Simulating Cyber-Physical Systems using SysML and Numerical Simulation Tools
Acknowledgements

- The tools and methods presented in this tutorial were developed as a joint project of IBM Research - Tokyo and IBM Rational Rhapsody Development Lab in Israel.

- Main contributors:
  - Takashi Sakairi, IBM Research - Tokyo
  - Chaim Cohen, IBM Rational
  - Eldad Palachi, IBM Rational
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACD</td>
<td>Activity Diagram</td>
</tr>
<tr>
<td>BDD</td>
<td>Block Definition Diagram</td>
</tr>
<tr>
<td>CAS</td>
<td>Computer Algebra System</td>
</tr>
<tr>
<td>CPS</td>
<td>Cyber Physical Systems</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain Specific Language</td>
</tr>
<tr>
<td>FMI</td>
<td>Functional Mockup Interface</td>
</tr>
<tr>
<td>IBD</td>
<td>Internal Block Diagram</td>
</tr>
<tr>
<td>INCOSE</td>
<td>International Council of Systems Engineering</td>
</tr>
<tr>
<td>MBSE</td>
<td>Model-Based Systems Engineering</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>PCE</td>
<td>Parametric Constraint Evaluator</td>
</tr>
<tr>
<td>PD</td>
<td>Parametric Diagram</td>
</tr>
<tr>
<td>SysML</td>
<td>Systems Modeling Language</td>
</tr>
<tr>
<td>UCD</td>
<td>Use Case Diagram</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
</tbody>
</table>

© 2012 IBM Corporation
Presentation Outline

- What is a Cyber Physical System?
- A short overview of SysML
- Short overview of Simulink and Modelica
- How to model and simulate physical environments/plant?
- How to model and simulate CPS?
- Summary
Cyber Physical Systems (CPS)

- The term cyber physical system refers to the integration of computation with physical processes.
Typical Cyber Physical System (CPS)

Developing CPS (usually) means doing Systems Engineering

- CPS are commonly complex multi-disciplinary systems:
  - Requires disciplined requirements analysis (functional analysis, trade-offs, etc.)
  - Requires coordination between domain specific engineering teams (managing engineering artifacts)
  - High risk of failure, esp. during integration, non-trivial emergent behavior -> simulations!

Table 1. The Twelve Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>Abbr.</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RO</td>
<td>Requirements Owner</td>
</tr>
<tr>
<td>2</td>
<td>SD</td>
<td>System Designer</td>
</tr>
<tr>
<td>3</td>
<td>SA</td>
<td>System Analyst</td>
</tr>
<tr>
<td>4</td>
<td>VV</td>
<td>Validation/Verification Engr.</td>
</tr>
<tr>
<td>5</td>
<td>LO</td>
<td>Logistics/Ops Engineer</td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>Glue Among Subsystems</td>
</tr>
<tr>
<td>7</td>
<td>CI</td>
<td>Customer Interface</td>
</tr>
<tr>
<td>8</td>
<td>TM</td>
<td>Technical Manager</td>
</tr>
<tr>
<td>9</td>
<td>IM</td>
<td>Information Manager</td>
</tr>
<tr>
<td>10</td>
<td>PE</td>
<td>Process Engineer</td>
</tr>
<tr>
<td>11</td>
<td>CO</td>
<td>Coordinator</td>
</tr>
<tr>
<td>12</td>
<td>CA</td>
<td>Classified Ads SE</td>
</tr>
</tbody>
</table>

Presentation Outline

- What is a Cyber Physical System?
- A short overview of SysML
- Short overview of Simulink and Modelica
- How to model and simulate physical environments/plant?
- How to model and simulate CPS?
- Summary
The Systems Modeling Language (SysML)

- SysML is a general purpose domain-specific language (DSL) for model-based systems engineering (MBSE)
- Originated as an initiative of the International Council of Systems Engineering (INCOSE) in January 2001
- Developed by Object Management Group (OMG)
- Current version: 1.3 (http://www.omg.org/spec/SysML/1.3/) – released in 2012
- Implemented as a UML profile
  - Allows easier hand-off for software engineering
- Subsets and extends UML

