Model-Based Design Overview

Edited and presented by
Janos Sztipanovits and Gabor Karsai
ISIS, Vanderbilt University
Model-based design focuses on the *formal representation, composition, and manipulation of models* during the design process.
# Core Modeling Aspects in System Composition

<table>
<thead>
<tr>
<th>Component Behavior</th>
<th>Modeled on different levels of abstraction:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Transition systems (FSM, Time Automata, Cont. Dynamics, Hybrid), fundamental role of time models</td>
</tr>
<tr>
<td></td>
<td>- Precise relationship among abstraction levels</td>
</tr>
<tr>
<td></td>
<td>- Research: dynamic/adaptive behavior</td>
</tr>
<tr>
<td>Structure</td>
<td>Expressed as a system topology:</td>
</tr>
<tr>
<td></td>
<td>- Module Interconnection (Nodes, Ports, Connections)</td>
</tr>
<tr>
<td></td>
<td>- Hierarchy</td>
</tr>
<tr>
<td></td>
<td>- Research: dynamic topology</td>
</tr>
<tr>
<td>Interaction</td>
<td>Describes interaction patterns among components:</td>
</tr>
<tr>
<td></td>
<td>- Set of well-defined Models of Computations (MoC) (SR, SDF, DE,..)</td>
</tr>
<tr>
<td></td>
<td>- Heterogeneous, but precisely defined interactions</td>
</tr>
<tr>
<td></td>
<td>- Research: interface theory (time, resources,..)</td>
</tr>
<tr>
<td>Scheduling/Resource Mapping</td>
<td>Mapping/deploying components on platforms:</td>
</tr>
<tr>
<td></td>
<td>- Dynamic Priority</td>
</tr>
<tr>
<td></td>
<td>- Behavior guarantees</td>
</tr>
<tr>
<td></td>
<td>- Research: composition of schedulers</td>
</tr>
</tbody>
</table>

"Model-Based Design Overview", J. Sztipanovits

Chess Review, Nov. 21, 2005
**Tool Composition Approaches**

<table>
<thead>
<tr>
<th>Domain-Specific Tools; Design Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MIC Metaprogrammable Tool Suite:</strong> (mature or in maturation program)</td>
</tr>
<tr>
<td>• Metamodeling languages</td>
</tr>
<tr>
<td>• Modeling Tools</td>
</tr>
<tr>
<td>• Model Transformations</td>
</tr>
<tr>
<td>• Model Management</td>
</tr>
<tr>
<td>• Design Space Construction and Exploration</td>
</tr>
<tr>
<td>• Tool Integration Framework</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metaprogrammable Tools, Integration Frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain-Specific Design Flows and Tool Chains:</strong></td>
</tr>
<tr>
<td>• ECSL - Automotive</td>
</tr>
<tr>
<td>• ESML - Avionics</td>
</tr>
<tr>
<td>• SPML - Signal Processing</td>
</tr>
<tr>
<td>• CAPE/eLMS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semantic Foundation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic Foundations (work in progress):</strong></td>
</tr>
<tr>
<td>• Semantic Anchoring Environment (SAE)</td>
</tr>
<tr>
<td>• Verification</td>
</tr>
<tr>
<td>• Semantic Integration</td>
</tr>
</tbody>
</table>
Intersection with Tool Composition Dimensions

Component Behavior

- Semantic Units and Semantic Anchoring
- Compositional Semantics

Structure

- Metamodels and Metaprogrammable Tool Suites

Interaction

- Domain-Specific Design Flows and Tool Chains

Resource Modeling (Schedule)

- Semantic Foundation;
- Metaprogrammable Tools, Environments

- Domain-Specific Tools, Tool Chains
Domain Specific Design Flows and Tool Chains

- ECSL - Automotive
- ESML - Avionics
- SPML - Signal Processing
- FCS - Networked Embedded Systems
- CAPE/eLMS - Courseware Design/Delivery
- Integration among tool frameworks: Metropolis, Ptolemy II, MIC, Simulink/Stateflow, ARIES, CheckMate, ...
- www.escherinstitute.org
Intersection with Tool Composition Dimensions

<table>
<thead>
<tr>
<th>Component Behavior</th>
<th>Semantic Units and Semantic Anchoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Compositional Semantics</td>
</tr>
<tr>
<td>Interaction</td>
<td>Metamodels and Metaprogrammable Tool Suites</td>
</tr>
<tr>
<td>Resource Modeling (Schedule)</td>
<td>Domain-Specific Design Flows and Tool Chains</td>
</tr>
</tbody>
</table>

Semantic Foundation; Metaprogrammable Tools, Environments

Domain-Specific Tools, Tool Chains
Metaprogrammable Tool Suite

Generic Model Editor (GME)

Unified Data Model (UDM)

Open Tool Integration Framework

• Simulators
• Verifiers
• Model Checkers
• Generators

Design Space Exploration (DESERT)

Meta Models

Model Transformation

GMe

UDM

OTIF

DESERT

GReAT

"Model-Based Design Overview", J. Sztipanovits
Intersection with Tool Composition Dimensions

- Component Behavior
  - Semantic Units and Semantic Anchoring

- Structure
  - Compositional Semantics
    - Metamodels and Metaprogrammable Tool Suites
      - Domain-Specific Design Flows and Tool Chains

- Interaction

- Resource Modeling (Schedule)
  - Semantic Foundation
    - Metaprogrammable Tools, Environments
      - Domain-Specific Tools, Tool Chains
Transformational Specification of Semantics

- Specify mapping to another language with well-defined semantics.

