Model-Based Design

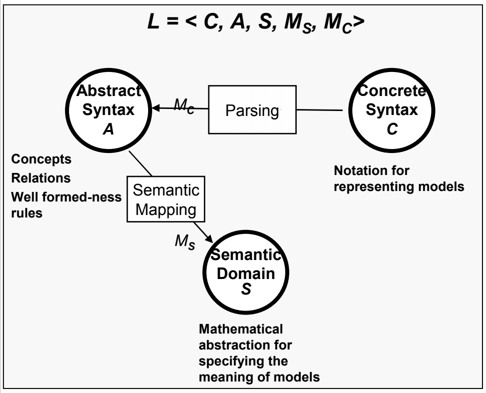
Edited and Presented by
Janos Sztipanovits, Co-PI
ISIS, Vanderbilt University

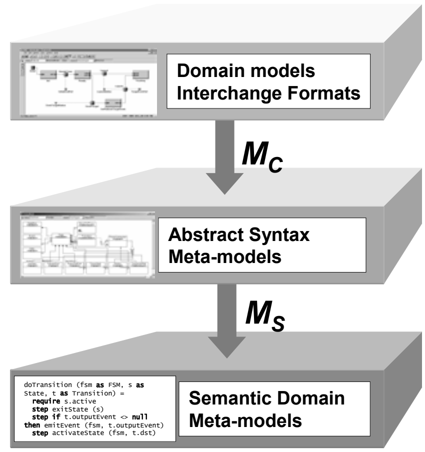
Chess Review
November 18, 2004
Berkeley, CA

Model-Based Design

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

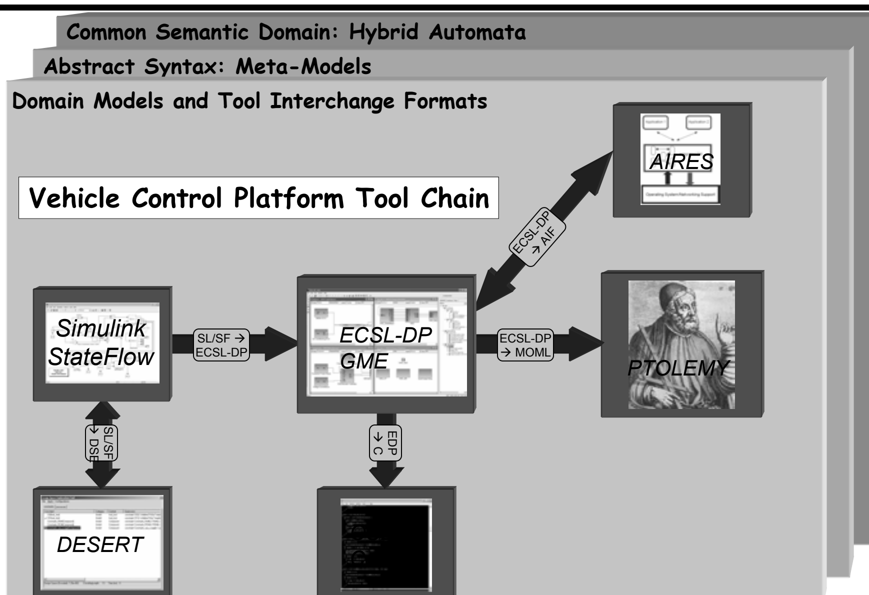
Domain Specific Modeling Languages (DSML)





DSML-s are the foundations for model-based design

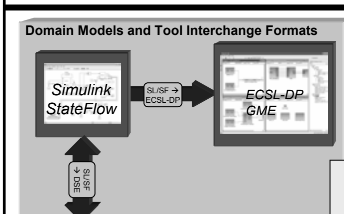
Tool Chain Example

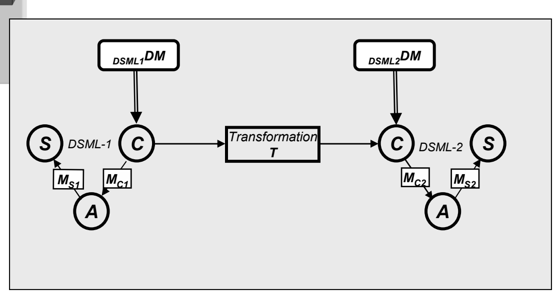


Research Agenda in Model-Based Design

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

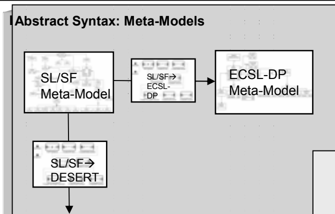
Constructing Design Flow: Modeling and Transformations



