«DigitalEngineering»

DEKonsult

SysML Version 2 – Final Stretch

Hans Peter de Koning (DEKonsult)

ESA MBSE2022 Workshop, 22-24 November 2022, Toulouse, France

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?

OMG Standards Development Organization

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)

- Systems Modeling Language
  - by Object Management Group (OMG)
  - A language supporting MBSE – Not an MBSE methodology
  - SysML v1 is a profile & extension of UML2
  - 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 completed and adopted – public release expected soon – will be the final version 1.x
  - Also used for system modelling / concepts in OMG UAF (Unified Architecture Framework)
    - UAF is for system-of-systems (SoS) modelling and enterprise architecture
    - UAF is the unification of DoDAF, MoDAF and NAF – successor to UPDM
    - UAF v2 will use SysML v2 for system modelling

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

2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024

SysML v2 RFP WG

Request for Proposal

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

SysML v2 Submission Team (SST)

- 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 of specifications, training material and pilot software
- Major stakeholders' review held in Feb 2021
- Revised submission delivered to OMG on 30 Aug 2021 for review
- Revised submission 2 delivered to OMG on 15 Sep 2022 for in-depth review
- Final submission to OMG 07 Nov 2022 of KerML and API & Services specs
- Final submission to OMG 20 Feb 2023 of SysML language spec
- Acceptance of 3 specs expected at March 2023 OMG technical meeting

Finalization Task Force

- Finalization of the specifications
- Monthly releases of pilot implementations and training material will continue
- Production tool developments & industrialization (both COTS and open-source)
- Official SysML v2.0 expected end 2023 / beginning 2024

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 ~215 members from 86 organizations
  - Large aerospace participation, 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
- U.S. Army DEVCOM Armaments Center
- BAE
- BigLever Software
- Boeing
- Budapest Univ of Tech and Economics (BME)
- CalTech CTME
- CEA
- Contact Software
- Defence Science and Technology Group
- DEKonsult
- Delligatti Associates
- Draper Lab
- Elparazim
- ESTACA
- Ford
- Fraunhofer FOKUS
- Galois
- General Motors
- George Mason University
- GfSE
- Georgia Tech/GTRI
- IBM
- Idaho National Laboratory
- IncQuery Labs
- Intercax
- Itemis
- Jet Propulsion Lab
- John Deere
- 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
- Naval Postgraduate School (NPS)
- 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
- SEI
- Siemens
- Sierra Nevada Corporation
- Simula
- Space Cooperative
- Sodius Willert
- System Strategy *
- Tata Consultancy Services
- TES
- 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

