Moving Forward with SysML v2

July 17, 2024

Sanford Friedenthal
safriedenthal@gmail.com

Copyright © 2019-2024 by Sanford Friedenthal

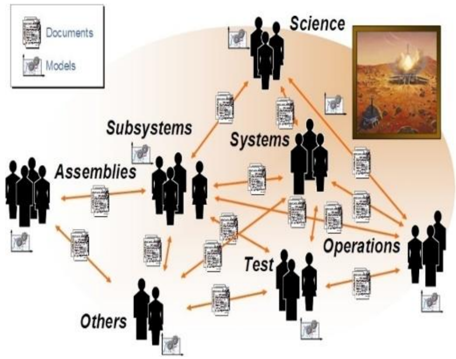
Agenda

- MBSE Background
- SysML v2 Status
- SysML v2 Overview & Comparison with SysML v1
- SysML v1 to SysML v2 Transition
- Summary

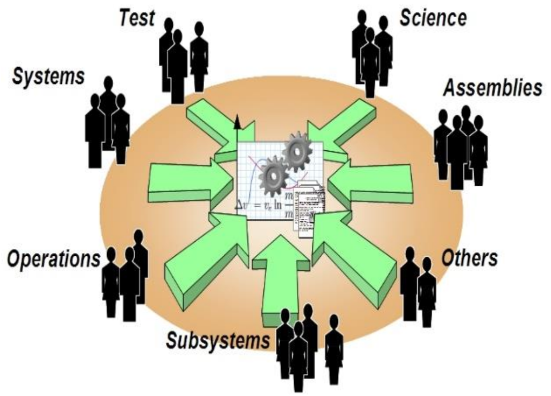
MBSE Background

Model-Based Systems Engineering (MBSE)

- A systems engineering approach where information about the system is captured in a system model
  - The model is the source of the data and managed throughout the lifecycle
- Contrasts with a document-based approach where the information is captured in a variety of documents, informal diagrams, and spreadsheets
- Provides a more complete, consistent, and traceable system design



From: System specification and design data related through documents

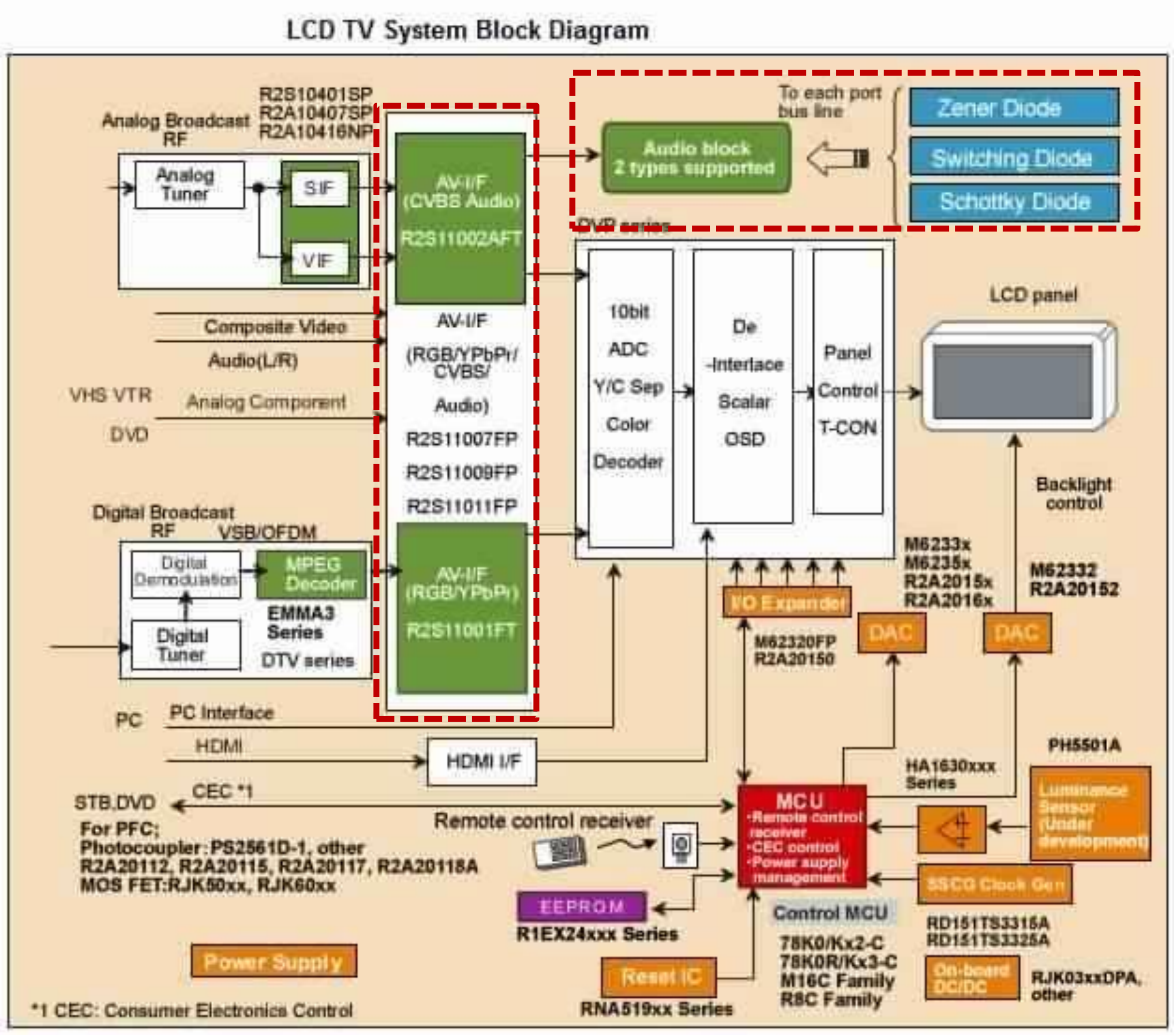


To: Shared system model with multiple views, and connected to discipline models

Source: Jet Propulsion Laboratory

Traditional System Block Diagram

- System architecture captured using informal diagramming notation
- Good domain content but imprecise description of:
  - Component hierarchy
  - Interfaces
  - Functions vs components
  - Succession vs connection
- Disconnected from other system views
- Lack of traceability



SysML v2 Status

Systems Modeling Language™ (SysML®)

Supports the specification, analysis, design, and verification and validation of complex systems that may include hardware, software, information, processes, personnel, and facilities

