Modeling Cyber-Physical Systems: Challenges and Recent Advances

Gabor Karsai
Institute for Software-Integrated Systems
Vanderbilt University

Model-Driven Development for Distributed Real-time Embedded Systems - 9/5/2014
Acknowledgements

- Janos Sztipanovits
- Ted Bapty
- Abhishek Dubey
- Larry Howard
- Xenofon Koutsoukos
- Zsolt Lattmann
- Tihamer Levendovszky
- Adam Nagel
- Sandeep Neema
- Joseph Porter
- Gabor Simko
- .... and many others at Institute for Software-Integrated Systems @ Vanderbilt University
Modeling CPS

- Definition
- Examples
- The three aspects of modeling
  - Modeling the physical system
  - Models of computation and communication
  - Modeling the platform
- Model integration
- Recent results
- Research challenges
- Conclusions
What is a Cyber-Physical System?

- An engineered system that *integrates* physical and cyber components where relevant functions are realized through the *interactions* between the physical and cyber parts.
  - Physical = some tangible, physical device + environment
  - Cyber = computational + communicational
## CPS Examples

<table>
<thead>
<tr>
<th>Sectors</th>
<th>Opportunities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Health and Biomedical</strong></td>
<td>In-home healthcare delivery. More capable biomedical devices for measuring health. New prosthetics for use within and outside the body. Networked biomedical systems that increase automation and extend the biomedical device beyond the body.</td>
</tr>
<tr>
<td><strong>Smart Grid</strong></td>
<td>Highway systems that allow traffic to become denser while also operating more safely. A national power grid that is more reliable and efficient.</td>
</tr>
</tbody>
</table>
## CPS Examples

<table>
<thead>
<tr>
<th>Sectors</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace</td>
<td>• Aircraft that fly faster and further on less energy.</td>
</tr>
<tr>
<td></td>
<td>• Air traffic control systems that make more efficient use of airspace.</td>
</tr>
<tr>
<td>Automotive</td>
<td>• Automobiles that are more capable and safer but use less energy.</td>
</tr>
<tr>
<td></td>
<td>• Highways that are safe, higher throughput and energy efficient.</td>
</tr>
<tr>
<td>Defense</td>
<td>• Fleets of autonomous, robotic vehicles</td>
</tr>
<tr>
<td></td>
<td>• More capable defense systems</td>
</tr>
<tr>
<td></td>
<td>• Integrated, maneuverable, coordinated, energy efficient</td>
</tr>
<tr>
<td></td>
<td>• Resilient to cyber attacks</td>
</tr>
</tbody>
</table>
The Good News…

Networking and computing delivers unique precision and flexibility in interaction and coordination

Computing/Communication

- Rich time models
- Precise interactions across highly extended spatial/temporal dimension
- Flexible, dynamic communication mechanisms
- Precise time-variant, nonlinear behavior
- Introspection, learning, reasoning

Integrated CPS

- Elaborate coordination of physical processes
- Hugely increased system size with controllable, stable behavior
- Dynamic, adaptive architectures
- Adaptive, autonomic systems
- Self monitoring, self-healing system architectures and better safety/security guarantees.
Computing/Communication

- Cyber vulnerability
- New type of interactions across highly extended spatial/temporal dimension
- Flexible, dynamic communication mechanisms
- Precise time-variant, nonlinear behavior
- Introspection, learning, reasoning

Integrated CPS

- Physical behavior of systems can be manipulated
- Lack of composition theories for heterogeneous systems: much unsolved problems
- Vastly increased complexity and emergent behaviors
- Lack of theoretical foundations for CPS dynamics
- Verification, certification, predictability has fundamentally new challenges.
Abstraction layers allow the verification of different properties.

Key Idea: Manage design complexity by creating abstraction layers in the design flow.

Abstraction layers define platforms.

Abstractions are linked through mapping.

Abstraction layers allow the verification of different properties.

Example for a CPS Approach
Software models

\[ f : [T \rightarrow In] \rightarrow 2^{[r \rightarrow Out]} \]

Real-time system models

\[ f_R : [T_R \rightarrow In] \rightarrow 2^{[r_R \rightarrow Out]} \]


In CPS, essential system properties such as stability, safety, performance are expressed in terms of physical behavior

- \( f \): reactive program. Program execution creates a mapping between logical-time inputs and outputs.

- \( f_R \): real-time system. Programs are packaged into interacting components. Scheduler control access to computational and communicational resources according to time constraints \( P \)

\[ \forall \rho \in E, \forall \pi \in f_R(\rho), (\rho, \pi) \in P \]
Abstraction layers: PHY-SW-RTS

Physical models
\[ p_R : [T_R \rightarrow In] \rightarrow 2^{[T_R \rightarrow Out]} ; f_R : [T_R \rightarrow In] \rightarrow 2^{[T_R \rightarrow Out]} \]

Software models
\[ f : [T \rightarrow In] \rightarrow 2^{[T \rightarrow Out]} \]

Real-time system models
\[ f_R : [T_R \rightarrow In] \rightarrow 2^{[T_R \rightarrow Out]} \]

Re-defined Goals:

- Compositional verification of essential dynamic properties
  - stability
  - safety

- Derive dynamics offering robustness against implementation changes, uncertainties caused by faults and cyber attacks
  - fault/intrusion induced reconfiguration of SW/HW
  - network uncertainties (packet drops, delays)

- Decreased verification complexity

\[ \forall \rho \in E, \forall \pi \in f_R(\rho), (\rho, \pi) \in P \]
Why is CPS Hard?