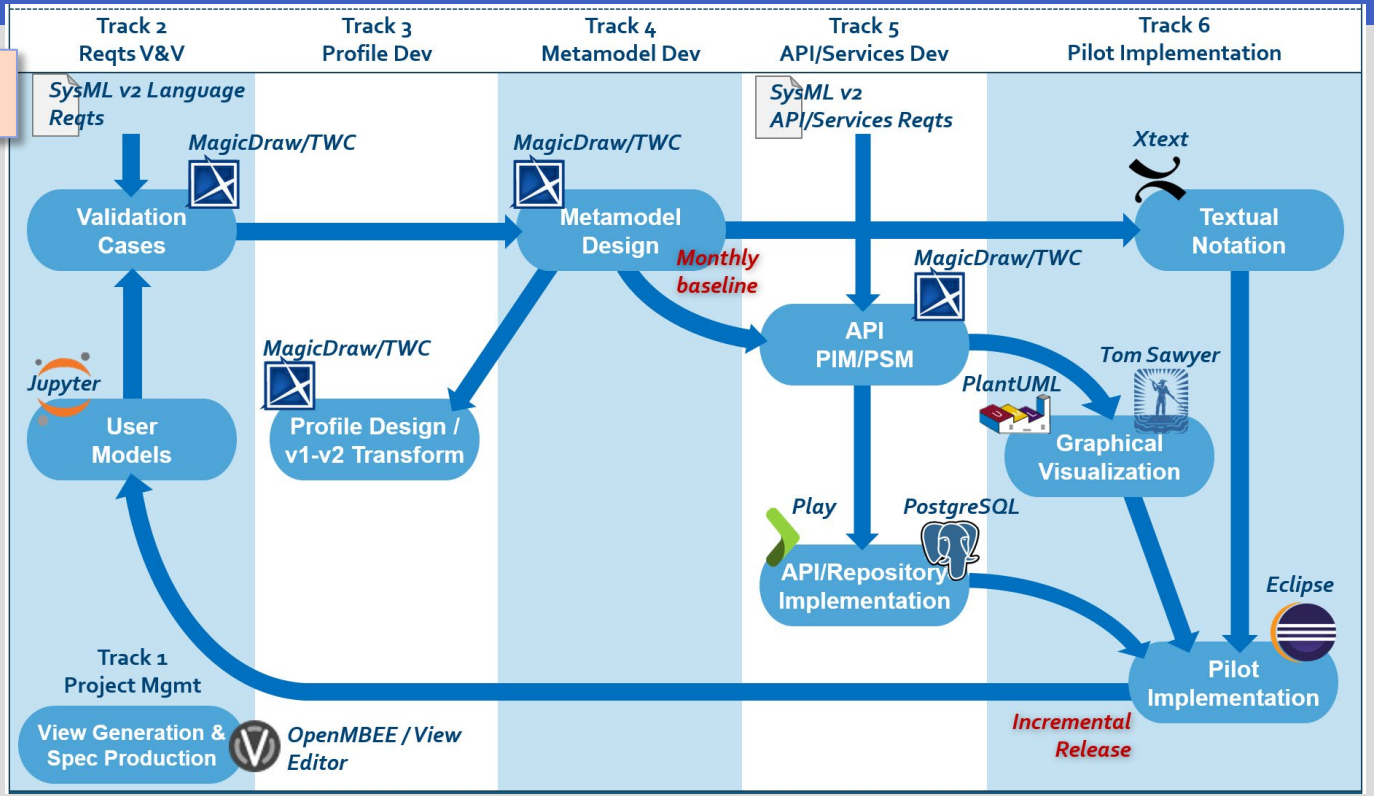
Academia/Research
End User

Tool/Training Vendors
Government Rep

INCOSE rep *

SST Agile / Incremental Development Workflow

Track 1 is project management



Revised Submission 3 = Release 2022-10
7 Nov 2022 – 3 specs

Date: November 2022

Kernel Modeling Language (KerML)

Version 1.0
Release 2022-10

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 Partners

Final submission KerML done 7 Nov 2022

Date: November 2022

OMG Systems Modeling Language ™ (SysML®)

