



SST

SysML v2 Overview & Demo

January 30, 2023

Sanford Friedenthal
SST Co-Lead
safriedenthal@gmail.com



Agenda

SST

- SysML v2 Overview
- Language Demo
- API Demo
- Summary



SysML v2 Objectives

SST

- **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

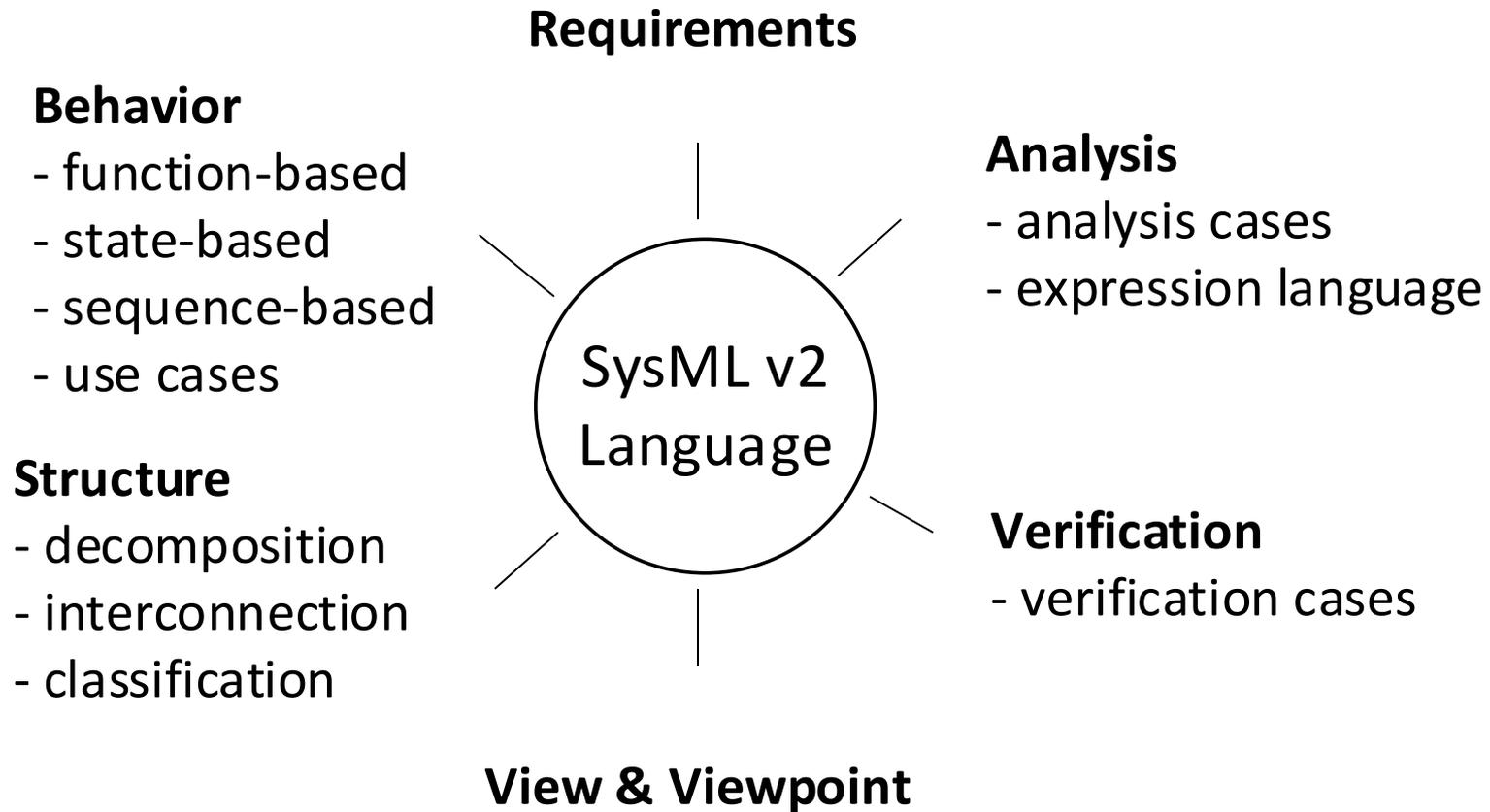
SST

- 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 Capabilities

SST





Vehicle Part Definition

Replaces SysML v1 Block

SST

- The vehicle part definition is characterized by different kinds of features including
 - Attributes
 - Ports
 - Actions
 - States
 - ...