Crosses Interdisciplinary Boundaries

- Disciplinary boundaries need to be realigned
- New fundamentals need to be created
- New technologies and tools need to be developed
- Education need to be restructured
CPS and Model-based Design

Design of CPS layers via MDE

- Software models
- Platform models
- Physical models

**Challenge:** How to integrate the models so that cross-domain interactions can be understood and managed?
Model Integration for CPS

- **Issues**
  - Cyber models are insufficient, physical models are insufficient
  - Many modeling paradigms for physical systems (consider engineering or physics!)

- **Universal modeling language with precisely defined semantics?**
  - All models are abstractions of reality from a specific point of view for a specific purposes. Universality is not pragmatic.

- **Universal modeling language with no/sparse semantics?**
  - [SysML] Enabler but not a solution; needs content semantics
Model Integration for CPS

- **Objective:** To support the model-based design of CPS
  - Represent the design: both physical and cyber, and the interface
  - Allow analysis of the design
    - Simulation-based evaluation and V&V
    - Discovering unintended interactions
    - Formal verification
  - Drive the implementation of the design
    - Compile to code, drive the fab

**Key:** understanding cross-domain interfaces and interactions
Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
    - Basic concepts: Vehicle Forge
    - Basic concepts: OpenMETA
  - Information Architecture Challenge
  - OpenMETA Design Flow Integration Challenge
  - Semantic Integration Challenge
    - Structural Semantics
    - Behavioral Semantics
A major DARPA program (a decade after MoBIES):
End-to-end model- and component-based design and integrated manufacturing of a new generation of vehicles; a complex, real-life cyber-physical system. From infrastructure to manufactured vehicle prototype in five yeas (2010-2014).

Engineering/economic goals:
• Decrease development time by 80% in defense systems (brings productivity consistent with other industries)
• Enable the adoption of fabless design and foundry concept in CPS
• “Democratize” design by open source tool chain, crowed-sourced model library and prize-based design challenges
**AVM Scientific Challenge**

- **Achieve AVM goals by pushing the limits of “correct-by-construction” design using**
  - **Model-based Technologies**
    - Computational models that predict properties of cyber-physical systems “as designed” and “as built”.
    - **Challenge:** Develop domain-specific abstraction layers for complex CPS that are evolvable, heterogeneous, yet semantically sound and supported by tools.
  - **Component-based Technologies**
    - Reusable units of knowledge (models) and manufactured components.
    - **Challenge:** Go beyond interoperability – find opportunities for composition where system-level properties can be computed from the properties of components
Technical Areas

Model Library; Curation

Curated Components

Vehicle Forge

Analysis Components, Designs, Design Spaces

OpenMETA Tools

Produces Design Data

MFG Feedback

Use Tools

Collaborate Using VF

FANG Competitors

FANG Competition Coordination

Requirements, Test Benches

Seed Designs, Scores

Produced Design Data

Foundry
Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
    - Collaborative environment: Vehicle Forge
    - Engineering environment: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics
Interface to OpenMETA: VehicleForge

**Components**
- Component discovery interface based on taxonomical- and faceted search
- Component view/visualization

**Design Projects**
- Self-provisioned collaboration tools
  - Wiki,
  - Discussion Forum,
  - Issue tracking for managing team work.
- Git/SVN repositories for design artifacts
- Project and tool-based permission control
- Notification and Messaging system (in e-mail or as Dashboard messages)
- Set of available tools is extensible

**Designers**
- Public profile to show recent activities and involvement in design projects
- Designer portfolio publishing résumé and for self-promotion
- Find designers based on expertise and résumé
- Private profile for customizing account and notification settings
- User dashboard showing feeds of activities from projects, public/private messages from other users, announcements from forge-message channels
VehicleForge Gateway
Service Integration Platform

Browser-based
- Coordination and Monitoring Tools
- Design-space Evaluation and Visualizers
- Team-collaboration Tools
- Component Discovery and Subscription
- Service and Resource Allocation

http://vehicleforge.org/
Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics
AVM Components

Component-based:
- Physical
- Cyber
- Cyber-Physical

Model-based:
- Model-Integrated
- Design and
- Manufacturing Process

Components span:
- Multiple physics
- Multiple domains
- Multiple tools
Caterpillar C9 Diesel Engine: AVM Component

**High-Fidelity Modelica Dynamics Model**
- Rotational Power Port
- Signal Port

**Low-Fidelity Modelica Dynamics Model**
- Rotational Power Port
- Signal Port

**Bond Graph Dynamics Model**
- Rotational Power Port
- Signal Port

**Detailed Geometry Model (CAD)**
- Structural Interface
- Structural Interface

**FEA-Ready CAD Model**
- Structural Interface
- Structural Interface

**Structural Interface**
- Named datums
- Surface/axis/point
- Mapped to CAD

**Power Interfaces**
- Acausal
- Physical phen. (torque/angle...)
- Power flow

**Signal Interfaces**
- Causal/directional
- Logical connect.
- No power transfer

**Param./Property Interfaces**
- Characterize
- Configure

**Dynamics**

**FEA Geometry**

**Detailed Geometry**

- Weight: 680 kg
- Height: 1070 mm
- Length: 1245 mm
- Width: 894.08 mm
- Number of Cylinders: 6
- Maximum Power: 330 kW
- Maximum RPM: 2300 rpm
- Minimum RPM: 600 rpm