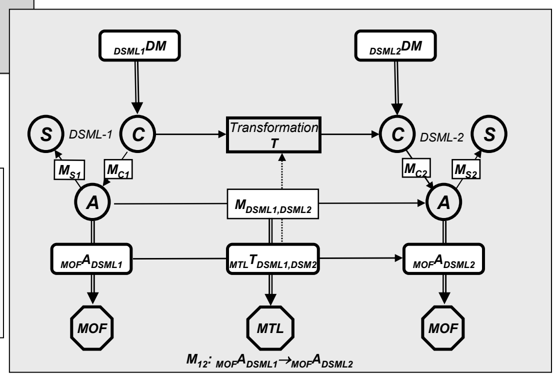


- Large influence of concrete syntax
- No clear role of semantics
- It is not clear what are we doing?

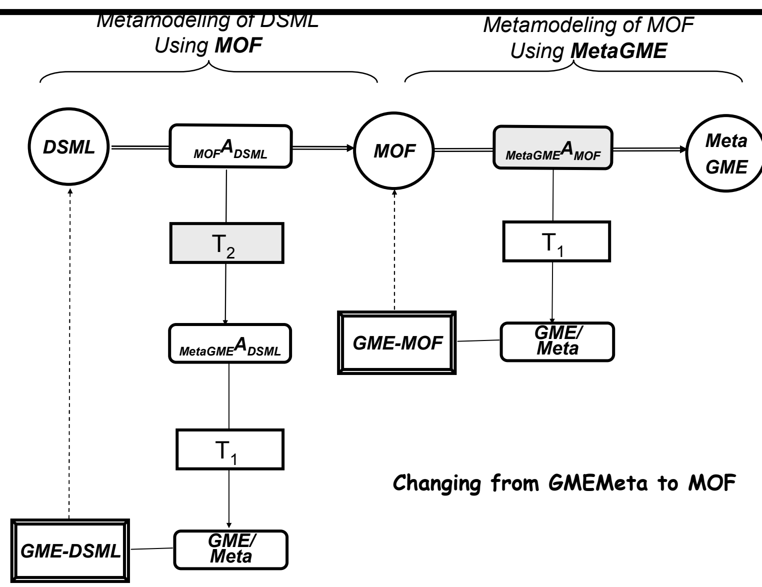
Abstract Syntax Metamodeling



- Gives structural semantics for the models
- Set-valued Semantic Domain for the metamodels and transformations



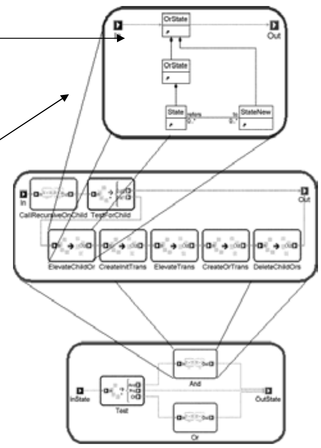
Metamodeling Languages



Changing from GMEMeta to MOF

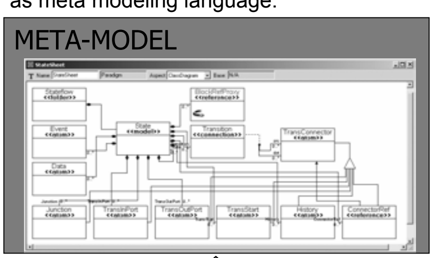
UMT:
A Simple Model Transformation Language

1. Pattern specification
   - Pattern variables are typed with their UML classes
   - Cardinality of association-ends is checked
   - Extra (OCL) constraints define guard conditions
2. Graph transformation and rewrite
   - Create new/delete/modify objects
   - Attribute mapping (procedural)
   - "Cross-links": edges between old/new objects
   - Input/output ports: pre-bound pattern variables
3. "High-level" control flow over the rules
   - Port connections imply "data flow" and control flow
   - Hierarchy/Sequencing/Recursion/Branching



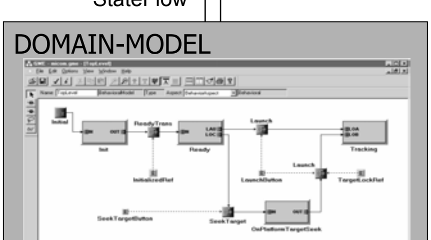
Results: MIC Metaprogrammable Tool

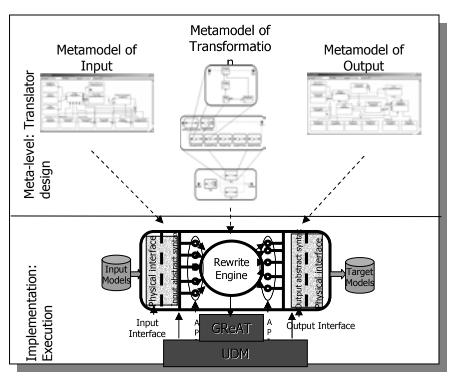
Meta-Model of StateFlow using uml/OCL as meta modeling language.