- SysML has evolved to address user and vendor needs
  - v1.0 adopted in 2006; v1.7 adopted 2022
- SysML v1 has facilitated awareness and adoption of MBSE
- Much has been learned from using SysML v1 for MBSE
- SysML v2 is the next generation systems modeling language intended to address some of the limitations of SysML v1

SysML v2 Status

- SysML v2 was developed by the SysML v2 Submission Team (SST) in response to the SysML v2 RFP issued by the OMG in December, 2017
- SysML v2 beta specifications (i.e., KerML, SysML v2, Systems Modeling API & Services) were approved by the OMG and are in the finalization phase
  - Finalization task force responds to issues raised by vendors as they develop their implementations
- Submit specifications for final adoption in 2024

SysML v2 Examples
Open-Source Pilot Implementation

- Examples of the SysML v2 textual syntax were created using the open-source reference implementation that was developed as part of the SysML v2 submission development effort
- The graphical views of the SysML v2 model were created using a drawing tool (draw.io) or the prototype visualization tool integrated with the reference implementation, based on an open-source application called Plant UML
- The quality and conformance of the graphical visualization is limited but will be substantially improved when commercial tools become available

SysML v2 Overview & Comparison with SysML v1

8 June, 2024

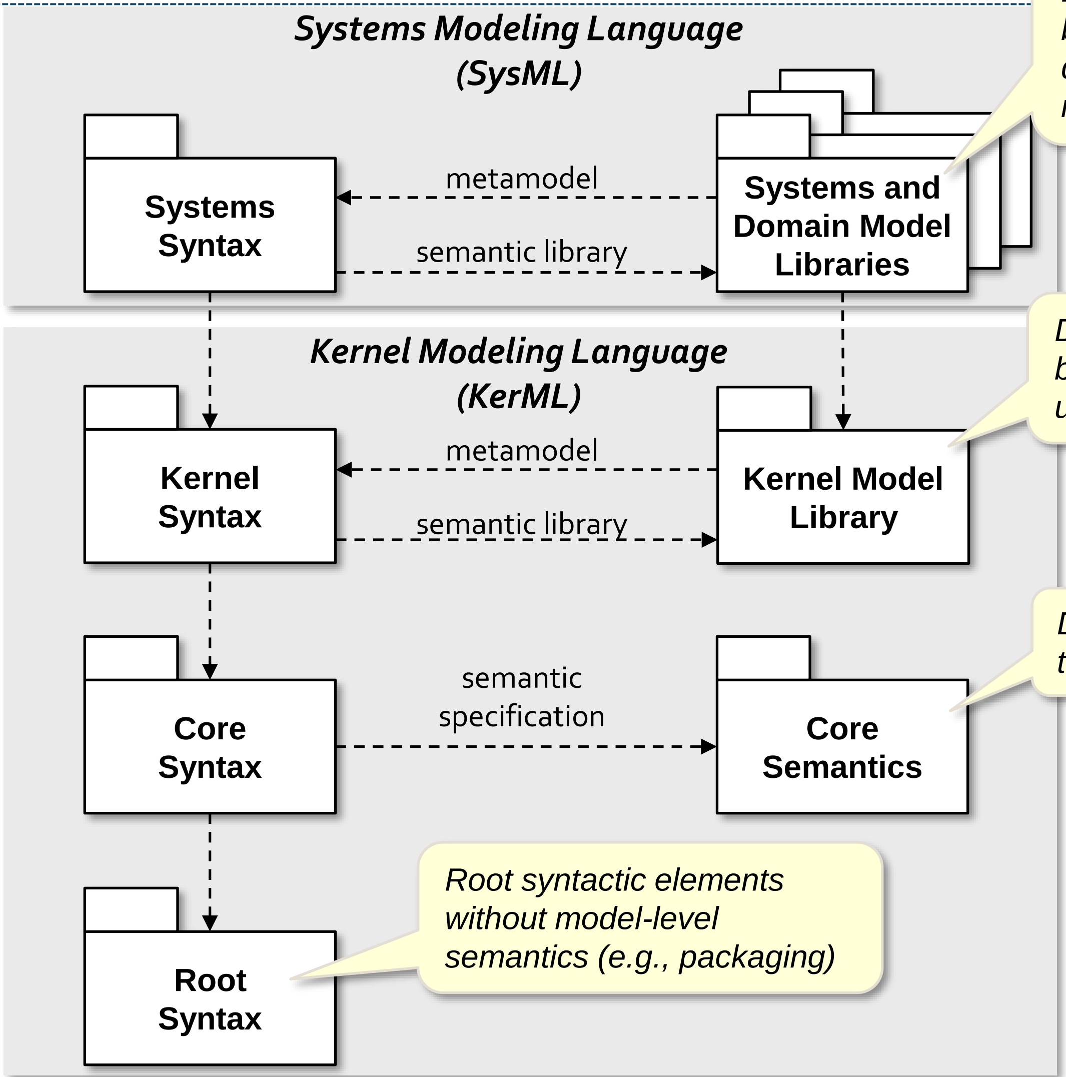
SysML v2 Objectives

- **Increase adoption and effectiveness of MBSE with SysML by enhancing…**
  - Precision and expressiveness of the language
  - Consistency and integration among language concepts
  - Interoperability with other engineering models and tools
  - Usability by model developers and consumers
  - Extensibility to support domain specific applications
  - Migration path for SysML v1 users and implementors

Key Elements of SysML v2

- New Metamodel that is not constrained by UML
  - Preserves most of UML modeling capabilities with a focus on systems modeling
  - Grounded in formal semantics
- Robust visualizations based on flexible view & viewpoint specification
  - Graphical, Tabular, Textual
- Standardized API to access the model