Version 2.0
Release 2022-10

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 Partners

Final submission SysML v2 scheduled for 20 Feb 2023

Date: November 2022

Systems Modeling Application Programming Interface (API) and Services

Version 1.0
Release 2022-10

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

Final submission API & Services done 7 Nov 2022

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

New Layered Architecture

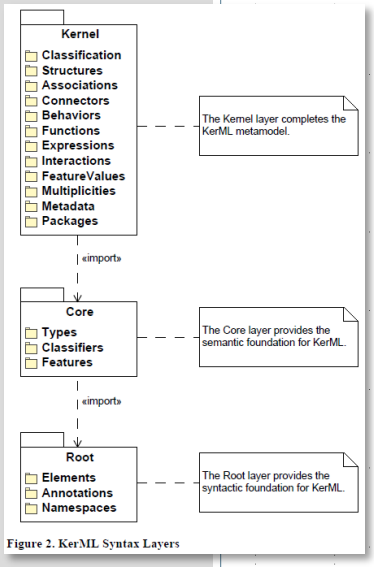
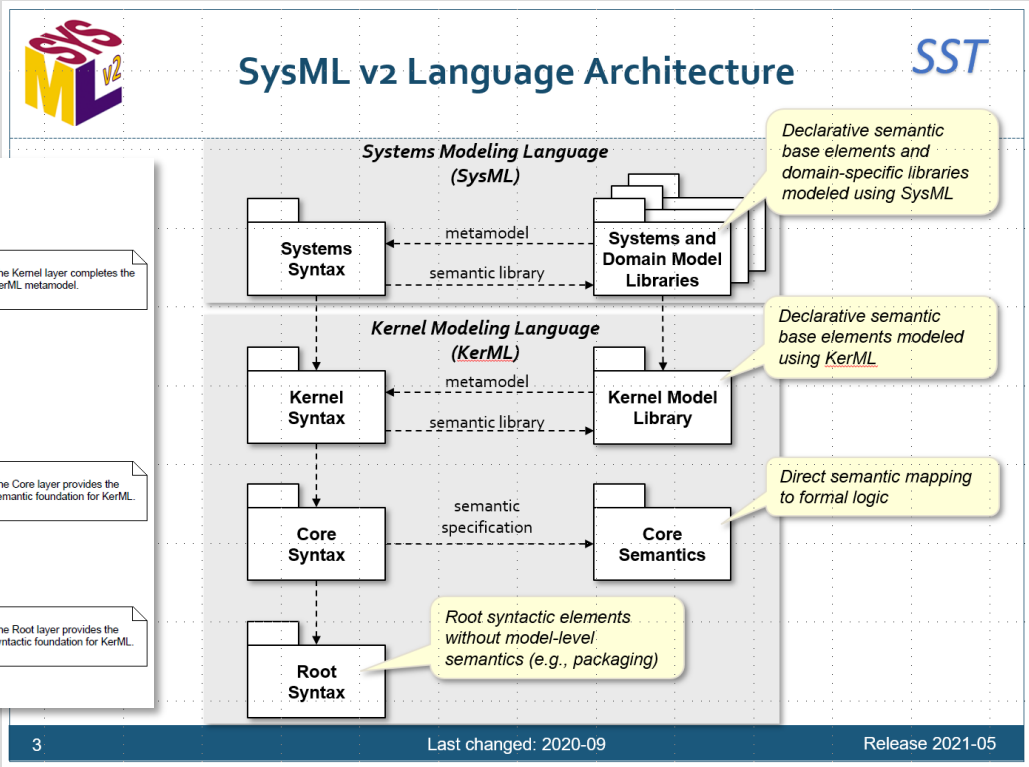


Figure 2. KerML Syntax Layers



- 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 decomposition, type specialisation are 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

