Institute for Software Integrated Systems

Vanderbilt University

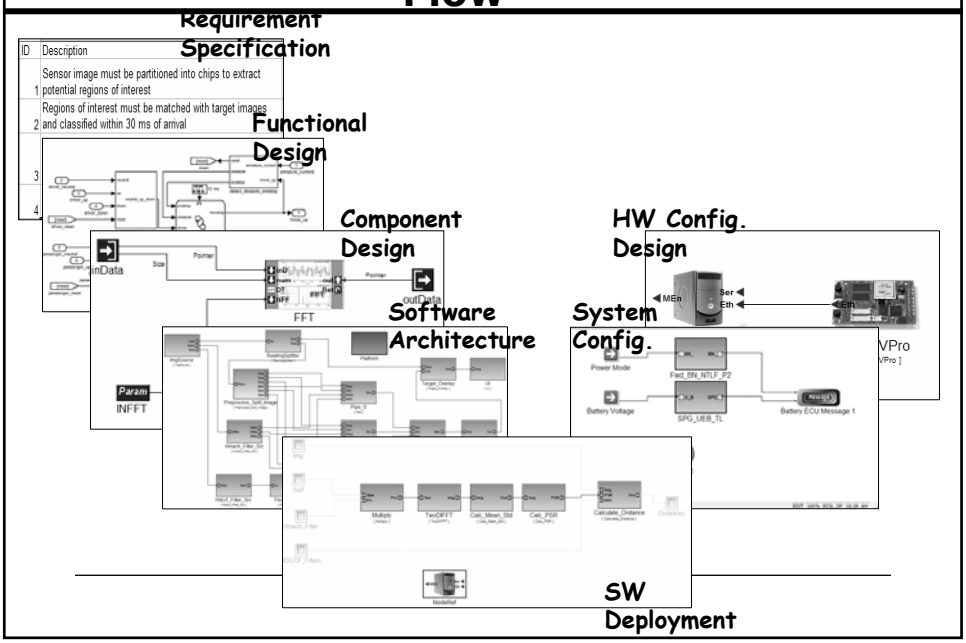
Semantics in Model-Based Design

Janos Sztipanovits
ISIS, Vanderbilt University

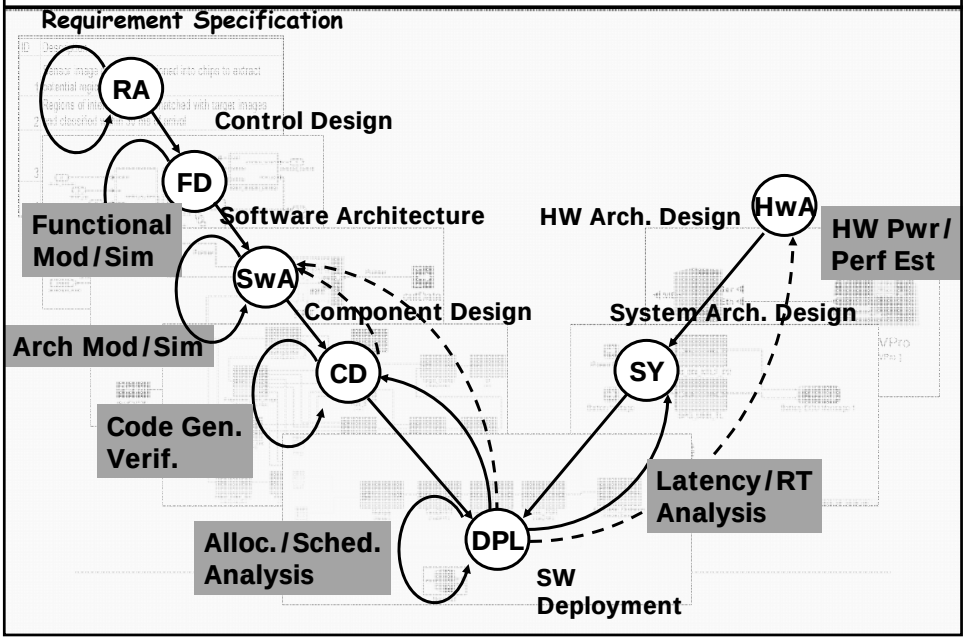
Content

- Application Drivers
- Composition on the Abstract Syntax Layer
- Composition on the Semantic Layer
- Road Ahead