DSML: StateFlow

Meta-model

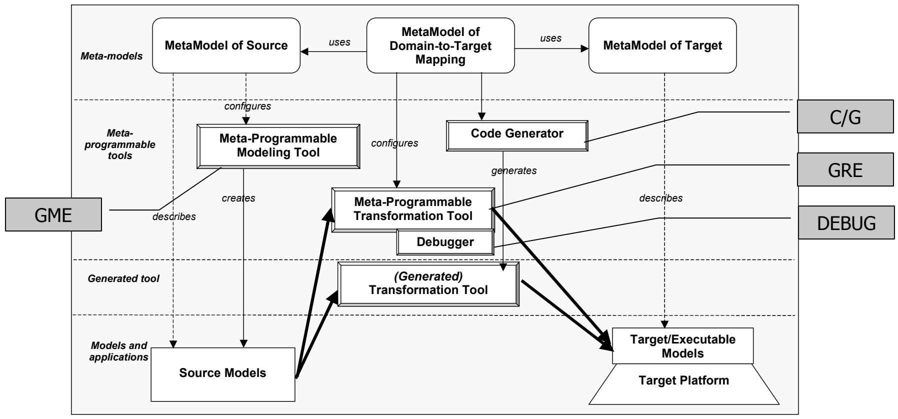




GME, UDM & GREAT

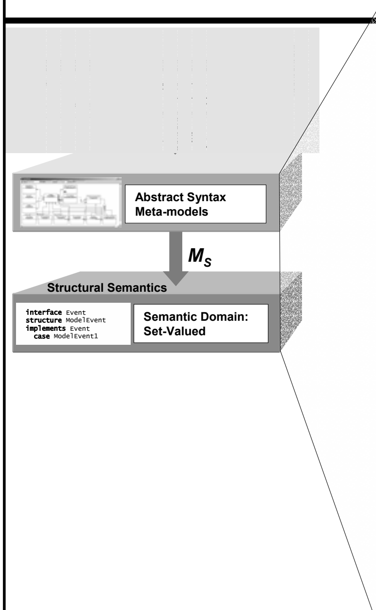
Completed tool suite, available through the ESCHER Repository

Modeling and Model Transformation Tool Chain



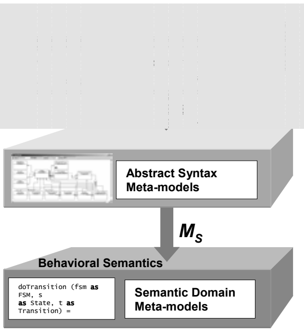
Tools: UMT Language, GRE (engine), C/G, GR-DEBUG

Ongoing Research on DSML-s and Model Transformations

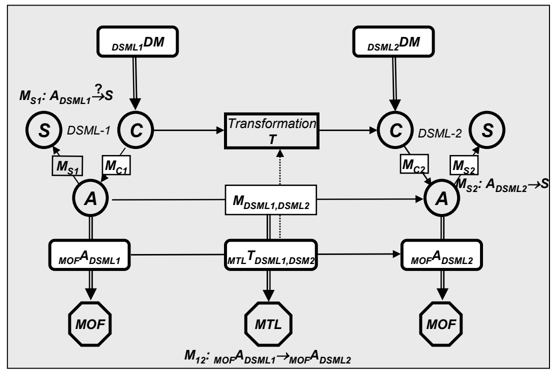


- Compositional construction of Metamodels (inheritance, packages, libraries, operators)
- Compositional construction of Model Transformations
- Multiple Aspect Modeling and modeling of aspect inter-dependences:
  - constraint-based,
  - transformation-based
- Reasoning about properties of transformations
- Formal semantics of transformations
- Platform modeling and use of embedded platform models in transformations
- Pushback reasoning in transformations
- Generation of efficient code from graph transformations
- Transformations for embedded system platforms
- Using graph transformations for embedded component adaptation
- Embedding graph transformations in the run-time platform