- Height: 1070 mm
- Width: 894.08 mm
- Number of Cylinders: 6
- Maximum Power: 330 kW
- Maximum RPM: 2300 rpm
- Minimum RPM: 600 rpm
## Components, Designs, Design Spaces

<table>
<thead>
<tr>
<th>Components</th>
<th>Designs</th>
<th>Design Spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Self-contained</strong> building block</td>
<td><strong>Instantiate</strong> and <strong>connect</strong> Components</td>
<td>Sets of <strong>parameterized architectures</strong></td>
</tr>
<tr>
<td><strong>Properties and Parameters</strong></td>
<td>Parameters, behaviors, geometry are <strong>composed</strong></td>
<td>Extended around <strong>seed designs</strong></td>
</tr>
<tr>
<td>Wrapper for <strong>detailed domain models</strong></td>
<td>Can be wrapped as a <strong>component</strong></td>
<td>Shaped by <strong>design and manufacturability constraints</strong></td>
</tr>
<tr>
<td>Aggregates the domain interfaces into a <strong>single set of component interfaces</strong></td>
<td>Aggregates the component interfaces into a <strong>single set of system interfaces</strong></td>
<td>Accumulates, evolves <strong>design and manufacturing knowledge</strong></td>
</tr>
</tbody>
</table>
## Design Flow

<table>
<thead>
<tr>
<th>Architecture Design</th>
<th>Integrated Multi-physics/Cyber Design</th>
<th>Detailed Design</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modeling</strong></td>
<td><strong>Modeling</strong></td>
<td><strong>Modeling</strong></td>
</tr>
<tr>
<td><strong>Exploration</strong></td>
<td><strong>Simulation</strong></td>
<td><strong>Analysis</strong></td>
</tr>
<tr>
<td><strong>V&amp;V</strong></td>
<td><strong>Modeling</strong></td>
<td><strong>Detailed Manuf. Modeling</strong></td>
</tr>
<tr>
<td><strong>Simulation</strong></td>
<td><strong>V&amp;V</strong></td>
<td><strong>RT SW modeling</strong></td>
</tr>
<tr>
<td><strong>Modeling</strong></td>
<td><strong>Analysis</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Rapid exploration
- Design Space + Constraint Constraint Modeling
- Architecture Modeling
- Static Component Modeling (multiphysics)

### Exploration with integrated optimization and V&V
- Design Space + Behavioral Constraint Modeling
- Architecture Modeling
- Dynamics Modeling (multiple abstractions and multiphysics)
- CAD/assembly modeling
- Coarse Manufacturing Constraint Modeling

### Deep analysis
- Architecture Modeling
- Detailed Domain Modeling
  - CAD
  - FEA; thermal, fluid...
  - Surrogate gen.
- Detailed Manuf. Modeling
- RT SW modeling
Using each component’s mappings to detailed domain models, system-level analyses is automatically composed.

- Static properties
- Multi-physics dynamics
- Geometry
- FEA

META Test Benches provide an analysis context, including stimulus, loading, and monitoring.

Test Benches include algorithms to produce Metrics, which are used to evaluate the design against Requirements.

META Design Models are mapped to these Test Benches.

Design Spaces can also be mapped to Test Benches, enabling rapid evaluation of a family of point designs.
Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics
## Design Flow Spans
### Heterogeneous Modeling Domains

<table>
<thead>
<tr>
<th>Architecture Design</th>
<th>Integrated Multi-physics/Cyber Design</th>
<th>Detailed Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Exploration</td>
<td>Modeling Simulation V&amp;V</td>
<td>Modeling Analysis</td>
</tr>
</tbody>
</table>

- **Rapid exploration**
- **Exploration with integrated optimization and V&V**

- **Design Space + Constraint Modeling**
- **Architecture Modeling**
- **Static Component Modeling (multiphysics)**

- **Design Space + Behavioral Constraint Modeling**
- **Architecture Modeling**
- **Dynamics Modeling (multiple abstractions and multiphysics)**
- **CAD/assembly modeling**
- **Coarse Manufacturing Constraint Modeling**

- **Architecture Modeling**
- **Detailed Domain Modeling**
  - **CAD**
  - **FEA; thermal, fluid…**
  - **Surrogate gen.**
- **Detailed Manuf. Modeling**
- **RT SW modeling**

### Domain Specific Modeling Languages
Modeling Domains

Key META Challenge:
Modeling cross-domain interactions
Information Flows Across Program Components

Model Library; Curation

Competition Coordination

Vehicle Forge

OpenMETA Tools

Foundry

- Component Model
- Design, Design Space, Test Bench Models
- Component, Design, Test Bench Models
- Use cases/Scenarios
- META/MFG Interface

Curated Components

Requirements, Test Bench Models

Seed Designs, Scores

Analysis

Components, Designs, Design Spaces

Produces Design Data

MFG Feedback

Uses Tools

Competitors

Collaborates Using VF
Information Architecture Challenges

- *Shared* conceptualization
- Semantically *sound* modeling languages
- Integration of *many* tools and their modeling languages

**Shared conceptualization**
(Vocabularies, ontologies)

**Integrated Modeling Languages**

- CyPhyML
- HBG
- SignalFlow
Information Architecture Challenges

How Should We Choose Vocabularies, Ontologies?
- Could not find standards covering even smaller part of the AVM domain…
- Grow and evolve vocabularies/ontologies during model library development
- Adopt vocabularies as defined by integrated tools (such as Modelica)

How Should We Choose Modeling Language(s)?
- Define yet another modeling language?
- Choose one that already exists and broad enough to cover the design domain?
- Create a new standard or update an old one?

