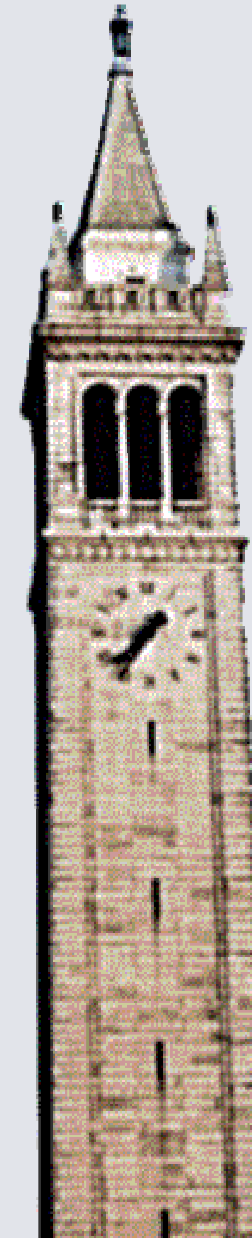
Model-Based Design Overview

Edited and presented by
Janos Sztipanovits and Gabor Karsai
ISIS, Vanderbilt University



Chess Review
November 21, 2005
Berkeley, CA

Model-Based Design

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

Core Modeling Aspects in System Composition

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

Tool Composition Approaches

Domain-Specific Tools; Design Environments	**Domain-Specific Design Flows and Tool Chains:** • ECSL - Automotive • ESML - Avionics • SPML - Signal Processing • CAPE/eLMS
Metaprogrammable Tools, Integration Frameworks	**MIC Metaprogrammable Tool Suite: (mature or in maturation program)** • Metamodeling languages • Modeling Tools • Model Transformations • Model Management • Design Space Construction and Exploration • Tool Integration Framework
Semantic Foundation	**Semantic Foundations (work in progress):** • Semantic Anchoring Environment (SAE) • Verification • Semantic Integration

Intersection with Tool Composition Dimensions

	Semantic Foundation;	Metaprogrammable Tools, Environments	Domain-Specific Tools, Tool Chains
Component Behavior	Semantic Units and Semantic Anchoring	Metamodels and Metaprogrammable Tool Suites	Domain-Specific Design Flows and Tool Chains
Structure	Compositional Semantics		
Interaction			
Resource Modeling (Schedule)			

Intersection with Tool Composition Dimensions

Component Behavior

Structure

Interaction

Resource Modeling (Schedule)

Semantic Units and Semantic Anchoring

Compositional Semantics

Metamodels and Metaprogrammable Tool Suites

Domain-Specific Design Flows and Tool Chains

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

Component Behavior

Structure

Interaction

Resource Modeling (Schedule)