SysML v2 Language Architecture



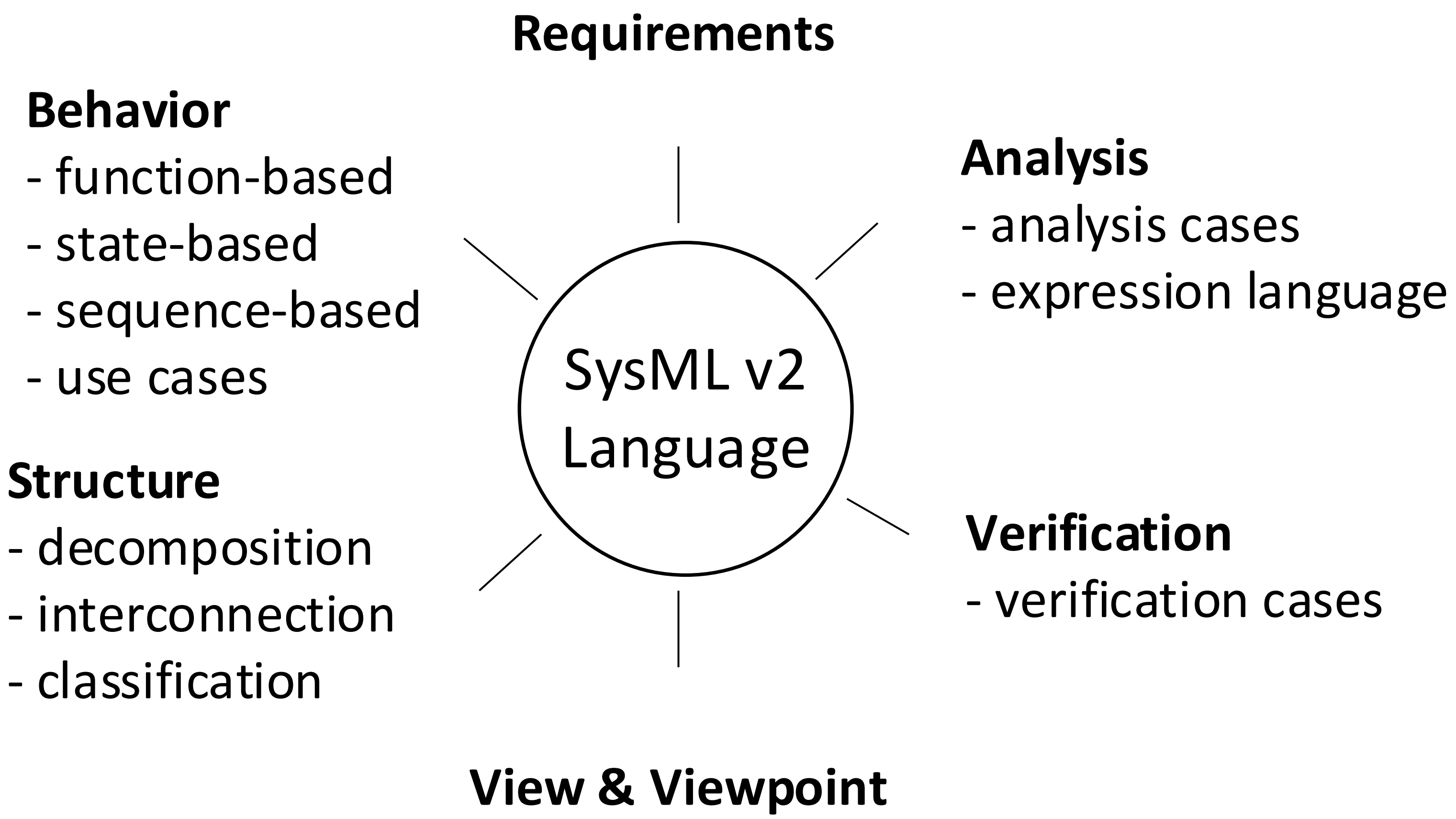
Declarative semantic base elements and domain-specific libraries modeled using SysML

Declarative semantic base elements modeled using KerML

Direct semantic mapping to formal logic

Root syntactic elements without model-level semantics (e.g., packaging)

SysML v2 Language Capabilities



Requirements

Behavior
- function-based
- state-based
- sequence-based
- use cases

Analysis
- analysis cases
- expression language

SysML v2 Language

Structure
- decomposition
- interconnection
- classification

Verification
- verification cases

View & Viewpoint

SysML v2 Reuse Pattern

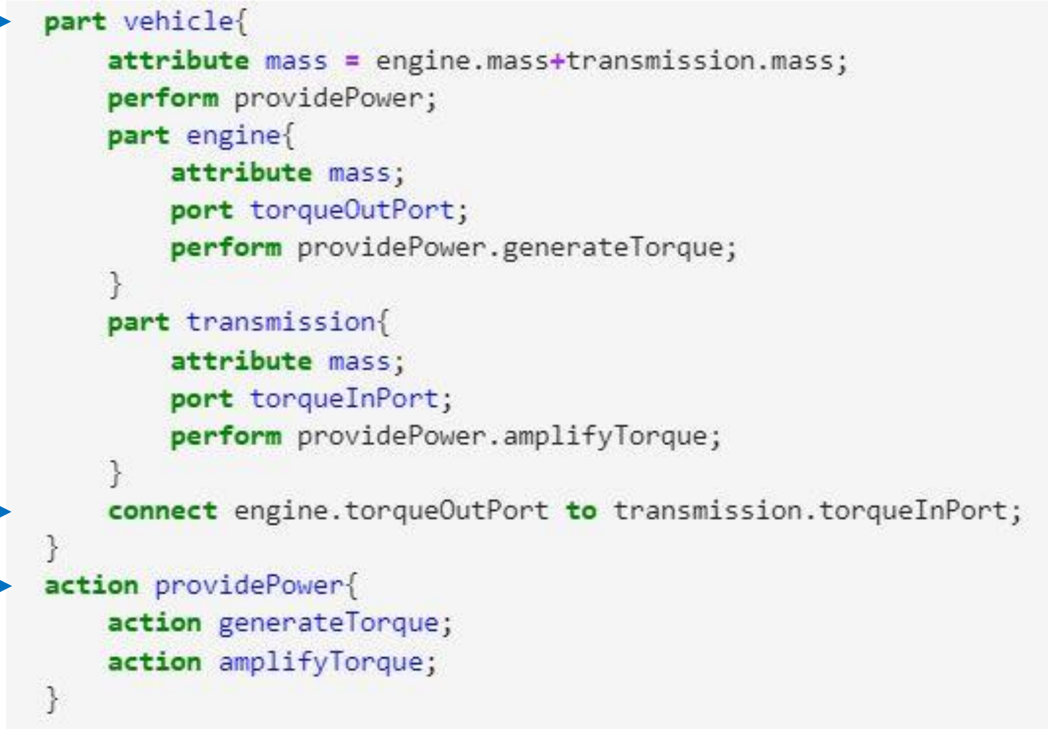
- Definition and usage
  - A definition element defines an element such as a part, action, or requirement
  - A usage element is a usage of a definition element in a particular context
  - Pattern is applied consistently throughout the language