Unintended consequences
- What are the implications on tools?
- How about “my freedom of abstractions”?
- What is the language evolution trajectory?
The Case for Model Integration Languages...

**Model Integration Language - CyPhy**
- Hierarchical Ported Models /Interconnects
- Structured Design Spaces
- Model Composition Operators

**Domain Specific Tools and Frameworks**
- MATLAB, Simulink
- Pro-E
- Calculix
- SAL
- Dymola
- MSC Software
- Delta3D

**Semantic Backplane**
- Semantic Interface
- Semantic Translators
  - CyPhy ↔ SL/SF
  - CyPhy ↔ SEER
  - CyPhy ↔ CAD

Model-Based Design

**Domain Specific Design Automation Environments:**
- Automotive
- Avionics
- Sensors...

**Tools:**
- Modeling
- Analysis
- Verification
- Synthesis

**Challenges:**
- Cost of tools
- Benefit only narrow domains
- Islands of Automation

**Key Idea:** Use models in domain-specific design flows and ensure that final design models are rich enough to enable production of artifacts with sufficiently predictable properties.

**Impact:** Significant productivity increase in design technology

---

Mathematical and physical foundations
**Metaprogrammable Design Tools:** “Freedom of Abstractions”

**Domain Specific Design Automation Environments:**
- Automotive
- Avionics
- Sensors…

**Key Idea:** Ensure reuse of high-value tools in domain-specific design flows by introducing a *metaprogrammable* tool infrastructure.

**VU-ISIS implementation:** Model Integrated Computing (MIC) tool suite ([http://repo.isis.vanderbilt.edu/downloads/](http://repo.isis.vanderbilt.edu/downloads/))

---

**Metaprogrammable Tool Infrastructure**
- Model Building
- Model Transf.
- Model Mgmt.
- Tool Integration

**Explicit Semantic Foundation**
- Structural
- Behavioral

**Semantic Foundation Component Libraries**
- Metaprogrammable Tools, Environments
- Domain-Specific Environments
- Design Requirements
- Production Facilities

**Meta Layer**
- Meta**: Ensure reuse of high-value tools in domain-specific design flows by introducing a *metaprogrammable* tool infrastructure.

**VU-ISIS implementation:** Model Integrated Computing (MIC) tool suite ([http://repo.isis.vanderbilt.edu/downloads/](http://repo.isis.vanderbilt.edu/downloads/))
## OpenMETA
### Information Architecture

#### Models and Modeling Languages

<table>
<thead>
<tr>
<th>Design Data Package (DDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component Model</strong></td>
</tr>
<tr>
<td><strong>Design Model</strong></td>
</tr>
<tr>
<td><strong>Design Space Model</strong></td>
</tr>
<tr>
<td><strong>Requirement Model</strong></td>
</tr>
<tr>
<td><strong>Result Package</strong></td>
</tr>
</tbody>
</table>

- **CyPhy Model Integration Language**
- **Embedded System Modeling Language (ESMOL)**
- **Modelica**
- **DESERT**
- **CAD**
- **FEA**
- **Relational Abstraction**
- **Probab. Analysis (PCC)**
- **Fault Modeling**

#### Standardized Vocabularies and Core Types

- **META Ontologies**
- **VehicleForge Ontology**
- **iFAB Ontology**

- **Interface & Composition Vocabulary**
- **Behavior Vocabulary**
- **Testing Vocabulary**
- **Vehicle Component Vocabularies**

<<note>>
In progress. Currently includes characterizations of supplier data (unknown source for this vocab)
Summary of OpenMETA – Approach to Information Architecture

- Model-Integration Language: *CyPhyML*
- Use of Metaprogrammable tools *(MIC Tool Suite of ISIS/Vanderbilt)*
- Use of *Semantic Integration* (see later)
Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics
# Design Flow