<p>«part def» Vehicle</p>
<p><i>attributes</i></p> <p>mass :> ISQ::mass = dryMass + cargoMass + fuelMass dryMass :> ISQ::mass cargoMass :> ISQ::mass fuelMass :> ISQ::mass position :> ISQ::length velocity :> ISQ::speed acceleration :> ISQ::acceleration avgFuelEconomy :> distancePerVolume electricalPower :> ISQ::power</p>
<p><i>ports</i></p> <p>fuelCmdPort : FuelCmdPort ignitionCmdPort : IgnitionCmdPort vehicleToRoadPort : VehicleToRoadPort</p>
<p><i>perform actions</i></p> <p>providePower</p>
<p><i>exhibit states</i></p> <p>vehicleStates</p>



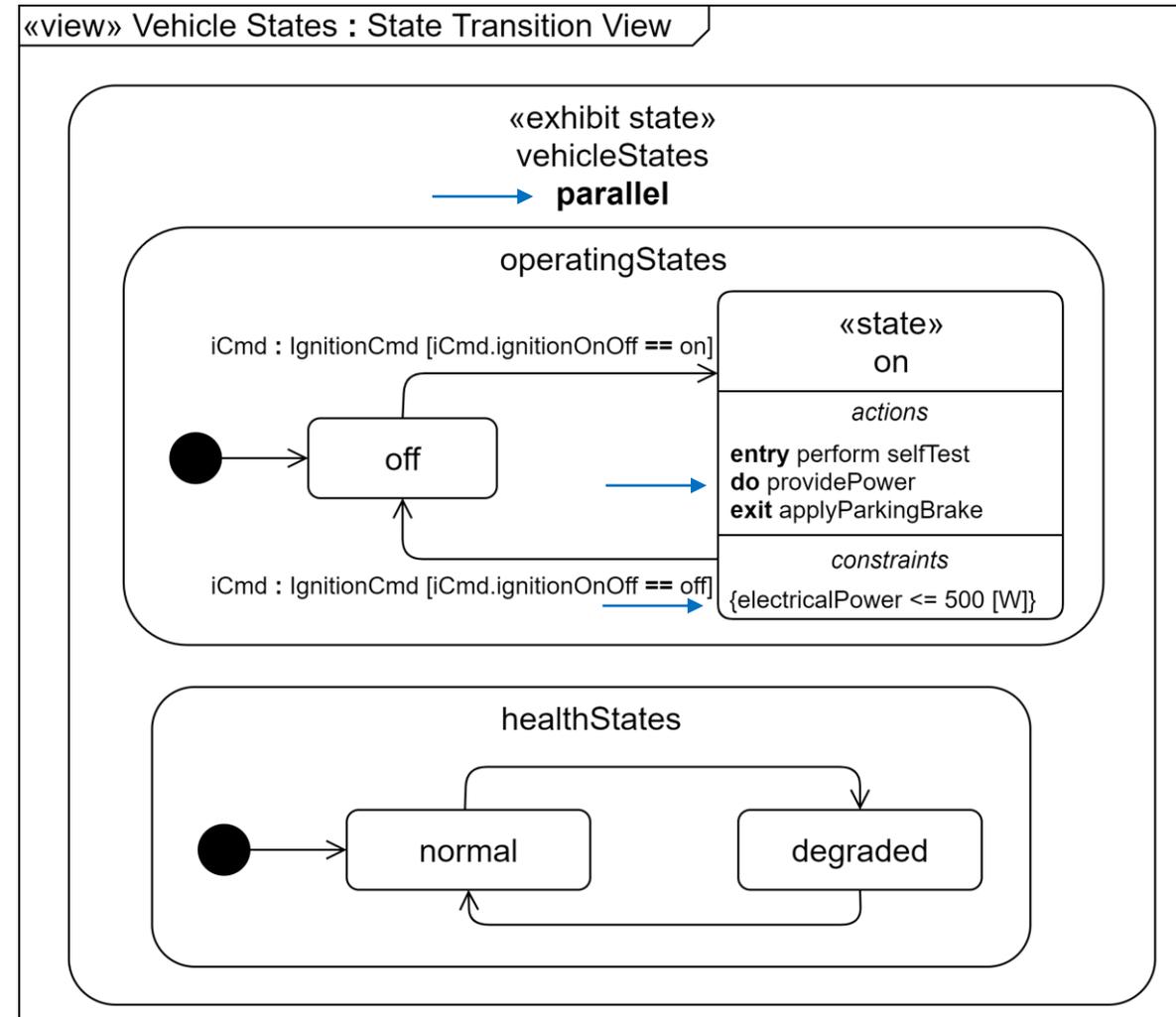
Vehicle Part Definition Textual Syntax

SST

- The textual syntax reflects the same model as the graphical syntax

```
part def Vehicle{
  attribute mass :> ISQ::mass = dryMass + cargoMass + fuelMass;
  attribute dryMass:>ISQ::mass;
  attribute cargoMass:>ISQ::mass;
  attribute fuelMass:>ISQ::mass;
  attribute position:>ISQ::length;
  attribute velocity:>ISQ::speed;
  attribute acceleration:>ISQ::acceleration;
  attribute avgFuelEconomy:>distancePerVolume;
  attribute electricalPower:> ISQ::power;
  port fuelCmdPort:FuelCmdPort;
  port ignitionCmdPort:IgnitionCmdPort;
  port vehicleToRoadPort:VehicleToRoadPort;
  perform action providePower;
  exhibit state vehicleStates parallel {↔}
}
```