SysML v2 to v1 Terminology Mapping (partial)

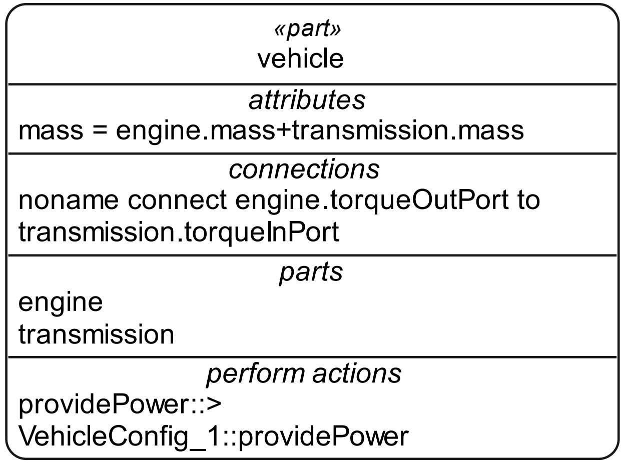
SysML v2	SysML v1
part / part def	part property / block
attribute / attribute def	value property / value type
port / port def	proxy port / interface block
action / action def	action / activity
state / state def	state / state machine
constraint / constraint def	constraint property / constraint block
requirement / requirement def	requirement
connection / connection def	connector / association block
view / view def	view

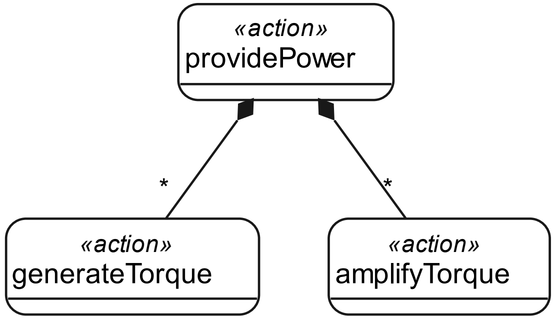
SysML v2 applies a consistent pattern of definition and usage

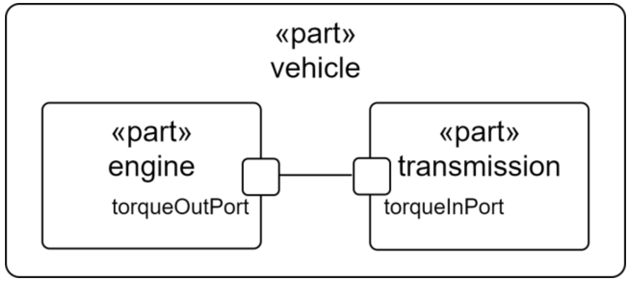
Simple Vehicle Model
SysML v2 Textual and Graphical Syntax



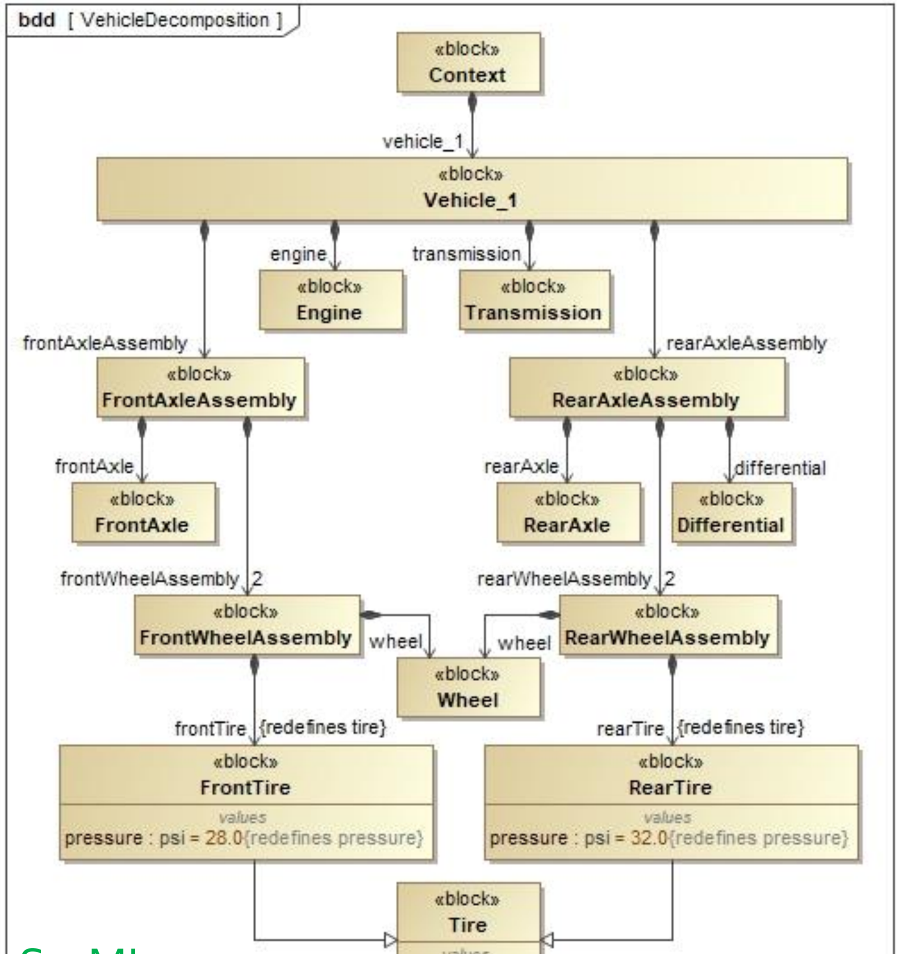
```
part vehicle{
    attribute mass = engine.mass+transmission.mass;
    perform providePower;
    part engine{
        attribute mass;
        port torqueOutPort;
        perform providePower.generateTorque;
    }
    part transmission{
        attribute mass;
        port torqueInPort;
        perform providePower.amplifyTorque;
    }
    connect engine.torqueOutPort to transmission.torqueInPort;
}
action providePower{
    action generateTorque;
    action amplifyTorque;
}
```


«part»
vehicle
attributes
mass = engine.mass+transmission.mass
connections
noname connect engine.torqueOutPort to transmission.torqueInPort
parts
engine
transmission
perform actions
providePower::> VehicleConfig_1::providePower