Source: SysML 1.3 Spec.
Simulating Cyber-Physical Systems using SysML and Numerical Tools

Requirement Diagram

- Introduced in SysML
  - Along with the notion of Requirement
- Describes a hierarchy of textual requirements
- Allows tracing from design elements to requirements using various standard relationships (satisfy, derived, etc.)
- SysML also supports tabular notations for requirements
- Requirements might be imported from requirements management tools

![Requirement Diagram](image-url)
Use Case Diagram (UCD)

- Same as in UML
- Lists the capabilities of the systems (use cases) and how they relate to actors and to each other
- Use Cases are commonly considered as the “chapters” of the SysML model
**Block Definition Diagram (BDD)**

- Based on UML Class Diagram
- Describes the system Blocks and their features (structural and behavioral)
Internal Block Diagram (IBD)

- Based on UML Internal Structure Diagram
- Describes the internal structure of Blocks
- Main concepts: Part, Connector, Port (various kinds)
Simulating Cyber-Physical Systems using SysML and Numerical Tools

**Parametric Diagram**

- Introduced in SysML
- Imposes mathematical constraints on properties of Blocks (in system’s context)
- Main concepts:
  - Constraint Block: groups non causal mathematical expressions (equations/inequalities)
  - Constraint Parameter: a variable of the math expressions that can be bounded to a design property
  - Constraint Property: a usage of a constraint block in a specific context
  - Binding Connector: declared that the value of the design property must be equal to the value of the constraint parameter
**Activity Diagram**

- Same as in UML 2 with a few extensions (mainly continuous flows)
- Token based semantics:
  - Every action has input/output pins through which control/object tokens flow
  - An action starts when all pins have tokens
  - As action offers its output tokens after its computation is done
- Commonly describes behavior of:
  - Use Cases: system level processes
  - Operations: block level procedures ("flow charts")
- Continuous semantics is not precise
- Not well-suited for describing dynamic time-dependent processes
State Machine Diagram (statechart)

- Same as in UML
- Describes reactive state based behavior
- Run-to-completion semantics:
  - One event at a time
  - A run-to-completion computation step takes the state machine from one stable state to another
    - Stable state: a concrete state configuration of the classifier after all internal actions are completed
- Commonly used to describe behavior of Blocks
Sequence Diagram

- Same as in UML
- Describes a specific interaction scenarios between parts
- Can be captured by model execution
- Commonly used to describe test cases in model-based testing (UML Testing Profile)
SysML overview summary

- Standardized language for model-based systems engineering
- Subsets and extends UML
- Consists of multiple diagram types to describe various aspects of the system
  - Requirements (requirement diagram, use case diagram)
  - Structure (block definition diagram, internal block diagram)
  - Constraints (parametric diagram)
  - Behavior (activity diagram, state machine diagram)
  - Interaction Traces (sequence diagrams)
- Lacks precise execution semantics
  - Not well suited for modeling plant and continuous control algorithms
- Parametric diagrams can be used to describe equations imposed on design attributes
Presentation Outline

- What is a Cyber Physical System?
- A short overview of SysML
- Short overview of Simulink and Modelica
- How to model and simulate physical environments/plant?
- How to model and simulate CPS?
- Summary
Simulating Cyber-Physical Systems using SysML and Numerical Tools

MATLAB Simulink

- Designed to model and simulate time-dependent transformational behavior
- Based on Block Diagrams
  - Every block transforms input signals to output signals
  - Signal: a time dependent function of a physical quantity
    - E.g. Amplitude of a radio signal, Voltage of an AC outlet, Car speed
- Simulink is widely used for control and signal processing designs
- Developed by Mathworks for modeling “multi-domain dynamic systems”
- Add-on to MATLAB
- For reactive behavior an add-on called Stateflow is used on top of Simulink
- Other add-ons:
  - Embedded coder: generate C and C++ code from models for real-time targets
  - HDL coder: generates VHDL
  - Simulink Design Verifier
Modelica