Metamodeling and Model Transformation Use Cases

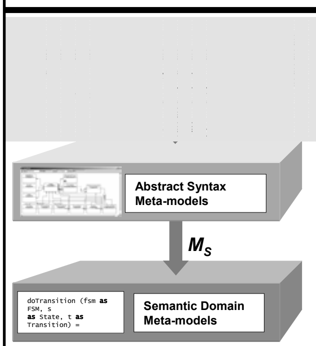


1. Transformational Specification of Behavioral Semantics



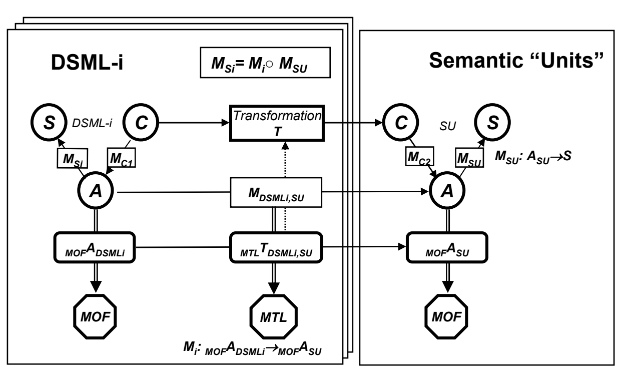
$$M_{S1} = M_{12} \circ M_{S2}$$

Metamodeling and Model Transformations Use Cases



- The "Semantic Units" are MoC-s
- DSML-s or their aspects are anchored to the MoC-s using transformations
- The "Semantic Units" are specified in a formal framework

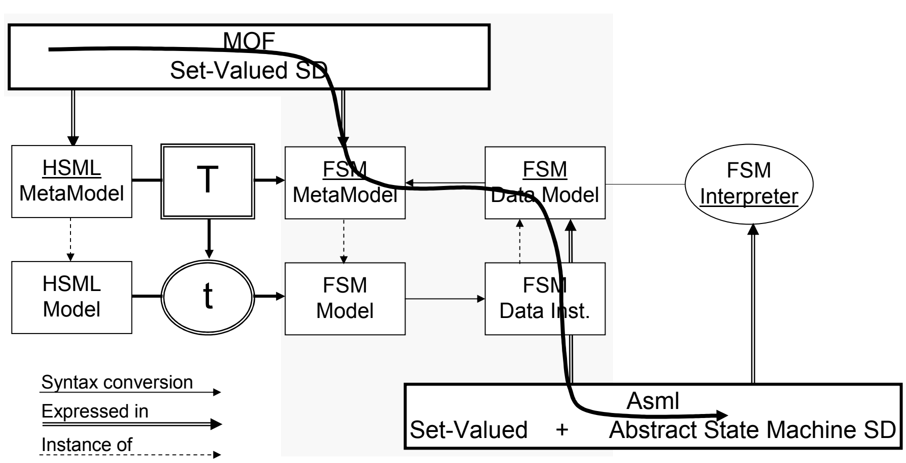
2. Semantic Anchoring of DSML-s



More On Semantic Anchoring

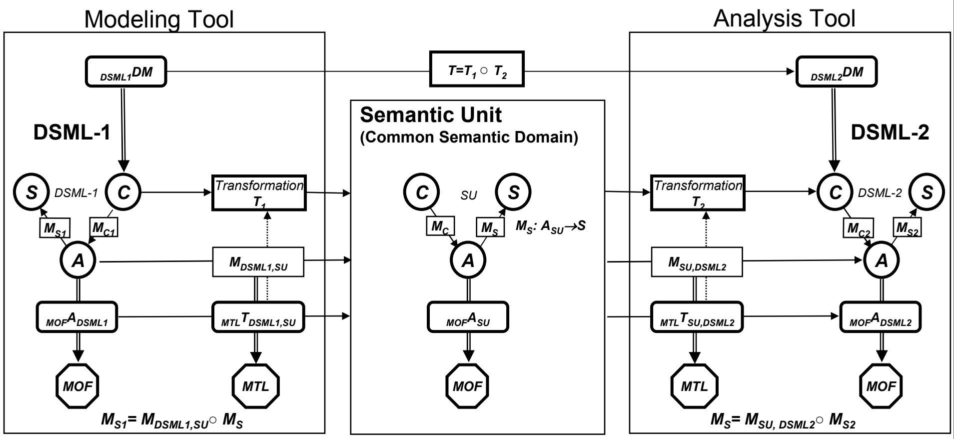
- Step 1
  - Define the DSML metamodel $\langle A, C, M_c \rangle$
- Step 2
  - Select a proper MoC as a "semantic unit" (MoC library): $L_i = \langle A_i, C_i, M_{Ci}, S_i, M_{Si} \rangle$
- Step 3
  - Anchor the semantics: $M_A = A \rightarrow A_i$
  - DSML semantics: $L = \langle A, C, M_c, S_i, M_A \circ M_{Si} \rangle$

