The OMG SysML-Modelica Transformation Specification:
Supporting Model-Based Systems Engineering with SysML and Modelica

Chris Paredis
Associate Director
Model-Based Systems Engineering Center
Georgia Tech
Chris.paredis@me.gatech.edu
SysML-Modelica Transformation Specification: Context & Objective

- Two complementary languages for Systems Engineering:
  - Descriptive modeling in SysML
  - Formal equation-based modeling for analyses and trade studies in Modelica

- Objective:
  - Leverage the strengths of both SysML and Modelica by integrating them to create a more expressive and formal MBSE language.
  - Define a formal Transformation Specification:
    » a SysML4Modelica profile
    » a Modelica abstract syntax metamodel
    » a mapping between Modelica and the profile
Presentation Overview

- What is SysML?
- What is Modelica?

- Motivating Example: Design & Analysis of Robot

- SysML-Modelica Transformation Specification
- Transformations in Model-Based Systems Modeling
- Summary
Acknowledgements

Working Group Members
- Yves Bernard (EADS)
- Roger Burkhart (Deere & Co)
- Wuzhu Chen (Univ. Braunschweig)
- Hans-Peter De Koning (ESA)
- Sandy Friedenthal (Lockheed Martin)
- Peter Fritzson (Linköping University)
- Nerijus Jankevicius (No Magic)
- Alek Kerzhner (Georgia Tech)
- Andreas Korff (Atego)
- Chris Paredis (Georgia Tech)
- Axel Reichwein (Georgia Tech)
- Nicolas Rouquette (JPL)
- Wladimir Schamai (EADS)

Students / post-docs
- Kevin Davies
- Sebastian Herzig
- Alek Kerzhner
- Ben Lee
- Roxanne Moore
- Axel Reichwein
- Wladimir Schamai

Sponsors
- Deere & Co
- Lockheed Martin
- National Science Foundation
Systems Modeling Language: SysML

- SysML is an extension of the Unified Modeling Language.
- Supports the modeling of physical systems – not just software.
- It is not a methodology or tool.
- Improves the ability to exchange systems engineering information among tools and people.
Pillars of SysML: 4 Main Diagram Types

1. Structure

2. Behavior

3. Parametrics

4. Requirements

(Source: Friedenthal, www.omgsysml.org)

(bdd [package] VehicleStructure [ABS-Block Definition Diagram])

«block» Library::Electronic Processor
«block» Anti-Lock Controller
«block» Traction Detector
«block» Brake Modulator

«block» Library::Electro-Hydraulic Valve

«block» Anti-Lock Controller
«block» Brake Modulator

m1

d1

definition
Pillars of SysML: 4 Main Diagram Types

1. Structure

2. Behavior

interaction

3. Parametrics

4. Requirements

(Source: Friedenthal, www.omgsysml.org)
Pillars of SysML: 4 Main Diagram Types

1. Structure

2. Behavior

4. Requirements

3. Parametrics

(Source: Friedenthal, www.omgsysml.org)
Pillars of SysML: 4 Main Diagram Types

1. Structure

2. Behavior

interaction
state
machine
activity/
function

3. Parametric

4. Requirements

(Source: Friedenthal, www.omgsysml.org)
Cross Connecting Model Elements

1. Structure

```
hibit [block] Anti-LockController
[Internal Block Diagram]
```

```
d1:Traction Detector
m1:Brake Modulator
```

```
c1:modulator interface
```

2. Behavior

```
act PreventLockup [Activity Diagram]
```

```
«allocate» : Traction Detector
«allocate» : Brake Modulator
```

```
allocatedTo
«connector» c: modulator Interface
```

3. Parametrics

```
par [constraintBlock] StraightLineVehicleDynamics [Parametric Diagram]
```

```
: Braking Force Equation
\[ f = (t^t_bf)(1-t_l) \]
```

```
: Acceleration Equation
\[ F = ma \]
```

```
: Distance Equation
\[ v = dx/dt \]
```

```
: Velocity Equation
\[ a = dv/dt \]
```