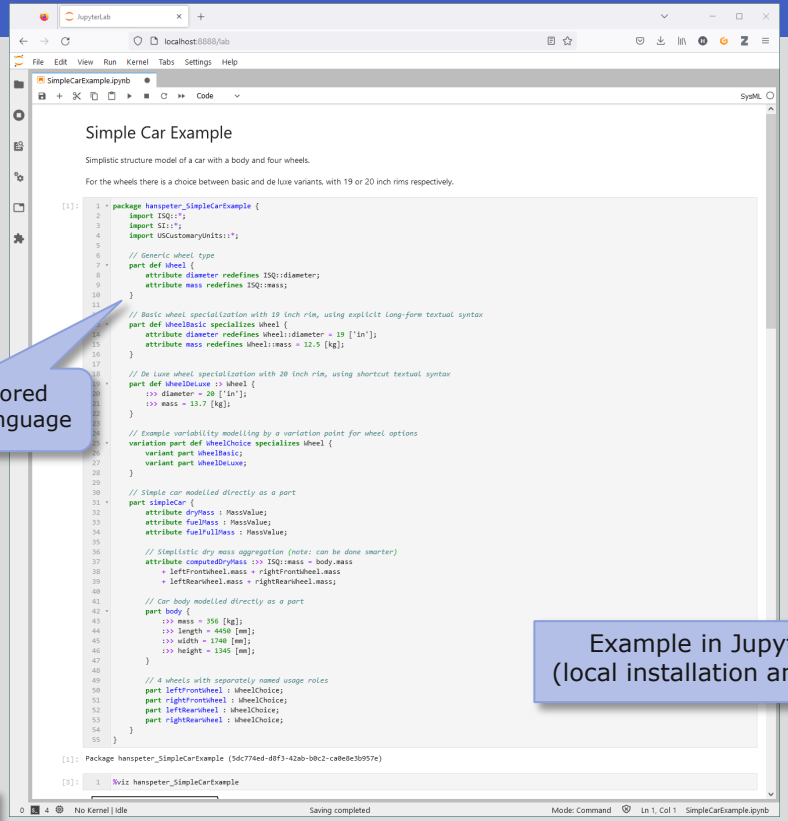
SysML v2 – Key Concepts and Innovations

- 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 natural, and much easier than in v1
  - SysML v2 still supports “Definition-Oriented” approach to achieve modularity when needed
- Very sophisticated (smart) package import and namespacing built into language
- 4D modelling of an object’s life and spatial extent as Occurrences & Snapshots
  - Included comprehensive, formal models of portions of life/extent, temporal logic, spatial topology
- Support for variation points and variants, at any level
  - Enables PLE, product configurations, design alternatives, options, trade-offs, ...
- Modelling of Individuals
  - E.g., for serial-numbered items, ‘digital twins’, analysis/simulation executions

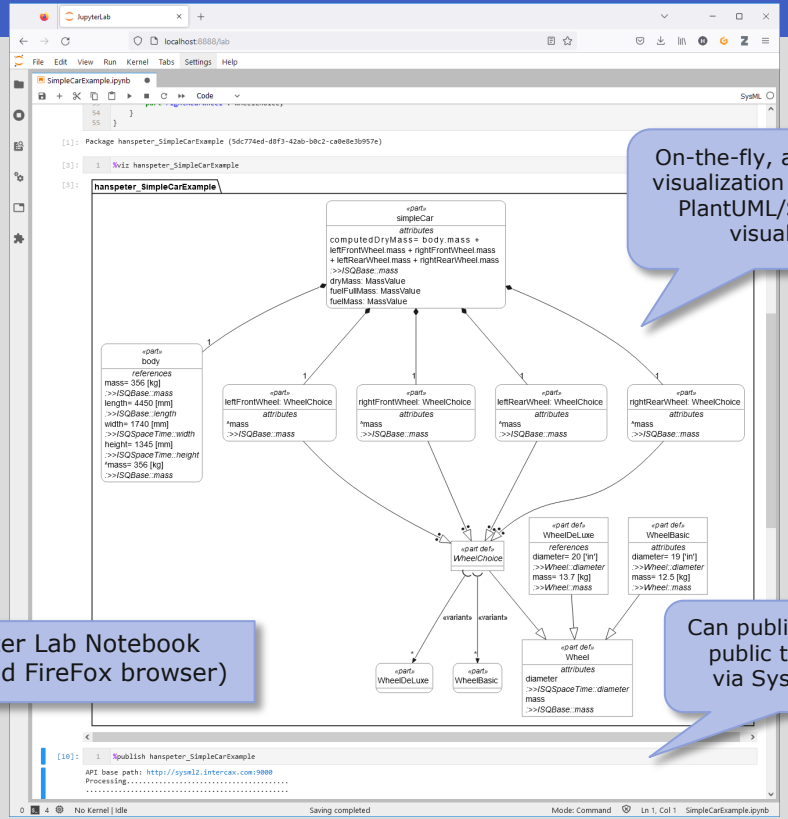
SysML v2 – Key Concepts and Innovations (cont'd)

- Integrated behaviour modelling: action control flow, state machines, sequences
  - Sync / async, serial / concurrent, signals, messages, events, aligned with ITU MSC
- Regularised specification of analysis or simulation cases, verification cases, use cases
  - Support for reusable calculations (calc's: similar to math / software functions)
- 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, modern API & Services for interoperability and fine-grained access
  - JSON and XMI object serialization
- Improved, flexible Viewpoints & Views aligned with ISO/IEC/IEEE 42010
  - ISO/IEC/IEEE 42010 "Software, systems and enterprise — Architecture description"

