

# Model-Based Design

Edited by Janos Sztipanovits,  
Presented by Gabor Karsai  
ISIS, Vanderbilt University



**CHES Review**  
May 10, 2004  
Berkeley, CA

**UC Berkeley: Chess**  
**Vanderbilt University: ISIS**  
**University of Memphis: MSI**



## Objectives

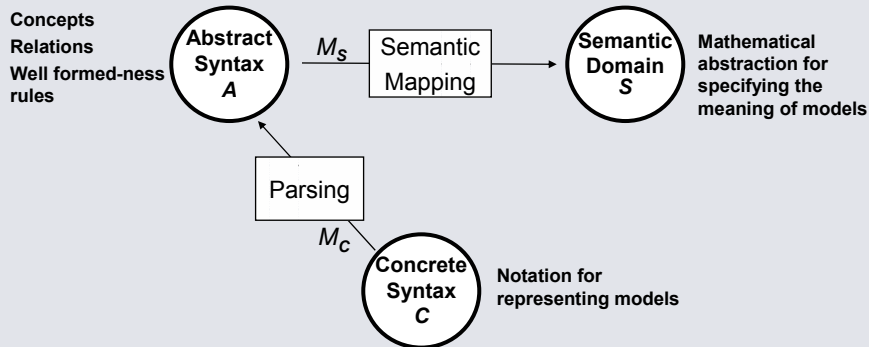
Model-based design focuses on the *formal representation, composition, and manipulation of models* during the design process.

Specific research areas:

1. Composition of Domain Specific Modeling Languages
2. Model Transformation
3. Model Synthesis

# 1. Composition of Domain Specific Modeling Languages

$$L = \langle C, A, S, M_S, M_C \rangle$$



CHES Review, May 10, 2004 3

## Research Agenda

### Baseline:

- Modeling of Abstract Syntax using UML class diagrams and OCL as meta language;
- Meta-programmable Graphical Model Editor (GME)

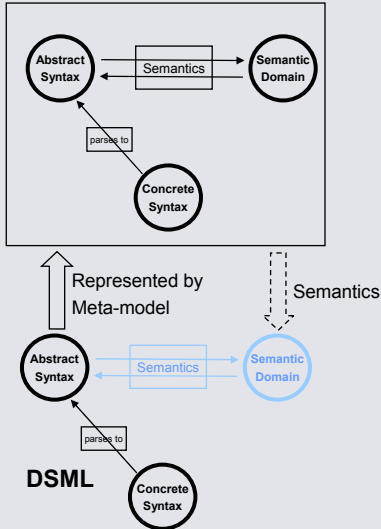
### Research Agenda:

1. Using the Meta-Object Facility (OMG) as the meta-metamodel
2. Specification of DSML semantics via meta-modeling
3. Foundation for composing DSML-s
4. New Semantic Domains
5. Extension of meta-programmable tools

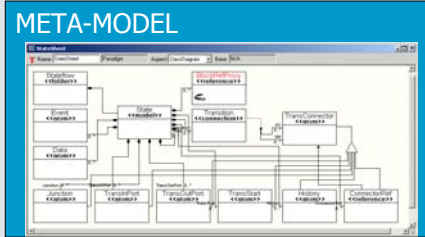
CHES Review, May 10, 2004 4

# Semantics via Meta-Modeling: Baseline

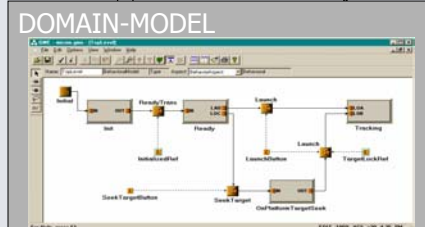
Meta-modeling language with well-defined semantics