- A standardized textual language for modeling physical systems
  - Annotations are also standardized now and can be used to render diagrams

- Developed since 1996 by the Modelica Association [https://www.modelica.org](https://www.modelica.org) – Current version 3.3 (May 2012)

- Modelica is Object-Oriented (see right side)

- Has a large (~30) set of free and commercial libraries for different domains (source: [https://www.modelica.org/ModelicaLibrariesOverview](https://www.modelica.org/ModelicaLibrariesOverview))

- Implemented by various free and commercial tools: ([Dymola](https://www.esym.com/products/dymola), [Open Modelica](https://www.openmodelica.org), [Math Modelica](https://www.mathmodelica.com))

- The [OMG SysML4Modelica](https://www.omg.org/spec/SysML/4Modelica/) profile extends SysML to model Modelica constructs in SysML (IBD) and roundtrip Modelica models back to SysML

```model Circuit
Resistor R1 (R=10), R2 (R=100);
Capacitor C (C=0.01);
Inductor L (L=0.1);
VsourceAC AC;
Ground G;

equation
connect(AC.p, R1.p); // Capacitor circuit
connect(R1.n, C.p);
connect(C.n, AC.n);
connect(R1.p, R2.p); // Inductor circuit
connect(R2.n, L.p);
connect(L.n, C.n);
connect(AC.n, G.p);
end Circuit;
```

Simulating Cyber-Physical Systems using SysML and Numerical Tools

Simulink vs. Modelica

Simulink

- Transformational semantics of signals
  - Language is well-suited for control algorithms and signal processing
- Causal semantics (inputs -> outputs)
- Mature: more features, highly usable, a lot of add-ons
- Well integrated into the “MATLAB universe”
- Widely used in industry (standard de-facto)
- Many existing tool integrations
- Code generation to C/C++/VHDL/Verilog

Modelica

- Object oriented approach for modeling physical components (mechanical, electrical, etc.)
  - Language is better suited for physical modeling (plant)
- Causal and A-Causal semantics (equations)
- Open standard (of the textual language) with collaboration in research and standard bodies
  - SysML for Modelica profile
- Multi tool support (although Dymola is dominant)
  - Tools are less mature than Simulink
- Few industry deployments but a lot of interest
- No export regulation restrictions (relevant for defense systems development)
- Currently no production code generation (to the best of our knowledge)
Presentation Outline

- What is a Cyber Physical System?
- A short overview of SysML
- Short overview of Simulink and Modelica
- How to model and simulate physical environments/plant?
- How to model and simulate CPS?
- Summary
Describing plant behavior

- Plant is the physical environment in which the CPS operates
- Plants can be described using partial time-dependent differential equations
  - These equations are a-causal: no notion of inputs and outputs
- “Hello, World” example – one dimensional spring and mass performing harmonic oscillation

\[ m \cdot a = -k \cdot x \quad \text{Analytic Solution: } x = A \cdot \sin(\omega \cdot t + \varphi) \]
\[ a = \dot{v} = \ddot{x} \]
\[ x(t = 0) = 5 \]
\[ v(t = 0) = 0 \]
\[ m = 10 \]
\[ k = 3 \]
\[ \omega = \frac{\sqrt{k}}{m} \]
\[ \omega = \sqrt{3/10} = 0.577 \]
\[ \varphi = \frac{\pi}{2} \]
\[ A = 5 \]
Modeling spring and mass dynamics using Modelica

```modelica
model Harm1
  annotation(Figure:visible = true, transformation({
    Modelica.Mechanics.Translational.Components.Mass mass1(m = 10, v(start = 0), x(start = 5))
    annotation(Figure:experiment(StartTime = 0.0, StopTime = 60.0, Tolerance = 1e-006))
  });

  Modelica.Mechanics.Translational.Components.Spring spring1(c = 3, s = m10 = 10)
  annotation(Figure:visible = true, transformation({
    equation
      connect(mass1.flange_a[-1], spring1.flange_b[1-1])
      annotation(Figure:Line(points = [(41.7482, 9.33522), (14.71, 9.33522),
        connect(spring1.flange_a[-1], fixed1.flange_b[-1])
        annotation(Figure:Line(points = [(-19.9082, 8.46608), (-57.7086, 9.46608),
         connect(spring1.flange_a, fixed1.flange_b)
         annotation(Figure:Line(points = [(-20.1152, 9.05233), (-57.7086, 9.05233), (-57.7,
         connect(mass1.flange_a, spring1.flange_b)
         annotation(Figure:Line(points = [(41.7482, 9.33522), (14.71, 9.33522), (14.71, 9.

end Harm1;
```
Modeling spring and mass dynamics using Simulink

