – 30 Aug 2021 – 3 specs

Date: August 2021

Kernel Modeling Language (KerML)

Version 1.0
Revised Submission

OMG Document Number: ad/2021-08-02

Machine Readable Files:
KerML Metamodel (XMI) ad/2021-08-05
KerML Model Library (textual notation) ad/2021-08-06

Submitted in partial response to Systems Modeling Language (SysML®) v2 RFP (ad/2017-12-02) by:

88Solutions Corporation	Lockheed Martin Corporation
Dassault Systèmes	MITRE
GfSE e.V.	Model Driven Solutions, Inc.
IBM	PTC
INCOSE	Simula Research Laboratory AS
InterCax LLC	Thematix

Date: August 2021

OMG Systems Modeling Language ™ (SysML®)

Version 2.0
Revised Submission

OMG Document Number: ad/2021-08-03

Machine Readable Files:
SysML Metamodel (XMI) ad/2021-08-07
SysML Model Library (textual notation) ad/2021-08-08
SysML v1 to v2 Transformation (XMI) ad/2021-08-09

Submitted in response to Systems Modeling Language (SysML®) v2 RFP (ad/2017-11-04) by:

88Solutions Corporation	Lockheed Martin Corporation
Dassault Systèmes	MITRE
GfSE e.V.	Model Driven Solutions, Inc.
IBM	PTC
INCOSE	Simula Research Laboratory AS
InterCax LLC	Thematix

Date: August 2021

Systems Modeling Application Programming Interface (API) and Services

Version 1.0
Revised Submission

OMG Document Number: ad/2021-08-04

Machine Readable Files:
API Platform Independent Model (XMI) ad/2021-08-10
HTTP/REST API Binding (OpenAPI) ad/2021-08-11
OSLC API Binding (OpenAPI) ad/2021-08-12

Submitted in response to Systems Modeling Language (SysML®) v2 API and Services RFP (ad/2018-06-03) by:

88Solutions Corporation	InterCax LLC
Dassault Systèmes	Lockheed Martin Corporation
GfSE e.V.	Model Driven Solutions, Inc.
IBM	PTC
INCOSE	Simula Research Laboratory AS

Latest release at https://github.com/Systems-Modeling/SysML-v2-Release/tree/master/doc

New Layered Architecture

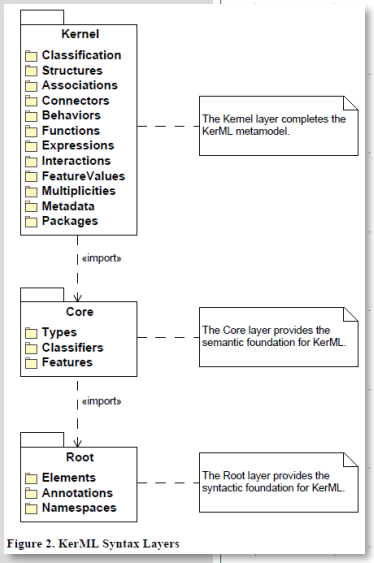
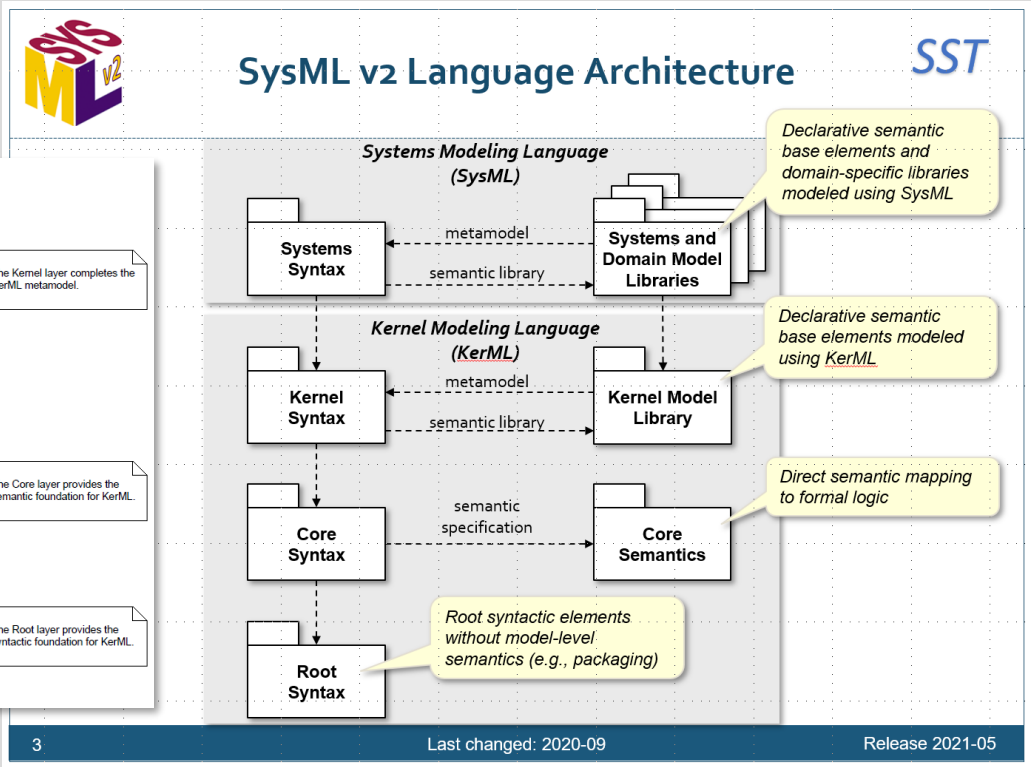


