SysML v2 Submission Team (SST)
SysML v2 Update

Sanford Friedenthal
SST Co-Lead
safriedenthal@gmail.com
Purpose

- Provide an update from the 2021 IW on the status of SysML v2

Agenda

- SysML v2 Overview
- Progress and Plans
- Summary
Model-Based Systems Engineering

- Part of the digital transformation
- Spans the systems engineering lifecycle
- Facilitates
  - managing complexity and risk
  - more rapidly responding to change
  - reuse and design evolution
  - reasoning about and analyzing systems
  - shared stakeholder understanding
  - documentation and reporting

Source: INCOSE SE Vision 2035
SysML v2 Overview
SysML v2 Track Leads

- Project Management – Ed Seidewitz, Sandy Friedenthal
  - Infrastructure – John Watson, Chris Delp
- Requirements V&V – Sandy Friedenthal
- Transformation – Yves Bernard, Tim Weikiens
- Metamodel Development – Karen Ryan
- API/Services Development – Manas Bajaj
- Pilot Implementation – Ed Seidewitz
SST Participating Organizations

- Aerospace Corp
- Airbus
- ANSYS medini
- Aras
- Army Aviation & Missile Center
- Army CBRND
- BAE
- BigLever Software
- Boeing
- U.S. Army DEVCOM Armaments Center
- CalTech CTME
- CEA
- Contact Software
- Defence Science and Technology Group
- DEKonsult
- Delligatti Associates
- Draper Lab
- ESTACA
- Ford
- Fraunhofer FOKUS
- Galois
- General Motors
- George Mason University
- GiSE
- 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
- UFREC
- 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
SysML has evolved to address user and vendor needs
  - v1.0 adopted in 2006; v1.6 current version; v1.7 in process

SysML has facilitated awareness and adoption of MBSE

Much has been learned from using SysML for MBSE
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

26 January 2022
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 and execution
  - Graphical, Tabular, Textual

- Standardized API to access the model
• Metamodel specifies the core concepts in the language and their semantics
• Systems model library instantiates the core concepts from the metamodel for use
• User creates the user model from the systems model library
• API accesses and operates on the user model from the repository
SysML v2 Language Capabilities

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

Structure
- decomposition
- interconnection
- classification

Requirements

Analysis
- analysis cases
- expression language

Verification
- verification cases

View & Viewpoint
The vehicle part definition is characterized by different kinds of features including:

- Attributes
- Ports
- Actions
- States
- ...

```
«part def»
Vehicle

attributes
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

ports
fuelCmdPort : FuelCmdPort
ignitionCmdPort : IgnitionCmdPort
vehicleToRoadPort : VehicleToRoadPort

perform actions
providePower

exhibit states
vehicleStates
```
The textual syntax reflects the same model as the graphical syntax.
Vehicle States

- 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

```plaintext
vehicleStates parallel {
    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;
    }
    healthStates {
        entry action initial;
        state normal;
        state degraded;
    }
}
```
SysML v2 Reuse Patterns

- **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
Vehicle Part Defined by Vehicle Part Definition

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

```
«part»
vehicle_1

attributes
- mass
  - dryMass => dryMass = sum (partMasses);
  - partMasses = (engine.mass, transmission.mass);
- cargoMass
- fuelMass
- position
- velocity
- acceleration
- avgFuelEconomy
- electricalPower

ports
- fuelCmdPort : FuelCmdPort
- ignitionCmdPort : IgnitionCmdPort
- vehicleToRoadPort : VehicleToRoadPort

perform actions
- providePower : ActionTree :: providePower : => Vehicle : providePower

exhibit states
- vehicleStates

engine : Engine
transmission : Transmission
```

```
«part def»
Vehicle

attributes
- 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

ports
- fuelCmdPort : FuelCmdPort
- ignitionCmdPort : IgnitionCmdPort
- vehicleToRoadPort : VehicleToRoadPort

perform actions
- providePower

exhibit states
- vehicleStates
```
Vehicle Part
Textual Syntax

```plaintext
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;
}
```
Vehicle Part Decomposition

- vehicle_1 : Vehicle
  - engine : Engine
  - transmission : Transmission
Vehicle Part Interconnection

- Part interconnection contains parts, ports, connections, and flows
Define vehicle configuration by specializing vehicle family with variation, identifying variants, and redefining features as required.
SysML v2 to v1
Terminology Mapping (partial)