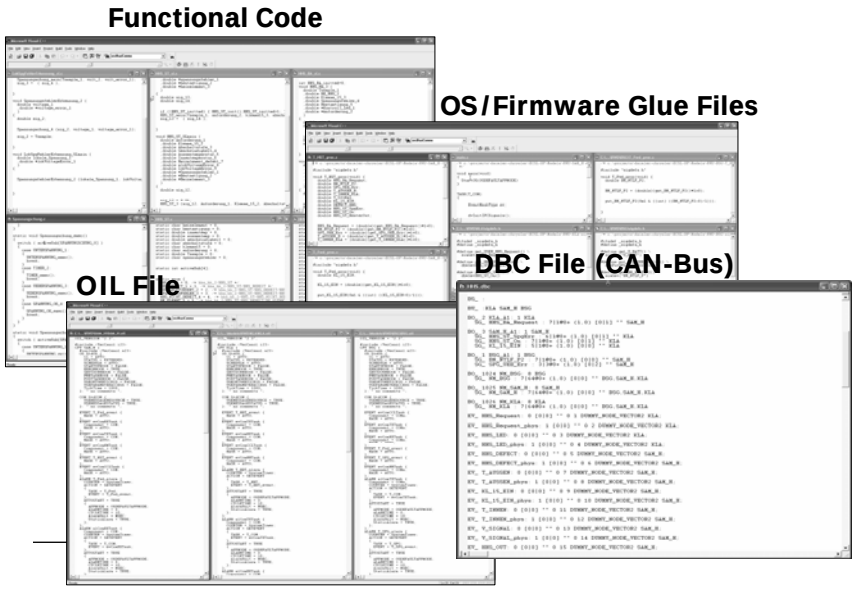
An Embedded System Design Flow



Related Tools and Analyses



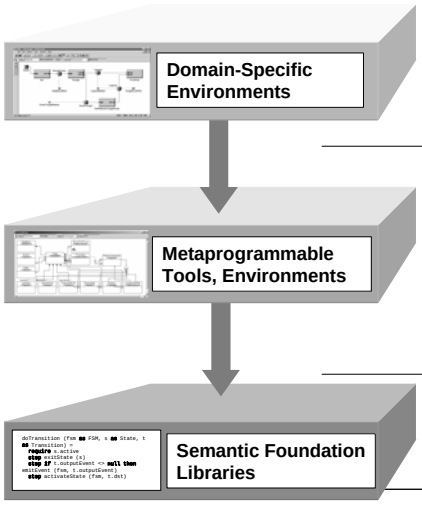
Tool Chain Outputs



Content

- Application Drivers
- Composition on the Abstract Syntax Layer
- Composition on the Semantic Layer
- Road Ahead

Layers in Model-Integrated Computing

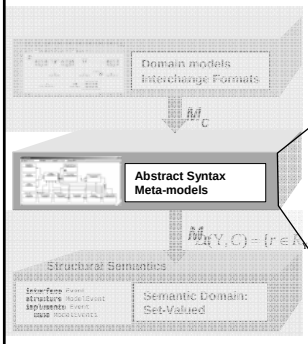


Modeling Domain Specific Design Flows:
Examples in MIC:
- **ECSL - Automotive**
- **ESML - Avionics**
- **SPML - Signal Processing**
- **CAPE/eLMS – Learning Technology**
- **AADL….**

Metamodeling and Metaprogrammable Tools (mature or in maturation program)
- **GME (Generic Model Editor)**
- **GReAT (Model Transformation)**
- **OTIF (Tool Integration Framework)**
- **UDM (Universal Data Model)**
- **DESERT (Design Space Exploration)**

Modeling Semantics (work in progress):
- **Semantic "Units"**
- **Semantic Anchoring**

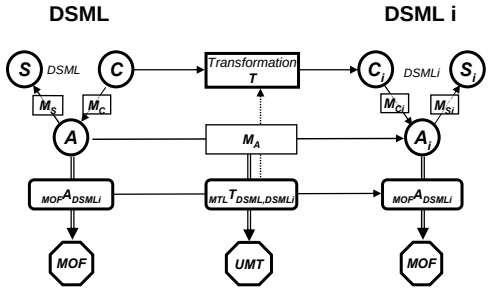
Syntactic Layer



- Modeling & Metamodeling
- Model Data Management
- Model Transformation
- Tool Integration
- Design-Space Exploration

