



Implementing SysML v2 – Part 2

Lilleigh Stevie
Systems Engineer, SPEC Innovations
lilleigh.stevie@specinnovations.com

Steven H. Dam, Ph.D., ESEP
President, SPEC Innovations
571-485-7805
steven.dam@specinnovations.com


Presentation Overview

- SysML v2 Overview
 - Background and Objectives
 - What is KerML?
 - What is SysML v2?
 - Plan for SysML v2
- SysML v2 with JupyterLab Pilot Implementation
- Research in Understanding and Applying SysML 2.0
 - GMU SYST 699 Research
 - LML to SysML v2 Conversion Guide
 - LML to SysML v2 Entity/Relationship Mapping
 - LML to SysML v2 Conversion Process
- Plans for Implementing SysML v2 in Innoslate[®]


SysML v2 Overview

Background and Objectives

- SysML v1 was adopted in 2006 and has been used across the MBSE industry
 - Learned both strengths and weaknesses
 - Outgrew UML
- Systems Engineering Domain Special Interest Group (SE DSIG)
 - OMG working group
 - Developed requirements for the next generation of SysML
- December 2017 - OMG RFP for SysML v2 Language
- June 2018 - OMG RFP for standardized SysML v2 API and Services
- Led to the establishment of the SysML v2 Submission Team (SST)
 - Over 200 people from over 80 organizations



SysML v2 Objectives



Increase adoption and effectiveness of MBSE by enhancing...

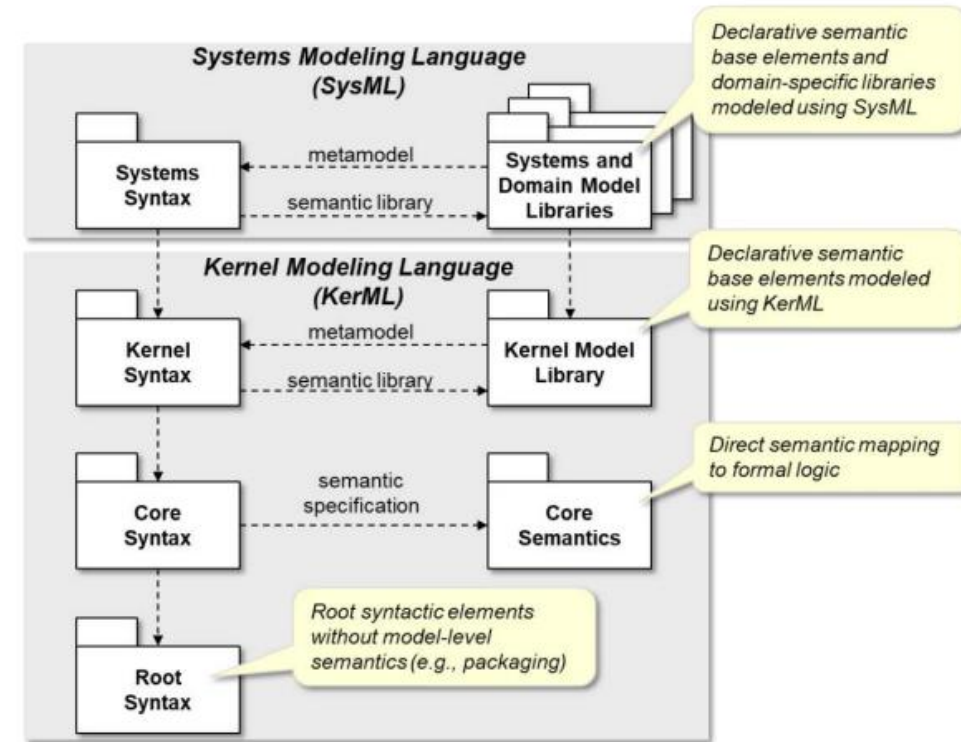
- Precision and expressiveness of the language
- Consistency and integration among language concepts
- Interoperability with other engineering models and tools
- Usability by model developers and consumers
- Extensibility to support domain specific applications
- Migration path for SysML v1 users and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

30 January 2021

From presentation by Sanford Friedenthal, SysML v2 Submission Team (SST), 30 January 2021, INCOSE International Workshop

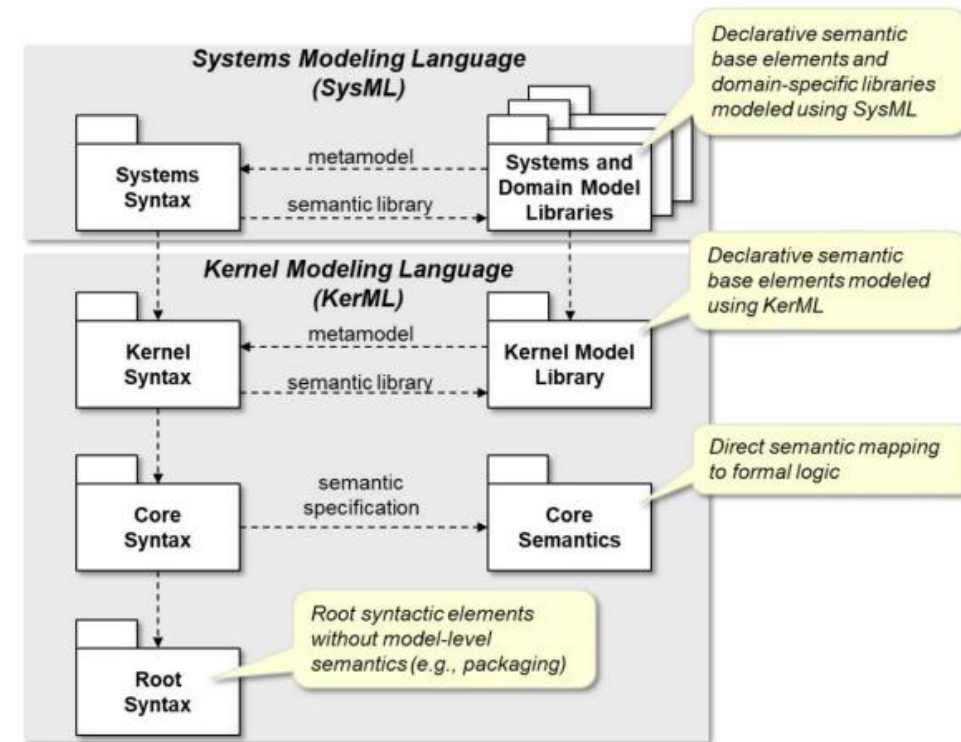
What is KerML?