- States are hierarchical and can include:
 - parallel states (e.g., concurrent states) and mutually exclusive states
 - entry, exit, and do actions
 - constraints





Vehicle States Textual Syntax

SST

```
exhibit state vehicleStates parallel {
  state operatingStates {
    entry action initial;
    state off;
    state on {
      entry action performSelfTest;
      do providePower;
      exit action applyParkingBrake;
      constraint {electricalPower<=500[W]}
    }
    transition initial then off;
    transition off_To_on
      first off
      accept ignitionCmd:IgnitionCmd via ignitionCmdPort
        if ignitionCmd.ignitionOnOff==IgnitionOnOff::on
      then on;
    transition on_To_off
      first on
      accept ignitionCmd:IgnitionCmd via ignitionCmdPort
        if ignitionCmd.ignitionOnOff==IgnitionOnOff::off
      then off;
  }
  state healthStates {
    entry action initial;
    state normal;
    state degraded;
  }
}
```

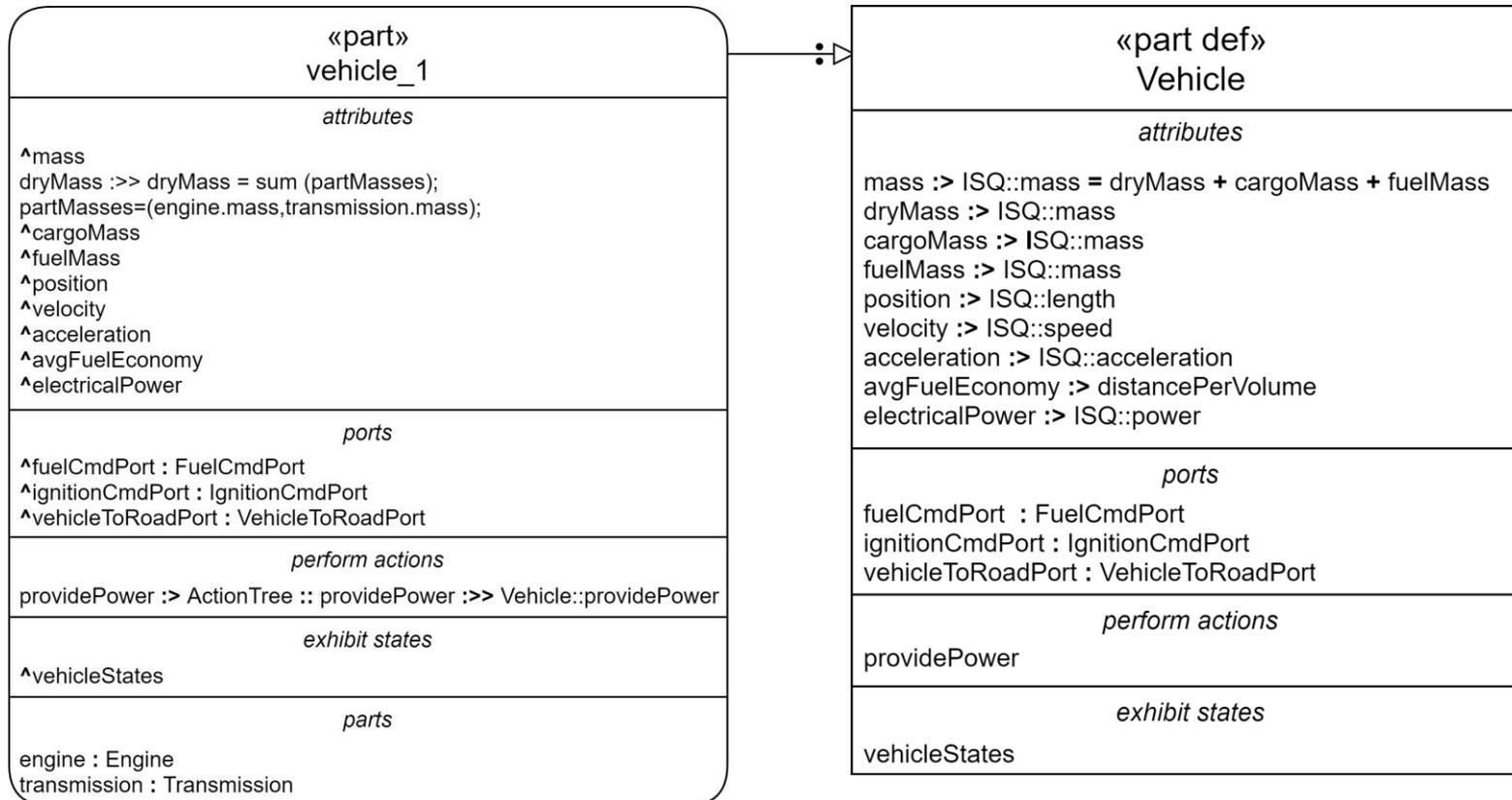


SysML v2 Reuse Patterns

SST

- **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
- **Variability**
 - Variation points represent elements that can vary
 - Variation applies to all definition and usage elements
 - A variant represents a particular choice at a variation point
 - A choice at one variation point can constrain choices at other variation points
 - A system can be configured by making choices at each variation point consistent with the specified constraints