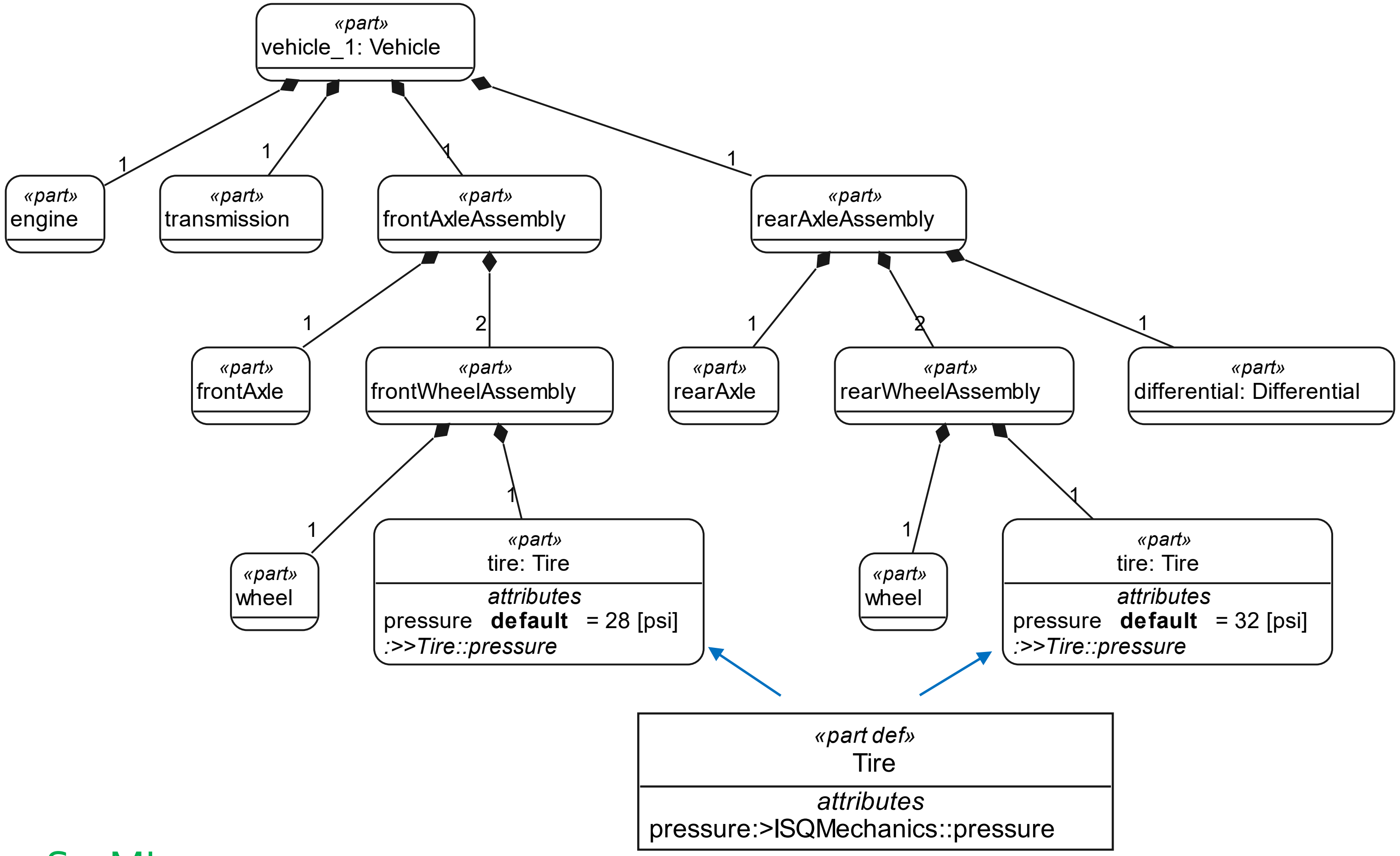




SysML v1 and v2
Vehicle Block vs Part Decomposition



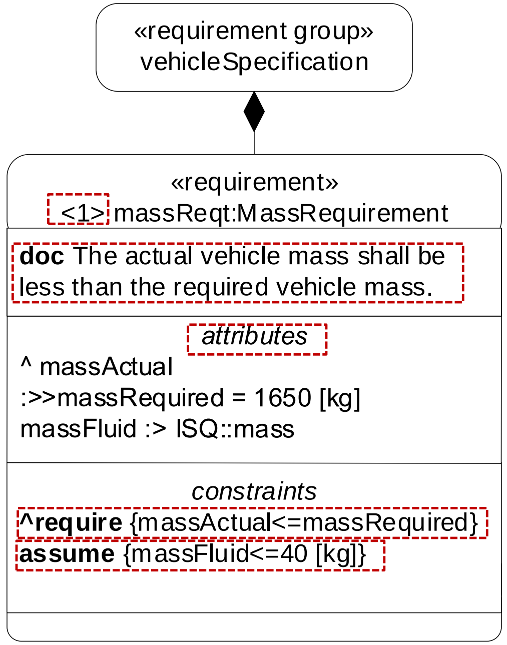
SysML v1
Block Decomposition



SysML v2
Part Decomposition

SysML v2 Requirement

- Builds on SysML v1 concept of a property-based requirement
- A constraint definition that a valid design solution must satisfy that can include:
  - Identifier
  - Shall statement
  - Constraint expression that can be evaluated to true or false
  - Attributes of the constraint expressions
  - Assumed constraint expression must be true for the requirement to be applicable



«requirement group»
vehicleSpecification

«requirement»
<1> massReqt:MassRequirement

doc The actual vehicle mass shall be less than the required vehicle mass.