Semantic Units and Semantic Anchoring

Compositional Semantics

Metamodels and Metaprogrammable Tool Suites

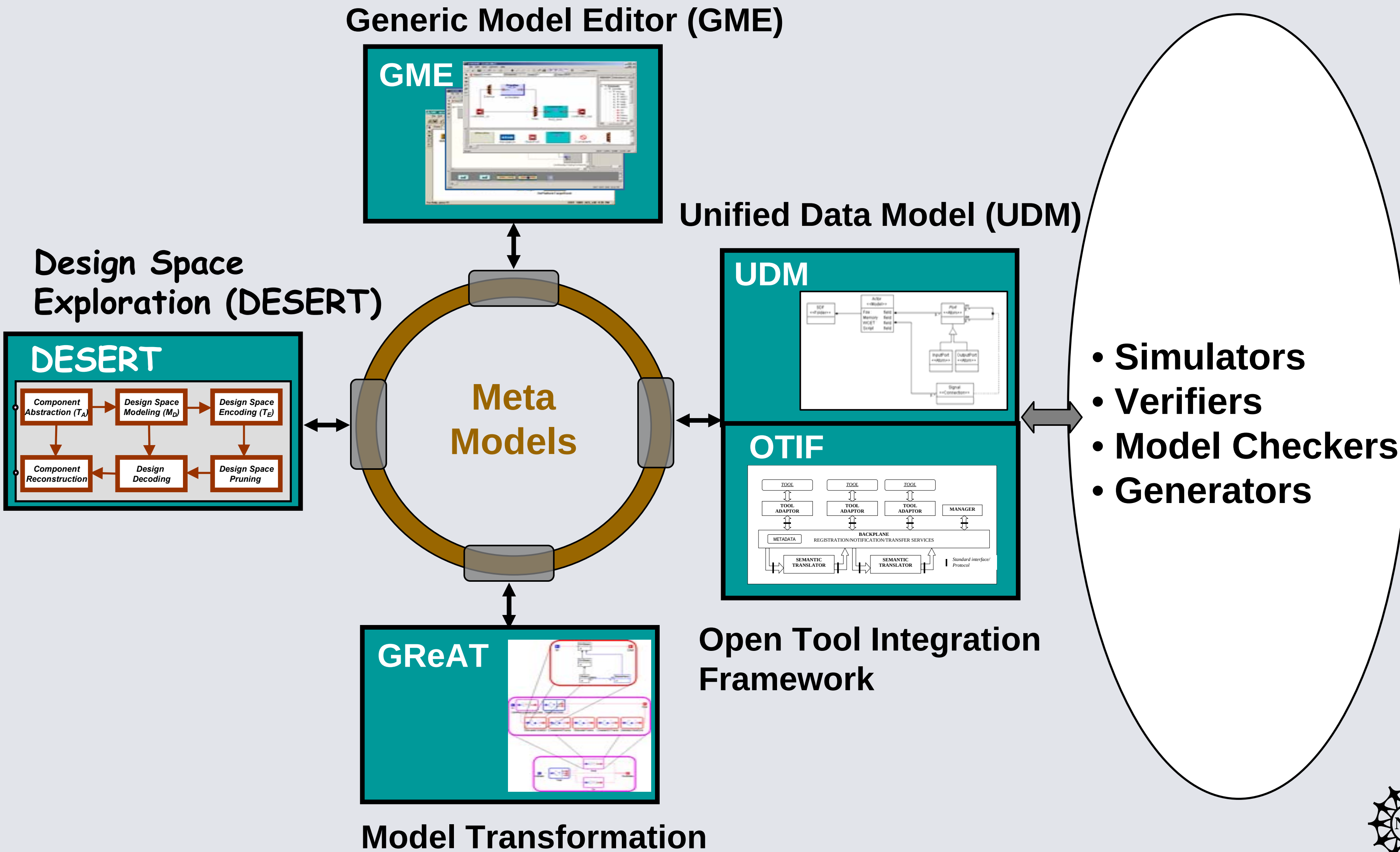
Domain-Specific Design Flows and Tool Chains

Semantic Foundation;