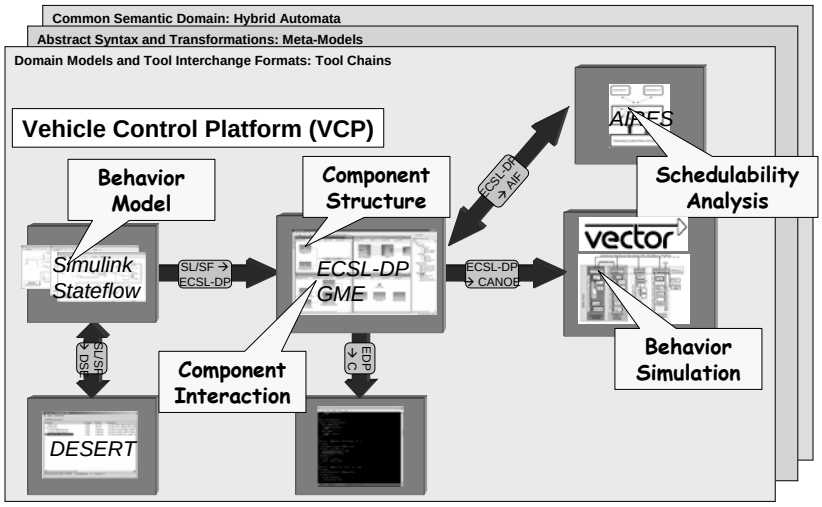
Core Components of the Syntactic Layer

- Methods and tools for modeling



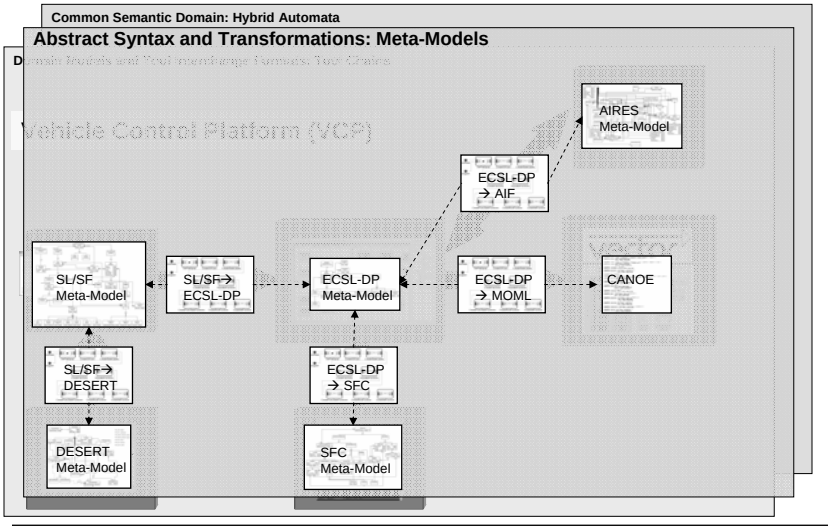
- Methods and tools for metamodeling
  - Industrial standard metamodeling language: MOF (or some others)
- Methods and tools for model transformations
  - Formal metamodel transformation language: Unified Model Transformation Language.

"Front-End View" of a Typical MIC Tool Chain

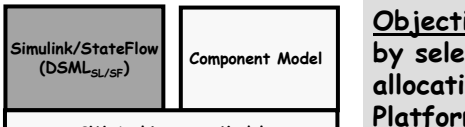


(Experimental tool chain)

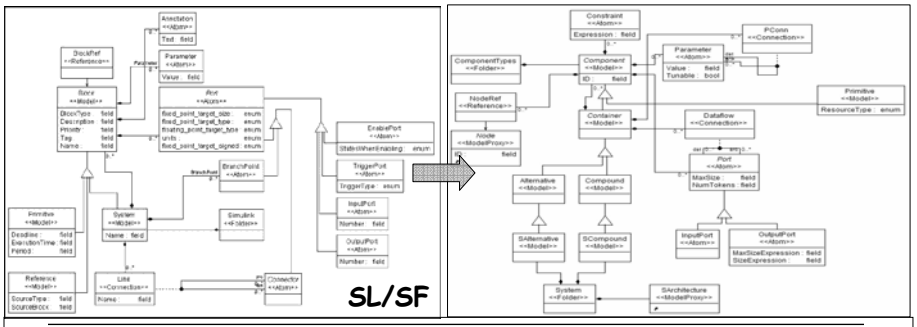
"Back-plane View" of a Typical MIC Tool Chain



Need for DSML Composition



Objective: Optimize the SW architecture by selecting a component model and by allocating functions to components.
Platform: Heterogeneous Dataflow Component Model
Tools:
GME, GReAT, C Compiler, WCET Analyzer

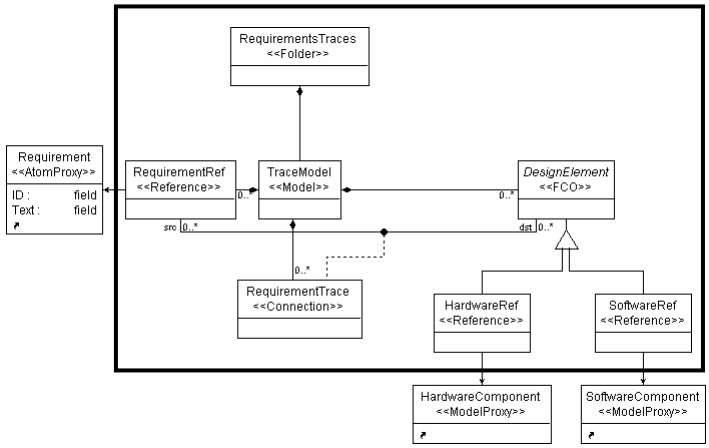


Functional blocks – SW Component Mapping

Compositional Metamodeling

- Goal: Composing modeling languages (not models)
- Metamodel composition methods in the Generic Modeling Environment (GME):
  - Class Merge
  - Metamodel Interfacing
  - Class Refinement
  - Template Instantiation
  - Metamodel Transformations