\[ m \cdot a = -k \cdot x \]
\[ a = \dot{v} = \ddot{x} \]
\[ x(t = 0) = 5 \]
\[ v(t = 0) = 0 \]
\[ m = 10 \]
\[ k = 3 \]
Modeling spring and mass dynamics using SysML parametric diagram
IBM Rational Rhapsody is a modeling tool capable of specifying and executing SysML models. PCE is an add-on to Rhapsody that allows solving sets of math expressions specified in parametric diagrams and update the design model based on the results.
Plant modeling summary

- We have shown three simulations of a simple harmonic oscillator using three languages and three tools:
  - Modelica (Open Modelica)
  - MATLAB Simulink
  - SysML Parametric Diagrams (Rhapsody PCE)
- PCE relies on having a symbolic solver (CAS)
Presentation Outline

- What is a Cyber Physical System?
- A short overview of SysML
- Short overview of Simulink and Modelica
- How to model and simulate physical environments/plant?
- How to model and simulate CPS?
- Summary
Extending SysML: «SimulinkBlock»

- The stereotype «SimulinkBlock» means the block’s behavior is specified in a Simulink model.
- Every input/output port in the Simulink model is represented as a SysML atomic flow port.
- Type matching rules need to be applied.
Extending SysML: «StructuredSimulinkBlock»

- The stereotype «StructuredSimulinkBlock» means the block has parts typed by Simulink blocks
  - A block that owns a part typed by a «StructuredSimulinkBlock» is also a «StructuredSimulinkBlock»
- A «StructuredSimulinkBlock» can be exported to Simulink for simulation
  - All non Simulink blocks are transformed to a single S-Function in Simulink
Simulating Cyber-Physical Systems using SysML and Numerical Tools

Simulink S-Function

- MATLAB/Simulink S-Function is a user defined block implemented in C/C++ or other programming language.
  - The S-Function code must conform to the S-Function standard in order for Simulink to understand its interfaces and to interact with it.

- Rhapsody can generate C/C++ code corresponding to blocks stereotyped «S-FunctionBlock» along with a mex option file to generate an S-Function simulink block.
  - The generated code conforms to the S-Function standard and transforms the ports accordingly.

- For more information on S-Function see http://www.mathworks.com/help/simulink/s-function-basics.html
Exchanging behavior via generated code

- Our approach uses generated C/C++ code to generate behavior of blocks brought to the simulator

- «SimulinkBlock» may reference C/C++ code generated by MATLAB Embedded Coder
  - This code is compiled with the rest of the code into an executable used by Rhapsody simulation

- «StructuredSimulinkBlock» is transformed to a Simulink model with an auto-generated S-Function Block that encapsulates the behavior of the native SysML blocks

- Modelica has adopted the Functional Mockup Interface (FMI) standard (see https://www.fmi-standard.org/) to exchange behavior using generated C code
  - Unlike S-Function, FMI is non-proprietary
Demo: Modeling a Cruise Control Vehicle

- We will demonstrate two simulations of a cruise control system of a vehicle:
  - Exporting a hierarchy of IBDs to Simulink and running the simulation in Simulink
  - Importing Simulink models + generated code to Rhapsody and building an executable that can be simulated