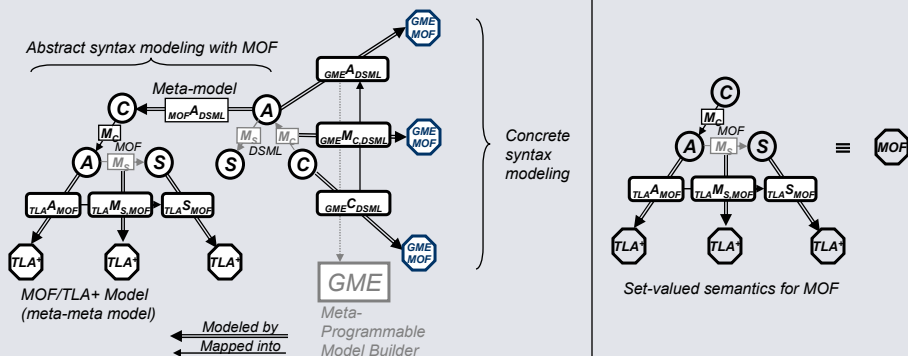
Meta-Model of StateFlow using uml/OCL as meta modeling language.



DSML: StateFlow  $\uparrow$  Meta-model  $\downarrow$  Structural Semantics



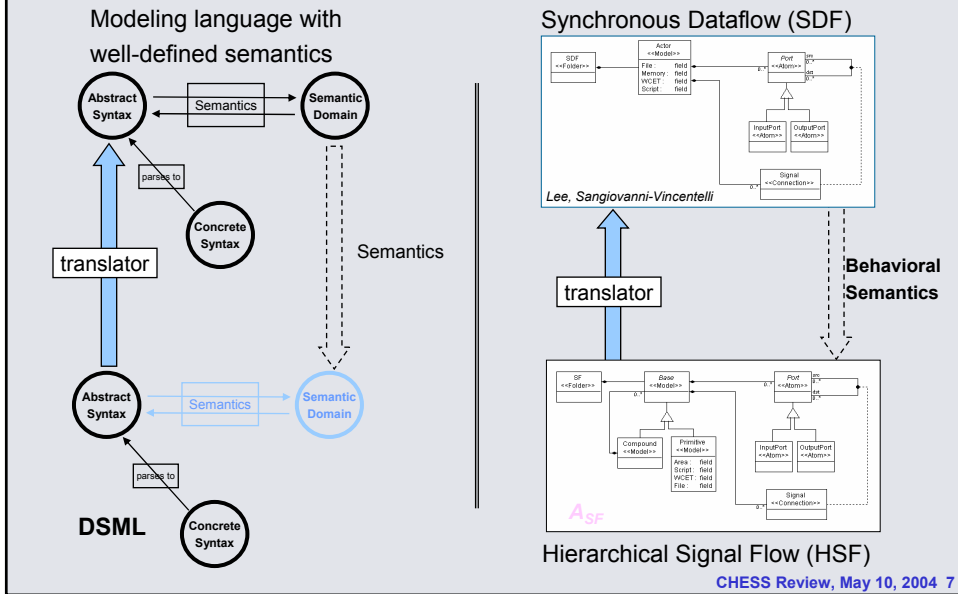
# New Direction: Precise Meta-Modeling



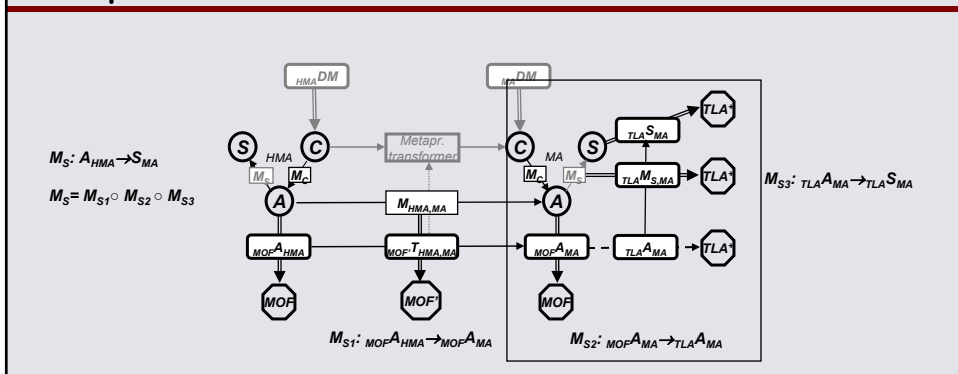
## Ongoing Work: (Poster: Compositional Metamodeling for DSMLs)