- Parts are specializations of their definitions (defined by)
 - Enables adaptation of each usage to its context by inheriting and redefining its features





Vehicle Part Textual Syntax

SST

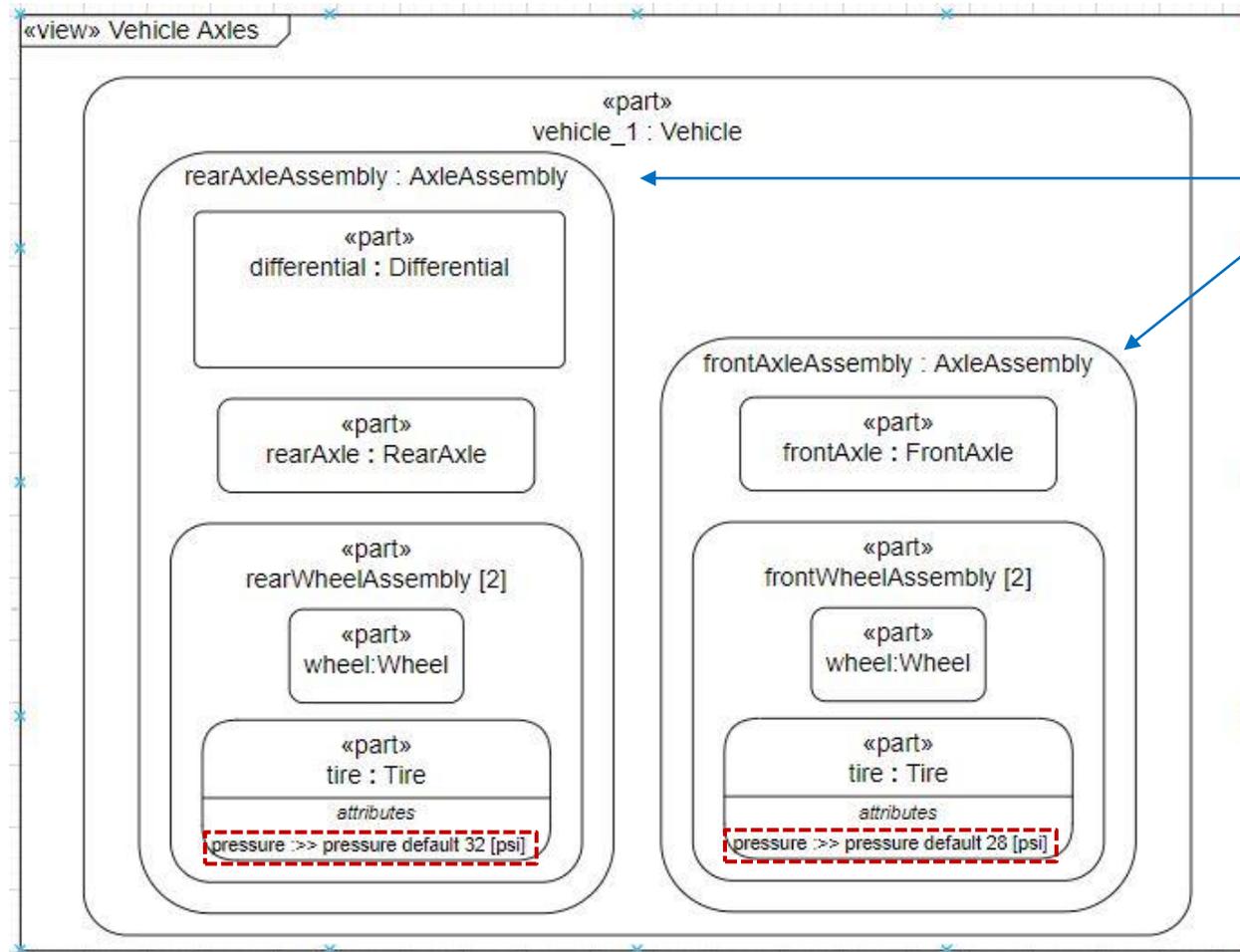
nested part →

```
part vehicle_1:Vehicle{
  attribute mass redefines mass;
  attribute dryMass redefines dryMass = sum (partMasses);
  attribute partMasses=(engine.mass,transmission.mass);
  perform ActionTree::providePower redefines providePower;
  part engine:Engine{
    attribute mass redefines mass default 200 [kg];
    port fuelCmdPort:>>fuelCmdPort=vehicle_1.fuelCmdPort;
    port ignitionCmdPort:>>ignitionCmdPort=vehicle_1.ignitionCmdPort;
    perform ActionTree::providePower.generateTorque;
    part cylinders[6]:Cylinder;
  }
  part transmission:Transmission{
    attribute mass redefines mass default 60 [kg];
    perform action amplifyTorque:>> amplifyTorque = ActionTree::providePower.amplifyTorque;
  }
  connect engine.drivePwrPort to transmission.drivePwrPort;
}
```