Figure 2. KerML Syntax Layers



- KerML: generic, rigorous, minimal, formal semantic foundation
  - Formal semantics, in OWL2 ontology style, but expressed in SMOF
  - Domain-independent
  - Not constrained by UML2
- SysML: adaptation and extension for systems engineering
  - Systems engineering concepts and terminology
  - Model libraries of essential and generally used concepts
- Extensible by design

Source: Intro to the SysML v2 Language-Textual Notation on GitHub

Impression of the new v2 terminology w.r.t. v1

Note: non-exhaustive list

SysML v1	SysML v2 (textual syntax keywords)	SysML v2 (metamodel concepts)
part property | block	part | part def	PartUsage | PartDefinition
value property | value type	attribute | attribute def	AttributeUsage | AttributeDefinition
proxy port | interface block	port | port def	PortUsage | PortDefinition
action | activity	action | action def	ActionUsage | ActionDefinition
state | state machine	state | state def	StateUsage | StateDefinition
constraint property | constraint block	constraint | constraint def	ConstraintUsage | ConstraintDefinition
requirement	requirement | requirement def	RequirementUsage | RequirementDefinition
connector | association block	connection | connection def interface | interface def	ConnectionUsage | ConnectionDefinition InterfaceUsage | InterfaceDefinition
use case	use case | use case def	UseCaseUsage | UseCaseDefinition

- Structure, behavior and any other decomposition fully regularised
- Consistent pattern of Usage and Definition for any concept that can be ‘typed’

Source: Intro to the SysML v2 Language-Graphical Notation on GitHub

SysML v2 – Key Concepts and Innovations

- Powerful textual language alongside graphical language
- Extensible Model Libraries (M1) rather than profiles with stereotypes (M2)
- “Usage-Focused” modelling approach
  - Makes modelling deeply-nested decomposition easy and natural
  - While still supporting “Definition-Oriented” approach to achieve modularity
- Sophisticated (smart) package import built into language
- 4D modelling object life and spatial extent as Occurrences & Snapshots
  - Including portions of life / extent, time-slices
- Support for variation points and variants
  - Enables PLE, product configurations, design alternatives, options, trade-offs, ...

SysML v2 – Key Concepts and Innovations (cont'd)

- Specification of reusable calculations as well as analysis / simulation and verification cases
- Modelling of Individuals
  - E.g., for serial-numbered items, 'digital twins', analysis/simulation executions
- Integrated behaviour modelling: control flow, state machines, sequences
  - Sync / async, serial / concurrent, signals, messages, events, aligned with ITU MSC
- Comprehensive set of extensible domain libraries
  - Mathematical, logical, utility functions, integrated with textual expression language
  - Quantities, Units, Scales and Quantity Dimensions (full ISO/IEC 80000 "SI", US Customary)
  - Time & Clocks, State-Space Representation, Basic Geometry
- Standardized API & Services for interoperability, XMI and JSON serialization
- Improved, flexible Viewpoints & Views aligned with ISO/IEC/IEEE 42010

Pilot Implementations
Try-out with Eclipse IDE or Jupyter Lab Notebook

JupyterLab — localhost:8888/lab

File Edit View Run Kernel Tabs Settings Help

Launcher | SimpleCarExample.ipynb

Simple Car Example

Simplistic structure model of a car with a body and four wheels.