attributes

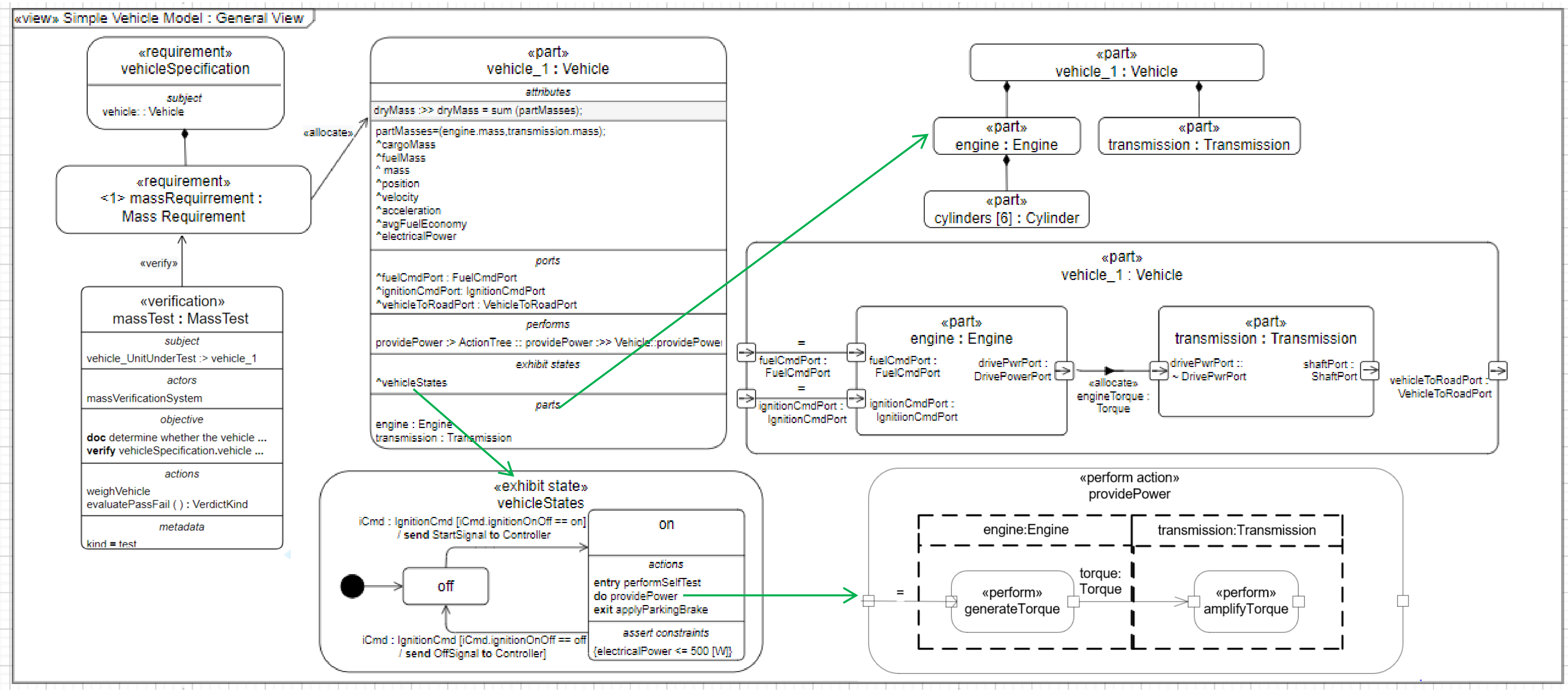
^ massActual
:>>massRequired = 1650 [kg]
massFluid :> ISQ::mass

constraints

^require {massActual<=massRequired}
assume {massFluid<=40 [kg]}

A SysML v2 Requirement Can be Evaluated by a Solver as Pass or Fail

Simple Vehicle Model



«view» Simple Vehicle Model : General View

«requirement»
vehicleSpecification
subject
vehicle: : Vehicle

«allocate»

«requirement»
<1> massRequirrement :
Mass Requirement

«verify»

«verification»
massTest : MassTest
subject
vehicle_UnitUnderTest :> vehicle_1
actors
massVerificationSystem
objective
doc determine whether the vehicle ...
verify vehicleSpecification.vehicle ...
actions
weighVehicle
evaluatePassFail ( ) : VerdictKind
metadata
kind **=** test

«part»
vehicle_1 : Vehicle
attributes
dryMass :>> dryMass = sum (partMasses);
partMasses=(engine.mass,transmission.mass);
^cargoMass
^fuelMass
^ mass
^position
^velocity
^acceleration
^avgFuelEconomy
^electricalPower
ports
^fuelCmdPort : FuelCmdPort
^ignitionCmdPort: IgnitionCmdPort
^vehicleToRoadPort : VehicleToRoadPort
performs
providePower :> ActionTree :: providePower :>> Vehicle::providePowe
exhibit states
^vehicleStates
parts
engine : Engine
transmission : Transmission

«part»
vehicle_1 : Vehicle

«part»
engine : Engine

«part»
transmission : Transmission

«part»
cylinders [6] : Cylinder

«part»
vehicle_1 : Vehicle
=
fuelCmdPort :
FuelCmdPort
=
ignitionCmdPort :
IgnitionCmdPort

«part»
engine : Engine
fuelCmdPort :
FuelCmdPort
drivePwrPort :
DrivePowerPort
ignitionCmdPort :
IgnitiionCmdPort

«allocate»
engineTorque :
Torque

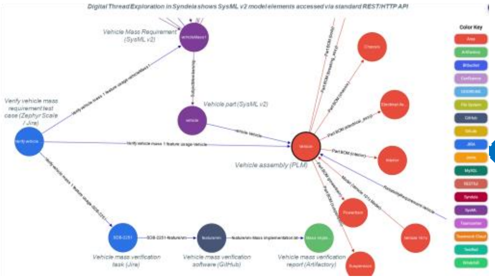
«part»
transmission : Transmission
drivePwrPort ::
~ DrivePwrPort
shaftPort :
ShaftPort

vehicleToRoadPort :
VehicleToRoadPort

«exhibit state»
vehicleStates
iCmd : IgnitionCmd [iCmd.ignitionOnOff == on]
/ **send** StartSignal **to** Controller
off
on
actions
entry performSelfTest
do providePower
exit applyParkingBrake
assert constraints
{electricalPower <= 500 [W]}
iCmd : IgnitionCmd [iCmd.ignitionOnOff == off
/ **send** OffSignal **to** Controller]

«perform action»
providePower
engine:Engine
transmission:Transmission
=
«perform»
generateTorque
torque:
Torque
«perform»
amplifyTorque