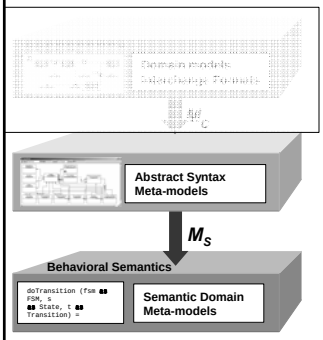
Example: Metamodel Interfacing



Content

- Application Drivers
- Composition on the Abstract Syntax Layer
- Composition on the Semantic Layer
- Road Ahead

Semantics of DSMLs: Current Status

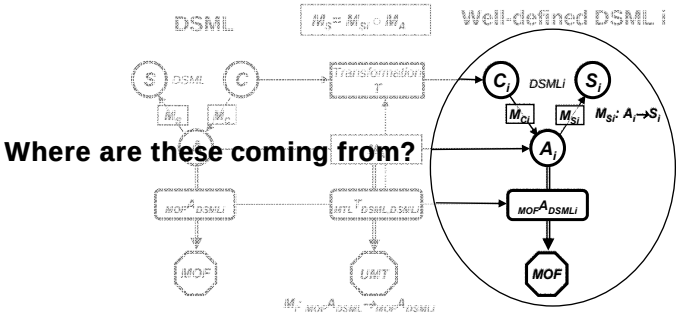


- Usually, specification stops at the level of abstract syntax metamodels ("static semantics")
- Specification of behavioral semantics (if done)
  - involve major effort due to overly complex modeling languages,
  - use a wide range of formalisms and
- Impact is far-reaching
  - tool chains are closed and built around loosely defined "conventions" and proprietary interpretations of semantics instead of standards
  - potential semantic mismatches create unacceptable risk for safety critical applications

Major roadblock that slows down acceptance of model-based design technology.

Transformational Specification of Semantics

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

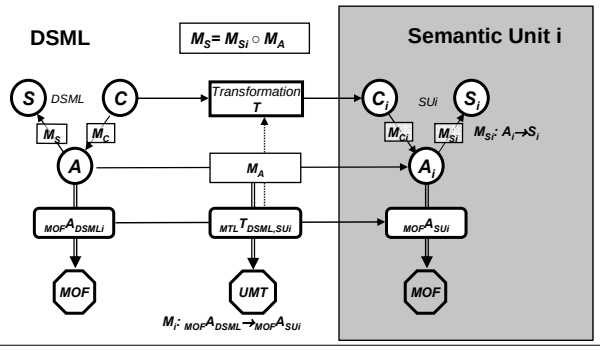


- Use a formal metamodeling framework
  - Industrial standard metamodeling language: MOF
  - Formal metamodel transformation language: Unified Model Transformation Language.

System Composition Dimension: Core Modeling Aspects

Component Behavior	**Modeled on different levels of abstraction:** **• State-based modeling (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 Allocation	**Mapping/deploying components on platforms:** **• Dynamic Priority** **• Behavior guarantees** **• Research: composition of schedulers**

Semantic Units



- Semantic Units
  - Capture the semantics of a finite set of basic dynamic behavioral categories, such as FSM, DES, TA.
  - Capture the semantics of basic interaction categories (Model of Computation, MoCs), such as SDF, PN, ...
  - Specified in a formal semantics framework.
  - Develop a set of simple modeling languages for the semantic units.

Search for a Formal Framework

- Specification style: Operational semantics
- Solid mathematical foundation
- Tool support for core use cases:
  - Readability (clear syntax and understandable semantics)
  - Validation/exploration of semantics (executable specification)
  - Verification of semantic equivalence (generation of "reference traces", integratability)

After evaluating several frameworks (Z, TLA+,..) we selected ASM and the AsmL tool suite (MSR).

Example: Synchronous Data Flow

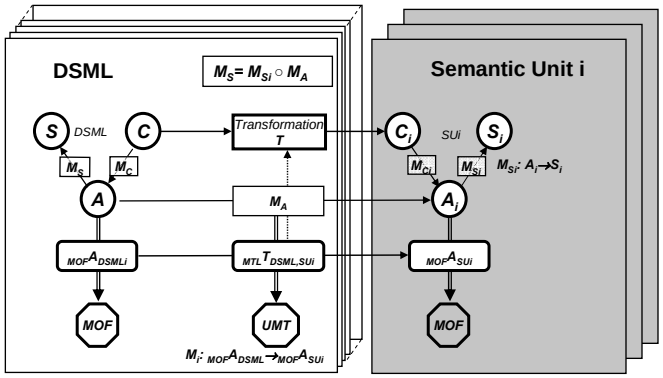
Abstract Data Model

```
structure Value
  case IntValue
    v as Integer
  case DoubleValue
    v as Double
  case BoolValue
    v as Boolean

//Data Token, it may contain a value or a null data
structure Token
  value as Value?

//Data Port, when exist is true, the port has an effective data token
class Port
  id        as String
  var token as Token   = Token (null)
  var exist as Boolean = false

//Data Channel connecting two data ports
class Channel
  id as String
  srcPort as Port
  dstPort as Port

//A Node is a basic unit is the Data Flow. It may be an action or a Guard
abstract class Node
  id as String

  abstract property inputPorts  as Seq of Port
    get
  abstract property outputPorts as Seq of Port
    get
  //The Run method takes tokens from its input ports, do actions and set output
  //tokens in the output ports
  abstract Fire ()


//Dynamic Data Flow Semantic Unit
abstract class SDF
  id as String

  abstract property nodes as Set of Node
    get
  abstract property channels as Set of Channel
    get
  abstract property inputPorts as Seq of Port
    get
  abstract property outputPorts as Seq of Port
    get
```