- Kernel Modeling Language
- Created by the SysML v2 Submission Team (SST) to be a modeling language to create modeling languages
- Consists of three metamodel layers:
 - Root Layer - most general syntactic constructs
 - Includes elements and relationships between them, annotations of elements, and membership of elements in namespaces
 - Core Layer - most general constructs that have semantics based on classification
 - Includes type, feature, and classifiers
 - Kernel Layer - provides commonly needed modeling capabilities
 - Include classes, data types, structure, associations, connectors, behaviors, interactions, functions, expressions, feature values, multiplicities, metadata, and packages
- Foundation for SysML v2



What is SysML v2?

- Fourth metamodel layer:
 - Systems Layer - specializes the KerML to include domain-specific concepts
- Metaclasses either reuse or extend the KerML metaclasses
- Usage-Oriented
- Provides a textual notation and a graphical notation
- Graphical notation
 - Diagram types will be defined to facilitate the transition from v1 to v2
 - Will not constrain what elements can be on a particular diagram, so long as it conforms to the metamodel
- Can be extended to build domain specific extensions



Plan for SysML v2

- SST finalized the specification documents and artifacts for their September OMG meeting to submit early
- Final submission is in November
- OMG adoption and finalization to start December/January
- Beta specification to be released at the beginning of next year
- Finalization process will take until 2024
- 1.0 Specification to release early to mid 2024

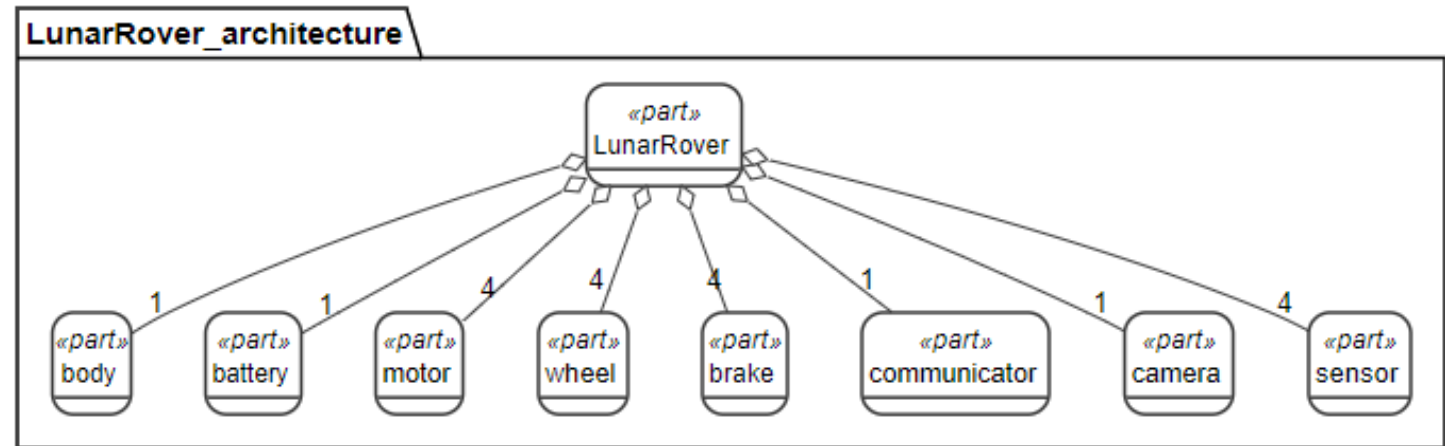
SysML v2 with JupyterLab Pilot Implementation

- Package
 - container for elements to organize a model
- Part
 - unit of structure
 - may have attributes, ports, perform actions, and exhibit states
- Decomposition

```
package LunarRover_architecture{
  part LunarRover{
    part body;
    part battery;
    part motor [4];
    part wheel [4];
    part brake [4];
    part communicator;
    part camera;
    part sensor [4];
  }
}
```