<table>
<thead>
<tr>
<th>SysML v2</th>
<th>SysML v1</th>
</tr>
</thead>
<tbody>
<tr>
<td>part / part def</td>
<td>part property / block</td>
</tr>
<tr>
<td>attribute / attribute def</td>
<td>value property / value type</td>
</tr>
<tr>
<td>port / port def</td>
<td>proxy port / interface block</td>
</tr>
<tr>
<td>action / action def</td>
<td>action / activity</td>
</tr>
<tr>
<td>state / state def</td>
<td>state / state machine</td>
</tr>
<tr>
<td>constraint / constraint def</td>
<td>constraint property / constraint block</td>
</tr>
<tr>
<td>requirement / requirement def</td>
<td>requirement</td>
</tr>
<tr>
<td>connection / connection def</td>
<td>connector / association block</td>
</tr>
<tr>
<td>view / view def</td>
<td>view</td>
</tr>
</tbody>
</table>
Contrasting 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 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, ..)
  - Query/filter expressions
  - Metadata

- **More extensible**
  - Simpler language extension capability
    - Based on model libraries

- **More interoperable**
  - Standardized API
Progress and Plans
# SysML v2 Spec (Clause 7)
## SysML v2 Language Description

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>Elements and Relationships</td>
</tr>
<tr>
<td>7.3</td>
<td>Annotations</td>
</tr>
<tr>
<td>7.4</td>
<td>Namespaces and Packages</td>
</tr>
<tr>
<td>7.5</td>
<td>Dependencies</td>
</tr>
<tr>
<td>7.6</td>
<td>Definition and Usage</td>
</tr>
<tr>
<td>7.7</td>
<td>Attributes</td>
</tr>
<tr>
<td>7.8</td>
<td>Enumerations</td>
</tr>
<tr>
<td>7.9</td>
<td>Occurrences</td>
</tr>
<tr>
<td>7.10</td>
<td>Items</td>
</tr>
<tr>
<td>7.11</td>
<td>Parts</td>
</tr>
<tr>
<td>7.12</td>
<td>Ports</td>
</tr>
<tr>
<td>7.13</td>
<td>Connections</td>
</tr>
<tr>
<td>7.14</td>
<td>Interfaces</td>
</tr>
<tr>
<td>7.15</td>
<td>Allocations</td>
</tr>
<tr>
<td>7.16</td>
<td>Actions</td>
</tr>
<tr>
<td>7.17</td>
<td>States</td>
</tr>
<tr>
<td>7.18</td>
<td>Calculations</td>
</tr>
<tr>
<td>7.19</td>
<td>Constraints</td>
</tr>
<tr>
<td>7.20</td>
<td>Requirements</td>
</tr>
<tr>
<td>7.21</td>
<td>Cases</td>
</tr>
<tr>
<td>7.22</td>
<td>Analysis Cases</td>
</tr>
<tr>
<td>7.23</td>
<td>Verification Cases</td>
</tr>
<tr>
<td>7.24</td>
<td>Use Cases</td>
</tr>
<tr>
<td>7.25</td>
<td>Views and Viewpoints</td>
</tr>
<tr>
<td>7.26</td>
<td>Language Extension (planned Q1 2022)</td>
</tr>
</tbody>
</table>

Most of the core language functionality is baselined

26 January 2022
High Priority Remaining Work

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
- Complete SysML v1 to v2 transformation

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

Additional work to be done during finalization

31 January 2022
SysML v2 Milestones

- 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)
- 2nd Qtr 2022     Final Submission (date to be confirmed)
- 2nd Qtr 2023     Finalized Specification (pending OMG approvals)
Summary
SST Public Repositories
Current Release: 2021-12

- Monthly release repository

- 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](https://github.com/Systems-Modeling)

- Google group for comments and questions
  - [https://groups.google.com/g/SysML-v2-Release](https://groups.google.com/g/SysML-v2-Release)
  (to request membership, provide name, affiliation and interest)
Summary

- SysML v2 is addressing SysML v1 limitations to improve MBSE adoption and effectiveness
  - Precision, expressiveness, usability
  - Interoperability with other engineering models and tools

- Approach
  - Simplified SysML v2 metamodel with formal semantics overcomes fundamental UML limitations
  - Flexible graphical notations and textual notation
  - Standardized API for interoperability

- Roadmap to final submission planned for Q2/Q3 2022
Attend the SysML v2 Demo
Sunday, January 30
10:00 – 3:00  PT
Salon F
Thank you!!