Example: HSML (Ptolemy II)->FSM

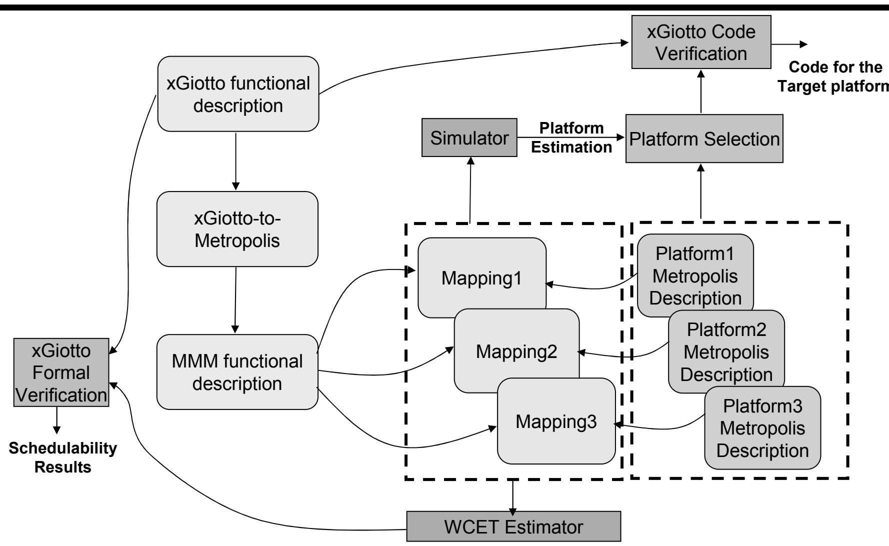


Metamodeling and Model Transformation Use Cases

3. Semantic Integration of Tools



xGiotto and Metropolis

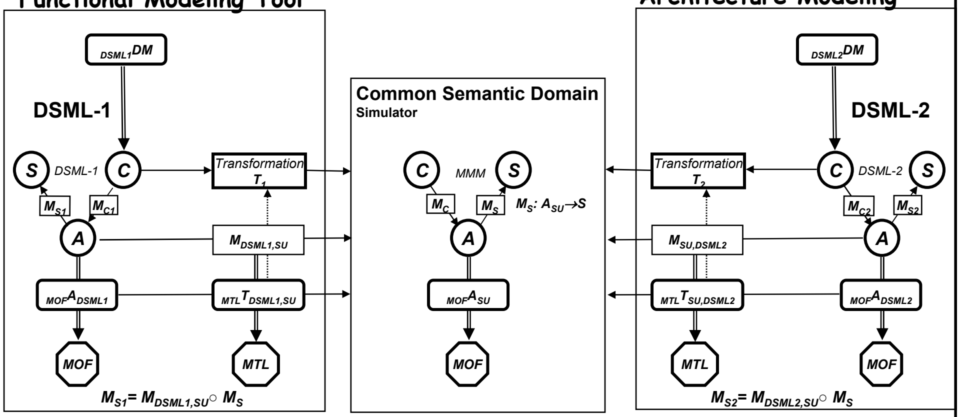


Metamodeling, Model Transformation and Analysis

4. xGiotto and Metropolis

xGiotto Functional Modeling Tool

Metropolis Platform Architecture Modeling



xGiotto Functional Description

Metropolis: Common Semantic Domain and Function-Architecture Mappings

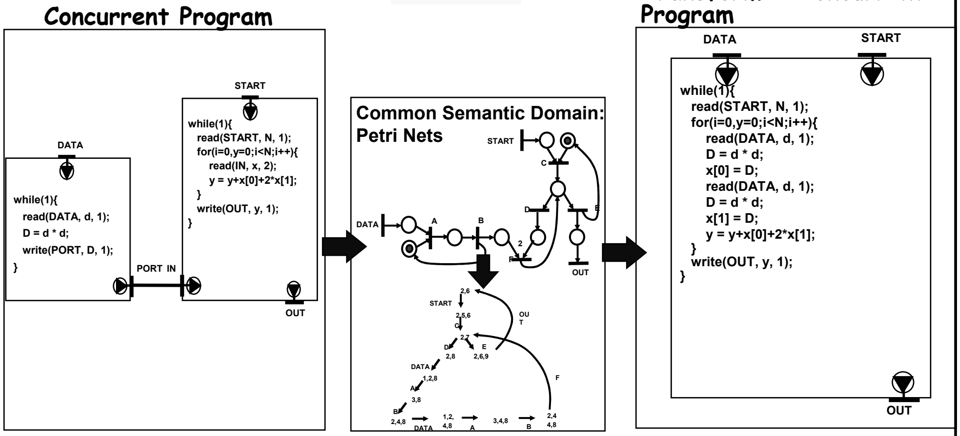
Platform Models

Metamodeling, Model Transformation and Analysis

5. QSS

Concurrent Program

Transformed Concurrent Program



Set of communicating processes

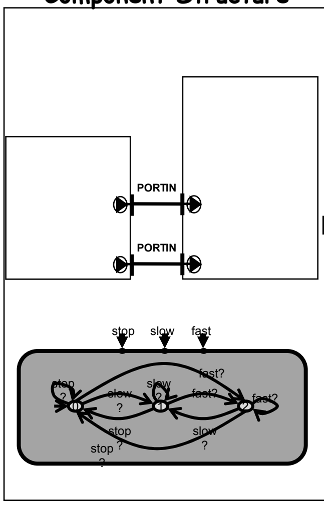
Find a single process that realizes a feasible execution of the original set under a bounded memory

Single process

New Semantic Domains: Resource Interfaces