Model Interpreter

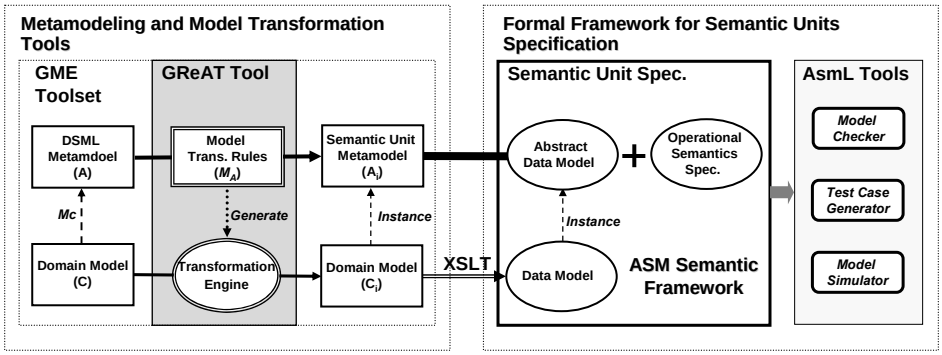
```
Run (n as Node)
    require n in me.EnabledNodes ()
    step
      n.Fire ()
    step
      if exists p in n.inputPorts where p.exist then
        error ("After the firing of a node, all input tokens should be consumed
by the node.")
    step
      if exists p in n.outputPorts where not p.exist then
        error ("After the firing of a node, each of its output port should have
one output token.")
    step
      forall c in me.channels where c.srcPort.exist
        if c.dstPort.exist then
          error ("A input port receives more than one token.")
        else
          WriteLine ("Channel " + c.id + " is sending data tokens.")
          c.dstPort.token := c.srcPort.token
          c.dstPort.exist := true
        c.srcPort.exist := false

  //Return all nodes in the SDF that have all its required data tokens to fire.
  EnabledNodes () as Set of Node
    return {n | n in me.nodes where forall p in n.inputPorts where p.exist}

  Initialize ()
    forall p in me.inputPorts where p.exist
      forall c in me.channels where p.id = c.srcPort.id
        c.dstPort.token := c.srcPort.token
        c.srcPort.exist := false
        c.dstPort.exist := true

  ClearPorts ()
    forall c in me.channels
      if c.srcPort.exist then
        c.srcPort.exist := false
      if c.dstPort.exist then
        c.dstPort.exist := false
```

DSML Design Through Semantic Anchoring



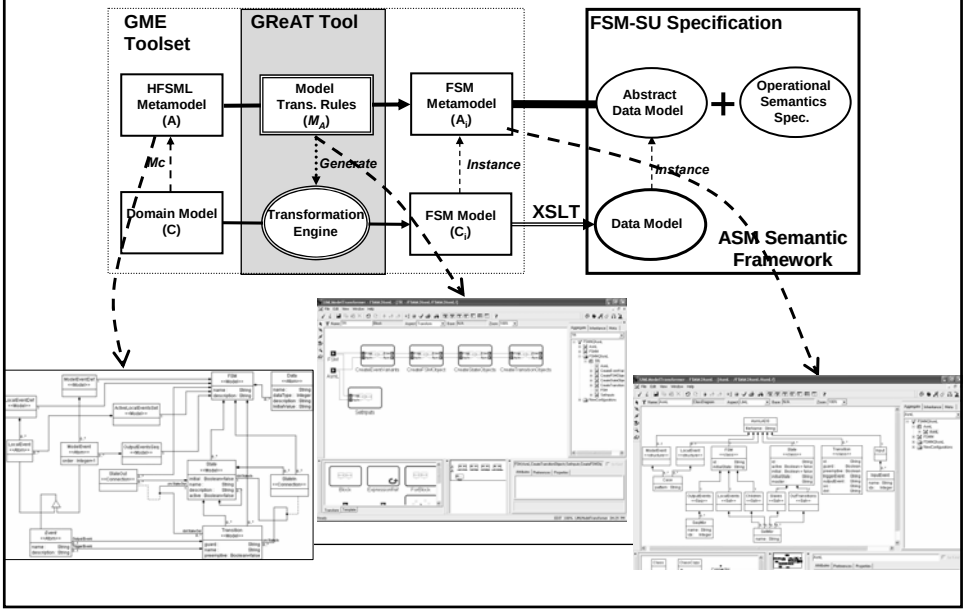
- Step 1
  - Specify the DSML <*A, C, M_c*> by using MOF-based metamodels.
- Step 2
  - Select appropriate semantic units *L = < A_i, C_i,* M_{Ci}, S_i, M_{Si}> for the behavioral aspects of the DSML.
- Step 3
  - Specify the semantic anchoring M_A = A -> A_i by using UMT.