Comment (723263a8-a0b0-4bb2-ba61-f2daa37f8a25)
 Package LunarRover_architecture (c7cb8c37-dddf-480a-a09d-9b79a4ca50ee)

```
%viz --view=Tree LunarRover_architecture
```



```

package LunarRover architecture{
import ScalarValues::*;

attribute def Resolution{
  resolution : Integer;
}

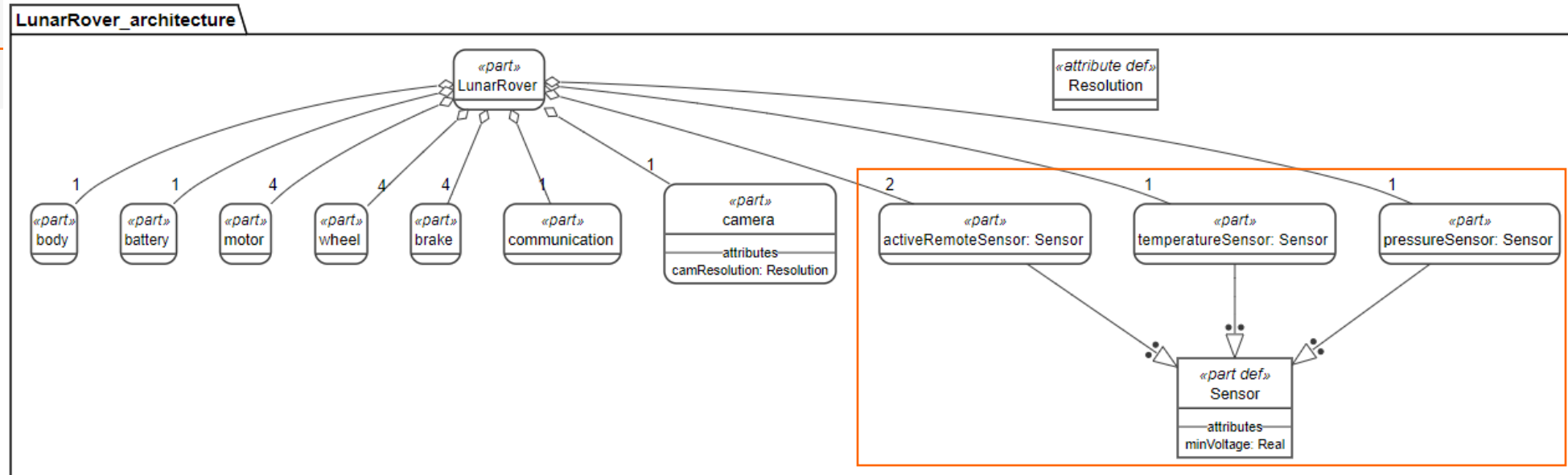
part def Sensor {
  attribute minVoltage : Real;
}

part LunarRover{
  part body;
  part battery;
  part motor [4];
  part wheel [4];
  part brake [4];
  part communicator;
  part camera{
    attribute camResolution : Resolution;
  }
  part activeRemoteSensor : Sensor [2];
  part temperatureSensor : Sensor;
  part pressureSensor : Sensor;
}
}

```

- Definition and Usage
 - Facilitates reuse in different context
 - Applies to all element types
 - Definition
 - Classifies or defines a type of element before its use
 - Usage
 - Usage of a definition element in a certain context
 - Must be defined by at least one definition element, default is most general definition
 - Colon connects a usage to its definition

- Import
 - Libraries and packages
- Multiplicity
 - Denoted by square brackets
 - Range or single number
- Attribute
 - Define a set of values

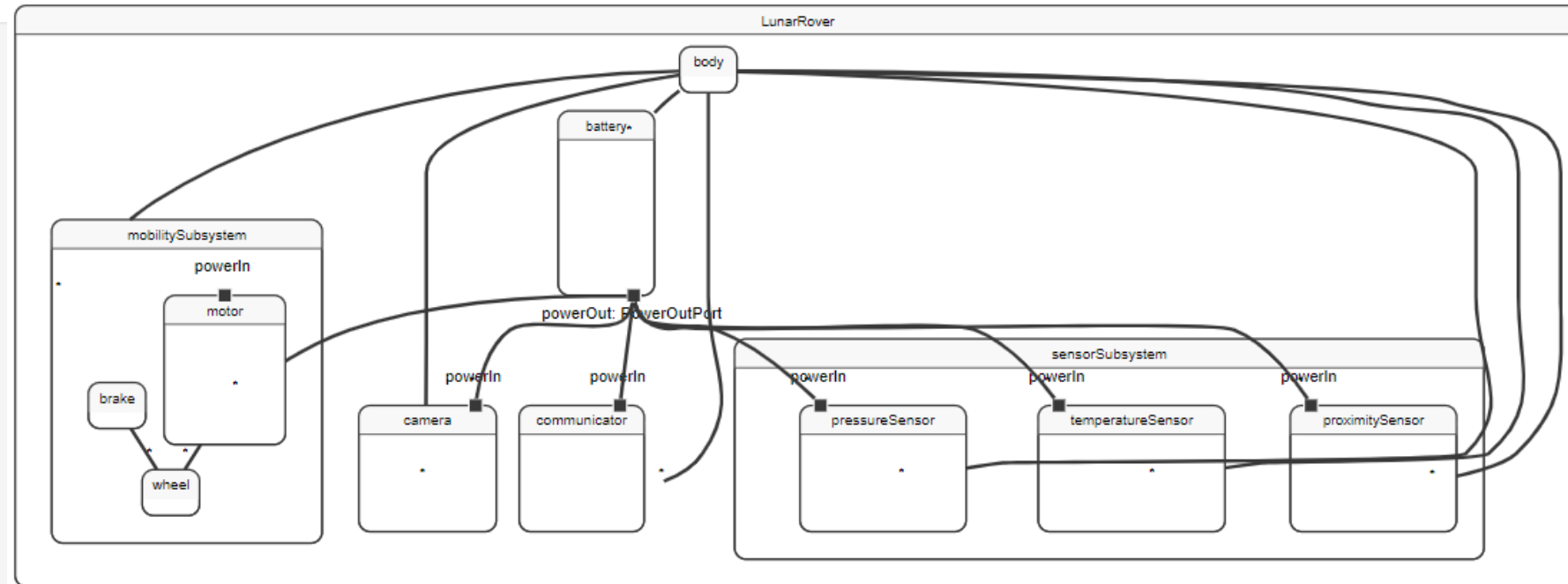


```
package LunarRover_architecture{
```

```
item def Energy;  
port def PowerOutPort{  
  out item powerSupply : Energy;  
}  
interface def PowerInterface{  
  end PowerOut : PowerOutPort;  
  end PowerIn : ~PowerOutPort;  
}  
}
```

```
part LunarRover{
```

```
  part mobilitySubsystem{  
    part wheel [4] ;  
    part brake [4] ;  
    part motor [4] {  
      port powerIn : ~PowerOutPort;  
    }  
  }  
  part body;  
  part battery{  
    port powerOut : PowerOutPort;  
  }  
  part communicator{  
    port powerIn : ~PowerOutPort;  
  }  
  part camera{  
    port powerIn : ~PowerOutPort;  
  }  
  part sensorSubsystem{  
    part proximitySensor [2]{  
      port powerIn : ~PowerOutPort;  
    }  
    part temperatureSensor{  
      port powerIn : ~PowerOutPort;  
    }  
    part pressureSensor{  
      port powerIn : ~PowerOutPort;  
    }  
  }  
}
```



```
interface : PowerInterface connect  
  PowerOut :> battery.powerOut to  
  PowerIn :> communicator.powerIn;
```

```
connect body to battery;  
connect body to mobilitySubsystem;  
connect body to communicator;  
connect body to camera;  
connect body to sensorSubsystem.proximitySensor;  
connect body to sensorSubsystem.temperatureSensor;  
connect body to sensorSubsystem.pressureSensor;  
connect battery.powerOut to mobilitySubsystem.motor;  
connect battery.powerOut to camera.powerIn;  
connect battery.powerOut to sensorSubsystem.proximitySensor.powerIn;  
connect battery.powerOut to sensorSubsystem.temperatureSensor.powerIn;  
connect battery.powerOut to sensorSubsystem.pressureSensor.powerIn;  
connect mobilitySubsystem.motor to mobilitySubsystem.wheel;  
connect mobilitySubsystem.brake to mobilitySubsystem.wheel;
```

```
  }  
}
```