Interface Theory (Henzinger et. al.)

Component Structure



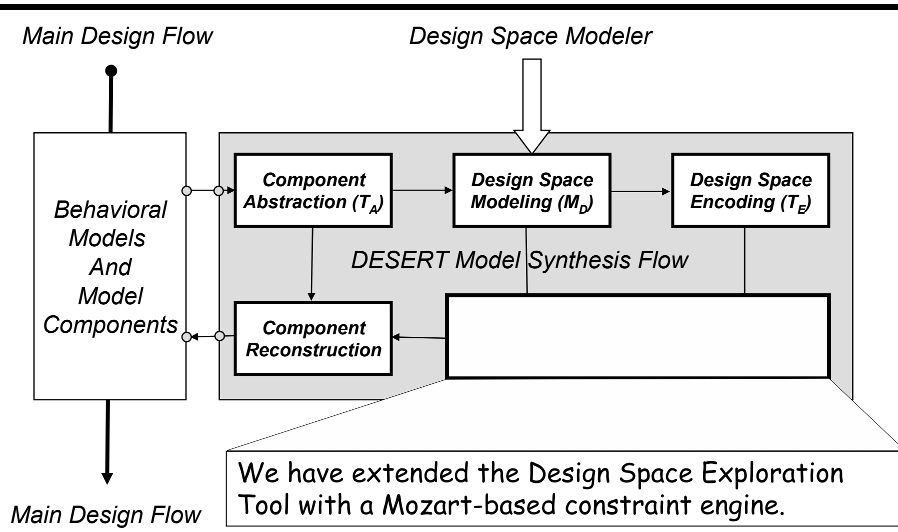
Common Semantic Domain

Resource Interfaces

Resource interfaces =
Methodology for compositional state-aware resource-usage analysis of open systems
+ Efficient algorithms for finding how a set of components can be made to work together using the least amount of a scarce resource
+ Algorithms implemented in the tool Chic (http://www.eecs.berkeley.edu/~tah/Chic/)

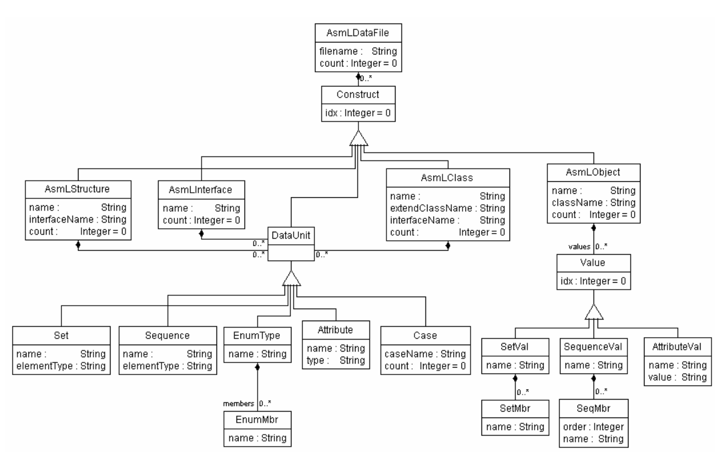
Two synthesis questions:
Strategy Synthesis. (e.g. resource scheduler, sensornet routing algorithm) Given a resource bound, how can player Environment achieve her objective ?
Resource Synthesis (e.g. necessary buffer size, battery capacity). What is the minimum resource requirement so that player Environment can achieve her objective ?
Game algorithms can be generalized to answer both.