For the wheels there is a choice between basic and de luxe variants, with 19 or 20 inch rims respectively.

[33]:
```
package SimpleCarExample {
    import ISQ::*;
    import SI::*;
    import USCustomaryUnits::*;

    // Generic wheel type
    part def Wheel {
        attribute diameter redefines ISQ::diameter;
        attribute mass redefines ISQ::mass;
    }

    // Basic wheel specialization with 19 inch rim, using explicit long-form textual syntax
    part def WheelBasic specializes Wheel {
        attribute diameter redefines Wheel::diameter = 19 ['in'];
        attribute mass redefines Wheel::mass = 12.5 [kg];
    }

    // De Luxe wheel specialization with 20 inch rim, using shortcut textual syntax
    part def WheelDeLuxe :> Wheel {
        :>> diameter = 20 ['in'];
        :>> mass = 13.7 [kg];
    }

    // Example variability modelling by a variation point for wheel options
    variation part def WheelChoice specializes Wheel {
        variant part WheelBasic;
        variant part WheelDeLuxe;
    }

    // Simple car modelled directly as a part
    part simpleCar {
        // Simplistic dry mass aggregation (note: can be done smarter)
        attribute dryMass :>> ISQ::mass = body.mass
            + leftFrontWheel.mass + rightFrontWheel.mass
            + leftRearWheel.mass + rightRearWheel.mass;

        // Car body modelled directly as a part
        part body {**}

        // 4 wheels with separately named usage roles
        part leftFrontWheel : WheelChoice;
        part rightFrontWheel : WheelChoice;
        part leftRearWheel : WheelChoice;
        part rightRearWheel : WheelChoice;
    }
}
```

No Kernel | Idle — Saving completed — Mode: Command — Ln 3, Col 18 — SimpleCarExample.ipynb

Model authored in textual language

Example in Jupyter Lab Notebook (local installation and FireFox browser)

JupyterLab — localhost:8888/lab

File Edit View Run Kernel Tabs Settings Help

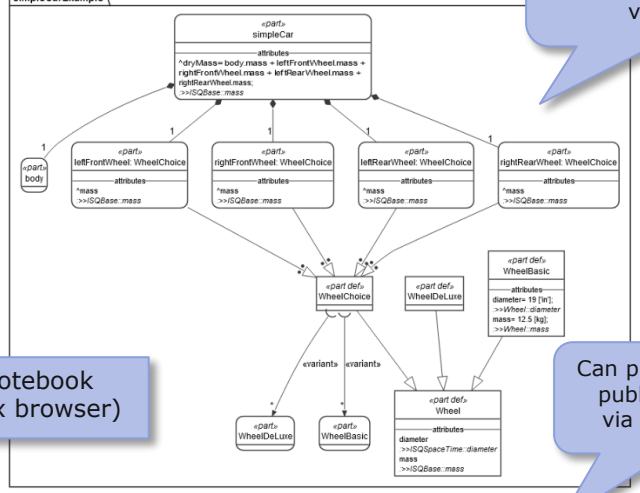
Launcher | SimpleCarExample.ipynb

```
        part leftRearWheel : WheelChoice;
        part rightRearWheel : WheelChoice;
    }
}
```

[33]: Package SimpleCarExample (ca8925e8-1437-4f8f-acbf-fc3c3647198b)

[34]:
```
%viz SimpleCarExample
```

[34]:



[*]:
```
%publish SimpleCarExample
```

API base path: http://sysml2.intercax.com:9000

No Kernel | Busy — Saving completed — Mode: Command — Ln 1, Col 1 — SimpleCarExample.ipynb