Connecting SysML v2 through the standard API

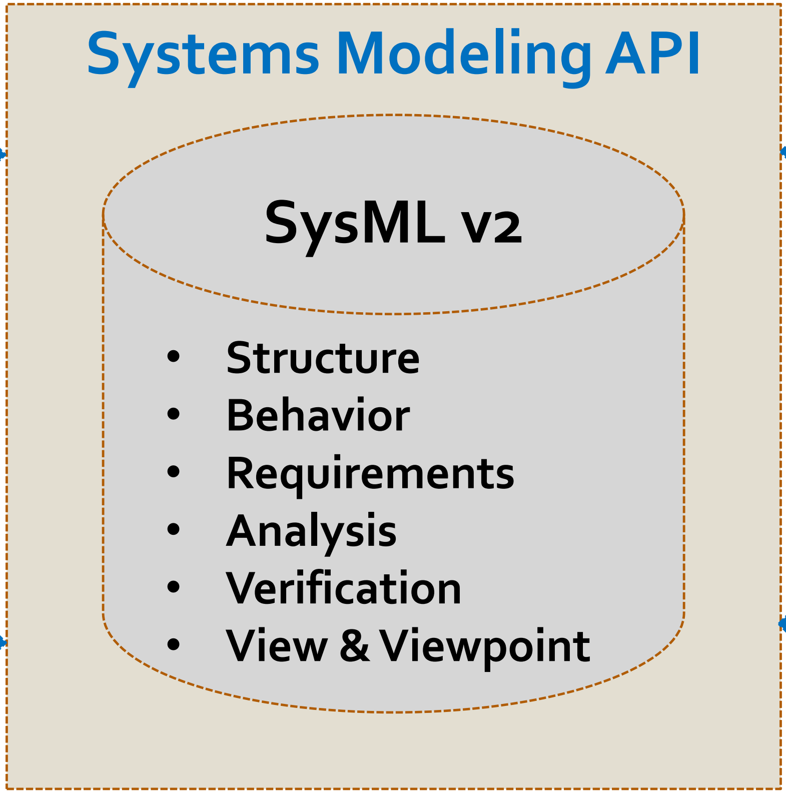


CM of the Digital Thread

Source: Syndeia with SysML v2

Graph Visualization

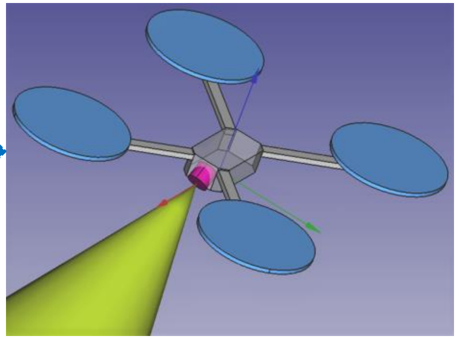
Source: Tom Sawyer with SysML v2



Systems Modeling API

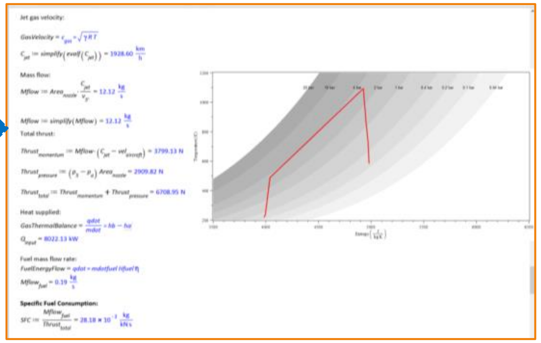
SysML v2

- Structure
- Behavior
- Requirements
- Analysis
- Verification
- View & Viewpoint



CAD/CAD Viewer

Source: FreeCAD with SysML v2



Analysis Solver

Source: Maple with SysML v2

Comparing SysML v2 with SysML v1

- **Simpler to learn and use**
  - Systems engineering concepts designed into metamodel versus added-on
  - Consistent application of definition and usage pattern
  - More consistent terminology
  - Ability to decompose parts, actions,
  - More flexible model organization with package filters
- **More precise**
  - Textual syntax and expression language
  - Formal semantic grounding
  - Requirements as constraints
- **More expressive**
  - Variant modeling
  - Analysis case
  - Trade-off analysis
  - Individuals, snapshots, time slices
  - More robust quantitative properties (e.g., vectors, ..)
  - Simple geometry
  - Query/filter expressions
  - Metadata
- **More extensible**
  - Simpler language extension capability
    - Based on model libraries
- **More interoperable**
  - Standardized API

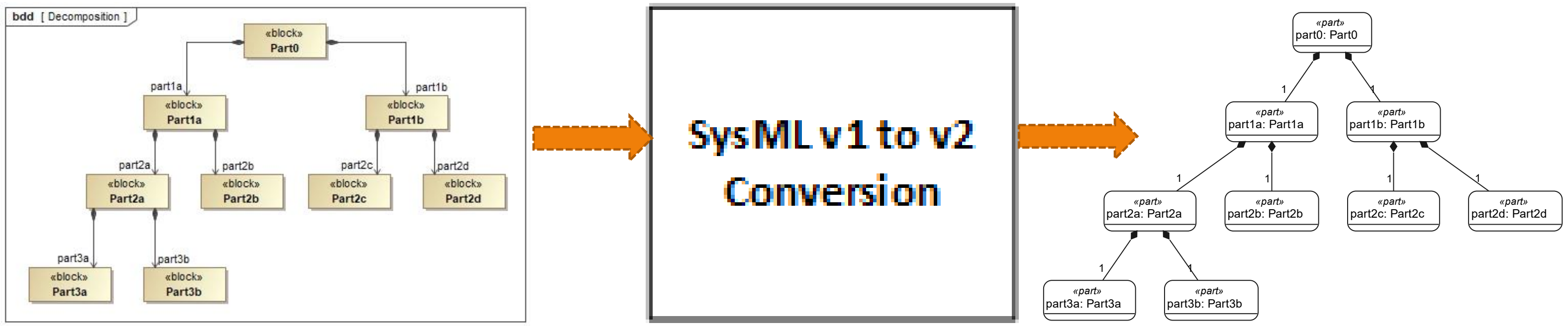
SysML v1 to SysML v2 Transition

SysML v1 to SysML v2 Model Conversion

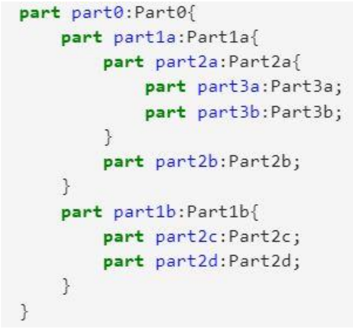
- Perform conversion incrementally
  - Select portion of model to convert
  - Pre-process as required
  - Perform transformation
  - Validate results
  - Reorganize and refactor