<table>
<thead>
<tr>
<th>Architecture Design</th>
<th>Integrated Multi-physics/Cyber Design</th>
<th>Detailed Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling Exploration</td>
<td>Modeling Simulation V&amp;V</td>
<td>Modeling Analysis</td>
</tr>
<tr>
<td><strong>Rapid exploration</strong></td>
<td><strong>Exploration with integrated optimization and V&amp;V</strong></td>
<td><strong>Deep analysis</strong></td>
</tr>
<tr>
<td>• Design Space + Constraint Modeling</td>
<td>• Design Space + Behavioral Constraint Modeling</td>
<td>• Architecture Modeling</td>
</tr>
<tr>
<td>• Architecture Modeling</td>
<td>• Architecture Modeling</td>
<td>• Detailed Domain Modeling</td>
</tr>
<tr>
<td>• Static Component Modeling (multiphysics)</td>
<td>• Dynamics Modeling (multiple abstractions and multiphysics)</td>
<td>• CAD</td>
</tr>
<tr>
<td></td>
<td>• CAD/assembly modeling</td>
<td>• FEA; thermal, fluid…</td>
</tr>
<tr>
<td></td>
<td>• Coarse Manufacturing Constraint Modeling</td>
<td>• Surrogate gen.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Detailed Manuf. Modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RT SW modeling</td>
</tr>
</tbody>
</table>
Design Flow Integration Challenges

- How to start the design process?
- How to help its convergence to a “good enough” solution?
- How to link all the tools?
Refined Design Flow

Seed design
- System as Component Assembly
  - Test bench for the full system
  - Test bench for a subsystems
  - Test bench for components

Run original design
- Look at the system in the tool
- Run test benches

Test benches
- Run test benches
  - ValueFlow
  - CAD
  - Dynamics

Design Space
- Turn original design into a design space
- Turn test benches into a test bench template

Configuration
- Generate configurations

Create constraints
- Auto-generated
  - Visual
  - Regular

Dashboard
- Show all generated configuration along with
  - Key Performance Parameters
  - Metrics
  - Requirements

Parametric Exploration
- PCC
- Optimization
- DOE

Verification tools
- QR
- HybridSAL
- Prismatic

Import new components
Expand Design Space
- Add alternatives
OpenMETA “Composers”

- Competition Coordinator
- Seed Design Spaces
- Requirements / Test Benches
- Evolve Design Spaces
- Compose with Test Benches
- META Composers
- Create Components
- Component Models
- Vehicle Forge (Component Exchange)
- OpenMETA Tools (used by Competitors)
Executable Requirements and Test Bench Concepts

Scenario Specification

Parameters

Environment Specification

Instrumentation

Test Article

Metrics & Requirements

Executable Requirements and Test Bench Concepts

Instrumentation

Test Article

Metrics & Requirements

Executable Requirements and Test Bench Concepts

Instrumentation

Test Article

Metrics & Requirements

Executable Requirements and Test Bench Concepts

Instrumentation

Test Article

Metrics & Requirements

Executable Requirements and Test Bench Concepts

Instrumentation

Test Article

Metrics & Requirements

Executable Requirements and Test Bench Concepts

Instrumentation

Test Article

Metrics & Requirements

Executable Requirements and Test Bench Concepts

Instrumentation

Test Article

Metrics & Requirements
Example for Test Benches to Evaluate FANG Requirements
Architecture Exploration Using Interface Abstractions

Components

Subsystem Design Spaces

Design Space Evolution

Requirements, Design Rules \(\rightarrow\) Constraints

Design Architectures

Constraint-guided Design Space Pruning

- Design Space + Behavioral Constraint Modeling
- Architecture Modeling
- Dynamic Modeling (multiple abstractions and multiphysics)
- CAD Assembly modeling
- Coarse Manufacturing
- Constraint Modeling
- Architecture Modeling
- Detailed Domain Modeling
- FEK, internal fluid
- Discretization
- Detailed H2G modeling
- RT-SySi modeling

Design Architectures
Design Space Exploration Using Multi-Fidelity ODEs

Uncertainty Propagation & Estimation

Multiple Physics Domains

Multiple Fidelity Behavior Models
Design Space Exploration Using Geometry and FEA

CAD Testbench for Physical Properties

1) Bounding box
2) Center of Gravity
3) Dimensions

FEA Testbench for Structural Properties

1) maximum shear stress,
2) maximum bearing stress,
3) maximum Von Mises stress
4) factor-of-safety
Design Architectures with **ideal component dynamics**

- Hybrid Dynamics Models
- SW Component Architecture Timing Model
- SW Component Architecture Synthesis
- System Architecture Synthesis
- Scheduling and Schedulability Analysis
- Implemented Dynamics Model Synthesis
- System Integration Code Synthesis
- System Platform Model
- Certificate

Design Architectures with **deployed component dynamics**

- Time-triggered Model of Computation
- TT bus (or emulated TT bus)
- Event-triggered Model of Computation
- CAN bus

OpenMETA Software Tool Chain
Design Space Evaluation Visualization
Pairwise Visualization of Metrics
Probabilistic Certificates of Correctness (PCC)
Geometric Reasoning: CAD Assembly Composition

BOM, Assembly, GD&T, ...