On-the-fly, auto-layout visualization via built-in PlantUML/SysMLv2 visualizer

Can publish model to public test server via SysMLv2 API

API & Services
Working prototype implementations

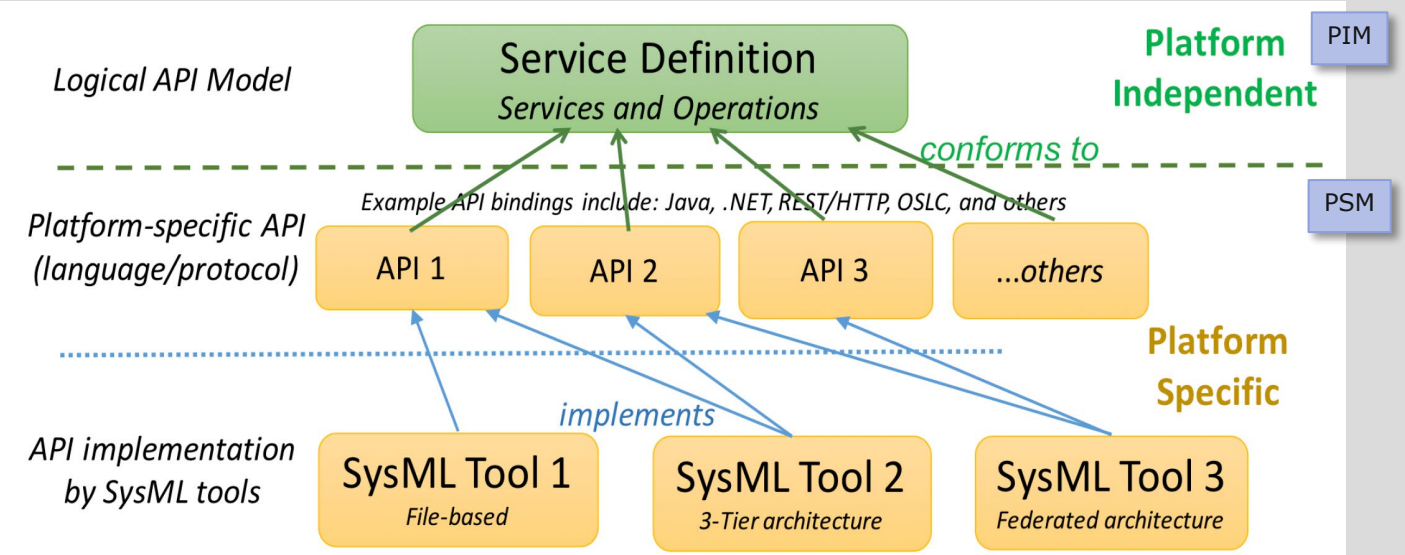


Figure 1. API and Services Architecture

- Current prototype API (PSM) implementations
  - HTTP/REST API
    - Conforms to OpenAPI spec, (https://www.openapis.org/)
    - Uses JSON or JSON-LD to serialize objects
    - Publicly accessible pilot server
    - Java and Python class libraries to facilitate client development
  - OSLC (Open Services for Lifecycle Collaboration)
    - Maps PIM concepts to OSLC resources / resource shapes
    - Uses JSON-LD to serialize objects
- Conformance Test Suite in Annex A of API & Services Spec

Formal Transformation from SysML v1 to v2

C Annex: SysML v1 to SysML v2 Transformation

C.1 General

C.1.1 Overview

This annex describes a transformation that specifies a semantic translation from SysML v1 [SysMLv1] to SysML v2 in a precise way. (In this annex, "SysML v1" refers to SysML v1.7, the last version of SysML prior to v2.0, and "SysML v2" refers to SysML as defined in this specification.)

C.2.3.4 Blocks

C.2.3.4.1 Overview

Table 75. List of all Overview Mapping Specfications

SysML v1 Concept	SysML v2 Concept	Mapping Class
AdjunctProperty	Feature	**AdjunctProperty_Mapping**
BindingConnector	BindingConnector	**BindingConnector_Mapping**
Block	PartDefinition	**Block_Mapping**
BoundReference	Feature	**BoundReference_Mapping**
ClassifierBehaviorProperty	Feature	**ClassifierBehaviorProperty_Mapping**
ConnectorProperty	Feature	**_ConnectorProperty_Mapping**
DirectedRelationshipPropertyPath		*** not specified yet ***
DistributedProperty		*** not specified yet ***
ElementPropertyPath		*** not specified yet ***
EndPathMultiplicity	Feature	**EndPathMultiplicity_Mapping**