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



Model authored in textual language



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

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

Can publish model to public test server via SysMLv2 API

API & Services
Working pilot implementations

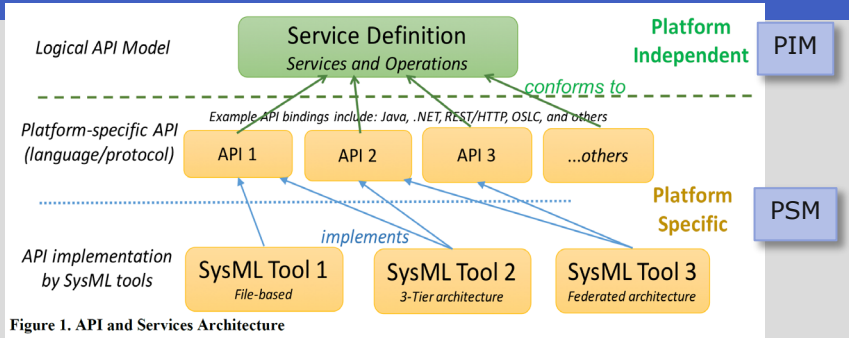
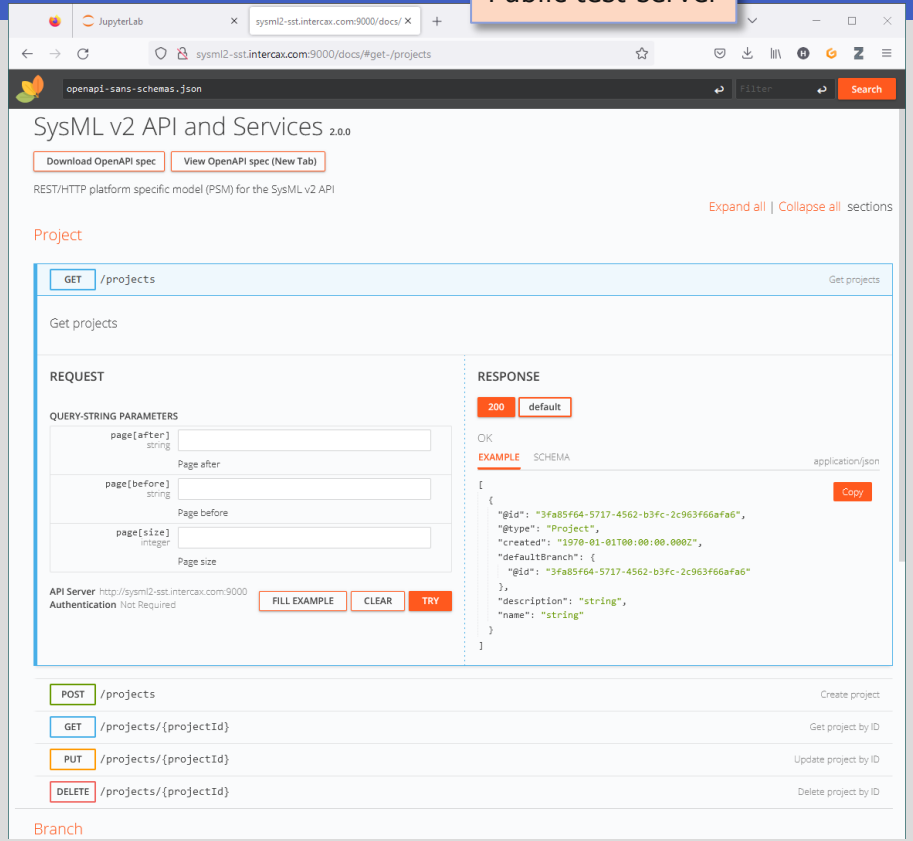


Figure 1. API and Services Architecture

- Two current pilot API (PSM) implementations
  - HTTP/REST API
    - Conforms to OpenAPI spec v3.1, (https://www.openapis.org/)
    - Contains full discoverable JSONSchema of SysML v2 metamodel
    - Uses JSON or JSON-LD to serialize objects
    - Provides paging and storable queries (to retrieve partial object graphs)
    - Publicly accessible pilot server – supports branching and tagging
    - 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

Public test server



Formal Transformation from SysML v1 to v2

Date: November 2022

OMG Systems Modeling Language™ (SysML®)
Annex C: SysML v1 to SysML v2 Transformation

Version 2.0
Release 2022-10

Submitted in response to Systems Modeling Language (SysML®) v2 RFP (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 Partners