Model Synthesis

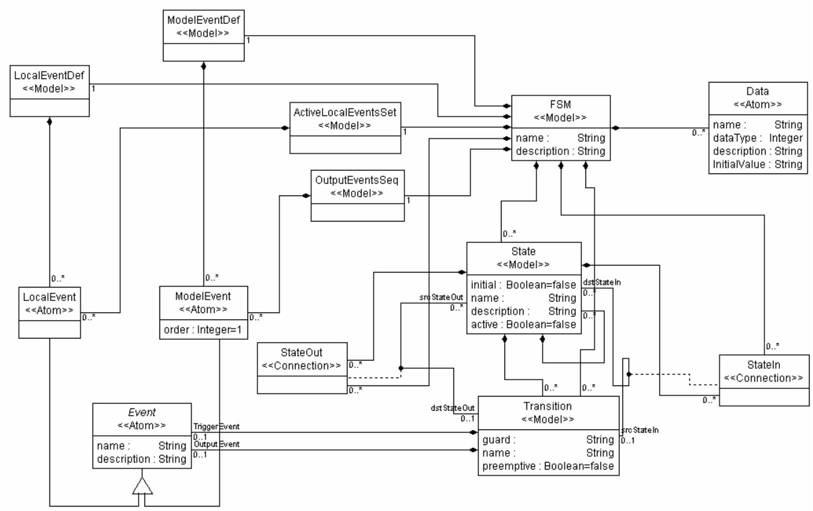


Backup Slides

Metamodel for a Subset of AsmL Data Structures



Metamodel for HSML



Ptolemy II Model

AsmL data Structure

- Event & FSM class

```
interface Event
structure ModelEvent implements Event
  case ModelEvent1
  case ModelEvent2
structure LocalEvent implements Event

class FSM
  var outputEvents as Seq of ModelEvent
  var localEvents as Set of LocalEvent
  var initialState as State
  var children as Set of State
```

AsmL data Structure (cont'd)

- State & Transition class

```
class State
  var active as Boolean = false
  var initial as Boolean
  var initialState as State?
  var parentState as State?
  var slaves as Set of State
  var outTransitions as Set of Transition

class Transition
  var guard as Boolean
  var preemptive as Boolean
  var triggerEvent as Event?
  var outputEvent as Event?
  var src as State
  var dst as State
```


Operational Semantics

- Top-level FSM model reaction

```
fsmReact (fsm as FSM, e as ModelEvent) =
  step
    let cs as State = getCurrentState (fsm, e)
  step
    let pt as Transition? = getPreemptiveTrasition (fsm, cs, e)
  step
    if pt <> null then
          doTransition (fsm, cs, pt)
    else
      step
        if isHierarchicalState (cs) then invocateSlaves (fsm, cs, e)
      step
        let npt as Transition? = getNonpreemptiveTranstion (fsm, cs, e)
      step
          if npt <> null then doTransition (fsm, cs, npt)
```

Operational Semantics (cont'd)

- Do transition

```
doTransition (fsm as FSM, s as State, t as Transition) =
  require s.active
  step exitState (s)
  step if t.outputEvent <> null then emitEvent (fsm, t.outputEvent)
  step activateState (fsm, t.dst)
```


Operational Semantics (cont'd)

- Activate state

```
activateState (fsm as FSM, s as State) =
  step s.active := true
  step
    if isAtomicState (s) then
      let t as Transition? = getInstantaneousTransition (s)
      step if t <> null then doTransition (fsm, s, t)
```

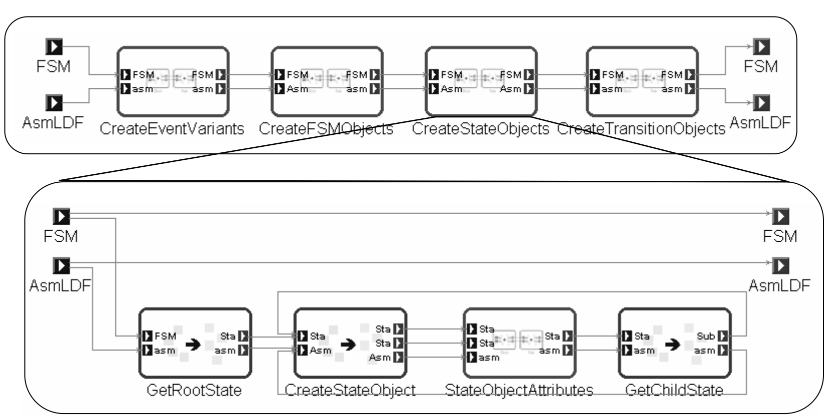
Operational Semantics (cont'd)