- Annex C of the SysML Language Spec
- Supported and validated by implementation of automated transformation using Eclipse Epsilon (Work in Progress)

Summary

Acknowledgement:
Many thanks to all my SST co-members
for a great project (with very limited funding)

- SysML v2 is now well on its way to industrial use
  - Easier-to-use, more regular and much more powerful than SysML v1
  - Very sophisticated textual language in addition to enhanced graphical notation
  - Modern API with already two technology implementations: HTTP(S)/REST and OSLC
  - XMI or JSON serialization
  - Final submission incorporating OMG review and fine-tuning scheduled for Feb 2022
  - Most vendors estimate industrial tool releases in 2~3 years from final submission
    - Source: INCOSE International Symposium July 2021 – Panel on “The Journey from SysML v1.7 to v2.0”
  - Monthly public releases of the specifications, training material, open-source pilot tools and software libraries
    - https://github.com/Systems-Modeling/SysML-v2-Release
    - https://github.com/Systems-Modeling/SysML-v2-Pilot-Implementation
    - https://github.com/Systems-Modeling/SysML-v2-API-Services

References

SysML v2 Submission Team (SST) public repositories on GitHub	https://github.com/Systems-Modeling/
General information on MBSE across all industry sectors, INCOSE/OMG MBSE Wiki	http://www.omgwiki.org/MBSE/doku.php
General information on the OMG Systems Modeling Language (SysML)	http://www.omgsysml.org
Friedenthal, S., and R. Burkhart, “Evolving SysML and the System Modeling Environment to Support MBSE”, INCOSE INSIGHT (August 2015 Volume 18 Issue 2, Pg 39-42)	link
Ed Seidewitz, “SysML v2 and MBSE: The Next Ten Years”, MODELS 2018 Conference, Copenhagen, Denmark, Oct 2018	link
Hans Peter de Koning, “What to Expect from SysML v2”, 2020, presented at MBSE2020	link
Hans Peter de Koning, “Progress on SysML v2”, 13th ESA Workshop on Avionics, Data, Control and Software Systems (ADCSS2019), Nov 2019, ESA/ESTEC	link
Systems Modeling Language (SysML®) v2 Request For Proposal (RFP), OMG, December 2017	https://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-2
Systems Modeling Language (SysML®) v2 API and Services Request For Proposal (RFP), OMG, June 2018	https://www.omg.org/cgi-bin/doc.cgi?ad/2018-6-3
Systems Modeling Language v1.6, OMG, November 2019	https://www.omg.org/spec/SysML/1.6/

Why do I think SysML v2 is Important?

SAME SLIDE AS LAST YEAR!

- It is the only global standardization effort that has the scale and vendor support to tackle the problem of fully digitalized open systems engineering
- It is thoroughly based on formal semantics / first order logic
  - Initial mapping to RDF/OWL2 DL done, but needs completion
  - Enables future use of OWL2 DL automated reasoners on SysML models
- Has powerful API with JSON(-LD) (REST and OSLC implementations)
  - Much better than XMI file-based exchange for many industrial use cases
- It maps quite well to the (conceptual) data models in European Space
  - ECSS E-TM-10-23, E-TM-10-25, EGS-CC
  - Capella

Part 2 – Elaborations

The following slides provide further detail
to the highlights in the main presentation.
These slides could not be presented due to time constraints.

Key SysML v2 Concepts and Innovations (1 of 5)