- Items

- Class of things that exist in space and time
- May be stored in and/or flow between parts of a system

- Ports

- Connection point for interactions
- Contained by part elements

- Interface

- Connection whose ends are ports

- Connection

- Relationship and kind of part that classifies connections between related things

```

import SI::*;
part LunarRover{
  attribute mass = 987 [kg];
}
package LunarRoverRequirements{
  requirement def LunarRoverMassReq{
    doc /* The lunar rover shall have a mass less than 12000 kg. */

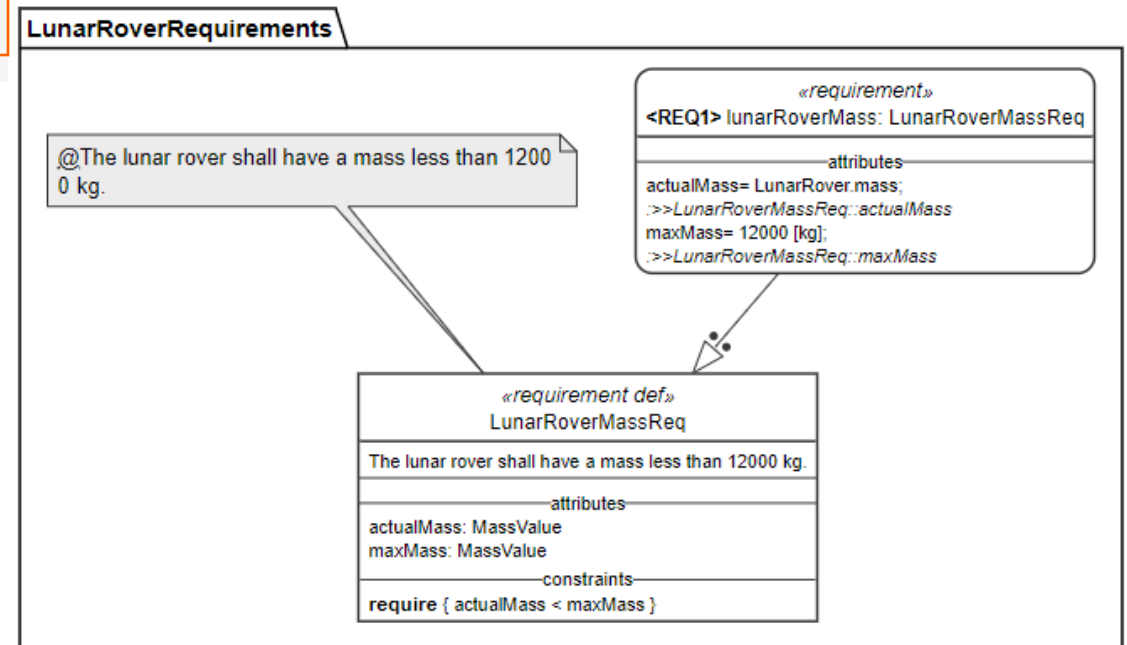
    attribute actualMass : MassValue;
    attribute maxMass : MassValue;

    require constraint{ actualMass < maxMass }
  }

  requirement <REQ1> lunarRoverMass : LunarRoverMassReq{
    attribute :>> actualMass = LunarRover.mass;
    attribute :>> maxMass = 12000 [kg];
  }
}