- Changing from UML-CD/OCL to MOF in abstract syntax modeling
- Developing set-valued semantics for MOF using TLA+ as meta-language
- Developing  $MOF^{A,DSML} \rightarrow TLA^{A,DSML}$  translator

# Semantics via Translation: Baseline



# New Direction: Translational Specification of DSML Semantics



## Ongoing Work: (Poster: Compositional Metamodeling for DSMLs)

- TLA+ specification of the semantics of selected models of computations (reference models)
- Specifying DSML semantics via specifying mapping between the abstract syntax of DSML and the reference models
- Developing modeling and translation tools

## New Semantic Domains: HyVisual - Hybrid System Modeling Tool Based on Ptolemy II

The screenshot displays the HyVisual software interface. At the top, a text box reads: "Refinement Solver This models the dynamics of a ball falling in a gravitational field." Below this, a block diagram shows a "Const" block with the value "-10" connected to a "Velocity" block, which is then connected to a "Position" block. A "ZeroCrossingDetector" block is connected to the "Position" block and has two outputs: "velocity" and "bump".

Below the block diagram is a state transition diagram with three states: "init", "free", and "stop". Transitions are labeled with conditions and actions:

- From "init" to "free": `true`, `free.initialPosition = initialPosition; free.initialVelocity = 0.0`
- From "free" to "stop": `abs(position) < stoppedThreshold && ...`
- From "stop" to "free": `bump_isPresent`, `free.initialVelocity = -elasticity * velocity; free.initialPosition = position`

On the right, a plot titled "Position" shows "Height (meters)" on the y-axis (0 to 10) and "time (sec)" on the x-axis (0 to 30). The plot shows a series of decaying oscillations, representing the ball's position over time as it bounces.

A yellow text box on the right states: "HyVisual is a DSML designed for hybrid system modeling, specialized from Ptolemy II by combining its CT and FSM domains."

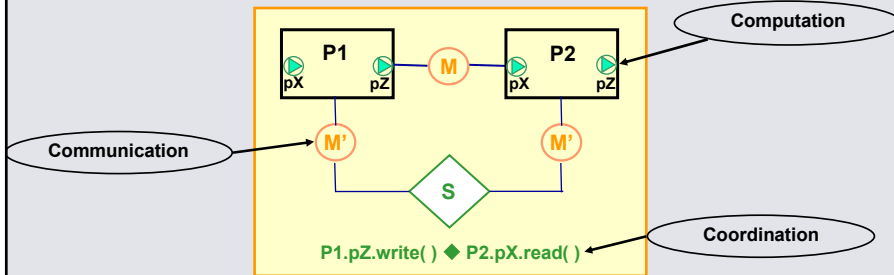
At the bottom right, the text reads: "CHESS Review, May 10, 2004 9"

## Major Applications of Meta Modeling: Metropolis Meta Model (Formal Model)

- How are designs represented in Metropolis?
  - Metropolis "meta-model": a language + modeling mechanisms
  - represents all key ingredients: function, architecture, refinement, platforms
  - parser and API to browse designs, interact with tools

# Metropolis Meta-Model

- Must describe objects at different levels of abstraction
  - Do not commit to the semantics of any particular model of computation
- Define a set of "building blocks"
  - specifications with many useful Models of Computations can be described using the building blocks
  - Processes, communication media and schedulers separate computation, communication and coordination