"Model-Based Design Overview", J. Sztipanovits

Chess Review, Nov. 21, 2005
Semantic Units

\[ M_S = M_{Si} \circ M_A \]

DSML

Semantic Unit i

\[ C_i \rightarrow S_i \]

\[ M_{Si} : A_i \rightarrow S_i \]

Ongoing Work

- Capture the behavioral semantics of a finite set of basic models of computations, such as FSM, DE, TA, SDF...
- Further exploration of the concept of abstract semantics with Berkeley.
Semantic Anchoring Tool Suite

Metamodeling and Model Transformation Tools

- GME Toolset
  - DSML Metamodel (A)
  - Domain Model (C)
- GReAT Tool
  - Model Trans. Rules ($M_A$)
  - Transformation Engine
  - Semantic Unit Metamodel ($A_s$)
- XSLT

Formal Framework for Semantic Units Specification

- Semantic Unit Spec.
  - Abstract Data Model
  - Operational Semantics Spec.
  - ASM Semantic Framework
  - Data Model

AsmL Tools

- Model Checker
- Test Case Generator
- Model Simulator

- Metaprogrammable tools
- Syntactic manipulations
- Abstract State Machine (ASM) (Evolving Algebras)
- AsmL tool suite

"Model-Based Design Overview", J. Sztipanovits
Chess Review, Nov. 21, 2005
## Intersection with Tool Composition Dimensions

<table>
<thead>
<tr>
<th>Component Behavior</th>
<th>Semantic Units and Semantic Anchoring</th>
<th>Compositional Semantics</th>
<th>Metamodels and Metaprogrammable Tool Suites</th>
<th>Domain-Specific Design Flows and Tool Chains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interaction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource Modeling (Schedule)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Semantic Foundation;
- Metaprogrammable Tools, Environments
- Domain-Specific Tools, Tool Chains

"Model-Based Design Overview", J. Szirahovits
DSML Design Through Semantic Anchoring

- **Ongoing Work**
  - Specification of component interaction semantics (MoC-s)
  - Design of a DSML Specification Environment
  - Compositional Semantics (with Berkeley and J. Sifakis, Verimag)

"Model-Based Design Overview", J. Sztipanovits

Chess Review, Nov. 21, 2005
Progress in Model Transformations: GReAT

Global “spaces” for Transformations

Source Models

Target Model

Global space <<Temps>>

InHost

RootRule

Lm

Lm

Lm

Out

PortBase

CreateQueue

Global spaces hold intermediate results of the transformation
Consequence: The transformations are simplified.

Additional new capabilities:
• Distinguished cross-product: a new built-in operator of the
  language that refines pattern matching semantics
• Match-any associations: “wild-card” pattern matching construct
  for matching arbitrary associations
• User code libraries
• Support for automatic connection of multi-ported objects in the
  modeling tool
• Integration with new development platform (Microsoft VS 7+)
• Support for XML namespaces
• Integration with Java in the data layer (UDM)
• Support for text output (UDM, using OCL for scripting)
• Support for structured text input: input is parsed using a
declarative parser that constructs the input data structures

GReAT Application Examples:
• Embedded system toolchains (see
  ESCHER)
• Large-scale architecture modeling and
  analysis tool for distributed embedded
  systems
• Semantic anchoring framework for
domain-specific modeling languages
• Model-based tools for CORBA
  component configuration

Sorting the transformation results

A transformation rule typically operates on a
sequence of matched objects that could be sorted
after the rule is applied.
Consequence: Model transformation results are
ordered by the sorting function.
Model Transformations for Hard-Real Time Languages: Control System Problem

Control system engineers model control problems as periodically triggered sensing, control calculation, and actuation actions, but traditional software implementations rarely comply with these strict requirements. Giotto (as a time-triggered language) and its execution platform offers a solution to bridge the gap between design and implementation. A prototype model transformation tool shows how the complex transformation from Giotto to E-code can be implemented in terms of graph transformation operations.

"Model-Based Design Overview", J. Sztipanovits
Model Transformations in Toolchains: Platform Modeling & Analysis

Metamodel of Source DSML

Source Domain Models

Model for Platform -> Analysis Mapping

Model Transformation Specification

Model in Analysis format
• Domain models
• Platform semantics
• Platform properties

Model of Platform

Analysis Tool:

Metamodel of Target (SYNTAX)

Target Model

Approach:
Analysis must take into consideration platform’s operational semantics, i.e. model of computation. To make this possible, an explicit platform model is required that proscribes how the platform’s concepts are mapped into the concepts of the analysis language. This mapping defines the model transformation needed.
Research on Model Transformations: Next steps

- (Better) formal semantics of GReAT
  - Goal: Reasoning about transformations
  - Use: in semantic anchoring

- Verification/certification of transformations
  - Proving that the generated output satisfies some interesting properties

- Traceability
  - Generated artifacts must be traceable back to their origins

- Model transformations for model evolution
  - Model management and engineering in large-scale embedded systems

- Transformations in mixed-mode development
  - Model-based components with hand-written code

- Experimentation with platform modeling and analysis
  - Event-driven and time-triggered platforms
  - Multiple analysis tools

- Transformations “guided” by platform restrictions
  - Example: resource-constrained platforms may influence the transformation process

- Using graph transformations for embedded component adaptation

"Model-Based Design Overview", J. Sztipanovits
Further Presentations

- Ethan Jackson: Coupled Interface Modules for Heterogeneous Composition

- Kai Chen: A Semantic Unit for Timed Automata Based Modeling Languages

- Tivadar Szemethy: Platform Modeling

- Trevore Meyerowitz, Ethan Jackson: Integration of Metropolis with GME and DESERT