Metaprogrammable Tools, Environments

Domain-Specific Tools, Tool Chains

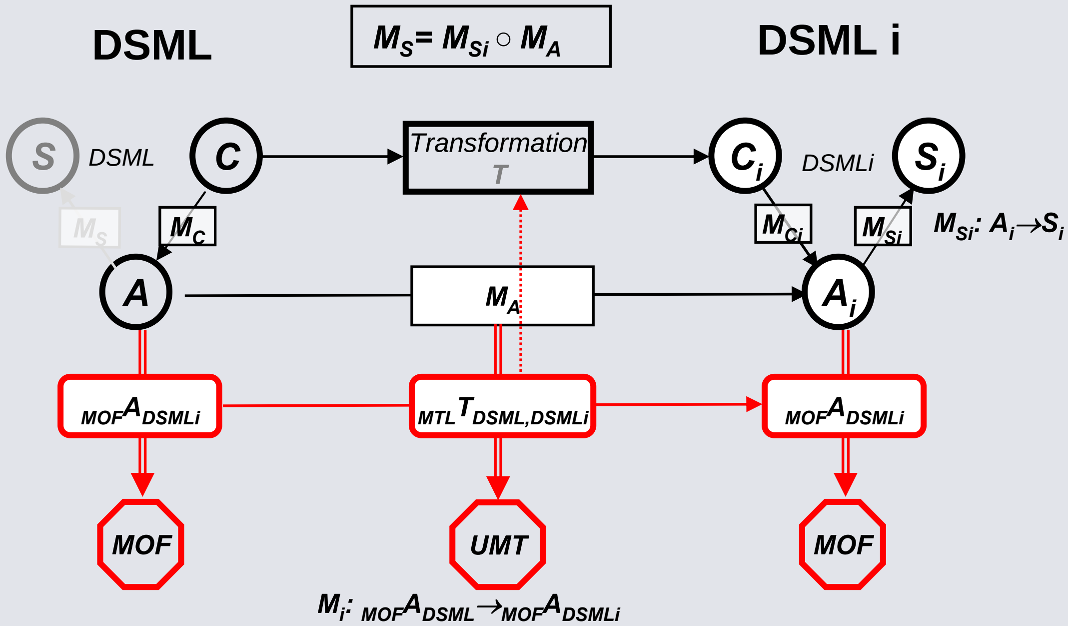
Metaprogrammable Tool Suite



Intersection with Tool Composition Dimensions

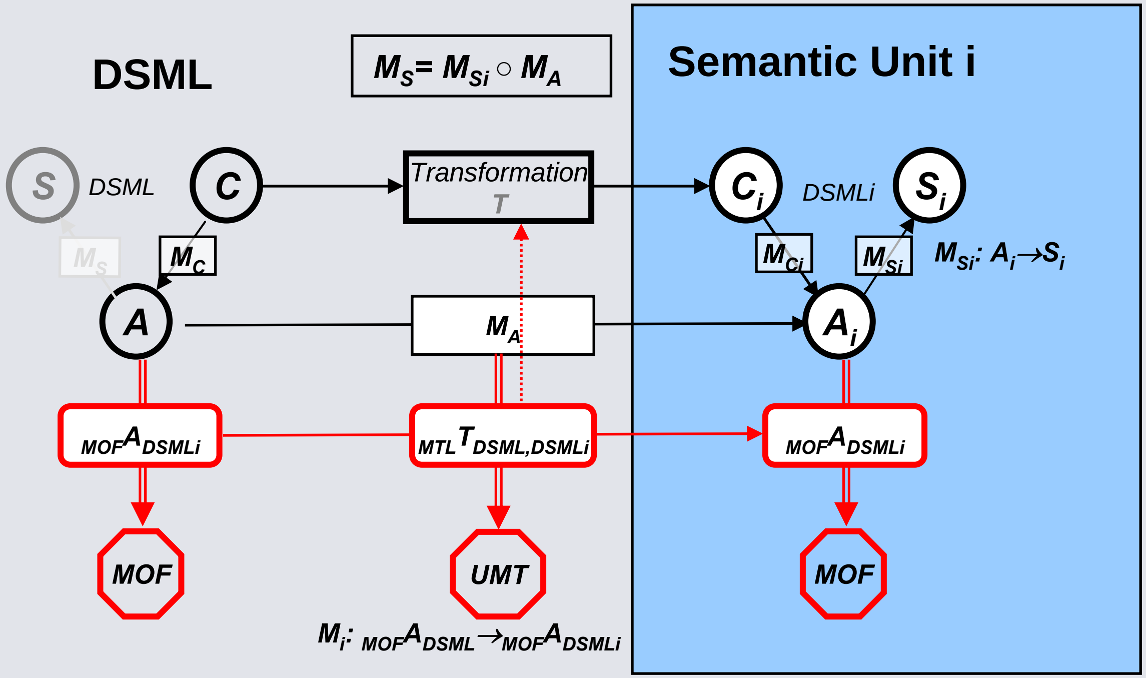
	Semantic Foundation;	Metaprogrammable Tools, Environments	Domain-Specific Tools, Tool Chains
Component Behavior	Semantic Units and Semantic Anchoring	Metamodels and Metaprogrammable Tool Suites	Domain-Specific Design Flows and Tool Chains
Structure	Compositional Semantics		
Interaction			
Resource Modeling (Schedule)			

