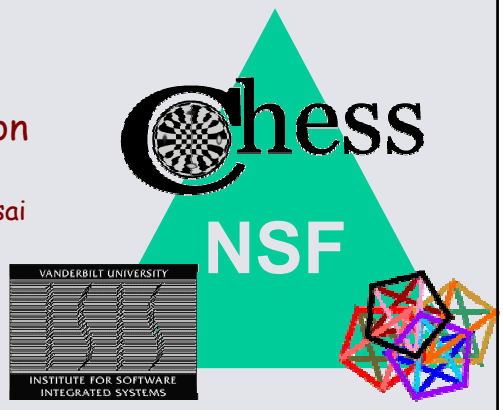
Model-Based Design

DSML Composition
Model Synthesis and
Model Transformation

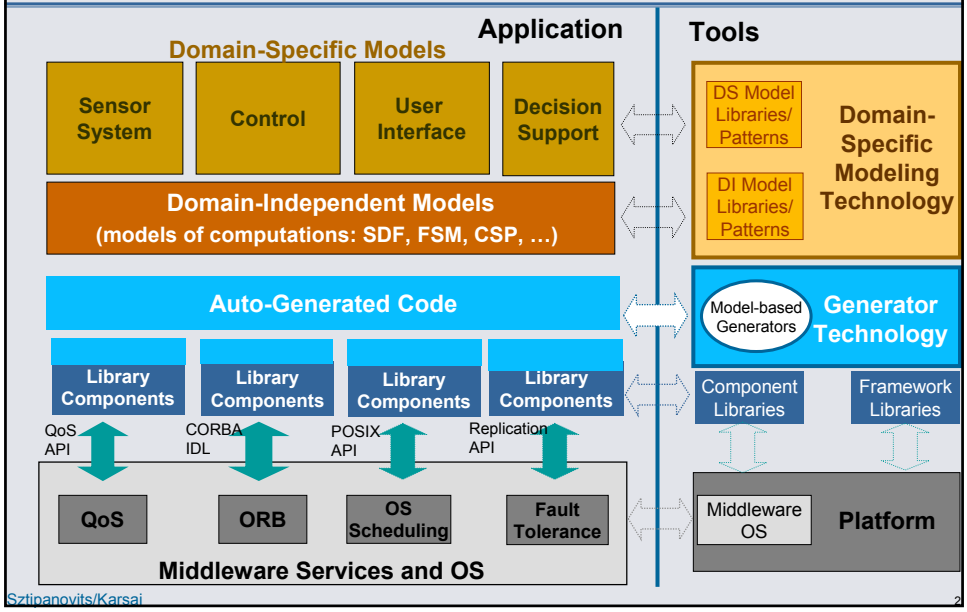
Janos Sztipanovits and Gabor Karsai
November 14,2002



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

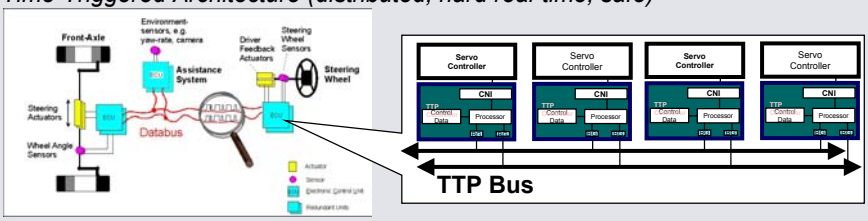
Foundations of Hybrid and Embedded Software Systems

Model-Based Design and Composition



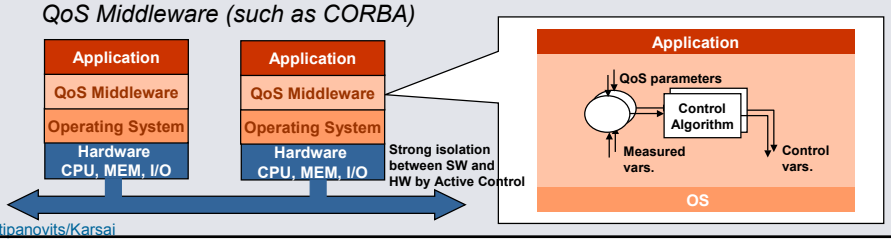
Platforms (There are many…)

Time-Triggered Architecture (distributed, hard real-time, safe)