SysML v1 Model

SysML v2 Model
Graphical & Textual Notation



SysML v1 to v2 Conversion

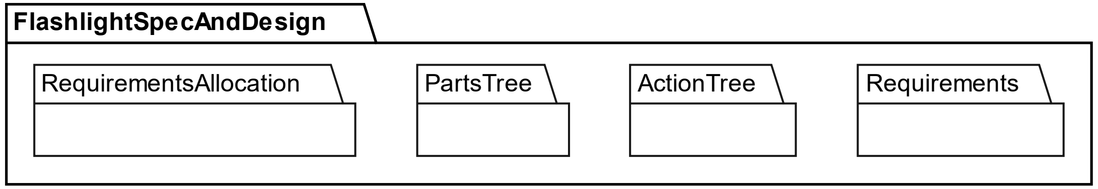


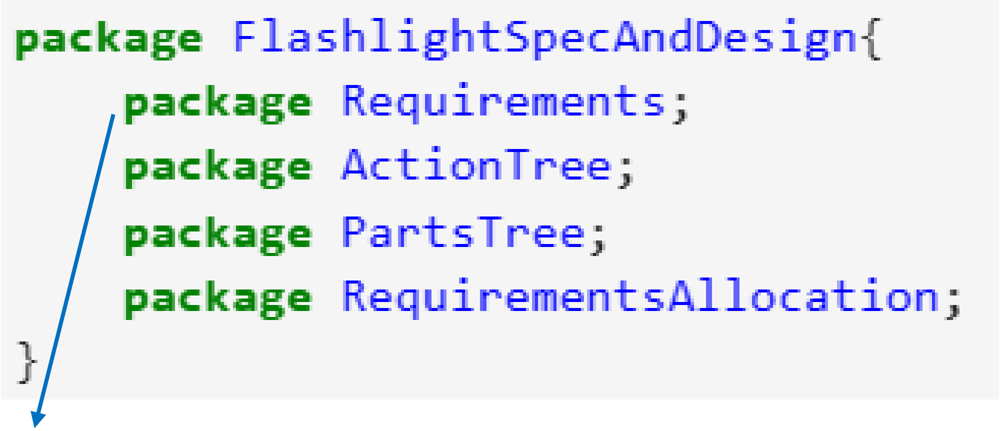
```
part part0:Part0{
    part part1a:Part1a{
        part part2a:Part2a{
            part part3a:Part3a;
            part part3b:Part3b;
        }
        part part2b:Part2b;
    }
    part part1b:Part1b{
        part part2c:Part2c;
        part part2d:Part2d;
    }
}
```

Starter Model

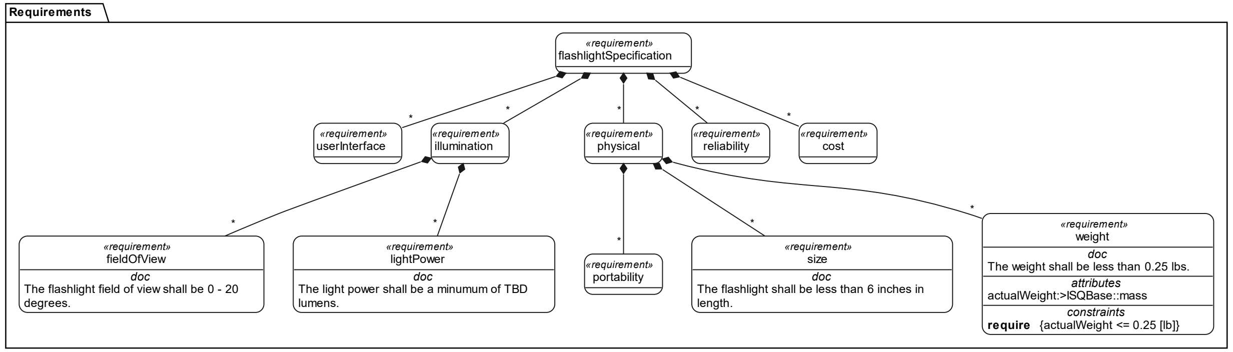
https://www.omgwiki.org/MBSE/doku.php?id=mbse:sysml_v2_transition

- Simple flashlight model
  - Packages / Model Organization
  - Requirements Tree
  - Action Tree
  - Parts Tree
  - State-based behavior
- jupyter modeling environment
  - Uses SysML v2 textual notation
  - Selected examples of SysML v2 graphical notation using Plant UML





```
package FlashlightSpecAndDesign{
    package Requirements;
    package ActionTree;
    package PartsTree;
    package RequirementsAllocation;
}
```



The flashlight field of view shall be 0 - 20 degrees.

The light power shall be a minumum of TBD lumens.

The flashlight shall be less than 6 inches in length.

The weight shall be less than 0.25 lbs.

actualWeight:>ISQBase::mass

require {actualWeight <= 0.25 [lb]}

Summary

Summary

- SysML v2 is addressing SysML v1limitations to improve MBSE adoption and effectiveness
  - New metamodel with both graphical and textual syntax and standardized API to access the model
  - More precise, expressive, usable, interoperable, and extensible than SysML v1
  - Consistent definition and usage pattern enables reuse, usability, and automation
- Progress/Plans
  - OMG approved SysML v2 beta specifications with specification to be submitted in 2024 for final adoption
  - Continue to evolve SysML v2 modeling practices, specifications, and domain specific extensions
- Organizations and practitioners should initiate their SysML v2 transition planning
  - Refer to SysML v2 Transition Wiki at https://www.omgwiki.org/MBSE/doku.php?id=mbse:sysml_v2_transition

SST Public Repositories
Current Release: 2024-05

- Monthly release repository
  - https://github.com/Systems-Modeling/SysML-v2-Release
- Release content
  - Specification documents (for KerML, SysML and API)
  - Training material for SysML textual notation
  - Training material for SysML graphical notation
  - Example models (in textual notation)
  - Pilot implementation
    - Installer for Jupyter tooling
    - Installation site for Eclipse plug-in
  - Web access to prototype repository via SysML v2 API
  - Web access to Tom Sawyer visualization tooling
- Open-source repositories
  - https://github.com/Systems-Modeling
- Google group for comments and questions
  - https://groups.google.com/g/SysML-v2-Release
    (to request membership, provide name, affiliation and interest)

Thank You!!