Transformational Specification of Semantics



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

Semantic Units

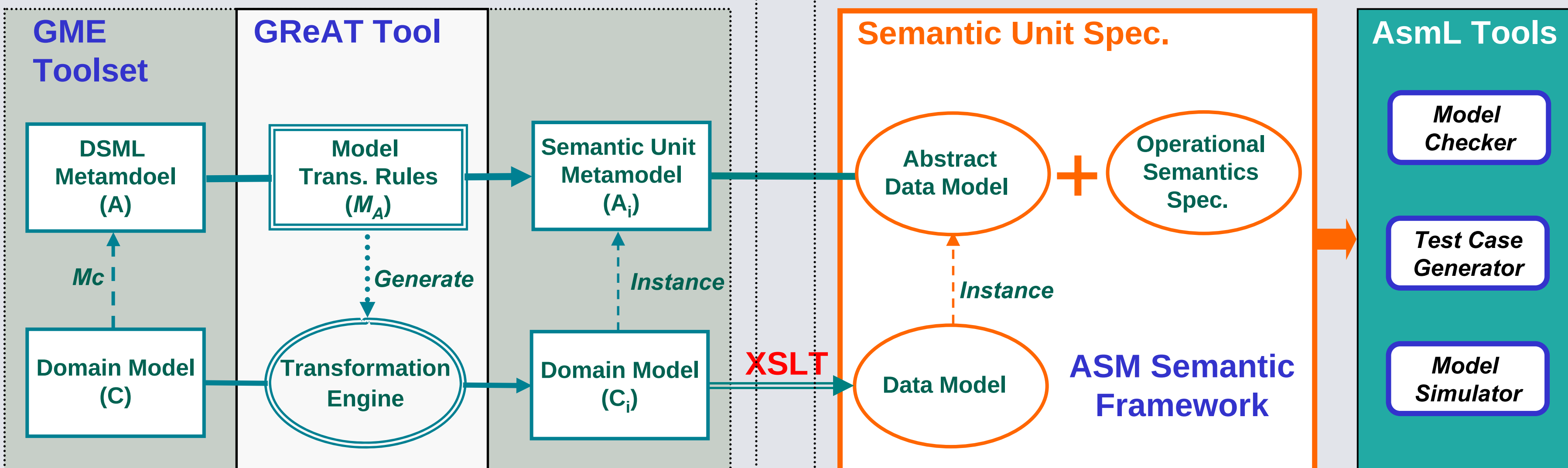


- Ongoing Work
  - Capture the behavioral semantics of a finite set of basic models of computations, such as FSM, DE, TA, SDF…
  - Further exploration of the concept of abstract semantics with Berkeley.

Semantic Anchoring Tool Suite

Metamodeling and Model Transformation Tools

Formal Framework for Semantic Units Specification



- Metaprogrammable tools
- Syntactic manipulations

- Abstract State Machine (ASM) (Evolving Algebras)
- AsmL tool suite

Intersection with Tool Composition Dimensions

Component Behavior

Structure

Interaction

Resource Modeling (Schedule)