connection →

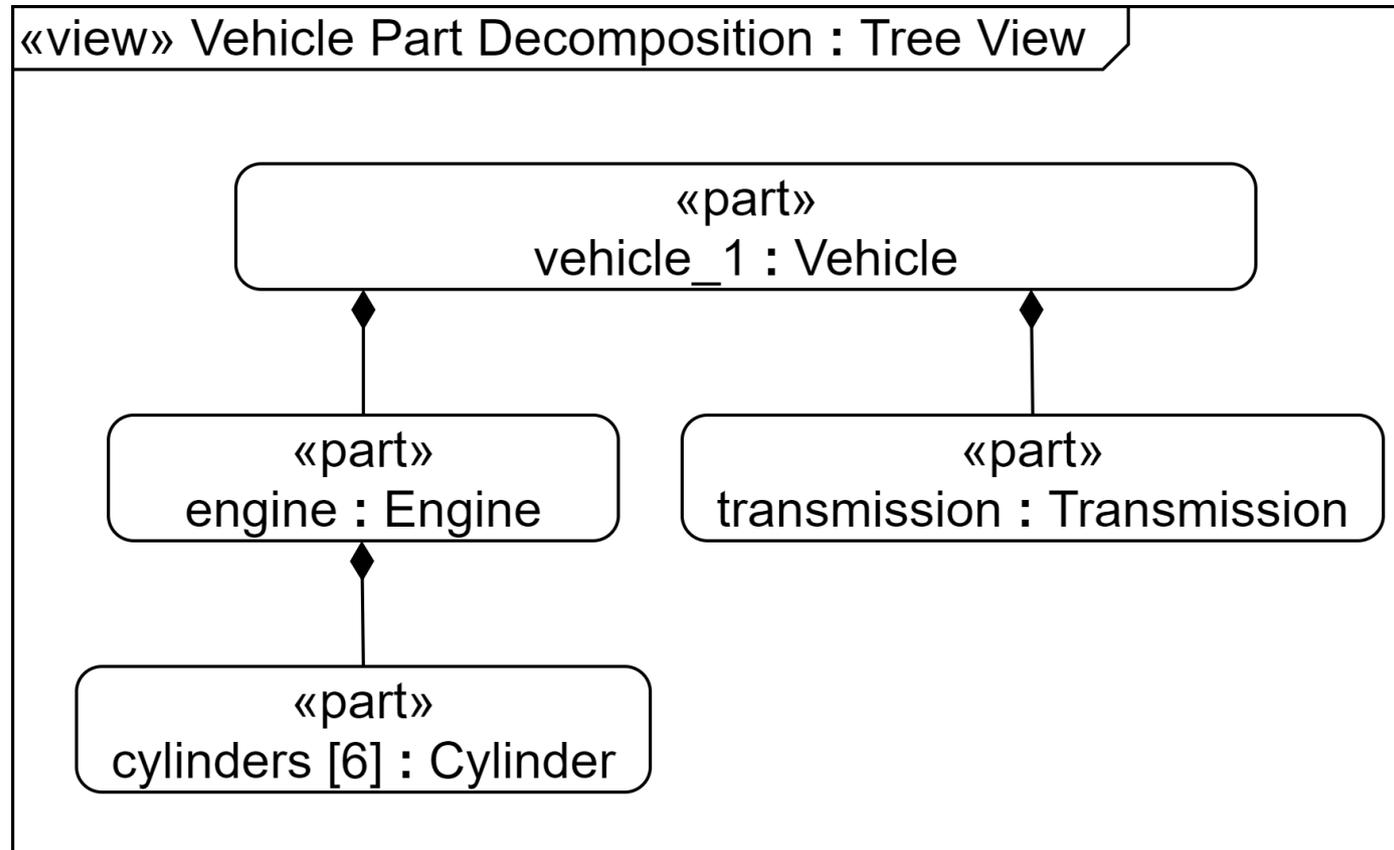
Vehicle Usage Example

Modifying Usages to their Context

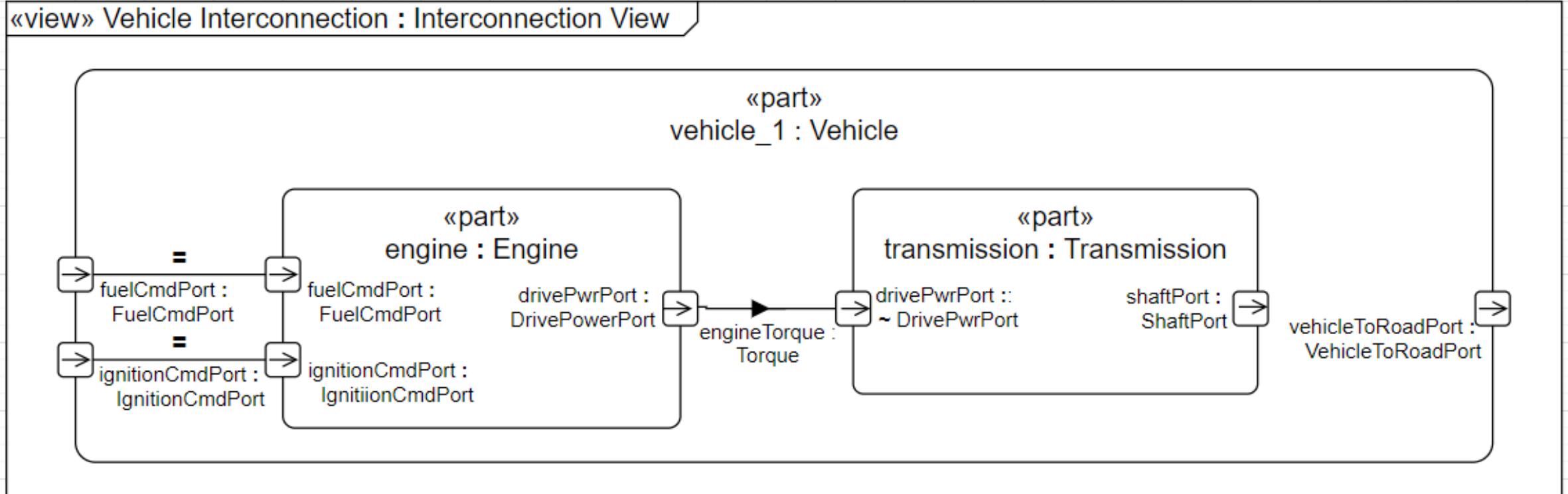