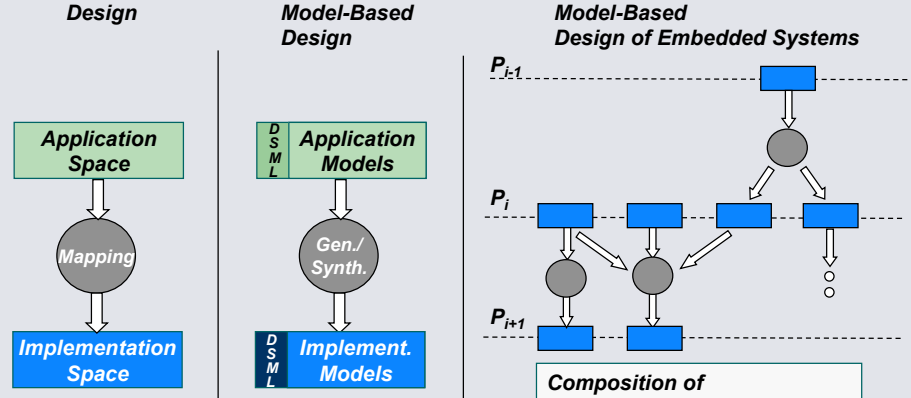


Integration framework, composition mechanisms, components

QoS Middleware (such as CORBA)



Challenges in Model-Based Design

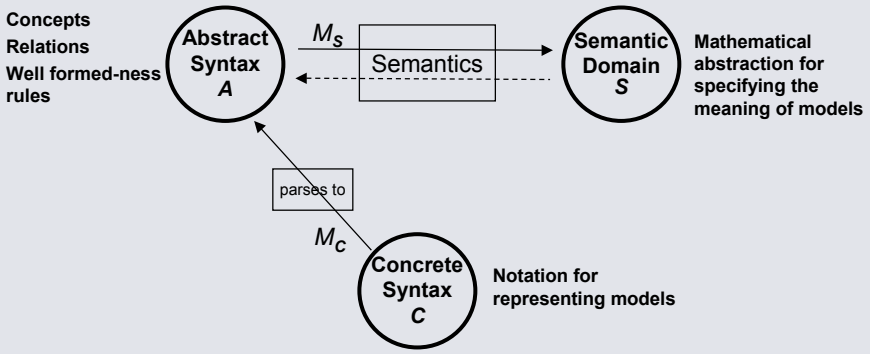


Composition of

- ***Domain Specific Modeling Languages (DSML)***
- ***Model Synthesis***
- ***Model Transformation***

Specification of Domain Specific Modeling Languages (DSML)

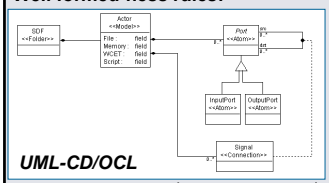
$L = < C, A, S, M_S, M_C >$



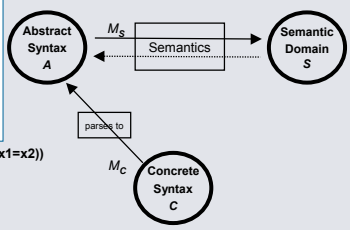
Concrete Syntax and Abstract Syntax

Concepts, Relations
Well formed-ness rules:

Signal Flow Language (SF)

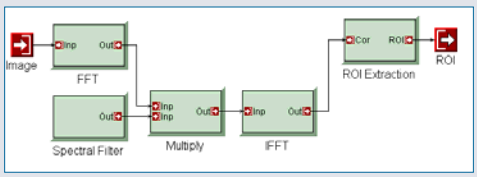


Self.InputPorts()→forAll(ip | ip.src()→forAll(x1,x2 | x1=x2))



Mathematical abstraction for specifying the meaning of models

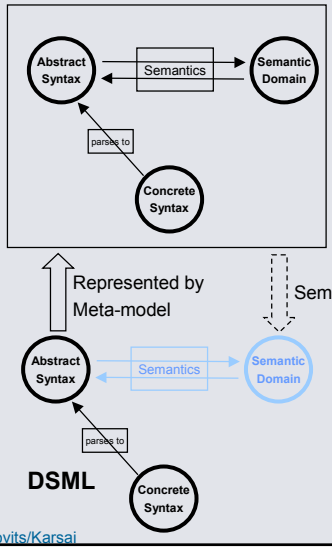
But What About S?