Semantic Units and Semantic Anchoring

Compositional Semantics

Metamodels and Metaprogrammable Tool Suites

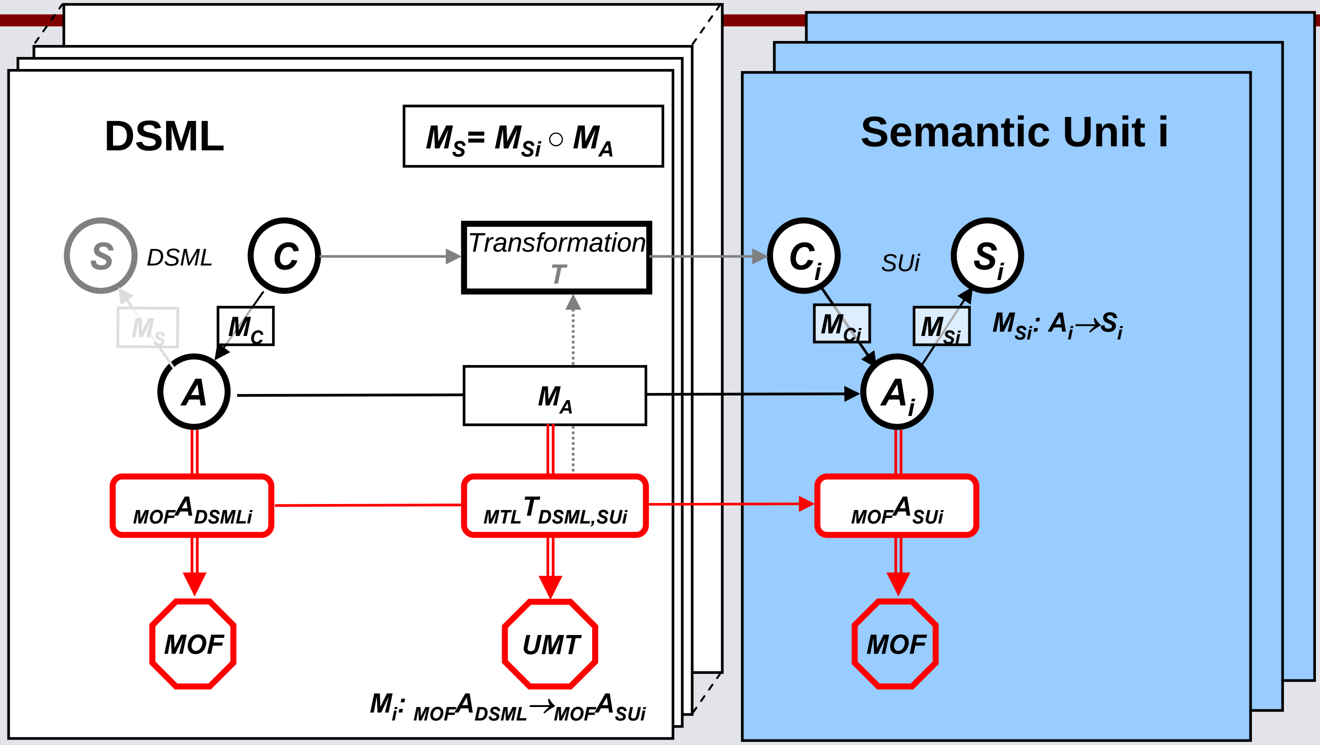
Domain-Specific Design Flows and Tool Chains

Semantic Foundation;