Different usages of Axle Assembly

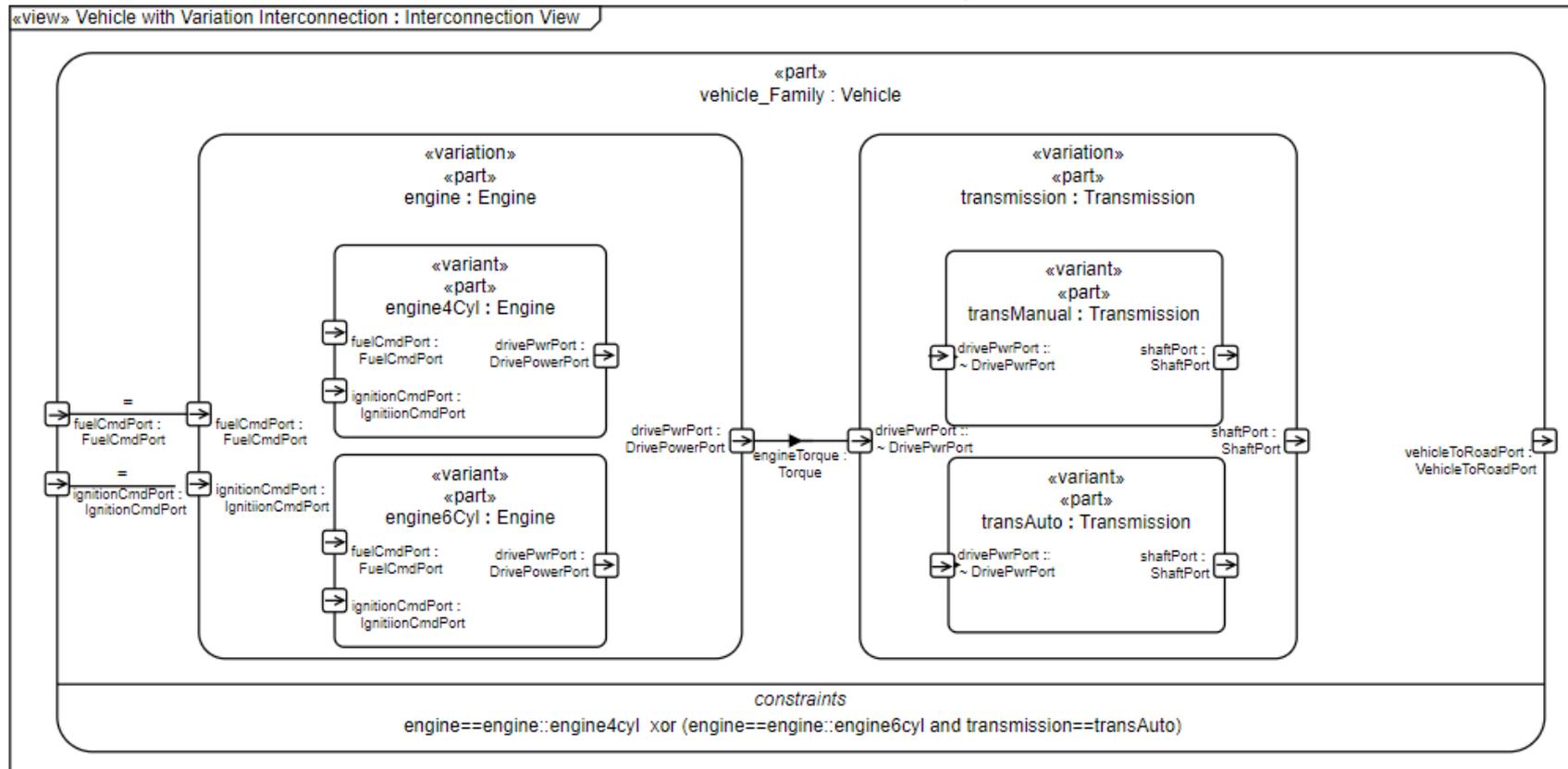
Tire pressure is different on front and rear tires