```

- Constraint
 - Reusable, parameterized Boolean expression
- Requirement
 - Specifies stakeholder-imposed constraints that must be satisfied for a design to be a valid solution
 - Has a textual statement, can require constraints, may include assumed constraints, can be parameterized, can specify the subject
 - Requirement usage will often bind the requirement definition parameters to specific values



```
package LunarRover_variant{
```

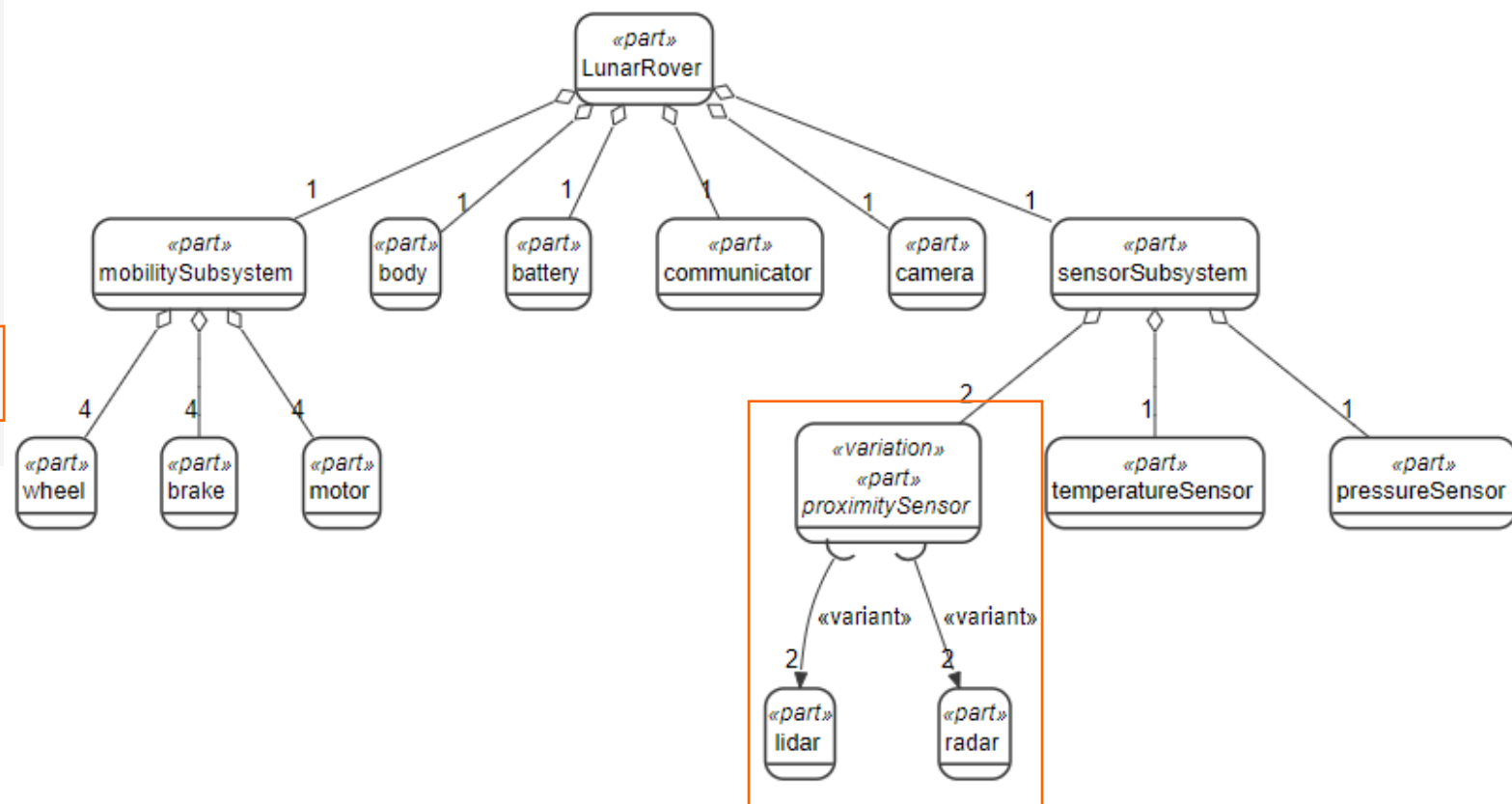
```
part lidar;  
part radar;
```

```
part LunarRover{  
  part mobilitySubsystem{  
    part wheel [4] ;  
    part brake [4] ;  
    part motor [4] ;  
  }  
  part body;  
  part battery;  
  part communicator;  
  part camera;
```

```
  part sensorSubsystem{  
    variation part proximitySensor [2]{  
      variant part lidar;  
      variant part radar;  
    }  
    part temperatureSensor;  
    part pressureSensor;  
  }  
}
```

```
part LunarRover11 :> LunarRover {  
  part redefines lidar :> sensorSubsystem.proximitySensor;  
}
```

- Variation
 - Identifies a variation point
 - Can apply to any kind of definition or usage
 - Can be nested
- Variant
 - Usage elements