Metaprogrammable Tools, Environments

Domain-Specific Tools, Tool Chains

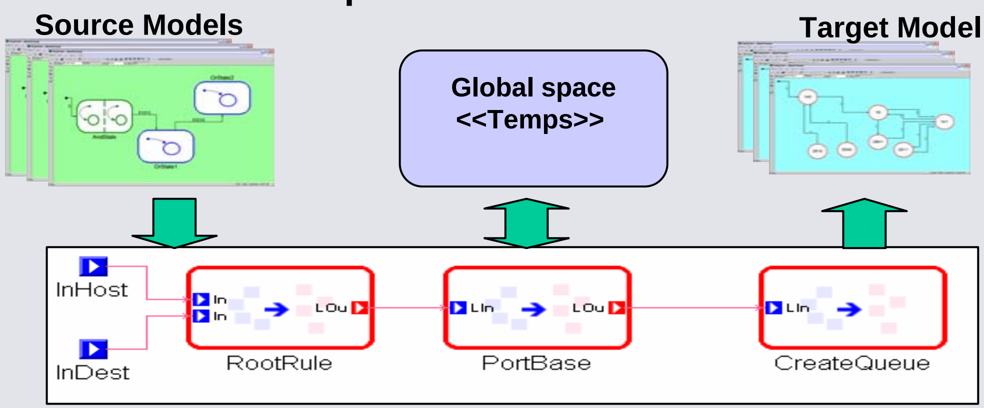
DSML Design Through Semantic Anchoring



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

Progress in Model Transformations: GReAT

Global "spaces" for Transformations

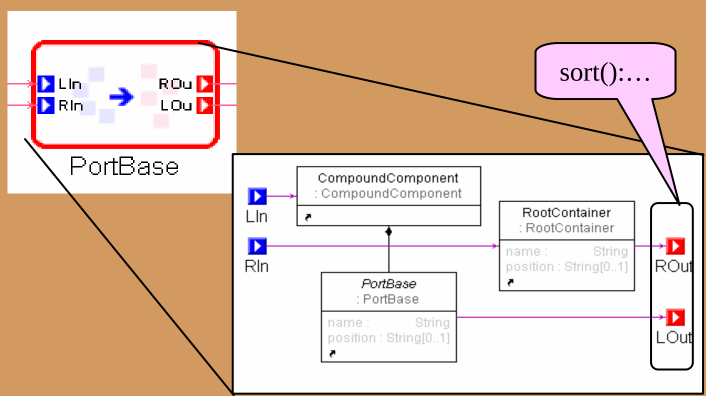


Global spaces hold intermediate results of the transformation
Consequence: The transformations are simplified.

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

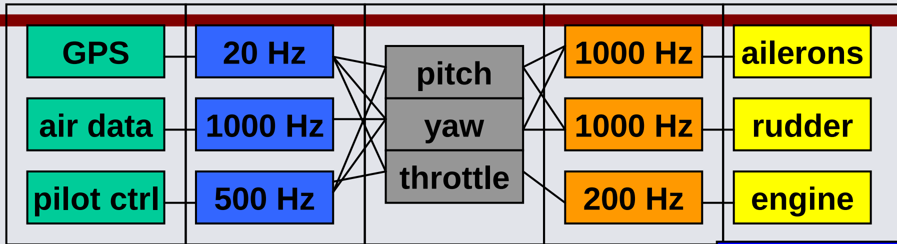


A transformation rule typically operates on a *sequence* of matched objects that could be sorted **after** the rule is applied.
Consequence: Model transformation results are ordered by the sorting function.

Additional new capabilities:

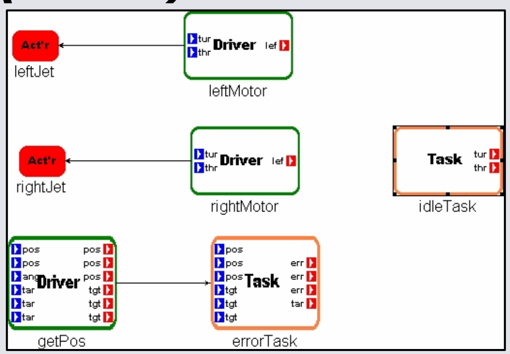
- Distinguished cross-product: a new built-in operator of the language that refines pattern matching semantics
- Match-any associations: "wild-card" pattern matching construct for matching arbitrary associations
- User code libraries
- Support for automatic connection of multi-ported objects in the modeling 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