4. Requirements

```
<requirement> Stopping Distance
id="102"
text="The vehicle shall stop from 60 mph within 150 ft on a clean dry surface."
```

```
<requirement> Anti-Lock Performance
id="337"
text="Braking subsystem shall prevent wheel lock up under all braking conditions."
```

(Source: Friedenthal, www.omg.sysml.org)
Cross Connecting Model Elements

1. Structure

2. Behavior

3. Parametrics

4. Requirements

(Source: Friedenthal, www.omg.sysml.org)
Cross Connecting Model Elements

1. Structure

<table>
<thead>
<tr>
<th>Model Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibd [block]</td>
<td>Anti-LockController [Internal Block Diagram]</td>
</tr>
<tr>
<td>d1:TractionDetector</td>
<td>allocatedFrom «ObjectNode» TractionLoss</td>
</tr>
<tr>
<td>m1:BrakeModulator</td>
<td>allocatedFrom «activity» Modulate BrakingForce</td>
</tr>
<tr>
<td>c1:modulator Interface</td>
<td></td>
</tr>
</tbody>
</table>

2. Behavior

<table>
<thead>
<tr>
<th>Model Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>act PreventLockup [Swimlane Diagram]</td>
<td></td>
</tr>
<tr>
<td>Traction</td>
<td>allocatedFrom «activity» DetectLossOf Traction</td>
</tr>
<tr>
<td>TractionLoss:</td>
<td></td>
</tr>
<tr>
<td>BrakeModulator</td>
<td>allocatedFrom «activity» Modulate BrakingForce</td>
</tr>
<tr>
<td>Anti-LockController</td>
<td>allocatedFrom «ObjectNode» TractionLoss</td>
</tr>
</tbody>
</table>

3. Parametrics

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F = ma )</td>
<td>Braking Force Equation</td>
</tr>
<tr>
<td>( f = (tf<em>bf)</em>(1-tl) )</td>
<td></td>
</tr>
<tr>
<td>( v = dx/dt )</td>
<td>Distance Equation</td>
</tr>
<tr>
<td>( a = dv/dt )</td>
<td>Velocity Equation</td>
</tr>
<tr>
<td>( x )</td>
<td></td>
</tr>
<tr>
<td>( v )</td>
<td></td>
</tr>
<tr>
<td>( f )</td>
<td></td>
</tr>
<tr>
<td>( F )</td>
<td></td>
</tr>
<tr>
<td>( c )</td>
<td></td>
</tr>
<tr>
<td>( tf ), ( tl ), ( bf )</td>
<td></td>
</tr>
</tbody>
</table>

4. Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StoppingDistance</td>
<td>The vehicle shall stop from 60 mph within 150 ft on a clean dry surface.</td>
</tr>
<tr>
<td>Anti-LockPerformance</td>
<td>Braking subsystem shall prevent wheel lockup under all braking conditions.</td>
</tr>
</tbody>
</table>

(Source: Friedenthal, www.omgsysml.org)
Cross Connecting Model Elements

1. Structure

ibd [block] Anti-LockController [Internal Block Diagram]

satisfies <requirement> Anti-Lock Performance

allocatedFrom <ObjectNode> TractionLoss:

allocatedTo <connector> c1:modulatorInterface

values DutyCycle: Percentage

allocate <allocate> :TractionDetector

allocate <allocate> :BrakeModulator

allocateFrom <activity> DetectLossOfTraction

allocateFrom <activity> ModulateBrakingForce

allocatedFrom <ObjectNode> TractionLoss:

d1:TractionDetector

m1:BrakeModulator

satisfy «requirement» Anti-Lock Performance

par [constraintBlock] StraightLineVehicleDynamics [Parametric Diagram]

allocateTo <connector> c1:modulatorInterface

allocateFrom <activity> DetectLossOfTraction

allocateFrom <activity> ModulateBrakingForce

allocatedFrom <ObjectNode> TractionLoss:

2. Behavior

act PreventLockup [Swimlane Diagram]

allocate <allocate> :TractionDetector

allocate <allocate> :BrakeModulator

DetectLossOfTraction

TractionLoss:

ModulateBrakingForce

allocatedFrom <activity> DetectLossOfTraction

allocatedFrom <activity> ModulateBrakingForce

allocatedFrom <ObjectNode> TractionLoss:

4. Requirements

Vehicle System Specification

<requirement> StoppingDistance

id="102" text="The vehicle shall stop from 60 mph within 150 ft on a clean dry surface."