Experimental Tool Suite for Semantic Anchoring

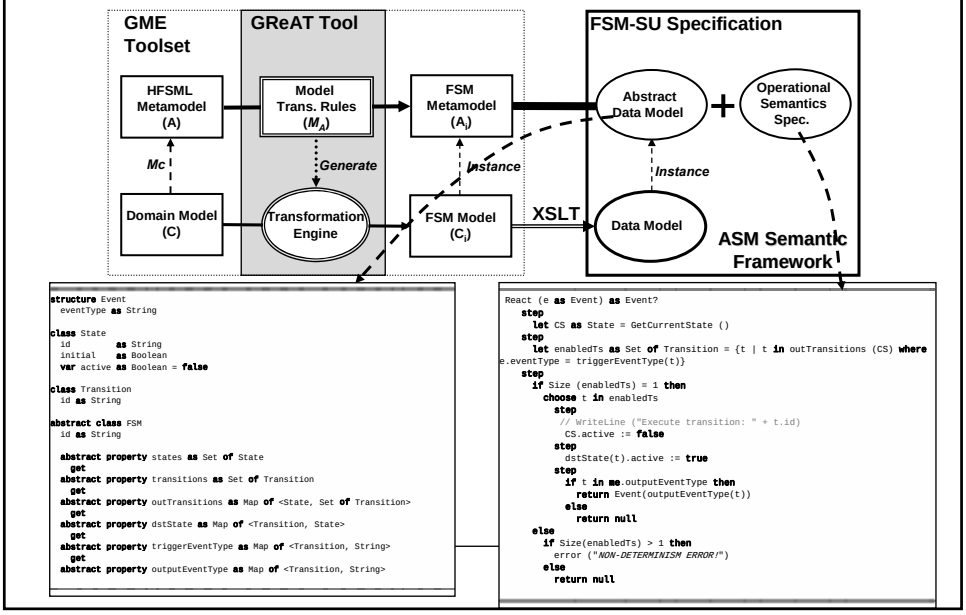


- Metamodeling and Model Transformation Tools
  - GME: Provide a MOF-based metamodeling and modeling environment.
  - GReAT: Build on GME for metamodel to metamodel transformation.
- Tools for Semantic Unit Specification
  - ASM: A particular kind of mathematical machine, like the Turing machine. (Yuri Gurevich)
  - AsmL: A formal specification language based on ASM. (Microsoft Research)