Overview of Other Elements

- Action - Occurrence over time that can coordinate with other actions and generate effects on items and parts involved
- Control Nodes - Merge, Decision, Fork, and Join
- Structured Control Actions - If Action, For Loop Action, While Loop Action
- Dependency - Kind of relationship between any number of source and target elements
- Occurrence - Class of elements that have an extent in time and may have spatial extent
- Time Slice - Represents a periods or phases of the lifetime of an occurrence over some period of time
- Snapshot - Represents an occurrence at a specific point in time
- Individual - An occurrence of any kind can be restricted to model a single individual and how it changes over its lifetime
- Viewpoint - Frames the concerns of one or more stakeholders regarding info about a modeled system or domain of interest
- State - Types of states and transitions are types of action; Can be exhibited by a part
- Calculation - Reusable, parameterized expression that returns a result
- Analysis Case - Defines the computation of an analysis of some subject in order to meet an objective and output a result
- Verification Case - Defines a process for verifying whether a subject satisfies one or more requirements
- Use Case - Specifies a sequence of interactions between the subject and various actors, modeled as part usages; Interactions are modeled as messages
- Specialization – Subclassification – Definition; Defines a subset of the classification of its generalization; Inherits features of general definitions; Can have multiple generalizations
- Specialization – Subsetting – Usage; Kind of generalization between features
- Redefines - Kind of subsetting; Replaces the redefined usage in the context of the redefining usage

Research in Understanding and Applying SysML 2.0

GMU SYST 699 Research

- Graduate-level capstone project
- Professor: Dr. Karla Hoffman
- Sponsor: INCOSE Automotive Working Group (represented by David Hetherington and Steve Dam)
- Students: Liza Kossobokova and Tim Lockhart
- Title: Initial Capability and Exploration and MBSE Model Translation
- Key Tasks:
 - Manually replicated LML model to SysML v2 textual notation
 - Developed LML to SysML v2 conversion guide and entity/relationship mapping
 - Developed LML to SysML conversion process

LML to SysML v2 Conversion Guide

Item	LML	SysML v2	Notes
1	Project named "Autonomous Vehicle Sample"	package AutonomousVehicleSample { }	All contents of project/package need to be in between { and }
1a		import ISQ::*; import SI::*; import USCustomaryUnits::*;	Rule of thumb to import packages necessary for units. Insert directly after defining a new package
2	Asset "ABC"	part def ABC { } }	All Assets need to first be defined as parts before they can be used. "part def ABC { }" defines ABC, which can later be used. For example: "part abc : ABC;" or "part Innoslate : ABC;" are usages of ABC.
3	Asset "ABC" has Asset "XYZ"	part def XYZ [] part def ABC { part xyz : XYZ; }	Alternate: part def XYZ [] part def ABC [] part abc : ABC; part xyz : XYZ;
4	Asset "ABC" has Asset "XYZ", which has Asset "EFG"	part def XYZ [] part def ABC { part xyz : XYZ; part efg : EFG; }	Can nest as much as necessary
5	Connection "CON" is used to connect Asset types "ABC" and "EFG"	connection def CON { end : ABC; end : EFG; }	
6	Assets abc and efg are connected, but the connection is not defined	connect abc to efg;	
7	Asset "abc" has Assets "xyz" and "efg". Assets xyz and efg are connected via connection "CON"	part abc : ABC { part xyz : XYZ; part efg : EFG; connection : CON connect xyz to efg; }	Must first define parts ABC, XYZ, and EFG (i.e. - "part def ABC { }"), as well as the connection (i.e. - "connection def CON { }"). Parts xyz and efg must be of type XYZ and EFG, which are defined as ends in connection def CON.
8	Asset "ABC" has attributes "range" and "weight", measured in meters and kilograms, respectively	part def ABC { attribute range > SI::m; attribute weight > SI::kg; }	Must first import the SI package

- Prioritized key modeling entities that could produce the best results
- Focused on closest "1-to-1" matches between the two languages

Item	LML	SysML v2	Notes
9	Model has Resource "Power". Power is stored in Asset "Battery".	item def Power; part def Battery { item power : Power; }	A SysML v2 "item" is a class of things that exist in space and time but are not necessarily considered parts of a system. Items may also model continuous materials that are stored in and/or flow between Assets. Just like parts, items must be defined, and then can be used.
10	Requirement number "1.1" named "REQ" with text "ABC XYZ" The subject of this requirement is "abc" of type "ABC"	requirement def <1.1> REQ { doc /* ABC XYZ. */ subject abc : ABC; }	A requirement definition is always about some subject, which may be implicit or specified explicitly
11	Model has Action "HU" with input "KLM" and output "NOP". KLM is a type of "stuff" and NOP is a type of "things"	action def HU (in KLM : stuff, out NOP : things);	"stuff" and "things" most both be items defined elsewhere in the model
12	Model has Action "HU", which does not have any inputs or outputs	action def HU;	
13	Asset "abc" performs Action "hj".	part abc : ABC { perform action hj;	abc is a usage of part definition ABC hj must be a defined action

LML to SysML v2 Entity/Relationship Mapping

- Entities of LML are more easily mapped to the entities of SysML v2
- LML relationship and entities can map to multiple SysML v2 entities depending on context

LML Entity Name	Parent Entity	Description	SysML v2 Entity Name
Action		An Action entity specifies the mechanism by which inputs are transformed into outputs.	Action
			Transition
			Case
			Trade-Off Case
			Use Case
			Analysis Case
			Verification Case
			Accept Action
Send Action			
Artifact		An Artifact entity specifies a document or other source of information that is referenced by or generated in the knowledgebase.	Textual Representation
Asset		An Asset entity specifies an object, person, or organization that performs Actions, such as a system, subsystem, component, or element.	Part Actor Stakeholder
Characteristic		A Characteristic entity specifies properties of an entity.	State Attribute
Conduit	Connection	A Conduit entity specifies the means for physically transporting Input/Output entities between Asset entities. It has limitations (attributes) of capability and latency.	Interface
Connection		A Connection entity specifies the means for relating Asset instances to each other.	Connection
			Binding Connection
			Flow Connection