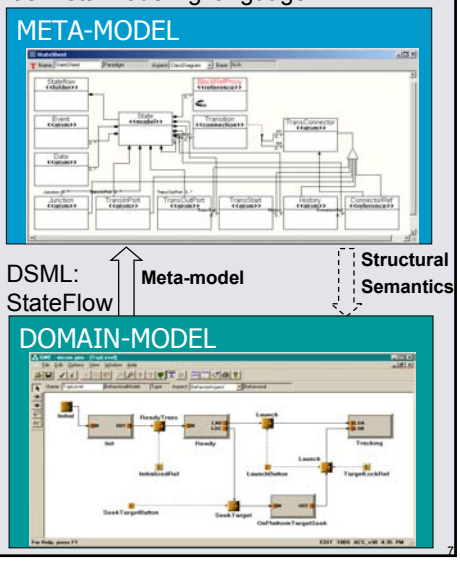
Notation for representing models:
E.g.: ***Block Diagram***

Semantics via Meta-Modeling

Meta-modeling language with well-defined semantics

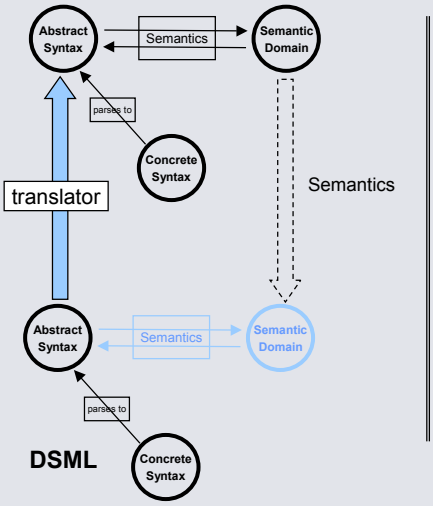


Meta-Model of StateFlow using uml/OCL as meta modeling language.

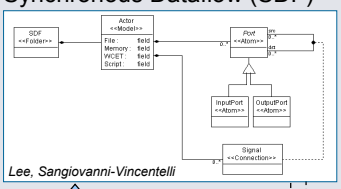


Semantics via Translation

Modeling language with well-defined semantics

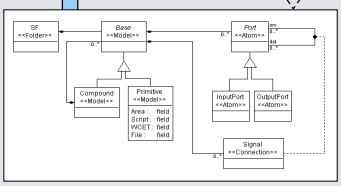


Synchronous Dataflow (SDF)



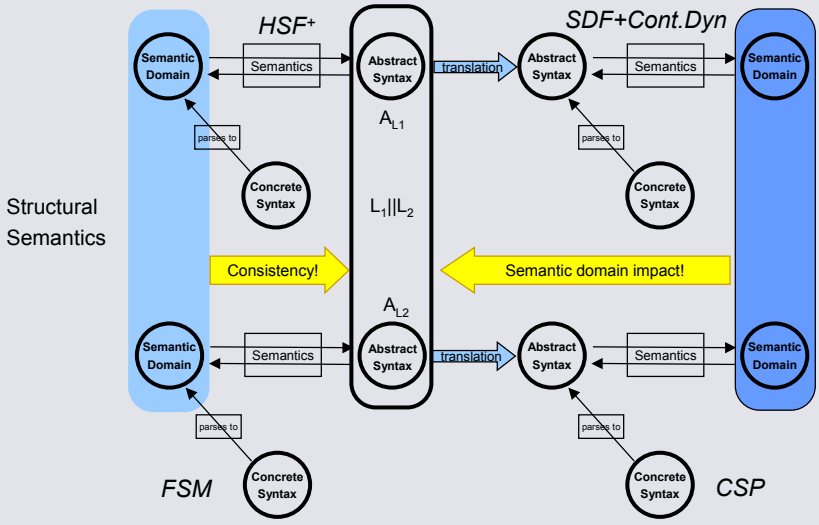
translator

Behavioral Semantics



Hierarchical Signal Flow (HSF)

DSML Composition

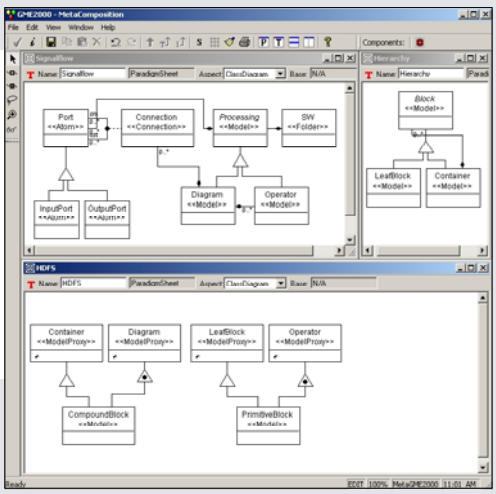


Simple Support for Compositional Meta-Modeling

Composition Operators

Operator	Symbol	Informal semantics
Equivalence		Complete equivalence of two classes
Implementation Inheritance		Child inherits all of the parent's attributes and those containment associations where parent functions as container.
Interface Inheritance		Child inherits all associations except containment associations where parent functions as container.