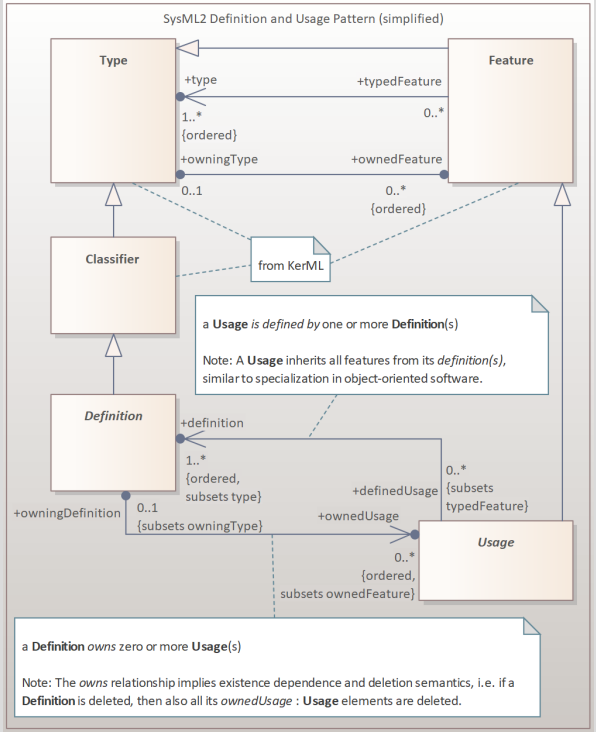
- Full textual language alongside graphical language
  - With bi-directional conversion either way
  - Integrated support for expressions and constraints
- SysML Specification contains many examples
  - Besides all formal definitions
  - Fully elaborated Vehicle example model in Annex B

Element	Graphical Notation	Textual Notation

Part
(can include any of the following compartments in the compartment stack: *actions, perform actions, allocations, attributes, connections, constraints, assert constraints, directed features, documentation, individuals, interfaces, items, metadata, namespaces, parts, ports, relationships, satisfy requirements, shapes, snapshots, states, exhibit states, timeslices, variants, variant elementusages*)
	«part»
	part1 : PartDef1
	
	«part»
	part1 : PartDef1
	compartment stack
	...
		`part part1 : PartDef1;`
		
		`part part1 : PartDef1 {`
		`  /* members */`
		`}`

Key SysML v2 Concepts and Innovations (2 of 5)

- "Usage-Focused Modeling Approach"
  - Enables direct modeling of nested hierarchical decomposition
    - More natural and quicker for most end-users – Similar to Capella and ViTech Genesys
    - Uses default Definitions (Types) in the background, without bothering the user
  - Enables redefinition directly at any deeply-nested Usage level
    - Override values, subset and/or narrow down multiplicities and/or types
    - Extend with additional structure, behaviour, attributes, ..., while keeping strong semantics
    - Resolves many cumbersome issues in SysML v1 (also in E-TM-10-25!)
    - Essential for modelling Individuals / Digital Twins
  - "Definition-Oriented Approach" still fully supported
    - For rigorously modular architectures, e.g., in product line libraries
    - Maintains compatibility with SysML v1's "Block-Definition-Oriented Approach"
- Usage is a 'first-class citizen'
  - Can declare a self-standing Usage (part, port, attribute, ...)
    - I.e. outside a particular (owning) Definition
    - Powerful pattern for re-usable libraries (e.g., Quantities and Units)



Key SysML v2 Concepts and Innovations (3 of 5)

- Model Libraries (normative and informative) at user-model level (M1)
  - Rather than profiles with stereotypes at language metamodel level (M2)
  - Root-level Classifiers and Features with Semantics in Libraries make tailoring & extension much simpler and cleaner
- Extensible support for Viewpoints & Views (Work in Progress)
  - A Viewpoint is a kind of requirement that frames concern(s) of stakeholder(s) regarding information from a model
  - A View addresses the concerns expressed in a Viewpoint
  - Aligned with latest update of ISO/IEC/IEEE 42010 “Software, systems and enterprise — Architecture description”
  - A View can specify conditions (what info to query from a model) and renderings
  - The Language Spec declares a minimum set of standardized views and renderings (subset compatible with SysML v1):
    Textual Notation, Element Table, Tree Diagram, Interconnection Diagram,
    Textual Rendering, Tabular Rendering, Graphical Rendering, ...
  - SysML v2 allows to combine different structure and behavior elements in single diagrams
- Standardized API & Services to access models / model repositories
  - Specification as Platform Independent Model (PIM) – i.e. independent from implementation technology
  - Two full Platform Specific Model (PSM) API pilot implementations: HTTP/REST and OSLC
  - Built-in (Git-like) life cycle support – with versioning, tagging, branching and merging
    - Orthogonal to KerML or SysML model – can also be used for non-KerML/SysML models

Key SysML v2 Concepts and Innovations (4 of 5)