Currently published as a separate document

- Annex C of the SysML Language Spec
- Fully explicit, machine readable mapping (or exclusion) of all SysML v1 concepts (v1.6 and v1.7) to v2
- Supported and validated by implementation of automated transformation using Eclipse Epsilon

C.2.4.8 Requirements

C.2.4.8.1 Overview

Table 13. List of all Overview Mapping Specfications

SysML v1 Concept	SysML v2 Concept	Mapping Class
AbstractRequirement		*** not specified yet ***
Copy		*** not specified yet ***
DeriveReqt		DeriveReqt_Mapping
Refine		Refine_Mapping
Requirement	RequirementDefinition	Requirement_Mapping
Satisfy	SatisfyRequirementUsage	Satisfy_Mapping
TestCase	VerificationCaseDefinition	TestCaseActivity_Mapping
Trace	Dependency	Trace_Mapping
Verify	RequirementVerificationMembership	Verify_Mapping

C.2.4.8.2 SysML v1 Requirements elements not mapped

Table 14. List of SysML v1 elements not mapped of this section

SysML v1 Concept	Rationale
Copy	The copy relationship is not covered by SysML v2.

Summary / Conclusions

Acknowledgement:
Many thanks to all my SST co-workers
for a great project

- SysML v2 is on the final stretch to become an international standard
  - Easier-to-use, more regular and much more powerful than SysML v1
  - Very sophisticated textual language in addition to enhanced graphical notation
  - Much improved and semantically precise language extension support
  - Modern API with already two technology implementations: HTTP(S)/REST and OSLC
  - Finalisation in progress
    - Adoption of submission expected at OMG meeting March 2023, then OMG FTF, and v2.0 release early 2024
  - Many vendors are working on implementation ... and have officially announced support
  - Monthly public releases of the specifications, training material, open-source pilot tools and software libraries will continue
    - 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/
SysML v2 Release group on Google Groups – Discussions on releases and pilot software – Apply for Membership	https://groups.google.com/g/sysml-v2-release
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, "SysML Version 2 Approaching Industrial Use", MBSE2021 Workshop, 29 Sep 2021, Virtual Event	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

Why do I think SysML v2 is Important?

- It is the only open international standard that has the scale and vendor support to tackle the problem of fully digitalised systems engineering across industry sectors and organisations
- It has a powerful, modern, open API
  - Validated with two technology implementations (HTTP(S)/REST/OpenAPI and OSLC)
  - Much better than XMI file-based exchange and tool-specific APIs for many industrial use cases
- It is thoroughly based on formal semantics / first order logic
  - Initial mapping to RDF/OWL2 DL done, further work underway
  - Enables future use of OWL2 DL automated reasoners on SysML models
- It maps quite well on the (conceptual) data models in European Space
  - OSMoSE, ECSS E-TM-10-23, E-TM-10-25, EGS-CC, Arcadia/Capella, ITU SDL & MSC
    ... with the big advantage of being a major cross-industry international standard

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

Table 16. Parts - Representative Notation

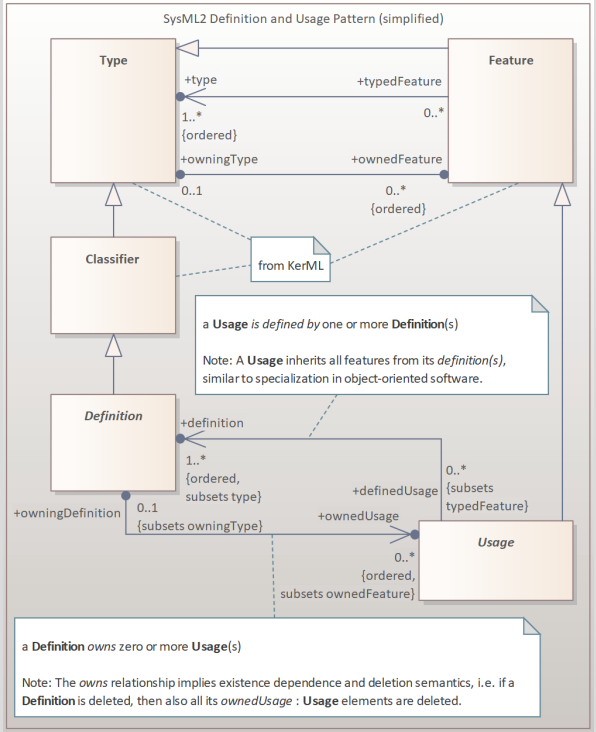
Element	Graphical Notation	Textual Notation