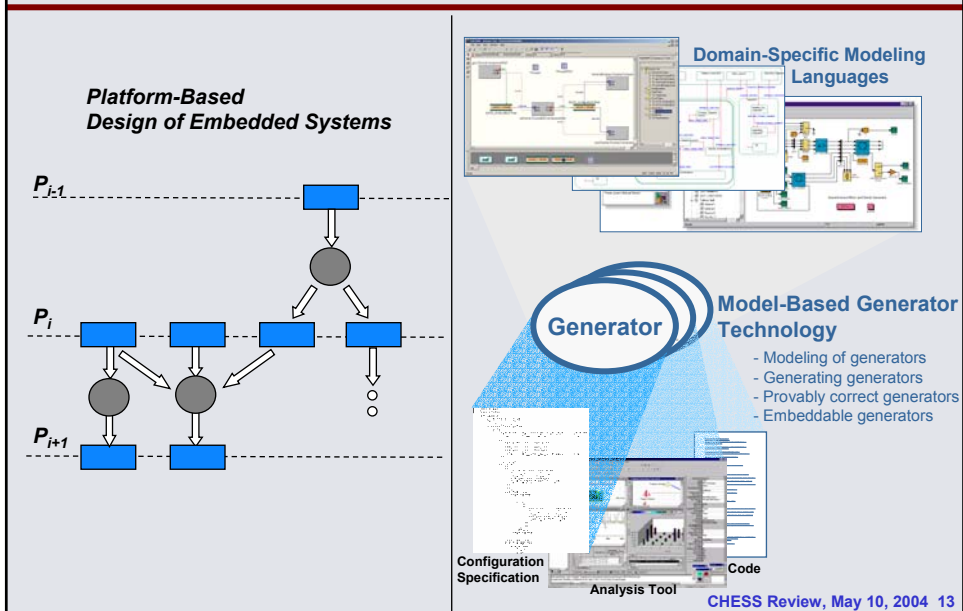
CHES Review, May 10, 2004 11

# Core Metropolis "Meta Model"

- Processes
  - Concurrent Computation, sequential execution within each process
  - Execution within a process can be suspended with the *AWAIT* statement
- Interfaces
  - The types of Ports, bundle function declarations
- Media
  - Model Communication by providing definitions of Interface Functions
  - Passive, cannot run in their own threads
- Netlists
  - *STRUCTURE* and *CONCURRENCY*
- Quantity Managers
  - Performance modeling
- Constraints
  - Specification & Synchronization
- Formal Semantics
  - Operational Semantics based on Action Automata

CHES Review, May 10, 2004 12

## 2. Model Transformations



## Research Agenda

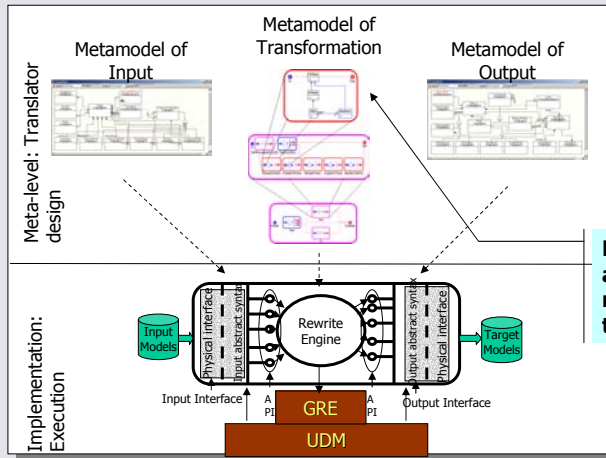
### Baseline:

- Meta-models of the Abstract Syntax of the source and target DSML-s;
- Tools helping the procedural implementation of translators

### Research Agenda:

1. *Meta Generators: Specification and generation of model transformations using graph rewriting technology*
2. *Embeddable generators*
3. *Generative extensions to modeling languages*

# Model Transformation Using Meta-Models and Graph Rewriting

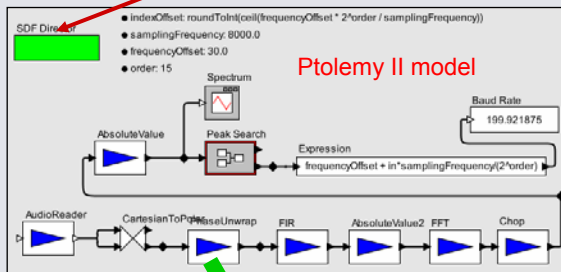


**Formal, explicit, and precise model of the transformations**

Karsai/Agrawal, ISIS

# Component Specialization

Model of Computation semantics defines communication, flow of control

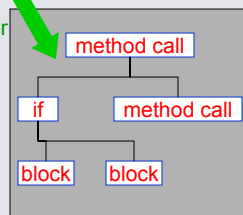


schedulener

- Schedule:
- fire Gaussian0
  - fire Ramp1
  - fire Sine2
  - fire AddSubtract5
  - fire SequenceScope10

parser

Java actor definitions are parsed and then specialized for their context.



abstract syntax tree

specializer

- Specialize for
- data types
  - parameter values
  - scheduling
- By
- token unboxing
  - inlining
  - partial evaluation
  - dead code elimination