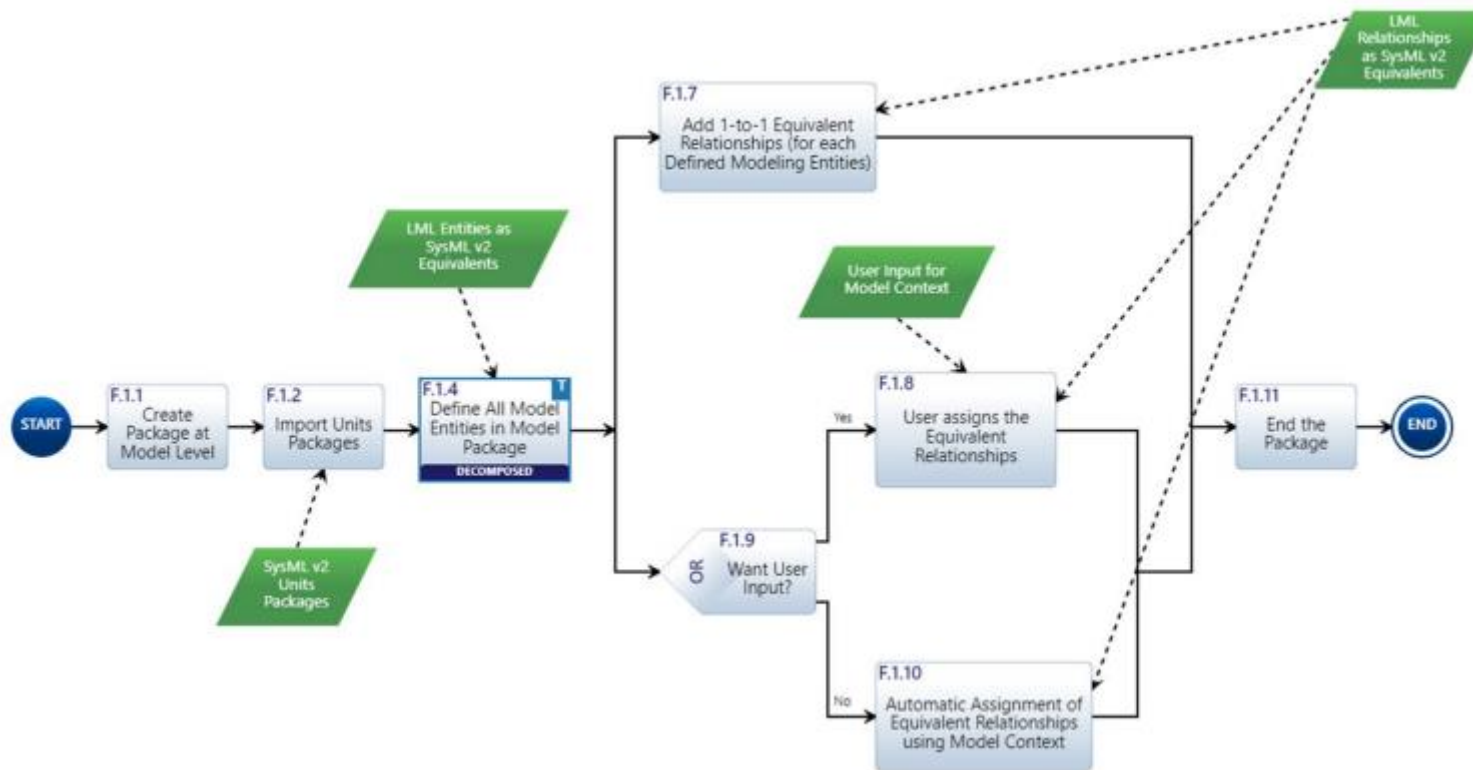
Entity Mapping

LML Relationship	LML Elements Related	SysML relationships
decomposes/decomposed by	All Elements to themselves	Definition, Element Group Membership, subsets, subclassifies, redefines
decomposes/decomposed by	Statement to Statement	Definition, Requirement Group Membership, derives/derived by
related to/relates	All Elements to themselves	Dependency, Association, references/is referenced by, refines/refined by, Element Group Membership, subsets, subclassifies, redefines
related to/relates	Statement to Statement	Relationship between the verification cases and their verification objectives, derives/derived by
references/referenced by	Artifact to All Elements	Dependency, is documented by, explains/explained by, about
specifies/specified by	Characteristic to All Elements	Defined as an attribute (property) of a part, conforms/constrained by
performs	Asset to Action	Allocation, Performs
consumes/consumed by	Asset as Resource to Action	receives/generates, references/is referenced by, flow (A kind of connection that transfers items from a source output to a target input)
produces/produced by	Asset as Resource to Action	causes/is caused by, connector, association, references/is referenced by, flow

Relationship Mapping

LML to SysML v2 Conversion Process

- How to do an LML to SysML v2 conversion
- Complements the conversion guide and entity/relationship mapping



Plans for Implementing SysML v2 in Innoslate®

Plan for Implementing SysML v2 in Innoslate

- With the finalization of SysML v2 specification, continue and improve the mapping between LML and SysML v2 in order to be as accurate as possible with translating between them
- Add/update diagrams for modeling SysML v2 in Innoslate
- Explore options for interoperability between tools with the SysML v2 API and/or JSON and/or XMI

Shortfalls

- Difficult to visualize models with only the textual notation
- Learning curve
 - Have to learn the textual notation for the modeling entities
 - Have to learn new concepts introduced with SysML v2
- Long time to implement in tools