- Powerful, robust model for name-space and package management via imports
  - Handles circular imports
  - Support for ‘smart packages’ using XPath-like import queries
- Possibility to declare and use metadata
  - Similar to stereotypes in UML and SysML v1 but now integrated in user language
- Specification of variability via variation points and variants
  - In support of PLE, product configurations, design alternatives, options, trade-offs, ...
  - Can add constraints to declare valid / invalid combinations
- Proper concept of modelling Individuals (distinguished from M0 instances)
  - E.g., to represent actual serial-numbered items, ‘digital twins’, analysis/simulation executions
- Specification of analysis/simulation/verification cases, calculations
  - Case, execution and results, including linking specification model with external solvers
  - Comes with execution semantics / legal execution traces (Work in Progress)

Key SysML v2 Concepts and Innovations (5 of 5)

- Modelling object lifetimes and spatial extent as Occurrences and Snapshots
  - 4D spatio-temporal extent model – and portions / time-slices thereof
  - 'Onto-behavior': rigorous modelling of time-dependent behaviour based on Allen's interval-based temporal logic: happens-before/during, succession, partial and total time ordering
- Support for synchronous & asynchronous messaging (Work in Progress)
  - Signals, required and provided interface ends (ports), events, messages, ...
  - Includes sequence diagram notation aligned with ITU MSC (Message Sequence Chart)
  - Validated with Service Oriented Architecture patterns
- Comprehensive set of extensible 'Domain Libraries' (largely done, some Work in Progress)
  - Mathematical, logical, programming functions integrated with textual expression language
  - Quantities, Units, Scales and Quantity Dimensions
    - Provides fully digitalized ISO/IEC 80000:2019 (ISQ & SI) as well as NIST SP-811 US Customary Units and CODATA constants
    - Supports scalar, vector and tensor quantities with automated unit/scale conversion
    - Supports coordinate systems and coordinate transformations
  - Time & Clocks, State-Space Representation, Basic Geometry (enveloping shapes, reference to CAD)

Example Textual Notation: Parts, Attributes, Quantities & Units, Redefinition, …

Expressions
Mass Rollup Example (2)

SST

```
import ScalarValues::*;
import MassRollup::*;

part def CarPart :> MassedThing {
   attribute serialNumber : String;
}
part car: CarPart :> compositeThing {
   attribute vin redefines serialNumber;
   part carParts : CarPart[*] redefines subcomponents;
   part engine :> simpleThing subsets carParts { … }
   part transmission :> simpleThing subsets carParts { … }
}

// Example usage
import SI::*;
part c :> car {
   redefines car::mass = 1000[kg];
   part redefines engine {
      redefines engine::mass = 100[kg];
   }
   part redefines transmission {
      redefines transmission::mass = 50[kg];
   }
}

// c.totalMass --> 1150.0[kg]
```

Units are identified on the *value, not* the type.

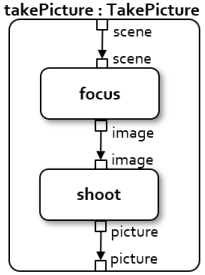
67 Last changed: 2021-03 (units notation) Release 2021-05

Source: Intro to the SysML v2 Language-Textual Notation on GitHub

Example Textual and Graphical Notation: Functional Behaviour Example (1 of 3)

Action Decomposition

SST




```
action def Focus(in scene : Scene, out image : Image);
action def Shoot(in image : Image, out picture : Picture);
action def TakePicture(in scene : Scene, out picture : Picture);

action takePicture : TakePicture {
  in item scene;
  out item picture;

  action focus : Focus {
    in item scene = takePicture::scene;
    out item image;
  }

  action shoot : Shoot {
    in item image flow from focus.image;
    out item picture = takePicture::picture;
  }
}
```

An action can also be directly decomposed into other actions.

takePicture is a usage, so dot notation could be used here, but it is unnecessary because picture is in an outer scope, not nested.

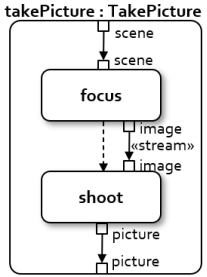
33 Last changed: 2021-05 (dot notation in flow) Release 2021-05

Source: Intro to the SysML v2 Language-Textual Notation on GitHub

Example Textual and Graphical Notation: Functional Behaviour Example (2 of 3)

Action Succession (1)

SST




```
action takePicture : TakePicture {
  in item scene;
  out item picture;

  action focus : Focus {
    in item scene = takePicture::scene;
    out item image;
  }

  succession focus then shoot;

  action shoot : Shoot {
    in item image stream from focus.image;
    out item picture = takePicture::picture;
  }
}
```

A *succession* asserts that the first action must complete before the second can begin.