Part Definition	«part def»
	PartDef1
	
	«part def»
	PartDef1
	compartment stack
	...
		`part def PartDef1;`
		
		`part def PartDef1 {`
		`  /* members */`
		`}`

Part	«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, …

Car Mass Rollup Example (2)

SST

```
import ScalarValues::*;
import MassRollup2::*;

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


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

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

The default of totalMass to simpleMass is inherited.

The default for totalMass is overridden as specified in compositeThing.

The totalMass default is *not* overridden for the nested carParts.

When computing the totalMass of c, c.engine and c.transmission, the relevant defaults are used.

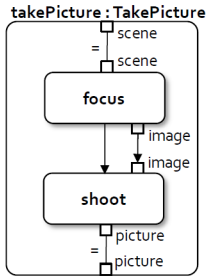
73 Last changed: 2021-08 (new) Release 2022-10

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;
  }

  flow focus.image to shoot.image;

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

An action usage can also be directly decomposed into other actions.

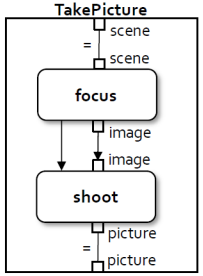
takePicture is a usage, so dot notation could be used here, but it is unnecessary because picture is in an outer scope, not nested.

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 def Focus{ in scene : Scene; out image : Image; }
action def Shoot{ in image : Image; out picture : Picture; }

action def TakePicture {
  in item scene : Scene;
  out item picture : Picture;

  bind focus.scene = scene;

  action focus : Focus { in scene, out image; }

  flow focus.image to shoot.image;

  first focus then shoot;

  action shoot : Shoot { in image, out picture; }

  bind shoot.picture = picture;
}
```

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

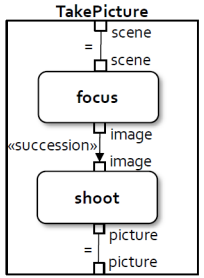
31 Last changed: 2022-07 (parameter declaration) Release 2022-10

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 def Focus{ in scene : Scene; out image : Image; }
action def Shoot{ in image : Image; out picture : Picture; }

action def TakePicture {
  in item scene : Scene;
  out item picture : Picture;

  bind focus.scene = scene;

  action focus : Focus { in scene; out ima
  succession flow focus.image to shoot.image;