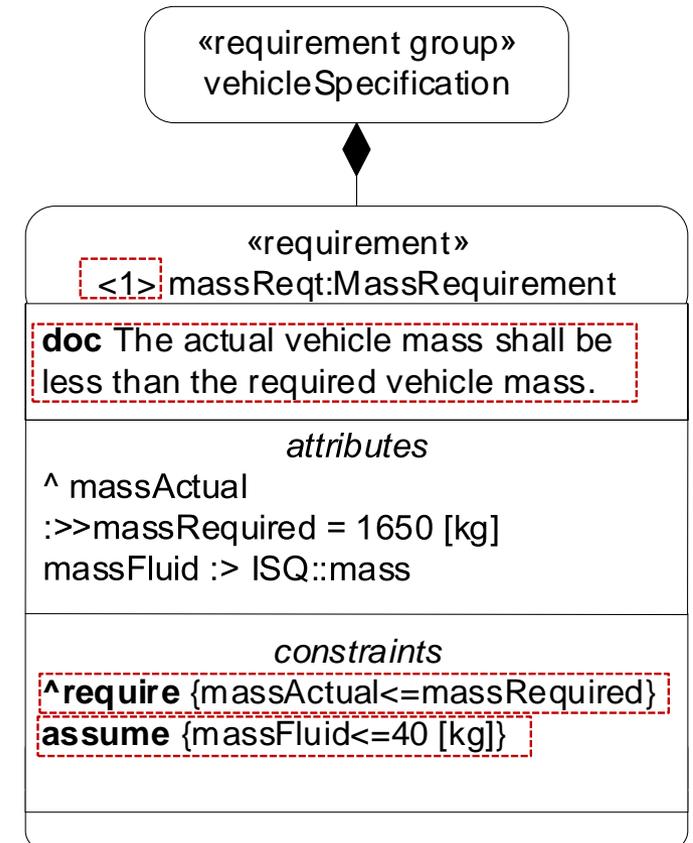
- Part interconnection contains parts, ports, connections, and flows



- Define vehicle configuration by specializing vehicle family with variation, identifying variants, and redefining features as required



- 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
 - can apply to performance, functional, interface and other kinds of requirements if desired
 - Assumed constraint expression that is asserted to be true for the requirement to be valid





4D Model & Occurrences

SST

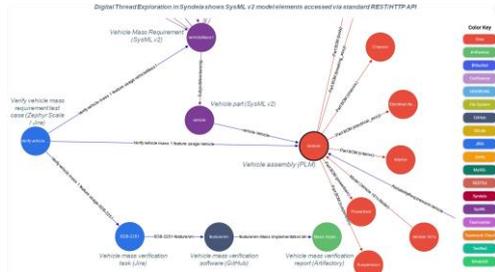
- Each entity called an occurrence has a lifetime
 - Distinct from attributes which do not have a lifetime
 - Reference clock
 - Can specify time slices and snapshots
- Spatial items are kinds of occurrences that have spatial extent that can change over their lifetime
 - Specified by shapes with position and orientation within coordinate frames
- Individuals
 - Unique occurrence with a lifetime



SysML v2 to v1 Terminology Mapping (partial)