Model Transformations for Hard-Real Time Languages: *Control System Problem*

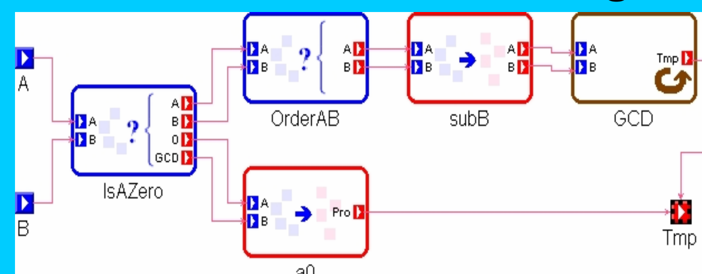


Proof of concept:
A full "compiler" formally specified

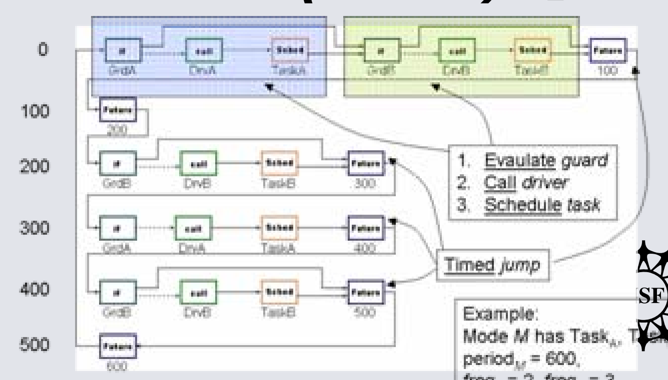
Time-triggered Model (Giotto)



Model Transformation "Program"

