Arquimedes Canedo* and Ling Shen
Siemens Corporation, Corporate Technology, Princeton, USA

Functional Debugging of Equation-based Languages
Status Quo

Detail Design Tools
- High complexity
- Intensive expert labor
Problems with the State-of-the-art CPS Design

1. Lack of vertical integration
2. Serialization of the engineering practice
3. Informal concept design
4. Holistic system-level validation
5. Disjointed practices (electro-mechanical vs control)
State-of-the-art Debugging Techniques for Equation-based Languages

- Low-level details for detail design
- Static
- Symbolic transformations.
- Dynamic
- Similar to classical debugging.
- Difficult to build a system-level context.

Source: Adrian Pop, Martin Sjölund, Adeel Asghar, Peter Fritzson, Francesco Casella, Static and Dynamic Debugging of Modelica Models, 9th International Modelica Conference, 2012
Functional Modeling

- **What** the system does
- Natural language
- Inter-disciplinary communication
- Visual syntax and well defined semantics
- Multi-disciplinary representation
- Functions and flows
- Standard practice in automotive
Functional Debugging

- The mechanism by which the variables of a running simulation are visualized through a high-level functional model to create an implementation independent understanding of the system.
- Coupling of behaviors to functions.
- High-level behavioral simulation.
- “What” not “how”
- System-level integration problems.
- Design space exploration.
- Visualization of system flows.
Functional Debugger

- **Level of Abstraction**
  - High (Concept-level)
  - Low (Equation-Level)

- **What the system does?**
  - Functional Editor (e.g. MS Visio)
  - Functional Model
  - Automatic or Manual Simulation Synthesis
  - What the system does?

- **How the system does it?**
  - Existing debugging techniques
  - Mapping Model
  - Simulation Model
  - Simulation Runtime (e.g. SystemModeler)
  - How the system does it?

- **User Interaction & Visualization**
  - Functions & Flows
  - Variables

- **Simulation Control**
  - Functional Debugger
  - Simulation Control
Functional Basis

- Well defined vocabulary by NIST
- Three flow categories
  - Material, energy, signal → 18 flow types
- Eight function categories
  - 32 elementary functions
- Execution flows from left-to-right
## Mapping between Functional and Behavioral Models

<table>
<thead>
<tr>
<th>Functional Modeling</th>
<th>Equation-based Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flow Class</strong></td>
<td><strong>Conjugate Vars. (Effort/Flow)</strong></td>
</tr>
<tr>
<td>Electrical</td>
<td>Electromotive Force / Current</td>
</tr>
<tr>
<td>Mechanical (Rotational)</td>
<td>Torque / Angular Velocity</td>
</tr>
<tr>
<td>Mechanical (Translational)</td>
<td>Force / Linear Velocity</td>
</tr>
<tr>
<td>Mechanical (Vibrational)</td>
<td>Amplitude / Frequency</td>
</tr>
<tr>
<td>Hydraulic</td>
<td>Pressure / Volumetric Flow</td>
</tr>
<tr>
<td>Pneumatic</td>
<td>Pressure / Mass Flow</td>
</tr>
<tr>
<td>Thermal</td>
<td>Temperature / Heat Flow</td>
</tr>
<tr>
<td>Electromagnetic</td>
<td>Intensity / Velocity</td>
</tr>
<tr>
<td>Magnetic</td>
<td>Mag. Force / Mag. Flux Rate</td>
</tr>
<tr>
<td>Chemical</td>
<td>Affinity / Reaction Rate</td>
</tr>
<tr>
<td>Biological</td>
<td>Pressure / Volumetric Flow</td>
</tr>
<tr>
<td>Human</td>
<td>Force / Motion</td>
</tr>
<tr>
<td>Acoustic</td>
<td>Pressure / Particle Velocity</td>
</tr>
<tr>
<td>Radioactive</td>
<td>Intensity / Decay Rate</td>
</tr>
<tr>
<td><strong>Signal</strong></td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td></td>
</tr>
<tr>
<td><strong>Material</strong></td>
<td></td>
</tr>
<tr>
<td>Human</td>
<td></td>
</tr>
<tr>
<td>Gas</td>
<td></td>
</tr>
<tr>
<td>Liquid</td>
<td></td>
</tr>
<tr>
<td>Solid</td>
<td></td>
</tr>
</tbody>
</table>

**Currently implemented**

R: Restricted © Siemens AG 2013 All rights reserved.

Canedo, Shen
Mapping Model

- Data structure that maintains the mapping between the functional model and the simulation model.
- Specifies the debugging actions to be taken in the functional debugger GUI.

<table>
<thead>
<tr>
<th>UID</th>
<th>Type</th>
<th>Func/Flow</th>
<th>Simulation Component</th>
<th>Debugging Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>f11</td>
<td>Flow</td>
<td>RME</td>
<td>Drivetrain.Engine.engineSpeed</td>
<td>if ($var &gt; 4000) f11.ChangeColor(Red)</td>
</tr>
<tr>
<td>f15</td>
<td>Flow</td>
<td>Signal</td>
<td>Drivetrain.driver1.brakeSignal</td>
<td>if ($var &gt; 0.05) f15.setInvisible();</td>
</tr>
<tr>
<td>f92</td>
<td>Function</td>
<td>Convert RME to TME</td>
<td>Drivetrain.body.flange_a.f</td>
<td>if ($var &gt; 60) f11.ChangeColor(Red)</td>
</tr>
</tbody>
</table>

- Proposal: describe what functions are realized by the components in the MSL
Our Implementation

Communication (e.g. FMI)

Front-end (e.g. SysML)

Runtime (e.g. OpenModelica)
Automotive Example

- Drivetrain of an internal combustion engine car.
Functional Debugging

- System configuration
- V6, V8
- Use-cases
- Driving cycles
- Performance targets
- Fuel economy
- NVH
Summary

- Functional debugging for concept design
  - Visualization of system flows
  - Mapping of functions to behavioral models through conjugate variables
  - Design space exploration tool
  - Orthogonal to existing debugging techniques
- Systems Engineering tools
  - Requirements, Functions, Logical, Physical
- Multi-tool debugging interface
  - e.g. Modelica, Nastran, VHDL
Thank you!

Dr. Arquimedes Canedo
Staff Scientist

Siemens Corporation
Corporate Technology
755 College Road East
Princeton, NJ, USA

arquimedes.canedo@siemens.com