- We will use the same two Simulink models in both simulations
  - A model specifying a proportional feedback setting the thrust based on the difference between the desired and actual speeds
  - A model simulating the drag force exerted on the vehicle

- SysML will be used to describe the composition of the system
  - The variant between the two SysML models is very small (will be explained)
Simulating Cyber-Physical Systems using SysML and Numerical Tools

Proportional Feedback in Simulink

\[
V_{\text{diff}} = \frac{V_{\text{desired}} - V_{\text{actual}}}{3.6}
\]

\[
\text{MaxThrust} = \left\lfloor \frac{147649}{V_{\text{actual}} / 3.6} \right\rfloor
\]

\[
\text{Th} = Kp \cdot V_{\text{diff}} + Kr \cdot \int V_{\text{diff}} \, dt
\]

\[
\text{Thrust} = \begin{cases} 
\text{MaxThrust} & \text{Th} > \text{MaxThrust} \\
\text{Th} & \text{MaxThrust} \geq \text{Th} \geq -\text{MaxThrust} \\
-\text{MaxThrust} & \text{Th} < -\text{MaxThrust}
\end{cases}
\]
Drag

- Thrust is converted to speed in m/sec, and then converted to Km/h
- Drag force is proportional to the square of the velocity

\[ F_D = \frac{1}{2} \rho v^2 C_d A, \]

where
- \( F_D \) is the force of drag,
- \( \rho \) is the density of the fluid,\(^{[12]} \)
- \( v \) is the speed of the object relative to the fluid,
- \( C_d \) is the drag coefficient (a dimensionless parameter, e.g. 0.25 to 0.45 for a car)
- \( A \) is the reference area,

Demo 1: Vehicle Cruise Control, Simulation in Simulink
Demo 1: Vehicle Cruise Control, Simulation in Simulink
Demo 2: Vehicle Cruise Control, Simulation in Rhapsody

 cpp / .h

 .cpp / .h

 cpp / .h
Demo 2: Vehicle Cruise Control, Simulation in Rhapsody
Simulating Cyber-Physical Systems using SysML and Numerical Tools

Demo summary: usage on the “V-Model”

- Requirements Analysis
- Functional Decomposition
- Design Synthesis
  - Trade Study
  - Component/Subsystem Spec.
- Systems Eng.

- Design
- Analysis
- Implementation
- Software Eng.

- Parametric Constraint Evaluation (PCE)*
- Simulation in Simulink
- Simulation in Rhapsody (algorithmic integration)

* Doing trade studies with PCE is not shown in this tutorial
Presentation Outline

- What is a Cyber Physical System?
- A short overview of SysML
- Short overview of Simulink and Modelica
- How to model and simulate physical environments/plant?
- How to model and simulate CPS?
- Summary
Tutorial Summary

- CPS are computerized systems the operate and interact with physical environment
- Complex CPS designs require systems engineering techniques involving modeling and simulations
- SysML is a standardized modeling language defining multiple diagram types to describe various aspects of systems
- SysML needs to be complemented by tools/languages such as Simulink/Modelica to specify continuous algorithms and plant behavior
- We showed how to model physical environments using Simulink, Modelica and SysML Parametric Diagrams
- We demonstrated three integration points between SysML and numerical tools:
  - Evaluation of parametric diagrams using Rhapsody PCE
  - Simulation of structured block by exporting it to Simulink for simulation
  - Simulation of hybrid models in Rhapsody using code generated from Simulink models
Some References and Further Readings

- Sakairi T., Palachi E., *Leveraging SysML parametric diagrams to perform trade studies and other quantitative analysis*, The Voice of the Systems, Vol 9, January 2012 (http://www.iltam.org/incose_il/Kol_hamaarahot9/)
- Simulink: http://www.mathworks.com/products/simulink/
- Open Modelica: https://openmodelica.org/
- FMI web site: https://www.fmi-standard.org

mailto:eldad.palachi@il.ibm.com