SatisfiedBy <block> Anti-LockController

<deriveReq>

Braking Subsystem Specification

<requirement> Anti-LockPerformance

id="337" text="Braking subsystem shall prevent wheel lockup under all braking conditions."

3. Parametrics

v.chassis.tire. Friction:

v.brake.abs.m1. DutyCycle:

v.brake.rotor. BrakingForce:

v.Weight:

tt: tf:

bt:

ft:

F:

m:

v: a:

:BrakingForce Equation

[v = dx/dt]

[v = dv/dt]

v: F:

:DistanceEquation

:VelocityEquation
Cross Connecting Model Elements

1. Structure

- **ibd** (block) Anti-LockController [Internal Block Diagram]
- **c1**: modulator Interface
- **d1**: TractionDetector
- **m1**: BrakeModulator
- **values**: DutyCycle: Percentage

2. Behavior

- **act** PreventLockup [Swimlane Diagram]
- **allocate** : TractionDetector
- **allocate** : BrakeModulator
- **allocate** : c1: modulator Interface

3. Parametrics

- **par** (constraintBlock) StraightLineVehicleDynamics [Parametric Diagram]

4. Requirements

- **req** (package) VehicleSpecifications [Requirements Diagram - Braking Requirements]

(Source: Friedenthal, www.omgsysml.org)
What is Modelica?
(www.modelica.org)

- State-of-the-art Modeling Language for System Dynamics
  - Differential Algebraic Equations (DAE)
  - Discrete Events

- Represents DAE models in an object-oriented, engineering-oriented language
- Multi-(physical)-domain modeling
- Ports represent energy flow (undirected) or signal flow (directed)
- Acausal, equation-based, declarative (f-m*a=0)
Modelica Semantics and Textual Syntax

Connections represent Kirchhoff semantics
- Across variables (voltage, pressure,...) are equal
- Through variables (current, flow rate,...) add to zero

```
model Spring "Linear 1D translational spring"
  extends Translational.Interfaces.PartialCompliant;
  parameter SI.TranslationalSpringConstant c(final min=0, start = 1)
    "spring constant ";
  parameter SI.Distance s_rel0=0 "unstretched spring length";

  equation
  f = c*(s_rel - s_rel0);
end Spring;
```
Presentation Overview

- What is SysML?
- What is Modelica?

Motivating Example: Design & Analysis of Robot

- SysML-Modelica Transformation Specification
- Transformations in Model-Based Systems Modeling
- Summary
A Robot Example in Modelica
SysML-Modelica Robot Example: Use Cases & Requirements

- Operator
- Maintenance
- Programmer

- Emergency Shutdown
- Program Trajectory
- Execute Robot Task

- Robot System
  - Cost
  - Performance
  - Reliability
  - Energy Consumption
  - Weight
  - Tracking Accuracy

- Block: Robot
- Requirement: Workspace Safety Violation
SysML-Modelica Robot Example: Robot Domain BDD & IBD
SysML-Modelica Robot Example: Robot BDD & IBD
Analysis models depend on descriptive models.
SysML4Modelica Analytical Model: Compose Model from Standard Library

Drag and drop into IBD «ModelicaModel»
SysML4Modelica Analytical Model: Detailed IBD
SysML4Modelica Analytical Model: Relation to Modelica Native Model
SysML-Modelica Robot Example: Modelica model with simulation results
 SysML-Modelica Robot Example: Analysis and Trade Study

Analysis results are incorporated in Trade Study
SysML-Modelica Robot Example: An Analysis Context
SysML-Modelica Robot Example: Dependencies between Models
SysML-Modelica Robot Example: Dependencies between Models
Presentation Overview

- What is SysML?
- What is Modelica?