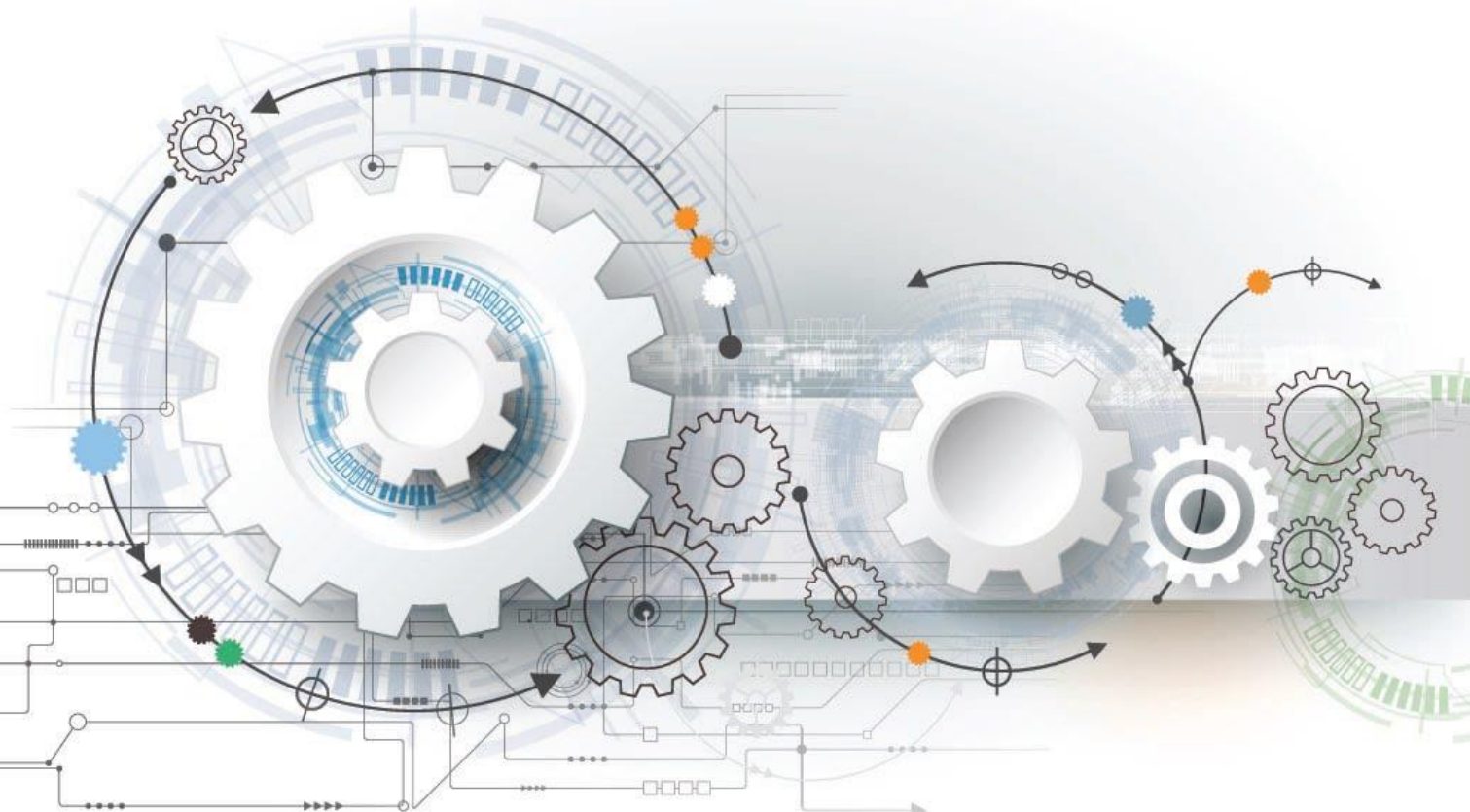
References

- A Preview of the Next Generation System Modeling Language (SysML v2), Sanford Friedenthal and Ed Seidewitz, pg. 6-22. <https://www.ppi-int.com/wp-content/uploads/2020/11/PPI-SyEN-95.pdf>
- Systems Modeling Language (SysML v2) Support for Digital Engineering, Manas Bajaj, Sanford Friedenthal, and Ed Seidewitz, pg. 19-24. https://connect.incose.org/Library/InsightMagazine/Practitioners%20Magazine/INSIGHT_v25-1_0331.pdf
- The MBSE Horizon: Advances and Challenges over the next few years, James Towers. <https://www.linkedin.com/pulse/mbse-horizon-advances-challenges-over-next-few-years-james-towers/>
- The MBSE Podcast: Episode 25 – SysML v2: A look behind the scenes with Ed Seidewitz. <https://mbse-podcast.rocks/episode-25-sysml-v2-a-look-behind-the-scenes-with-ed-seidewitz/>
- INCOSE Sweden Webinar 20220209 SysML v2 with Tim Weilkiens. <https://www.youtube.com/watch?v=ggXknIHSU7Q>
- MBSE4U SysML v2 JupyterBook. <https://www.sysmlv2lab.com/lab/workspaces/auto-X>
- SysML v2 Release GitHub. <https://github.com/Systems-Modeling/SysML-v2-Release>
- Kossobokova, L., Lockhart, T. (2022). SysML v2: Initial capability exploration and MBSE model translation.

Questions and Answers

Contact us on LinkedIn
(<https://www.linkedin.com/groups/7415373/>) or
at support@innoslate.com

Thank you!
Visit cloud.innoslate.com for a trial.



SPEC Innovations



@SPECInnovations



Innoslate User Group



Innoslate.com/blog



571.485.7800



specinnovations.com