META Model of Structural Connections

CAD-Independent Assembly CAD Tool

Specific Drivers

iF AB Interface (partial)

BOM, Assembly, GD&T, …
Tools for CPS Design

- A Cyber-Physical Systems Design Project: AVM
  - Goals
  - Basic concepts: Vehicle Forge
  - Basic concepts: OpenMETA
- Information Architecture Challenge
- OpenMETA Design Flow Integration Challenge
- Semantic Integration Challenge
  - Structural Semantics
  - Behavioral Semantics
The Need for Formal Semantics

- **META Semantic Integration**
- **Formal Verification**
  - Qualitative reasoning
  - Relational abstraction
  - Model checking
  - Bounded model checking
- **Stochastic Co-Simulation**
  - Open Modelica
  - Dymola
- **Distributed Simulation**
  - NS3
  - OMNET
  - Delta-3D
  - CPN
## Concept of “Semantic Integration”

<table>
<thead>
<tr>
<th>Physical Component Structure</th>
<th>CyPhy Integration Model</th>
<th>Tool Integration</th>
<th>Semantics Model /FORMULA/</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="bond_graph.png" alt="Bond Graph" /></td>
<td><img src="modelica.png" alt="Modelica" /></td>
<td><img src="cyphy_structure.png" alt="CyPhy Structure Model" /></td>
<td><img src="bg_semantics.png" alt="BG Semantics" /></td>
</tr>
<tr>
<td><img src="bg_model.png" alt="BG Model" /></td>
<td><img src="bg2mod_semantics.png" alt="BG2Mod Semantics" /></td>
<td><img src="modelica_equation.png" alt="Modelica Equation Model" /></td>
<td><img src="bg2mod_semantics.png" alt="BG2Mod Semantics" /></td>
</tr>
<tr>
<td><img src="cyphy_component.png" alt="CyPhy Component" /></td>
<td><img src="cyphy_comp_semantics.png" alt="CyPhy Comp Semantics" /></td>
<td><img src="cyphy_component.png" alt="CyPhy Component" /></td>
<td><img src="cyphy_comp_semantics.png" alt="CyPhy Comp Semantics" /></td>
</tr>
</tbody>
</table>

- **Bond Graph**: ![Insert BG/](bond_graph.png)
- **Modelica**: ![/Insert Mod/](modelica.png)
- **CyPhy Structure Model**: ![CyPhy Structure Model](cyphy_structure.png)
- **Modelica Equation Model**: ![Modelica Equation Model](modelica_equation.png)
- **CyPhy Comp User**: ![CyPhy Comp User](cyphy_comp_user.png)
- **Composed Modelica Equation Model**: ![Composed Modelica Equation Model](composed_modelica_equation.png)
- **VER**: ![VER](VER.png)
- **SIM**: ![SIM](SIM.png)
- **BG Semantics**: ![BG Semantics](bg_semantics.png)
- **BG2Mod Semantics**: ![BG2Mod Semantics](bg2mod_semantics.png)
- **Mod. EQ Semantics**: ![Mod. EQ Semantics](mod_eq_semantics.png)
- **CyPhy Semantics**: ![CyPhy Semantics](cyphy_semantics.png)
- **CyPhy Comp Semantics**: ![CyPhy Comp Semantics](cyphy_comp_semantics.png)
Cost of Model Integration Languages: “Semantic Backplane”

- Tight integration from architecture modeling to physics-based modeling
- Integrated multi-physics modeling
- Bridging gap between computation and physics-based domains
- Tight integration of structural and behavioral models
- Emphasis is on automation and scaling
- The META tool suite must be designed for rapid evolution

Agility is achieved by introducing a Semantic Backplane

The Semantic Backplane is implemented by:
- tools and methods for modeling language specification, validation and transformation
- tools and methods for explicit representation of and computation with structural and behavioral semantics
- metamodel and transformation libraries
- metaprogrammable tools

**SEMANTIC BACKPLANE**

- Model Transformers
- Metagenerators
- Metamodel Analysis, Verification & Validation
- Metamodeling

<table>
<thead>
<tr>
<th>Languages</th>
<th>Models</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRaAT-Model Transf.</td>
<td>GRaAT Transform. Libraries</td>
<td>GRaAT (FORMULA)</td>
</tr>
<tr>
<td>ASML</td>
<td>ASML</td>
<td>FORMULA Consistency Proofs</td>
</tr>
<tr>
<td>FORMULA</td>
<td>FORMULA</td>
<td>MetaGME++-2-FORMULA</td>
</tr>
<tr>
<td>FORMULA</td>
<td>ASML</td>
<td>FORMULA</td>
</tr>
<tr>
<td>MetaGME++</td>
<td>FORMULA</td>
<td>MetaGME++-ASML</td>
</tr>
<tr>
<td>DSML Metamodels</td>
<td>CrossDomain Metamodels</td>
<td>GME</td>
</tr>
</tbody>
</table>

FORMULA: http://research.microsoft.com/formula
Convergence to a Formal Framework: FORMULA

- History: Foundations for Embedded Systems ITR; Ethan Jackson at VU 2005-2008
- Microsoft Research (Bellevue & Aachen); Satisfiability Modulo Theory Solver (Z3); VS distribution
- http://research.microsoft.com/formula

- Foundation: Algebraic Data Types (ADT) and First-order logic with fixpoints (FPL)
- Parameterized with background theories (bit vectors, term algebras, etc.
- Semantics is defined by constraint logic programming (CLP)
- Evolving structures; temporal logic
Formalization of Semantics - Structural

**Structural Semantics** defines modeling domains using Algebraic Data Types and First-Order Logic with Fixpoints. Semantics is specified by Constraint Logic Programming.