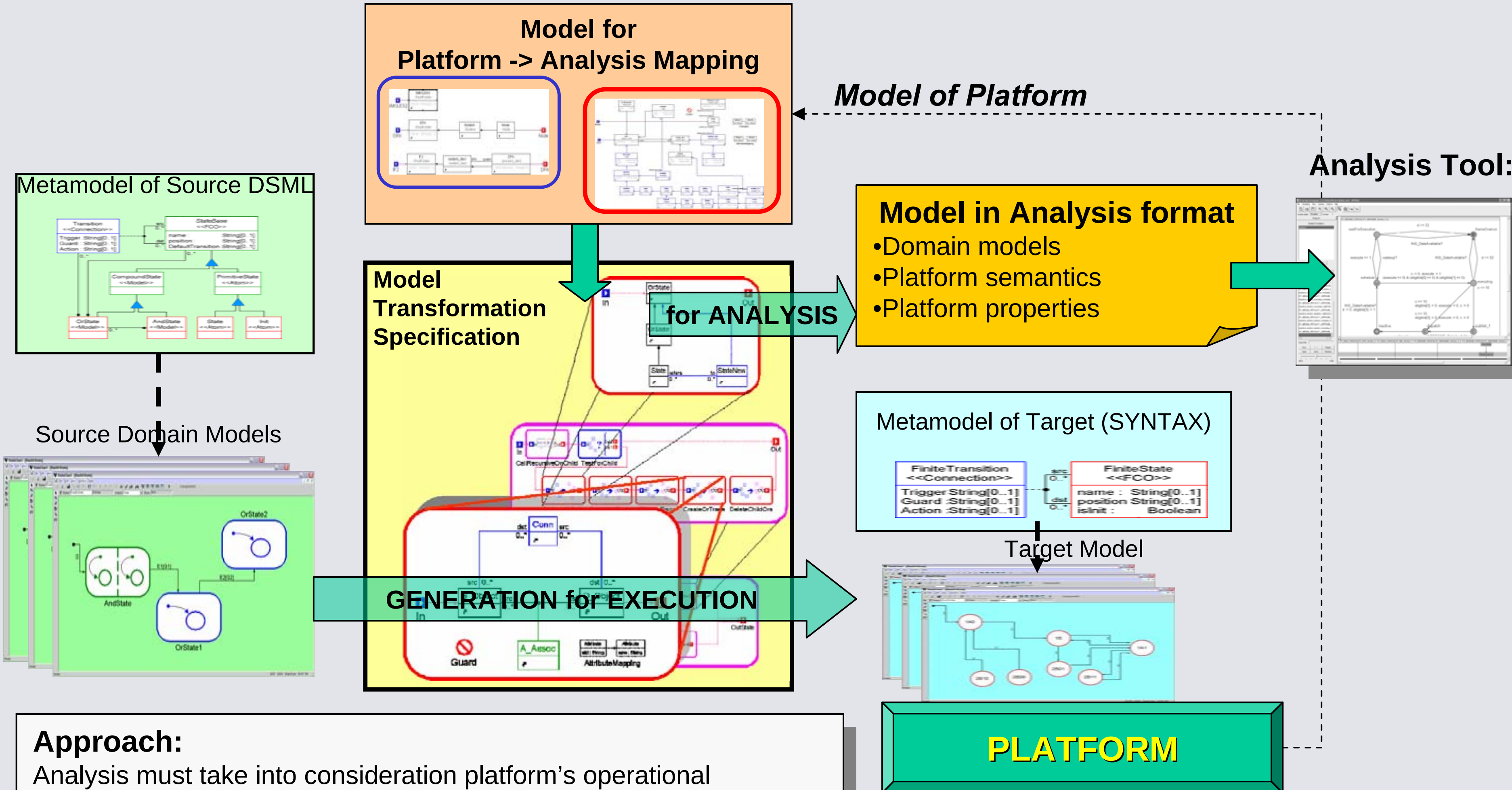


Schedule (E-code)



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 it's the E-machine (as 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.

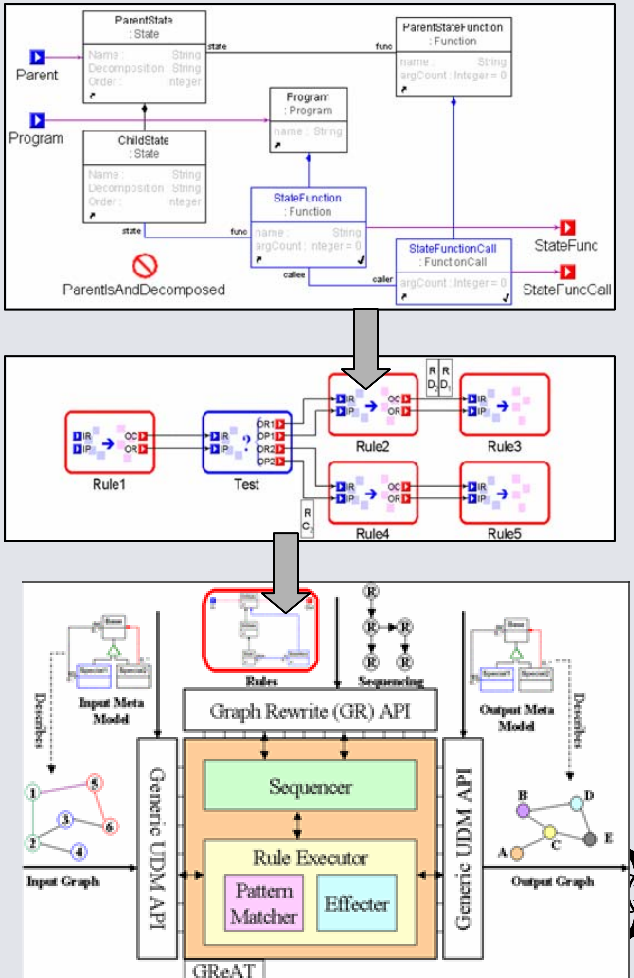
Model Transformations in Toolchains: Platform Modeling & Analysis



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



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