  action shoot : Shoot { in image; out picture; }

  bind shoot.picture = picture;
}
```

A *succession flow* requires the first action to finish producing its output before the second can begin consuming it.

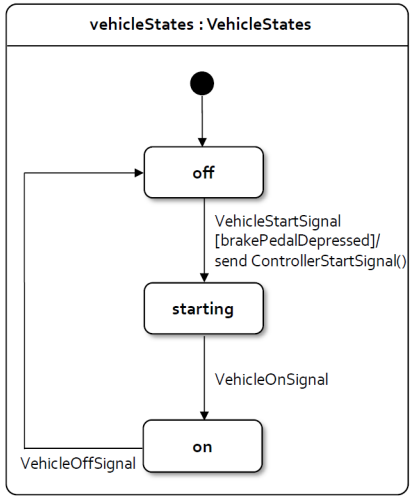
32 Last changed: 2022-07 (parameter declaration) Release 2022-10

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{in vehicle : Vehicle;}

state def VehicleStates {
  in operatingVehicle : Vehicle;
  in controller : VehicleController; }

state vehicleStates : VehicleStates {
  in operatingVehicle : Vehicle;
  in controller : VehicleControll

  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.

51 Last changed: 2022-07 (parameter declaration) Release 2022-10

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 <'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.

ⓘ Actually, any identifiable element can have a human id, not just requirements.

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*.

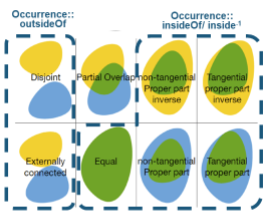
Features of the subject can be used in the requirement definition.

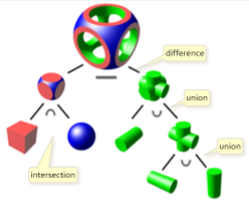
Source: Intro to the SysML v2 Language-Textual Notation on GitHub

Support for Basic Geometry, Spatial Extent, Coordinate Frames

Logical Space Modeling Sources

SST





- Region Connection Calculus (RCC)
  - Analogous to Allen's logical relations for time

See https://en.wikipedia.org/wiki/Region_connection_calculus
RCC diagram from John G. Stell, Spatial Connections An Informal Introduction, 2015

- Constructive Solid Geometry (CSG)
  - Union, intersection, and subtraction of shapes
  - Not assembly, shapes overlap
  - Enables automatic derivation of resulting geometry and topology.

See https://en.wikipedia.org/wiki/Constructive_solid_geometry

14 2022-06-21

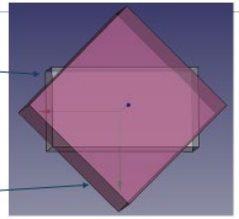
dcopter – MainBody

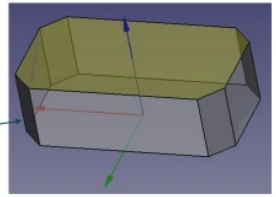
SST

```
ular box */
}
attribute :>> coordinateFrame {
    :>> transformation : TranslationRotationSequence {
        :>> elements = (Translation( (0, shape.width/2, 0)[source]));
    }
}
}

/* cuttingBox is a construction shape: the enveloping rectangular box */
part cuttingCornersBox : SpatialItem {
    item :>> shape : Box {
        :>> length = 105 [mm];
        :>> width = 105 [mm];
        :>> height = 60 [mm];
    }
    attribute :>> coordinateFrame {
        :>> transformation : TranslationRotationSequence {
            :>> elements = (Translation( (0, -shape.length/sqrt(2), -10)[source]),
                            Rotation((0, 0, 1)[source], 45["°"]));
        }
    }
}

/* Main body shape is the CSG intersection of rawBody and cuttingCornersBox */
attribute :> intersectionsOf[1] {
    item :>> elements = (rawBody, cuttingCornersBox);
}
// Current syntax is not end-user friendly
// It will be possible to specify following simple CSG expression:
// item :>> shape = rawBody & cuttingCornersBox;
}
```





10 2022-06-21

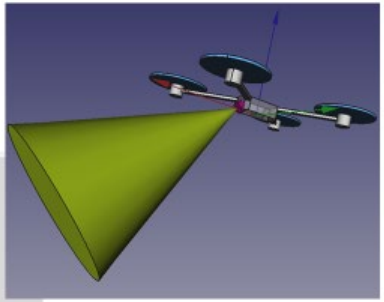
SimpleQuadcopter

SST

Quadcopter Example of an Assembly of Parts

Contains examples of:

- Specification of primitive 3D shapes: box, cylinder, cone
- Enveloping geometry of a simple quadcopter concept
- Use of CSG union, difference and intersection
- Parameterized construction of shapes
- Coordinate frame transformations, using translation and rotation sequences



```
1   package hpdekoning_SimpleQuadcopter {
2       import ISQ::*;
3       import SI::*;
4       import SpatialItems::*;
5       import ShapeItems::*;
6       import RealFunctions::sqrt;
7       import TrigFunctions::pi;
8       import TrigFunctions::tan;
9       import MeasurementReferences::CoordinateFrame;
10      import MeasurementReferences::TranslationRotationSequence;
11      import MeasurementReferences::Translation;
12      import MeasurementReferences::Rotation;
13
14      part def Strut :> CompoundSpatialItem {...}
51
52      part def PropellerMotorAssy :> CompoundSpatialItem {...}
83
84      part def Camera :> CompoundSpatialItem {...}
113
114     part quadCopter : CompoundSpatialItem {...}
247 }
```

8 2022-06-21