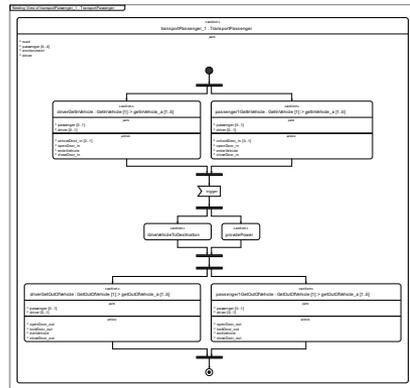
SST

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



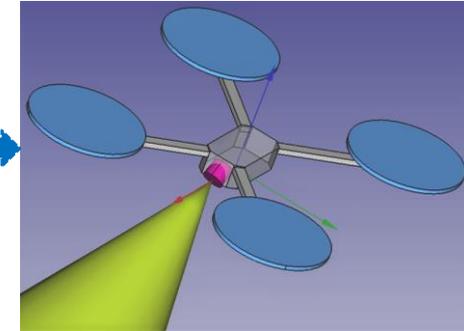
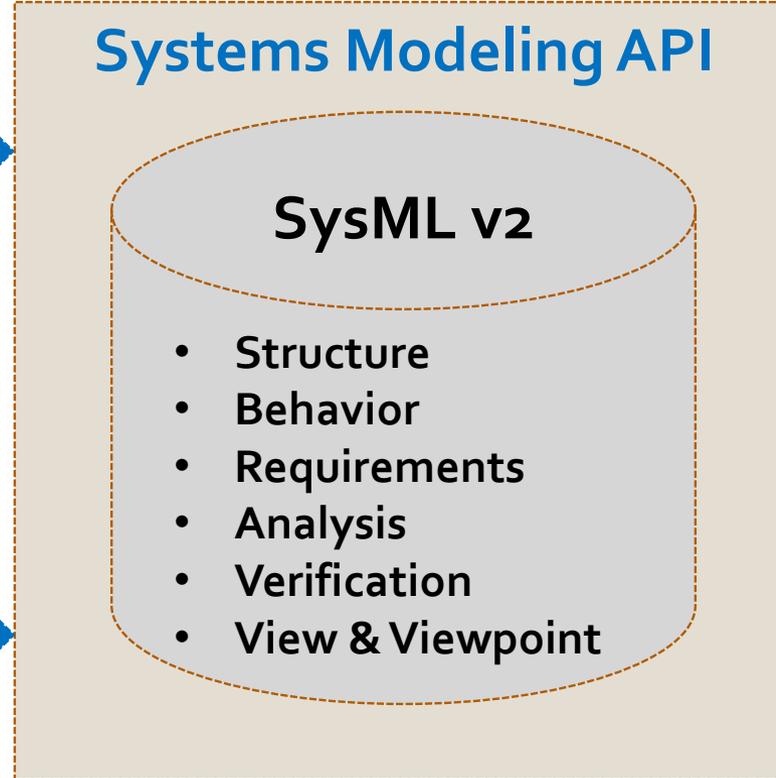
CM of the Digital Thread

Source: Syndeia with SysML v2



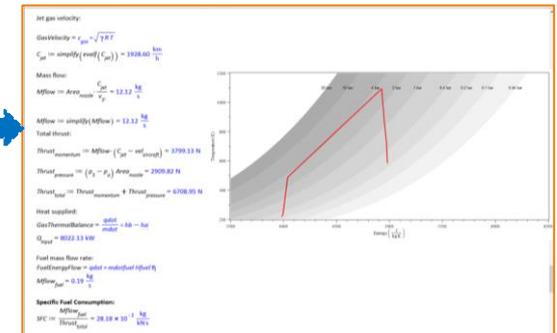
Graph Visualization

Source: Tom Sawyer with SysML v2



CAD/CAD Viewer

Source: FreeCAD with SysML v2



Analysis Solver

Source: Maple with SysML v2



Contrasting SysML v2 with SysML v1

SST

- **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 (unowned members, 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



Planned vs Completed Work Since IW 2022

SST

Language

- Finalize specification of graphical syntax
- Time semantics and change/time events
- Simple geometry (spatial semantics and shape library)
- Language extension
- Behavior execution guidance
- Model interchange
- Conformance cases
- SysML v1 to v2 transformation

API & Services

- Cross project element referencing
- Conformance tests
- OSLC PSM
- Query specification updates
- API Recipes

Work completed in 2022

Additional work to be done during finalization



SysML v2 Milestones

SST

December, 2017	SysML v2 RFP issued
June, 2018	SysML v2 API & Services RFP issued
August, 2020	Initial Submission
February, 2021	Stakeholder Review
August, 2021	Revised Submission
November, 2021	2nd Revised Submission (OMG evaluation initiated)
September, 2023	Specification Review at OMG
November, 2022	3 rd Revised Submission
1st Qtr 2023	Final Submission (beta specification pending OMG approval)
2024	Adopted Specification (pending OMG approvals)



Language Demo



SST

API & Services Demo



Summary



Summary

SST

- SysML v2 is addressing SysML v1 limitations to improve MBSE adoption and effectiveness
 - Precision, expressiveness
 - Regularity, usability
 - Interoperability with other engineering models and tools
- Approach
 - SysML v2 metamodel with formal semantics architected to overcome fundamental UML limitations
 - Flexible graphical notations and textual notation
 - Standardized API for interoperability
 - Transformation specification from SysML v1 to SysML v2
- Plan
 - Final submission (beta specification) - Q1 2023
 - Final adopted specification - 2024



SST Public Repositories

Current Release: 2022-12

SST

- 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)



SST

Thank You!!