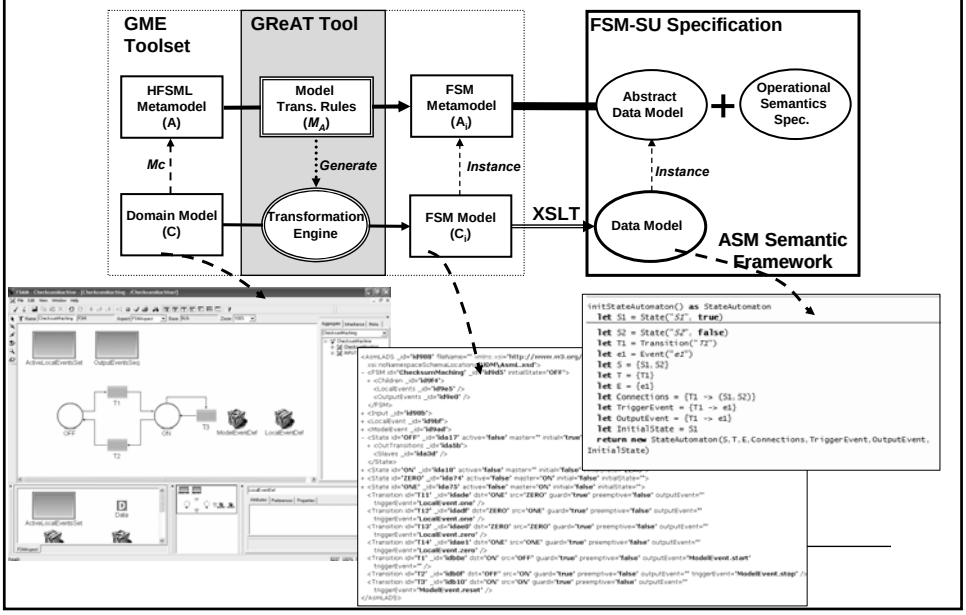
Example: HFSML => FSM-SU



Example: HFSML => FSM-SU



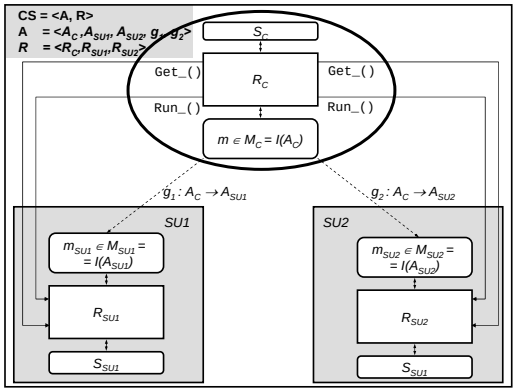
Example: HFSML => FSM-SU



Heterogeneous DSMLs

- Heterogeneity of systems
  - Complex systems are composed of heterogeneous components using heterogeneous interactions. Modeling and design of heterogeneous systems is a significant challenge.
- Heterogeneity of tool chains
  - Tool chains supporting domain-specific design flows integrate modeling, analysis and synthesis tools using DSMLs with overlapping semantics.
- The semantics of a heterogeneous DSML is probably not captured by a single predefined semantic unit.

Compositional Specification of Semantics

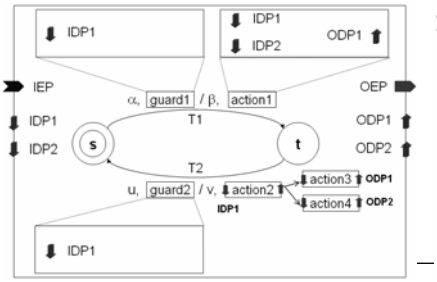


Remark: The behavioral composition specifies a controller, which restricts the executions of actions. Since the behavior of the component semantic units can be described as partial orders on the sets of actions they can perform, the behavioral composition can be modeled mathematically as a composition of the partial orders.

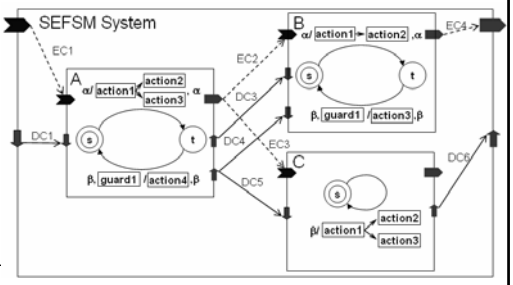
- Structural Composition yields the composed Abstract Data Model, $A =< A_C, A_{SU1}, A_{SU2}, g_1, g_2 >$ where *g1*, *g2* are the partial maps between concepts in A_C, A_{SU1}, and A_{SU2} .
- Behavioral composition is completed by the R_C set of rules that together with R_{SU1} and R_{SU2} form the R rule set for the composed semantics.

Example: EFSM

- EFSM has been developed by GMR to specify vehicle motion control (VMC) software.
- The SEFSM model is a synchronous reactive system including a set of components communicating through event channels and data channels.
- A SEFSM component is an FSM-based model, which integrates a set of stateless computational functions that consume input data and produce output data.
- Events determine which components are to be activated and the order of activations.
- Primary contributor: Kai Chen, Motorola Research

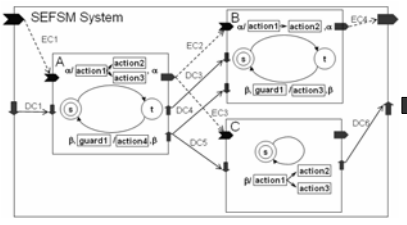


A SEFSM Component Model

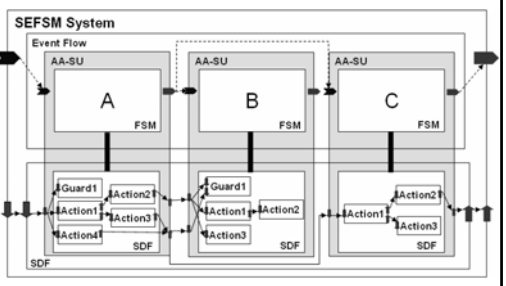


A SEFSM System Model

The Compositional Structure for SEFSM Systems



A SEFSM Conceptual Structure



A Compositional Structure

- A SEFSM system is composed of a set of components, which communicate with each other through event channels and data channels.
- The semantics of SEFSM systems is defined as the composition of FSM-SU and SDF-SU