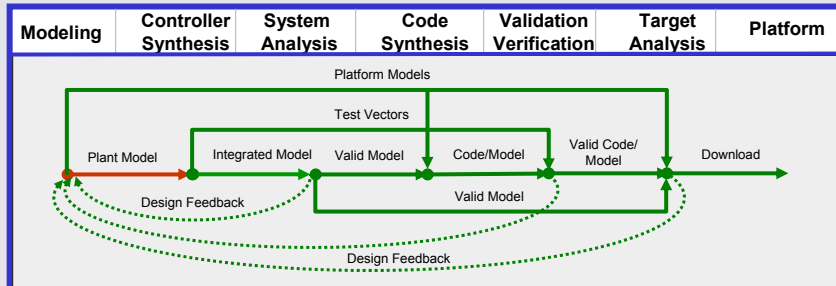
```
for (int i = 0; i < plus.getWidth(); i++) {
  if (plus.hasTaken(i)) {
    if (sum == null) {
      sum = plus.get(i);
    } else {
      sum = sum.add(plus.get(i));
    }
  }
}
```

target code



## 3. Model Synthesis

Example design flow:



### Challenge:

Compose plant and controller models using library components with alternatives so as to satisfy design constraints

CHES Review, May 10, 2004 17

## Research Agenda

Baseline:

- Constraint-based design space exploration coupled to a dataflow language

Research Agenda:

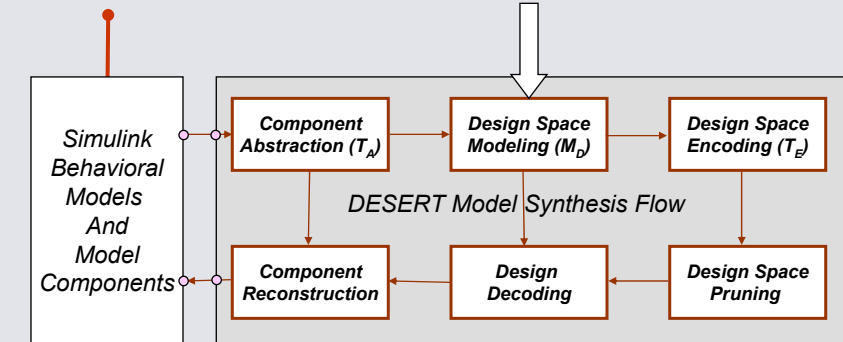
1. Pattern-based model synthesis
2. Models of Computations as design patterns
3. Design constraints and design patterns for multiple Models of Computations

CHES Review, May 10, 2004 18

# Pattern-based Model Synthesis

Main Design Flow

Design Space Modeler

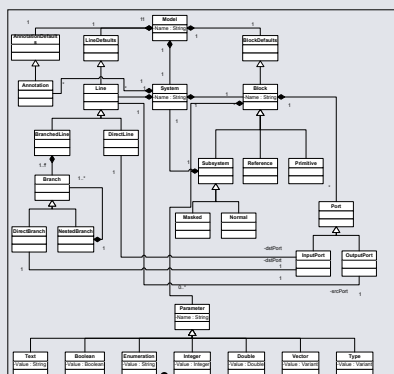


Nezma, Sztipanovits, Karsai at EMSOFT 2003

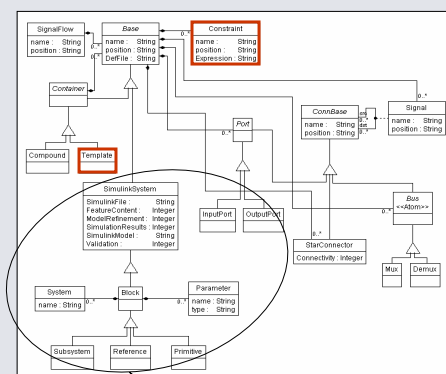
Using meta-modeling and model transformations we made the constraint based design space exploration tool suite composable with different design flows.

Main Design Flow

## Example: Use of Model Translation in the DESERT Tool Chain



Simulink® Metamodel



Design Space Metamodel

Component Abstraction  $T_A$

Abstracted Simulink Model Components