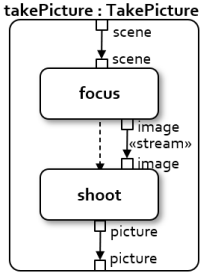
With the succession modeled explicitly, a stream item flow can be used here.

Source: Intro to the SysML v2 Language-Textual Notation on GitHub

Example Textual and Graphical Notation: Functional Behaviour Example (3 of 3)

Action Succession (2)

SST




```
action takePicture : TakePicture {
  in item scene;
  out item picture;

  action focus : Focus {
    in item scene = takePicture::scene;
    out item image;
  }

  then action shoot : Shoot {
    in item image stream from focus.image;
    out item picture = takePicture::picture;
  }
}
```

This is a shorthand for a succession between the lexically previous action and this action.

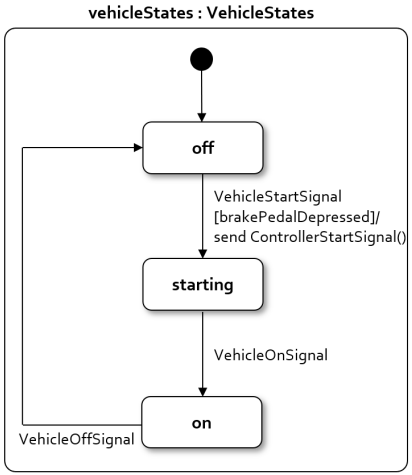
35 Last changed: 2021-05 (dot notation in flow) Release 2021-05

Source: Intro to the SysML v2 Language-Textual Notation on GitHub

Example Textual and Graphical Notation: State machine with transition guards

Transition Guards and Effect Actions

SST




```
action performSelfTest(vehicle : Vehicle);

state def VehicleStates(operatingVehicle : Vehicle);

state vehicleStates : VehicleStates (
  operatingVehicle : Vehicle,
  controller : VehicleController)

  entry; then off;

  state off;
  accept VehicleStartSignal
    if operatingVehicle.brakePedalDepressed
    do send ControllerStartSignal() to controller
    then starting;

  state starting;
  accept VehicleOnSignal
    then on;

  state on { … }
  accept VehicleOffSignal
    then off;
}
```

A *guard* is a condition that must be true for a transition to fire.

An *effect action* is performed when a transition fires, before entry to the target state.

50 Last changed: 2021-04 (dot notation) Release 2021-05

Source: Intro to the SysML v2 Language-Textual Notation on GitHub

Example Textual Notation: Requirement with properties and constraints

Requirement Definitions (1)

SST

A *requirement definition* is a special kind of constraint definition.

A textual statement of the requirement can be given as a documentation comment in the requirement definition body.

```
requirement def MassLimitationRequirement {
  doc /* The actual mass shall be less than or equal
       * to the required mass. */

  attribute massActual : MassValue;
  attribute massReqd : MassValue;

  require constraint { massActual <= massReqd }
}
```

Like a constraint definition, a requirement definition can be parameterized using features.

The requirement can be formalized by giving one or more component *required constraints*.

Requirement Definitions (2)

SST

```
part def Vehicle {
  attribute dryMass: MassValue;
  attribute fuelMass: MassV
  attribute fuelFullMass: M
  …
}

requirement def id '1' VehicleMassLimitationRequirement :> MassLimitationRequirement {
  doc /* The total mass of a vehicle shall be less than or equal to the required mass. */

  subject vehicle : Vehicle;

  attribute redefines massActual = vehicle.dryMass + vehicle.fuelMass;

  assume constraint { vehicle.fuelMass > 0[kg] }
}
```

A requirement definition may have a modeler specified *human id*, which is an alternate name for it.

A requirement definition is always about some *subject*, which may be implicit or specified explicitly.

A requirement definition may also specify one or more *assumptions*.

Features of the subject can be used in the requirement definition.

Source: Intro to the SysML v2 Language-Textual Notation on GitHub