Content

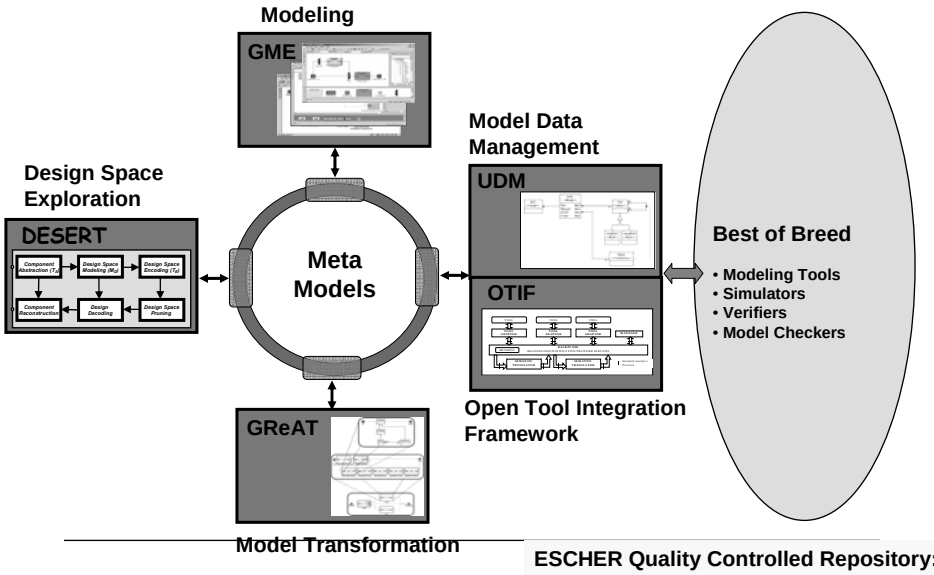
- Application Drivers
- Composition on the Abstract Syntax Layer
- Composition on the Semantic Layer
- Road Ahead

Road Ahead

- Continue the selection and specification of a suite of Semantic Units.
- Explore and develop theory for compositional specification of semantics.
- Initiate standardization with strong involvement of IT and Systems industry.
- Strengthen/expand international collaboration.
- Develop an experimental DSML Design Tool Suite.

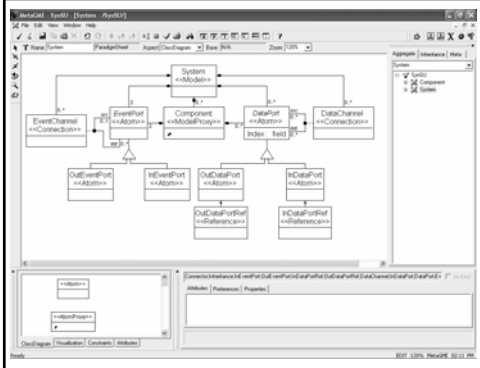
Appendices

Integrated MIC Tool Suite

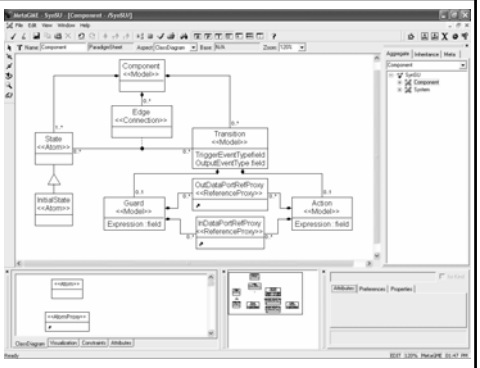


ESCHER Quality Controlled Repository:
http://escher.isis.vanderbilt.edu

SEFSM Metamodel



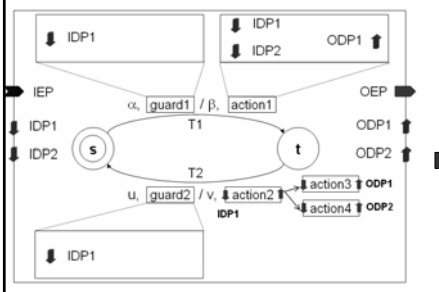
Metamodel defining the SEFSM system structure



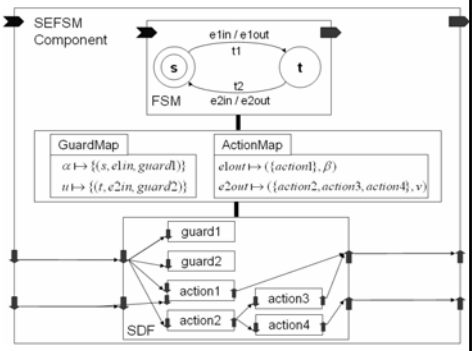
Metamodel defining the SEFSM component structure

- The abstract syntax metamodels and the textual description of the behavior are insufficient for the precise understanding of the semantics of SEFSM.
  - For example, the metamodel specification does not reveal the complex interdependency between the event flow and the data flow structure of the components.

The Compositional Structure for SEFSM Components



A SEFSM Component Model



A Compositional Structure

- The behavior of SEFSM components can be divided into two different behavioral aspects: the FSM-based behavior expressing reactions to events and the SDF-based behavior controlling the execution of computational functions (actions and guards).

A Derived Semantic Unit: AA-SU

- The behavioral semantics specification for SEFSM components actually specifies the semantics of a common behavioral category that captures elementary reactive computation behaviors.
- We can consider the compositional semantics specification of SEFSM components as a new derived semantic unit, called Action Automaton Semantic Unit (AA-SU).