Metamodel composition with GME

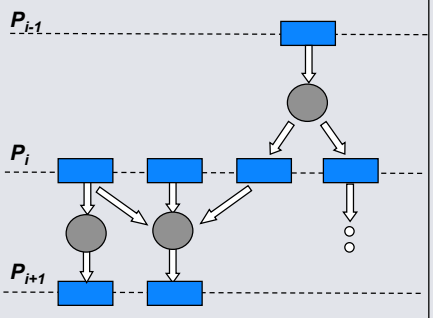


Research Agenda on Domain Specific Modeling Languages

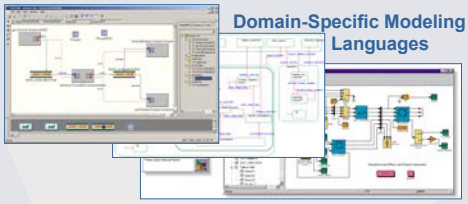
- Precise, compositional meta-modeling
- Multiple aspect modeling in the compositional meta-modeling framework
- Practical issues:
  - Examples, meta-model libraries
  - Meta-programmable tools
  - Link to UML-2

Model Synthesis and Transformations

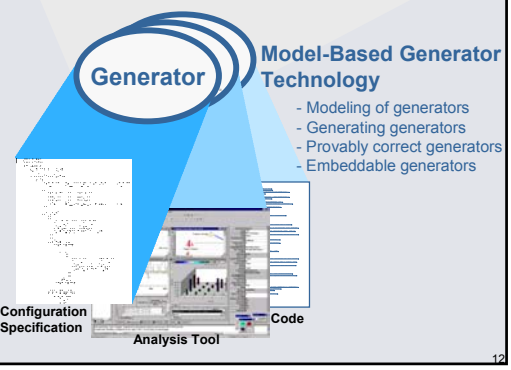
Model-Based Design of Embedded Systems



- ***Model Synthesis***
- ***Model Transformation***



Domain-Specific Modeling Languages



Model-Based Generator Technology

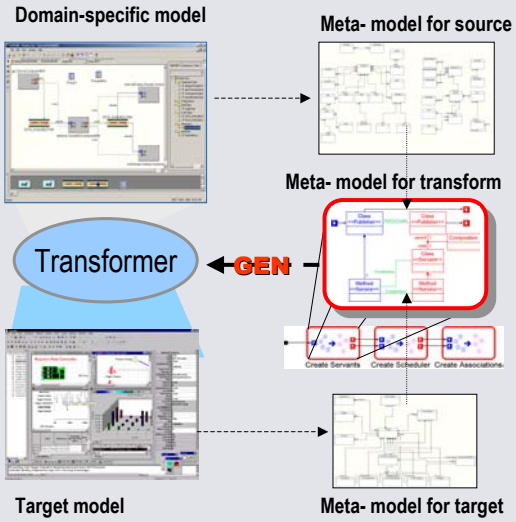
- Modeling of generators
- Generating generators
- Provably correct generators
- Embeddable generators

Meta-generators: *Model Transformations in Tool Integration*

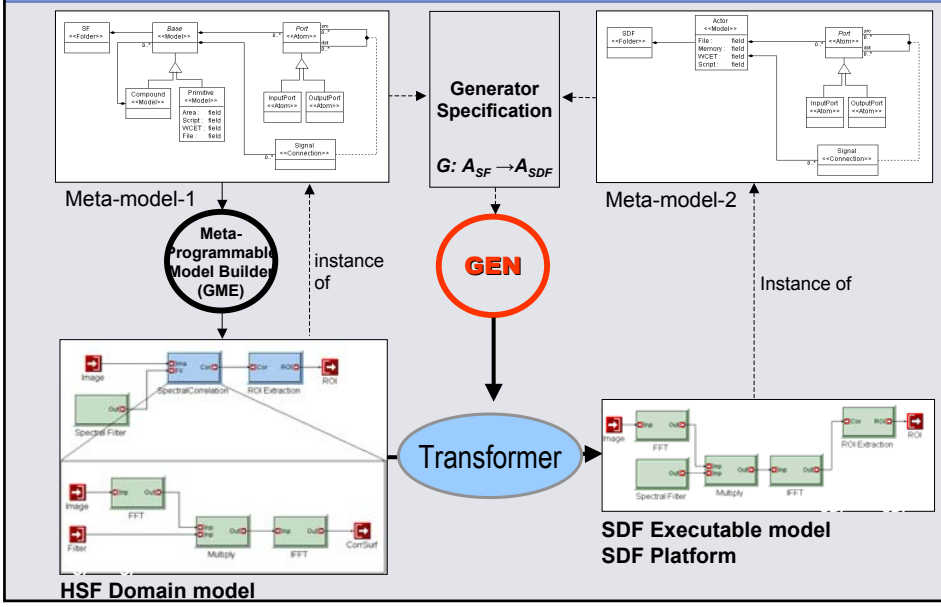
Roles transformations play in model-based design:

- Refining a design into an implementation
- Code generation
- PIM -> PSM mapping
- Support for model interchange for tool integration

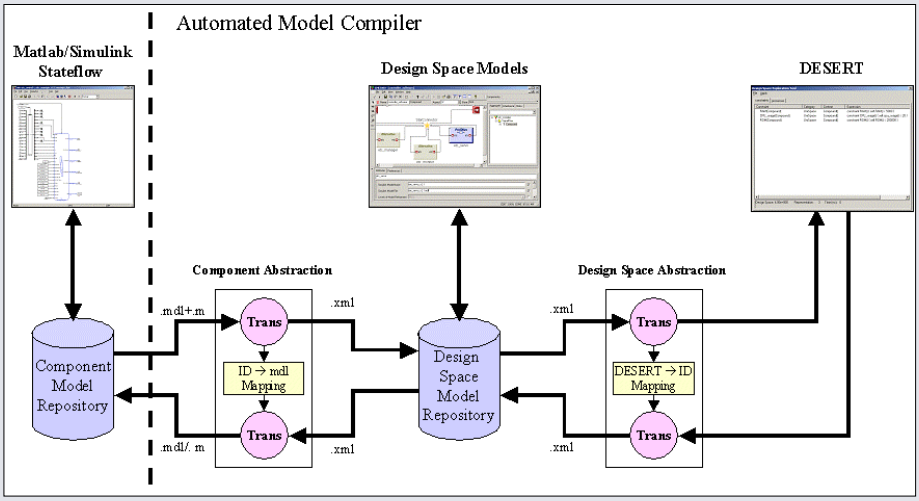
Approach (Karsai): **Meta-models for source and target models plus transformations, then generating the transformer**



Meta-generators: *Model Transformations in Component Integration*

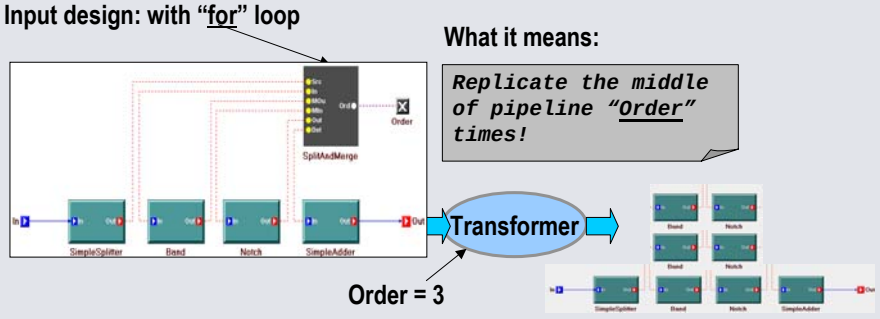


Constraint-Based Model Synthesis



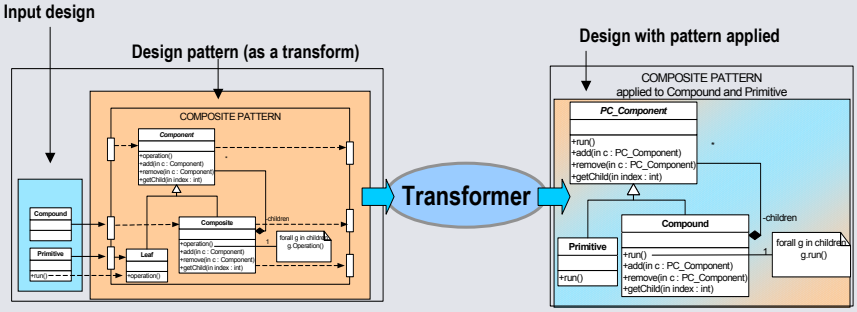
Generative modeling:
Extensions to Meta-Modeling Languages

Iterative ("for") and conditional ("if")constructs in models can greatly enhance expressiveness while reducing complexity.



Other Uses of Transforms on Models: Design Patterns

Design patterns capture prototypical solutions to a design problem. As such, they are *parametric* and *reusable*.



Research Agenda on Model Transformations

- Languages and tools for meta generators
- Model synthesis using explicit design patterns
- Model synthesis using constraint-based design-space exploration
- Generative modeling extensions to languages
- Embeddable generators