**Use of structural semantics:**
- Conformance testing: \( x \in D \)
- Non-emptiness checking: \( D(Y, C) = \{\text{nil}\} \)
- DSML composing: \( D_1 \star D_2 | D_1 + D_2 | D' \) includes \( D \)
- Model finding: \( S = \{s \in D | s| = P\} \)
- Transforming: \( m' = T(m); m' \in X; m \in Y \)
Behavioral Semantics defines exhibited behavior of models by
1. Specifying a translation to a domain with well-understood operational semantics
2. Specifying a translation to a mathematical domain defining behaviors denotationally (e.g. symbolic DAEs)

Use of Behavioral Semantics Specifications:
• Validating/understanding behaviors via simulation
• Generating behaviors using “reference semantics” and testing tools w.r.t. reference semantics
• Invariance checking
• Formalization → first steps toward proofs
• Tracking dependences in tool suites
# Layers of the Semantic Backplane

<table>
<thead>
<tr>
<th>Functions</th>
<th>(Meta)Models</th>
<th>Languages</th>
<th>Tools</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metamodeling</td>
<td><img src="image" alt="UML Diagram" /></td>
<td>MetaGME</td>
<td>• GME</td>
<td>• DSML spec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• MetaGME-2-Formula</td>
<td>• Constraint Checking</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Metaprog.</td>
</tr>
<tr>
<td>Transformation Modeling</td>
<td><img src="image" alt="UML Diagram" /></td>
<td>UMTL</td>
<td>• GReAT</td>
<td>• Transf. spec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• UDM</td>
<td>• Compiling spec to transformer</td>
</tr>
<tr>
<td>Formal Metamodeling</td>
<td><img src="image" alt="Language Example" /></td>
<td>Formula (MSR)</td>
<td>• Domain Comp.</td>
<td>• Metamod. checking</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Trace Gen.</td>
<td>• Example gen.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Semantic units</td>
</tr>
<tr>
<td>Formal Transformation Model</td>
<td><img src="image" alt="Language Example" /></td>
<td></td>
<td>• Semantic Anchoring</td>
<td>• Semantics for complex DSMLs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Composition</td>
</tr>
</tbody>
</table>
Structure of the Semantic Backplane

- Informal models/metamodels:
  - Instance Model
  - Instantiates
  - Metamodel
  - Updates, Revisions, Extensions

- Formal models/metamodels:
  - Instance Model (Term Algebraic Representation)
  - Defines
  - Structural Constraints Metamodel (Term Algebraic Representation)
  - Defines
  - Semantic Mapping (Execution of Logic Programs)
  - Specifies
  - Structural Constraints Semantic Domain (Term Algebraic Representation)
  - Semantic Mapping Rules (Logic Programs)
  - Testing and Verification

- Instance Models (Formal Structures in Term Algebra)
- Grounding Denotational Semantics in Mathematics
- Specification for Operational Semantics (Logic Program)
Metamodel of a simplified acausal Bond Graph DSML

Formal Metamodel of a simplified Bond Graph DSML

domain AcausalBG_elements
{
  primitive Sf ::= (id: String).
  primitive Se ::= (id: String).
  primitive R ::= (id: String).
  primitive C ::= (id: String).
  primitive I ::= (id: String).
  primitive TF ::= (id: String).
  primitive GY ::= (id: String).
  primitive ZeroJunction ::= (id: String).
  primitive OneJunction ::= (id: String).
  Source ::= Sf + Se.
  Storage ::= C + I.
  OnePort ::= Source + R + Storage.
  TwoPort ::= TF + GY.
  BGELEMENT ::= OnePort + TwoPort.
  Junction ::= ZeroJunction + OneJunction.
  BGNode ::= BGELEMENT + Junction.
  primitive Bond ::= (id: String).
  [Closed] primitive Src ::= (Bond,BGNode).
  [Closed] primitive Dst ::= (Bond,BGNode).
}
Part of Structural Semantics for acausal Bond Graphs

```prolog
domain AcausalBG extends AcausalBG_elements {
  invalidBondDef := a is Bond, no Src(a,_).
  invalidBondDef := a is Bond, no Dst(a,_).
  ...
  bondConn(a,x) :- Src(a,x); Dst(a,x).
  atLeastOneConnection(x) :- bondConn(_,x).
  atLeastTwoConnections(x) :-
    bondConn(a,x), bondConn(b,x), a != b.
  exactlyOneConnection(x) :-
    atLeastOneConnection(x),
    no atLeastTwoConnections(x).
  ...
  invalidBlock := x is OnePort,
    no exactlyOneConnection(x).
  invalidBlock := x is TwoPort,
    no exactlyTwoConnections(x).
  invalidBlock := x is R, Src(_,x);
    x is C, Src(_,x);
    x is I, Src(_,x).
  invalidBlock := x is TwoPort, no Src(_,x);
    x is TwoPort, no Dst(_,x).
  ...
  conforms := !invalidBlock &
    !invalidBondDef &
    !invalidSrcDef &
    !invalidDstDef.
}
```