- Motivating Example: Design & Analysis of Robot

SysML-Modelica Transformation Specification

- Transformations in Model-Based Systems Modeling
- Summary
Model Transformation

- Model Driven Architecture / Engineering
- Transformation Specification = Model → automated generation of transformation engine code
- Tools: MOFLON, QVTo, ATL, Kermeta, VIATRA2, …

(Czarnecki, K., & Hellen, S., 2006)
SysML-Modelica Transformation Specification

SysML-Modelica Transformation follows the principles of Model-Driven Architecture (MDA)
SysML4Modelica Profile
model Spring "Linear 1D translational spring"
   extends Translational.Interfaces.PartialCompliant;
   parameter SI.TranslationalSpringConstant c(final min=0, start = 1)
      "spring constant ";
   parameter SI.Distance s_rel0=0 "unstretched spring length";
   equation
      f = c*(s_rel - s_rel0);
end Spring;

SysML4Modelica

Formal mapping

Modelica
Reference implementation: Based on OMG QVT

QVT = Query / View / Transformation
Presentation Overview

- What is SysML?
- What is Modelica?

- Motivating Example: Design & Analysis of Robot

- SysML-Modelica Transformation Specification

- Transformations in Model-Based Systems Modeling
- Summary
Model Transformations in MBSE

- Model Object
- Model Dependency

System Model
Model Transformations in MBSE: Federated Models
Model Transformations in MBSE

- Requirements
  - Functional/Behavioral Model
  - Performance Model
- Formal System Model
- System Model
- Structural/Component Model
- Other Engineering Analysis Models
- Transformation
- Stage-Gate Documents
- Transformation
- Transformation
- Project Management Metrics
- Transformation
- Verification & Consistency Management
Model Transformations in MBSE: Reusable Models

Requirements

Functional/Behavioral Model

- Start
- Shift
- Accelerate
- Brake

Performance Model

- Control Input
- Power Equations
- Vehicle Dynamics

System Model

- Engine
- Transmission
- Transaxle

Creating Models is Expensive and Time-Consuming
Model Transformations in MBSE: Applying Reuse Patterns

- Physical components are reused
- Portions of the system model repeat
- Patterns for instantiating these portions

- Component models $\rightarrow$ Domain specific model libraries
- Application of pattern $=$ model transformations
Model Transformations in MBSE: Applying Reuse Patterns

- When cylinder is used, other corresponding models are often used also
  - Capture the reuse pattern
Model Transformations in MBSE: Applying Reuse Patterns
Model Transformations in MBSE: Descriptive to Analytical Transformation

SysML
Descriptive

SysML4Modelica
Analytical

Correspondence Models

Descriptive Models

Analytical Models

2008-2011 Copyright © Georgia Tech. All Rights Reserved.
Model Transformations in MBSE: Architecture Exploration

Problem Definition
- Generate Architecture
  - Components
    - SysML
  - Algebraic Models
    - SysML
  - Dynamic Models
    - SysML

Generate Algebraic Design Problem
- Generate Dynamic Design Problem
  - Algebraic Models
  - Dynamic Models

Problem Formulation
- MagicDraw SysML Editor
- Transformation Engine
- GAMS Solver
- Variable Fidelity Model Selection

Problem Solution
- Topology Analysis
- Mixed-Integ Nonlin Solver
- Algebraic Analysis
- Optimization Solver
- Uncertainty Quantification
- Dynamic Analysis
- Design Explorer
- Monte Carlo + Kriging
- Modelica
Summary

- **Objective:**
  - Leverage the strengths of both SysML and Modelica by integrating them to create a more expressive and formal MBSE language.

Descriptive Modeling in SysML

+ Formal Equation-Based Modeling for Analyses and Trade Studies in Modelica

http://www.omg.org/spec/SyM/