- Structural semantics is composed of constraints on model structure
- Modeling tools need to check constraints during modeling
- A well-formed model can be mapped into some behavior
Specifying Behavioral Semantics

\[ D(Y, C) = \{ r \in R_Y \mid r \models C \} \]

\[ [\quad ] : R_Y \mapsto R_{Y'} \]

\[ D(Y', C') = \{ r \in R_{Y'} \mid r \models C' \} \]

\[ [\quad ] : R_{Y'} \mapsto R_{Y''} \]

domain AcausalBG_elements

\[
\begin{align*}
\text{primitive } \text{Sf} &::= (\text{id}: \text{String}). \\
\text{primitive } \text{Se} &::= (\text{id}: \text{String}). \\
\text{primitive } \text{R} &::= (\text{id}: \text{String}). \\
//...
\text{primitive } \text{TF} &::= (\text{id}: \text{String}). \\
\text{primitive } \text{GY} &::= (\text{id}: \text{String}). \\
\text{primitive } \text{ZeroJunction} &::= (\text{id}: \text{String}). \\
\text{primitive } \text{OneJunction} &::= (\text{id}: \text{String}). \\
\text{Source} &::= \text{Sf} + \text{Se}. \\
//..
\end{align*}
\]

domain DAEquations

\[
\begin{align*}
\text{primitive } \text{Variable} &::= (\text{name}: \text{String}, \text{id}: \text{String}). \\
\text{primitive } \text{Param} &::= (\text{id}: \text{String}). \\
\text{primitive } \text{Neg} &::= (\text{Term}). \\
\text{primitive } \text{Inv} &::= (\text{Term}). \\
//.. 
\text{Term} &::= \text{Variable} + \text{Param} + \text{Neg} + \text{Inv} + \text{Mul} + \text{Sum}. \\
\text{primitive } \text{Eq} &::= (\text{Variable}, \text{Term}). \\
\text{primitive } \text{DiffEq} &::= (\text{Variable}, \text{Term}). \\
\text{primitive } \text{SumZero} &::= (\text{Sum}). \\
\text{Equation} &::= \text{Eq} + \text{DiffEq} + \text{SumZero}. \\
\end{align*}
\]
Operational Behavioral Semantics for Finite Automata

domain DFA {
    primitive Event ::= (lbl: Integer).
    primitive State ::= (lbl: Integer).
    primitive Transition ::= (src: State, trg: Event, dst: State).
    primitive Current ::= (st: State).
    nonDeterTrans ::= Transition(s, e, sp), Transition(s, e, tp), sp != tp.
    conforms ::= !nonDeterTrans.
}

transform Step<fire: in1.Event> from in1::DFA to out1::DFA {
    out1.Event(x) :- in1.Event(x).
    out1.Transition(s, e, sp) :- in1.Transition(s, e, sp).
    out1.Current(sp) :- in1.Current(s), in1.Transition(s, fire, sp).
    out1.Current(s) :- in1.Current(s),
    fail in1.Transition(s, fire, _).
}
CyPhy Languages

Structural Specification
Composition rules, Signal Flow Directionality, Port Type, Coercions

Behavioral Specification
High-Level Equation with Timing (Denotational, DAE)

Component Interchange
Connectivity rules, multiplicities

HLE via CyPhy (Denotational)

Design Interchange
Connectivity rules, multiplicities

HLE via CyPhy

CyPhy Design Space
Consistency of the variation points

Logical Expressions

CyPhy Signal Flow - Structure
Block Interface Specification, Input/Output Constraints, Connectivity

High-Level Equation with Timing (Denotational)

CyPhy Signal Flow - State
Initial states, MAAB requirements

Structural Operational Semantics (Operational)

High-Level Lambda Expressions (Denotational)

Cyber Deployment
Required mapping, Unique elements, naming conventions

Cyber Execution
Schedulability Constraints, Execution Assignments, Logical Execution Time (LET)

Timed Automata (Operational)

CyPhyML - Modelica Power Connections
The behavioral semantics of Modelica power ports is the same as that of CyPhyML. For example, in electrical domain effort is voltage and flow is current in both CyPhyML and Modelica.

\[ \forall (a, y) \in P, (x_2 = e_y, f_x = f_y) \]

where \( P \) is the set of Modelica - CyPhyML power port mappings.

CyPhyML - Modelica Signal Connections
The behavioral semantics of Modelica signal ports is the same as that of CyPhyML.

\[ \forall (x, y) \in P, (s_x = s_y) \]

where \( P \) is the set of Modelica - CyPhyML signal port mappings.

CyPhyML - SignalFlow Signal Connections
While signal ports in signal-flow are discrete-time ports, signal ports in CyPhyML are continuous-time. Thus, signal-flow output signals are integrated into CyPhyML by means of a \textit{hold} function.

\[ \forall (x, y) \in P, (e_y := \text{hold}(e_x)) \]

where \( P \) is the set of SignalFlow output - CyPhyML signal port mappings.
Summary
Lessons Learned building CPS Tools

- Understanding the current limits of correct-by-construction design using model-based verification
  - Significant scalability problems even in relatively simple (but real) systems
  - Scalable verification requires strong restrictions on modeling abstractions (e.g. linear hybrid dynamics, order reduction) and NOT high data fidelity
  - The resulting uncertainty is epistemic and cannot be characterized probabilistically
Links

- **AVM Publications:**
  [http://www.isis.vanderbilt.edu/biblio/keyword/183](http://www.isis.vanderbilt.edu/biblio/keyword/183)

- **AVM Tools:**
  [https://vehicleforge.vf.isis.vanderbilt.edu/p/metaresources/home/](https://vehicleforge.vf.isis.vanderbilt.edu/p/metaresources/home/)