- Get instantaneous transition

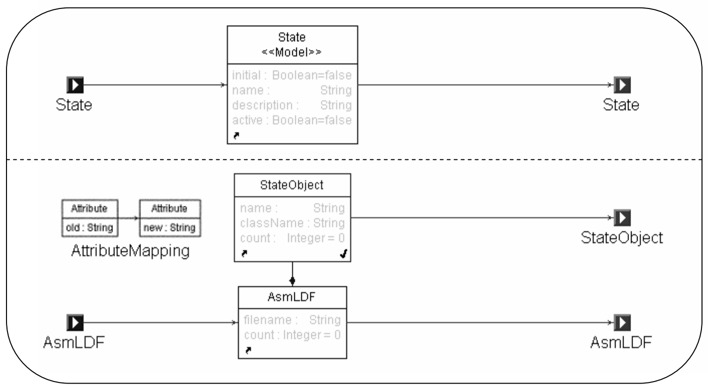
```
getInstantaneousTransition (s as State) as Transition? =
  require isAtomicState (s)
  step
    let ts = {t|t in s.outTransitions where t.triggerEvent = null
    and t.guard }
  step if Size (ts) > 1 then error "non-deterministic error"
  step
    choose t in ts
      return t
    ifnone
      return null
```


Semantic Mapping Specification

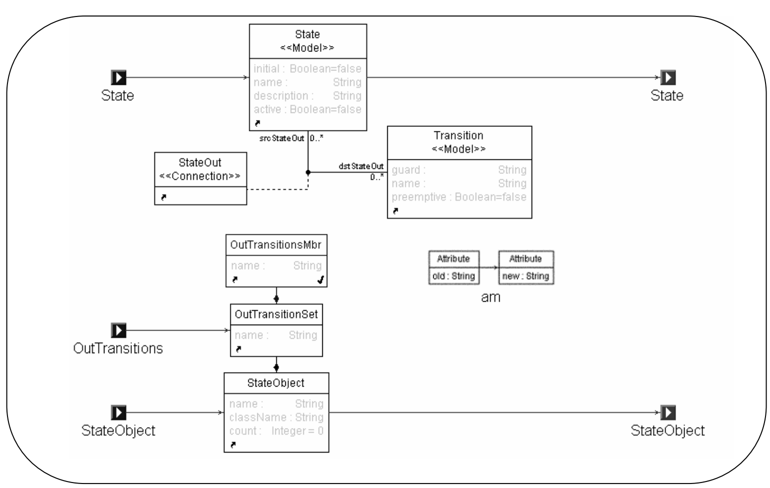
- The semantic mapping specification for FML consists of a sequence of mapping rules.



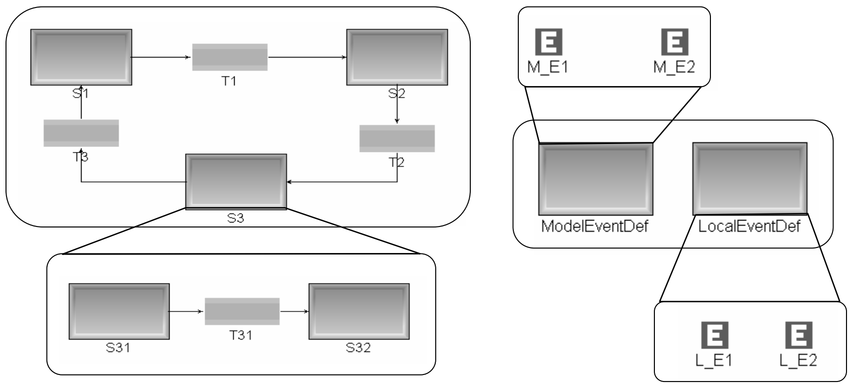
Semantic Mapping Specification (cont'd)



Semantic Mapping Specification (cont'd)



A Hierarchical FSM Model



Output XML file


```
- <AsmLDataFile _id="id671" count="19" filename="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="C:\research\semantics\new\UDM\ASMSfw.xsd">
  - <AsmLObject _id="id7ab" idx="6" name="FSM_Example" count="5" className="FSM">
      <AttributeVal _id="id7c0" idx="0" name="name" value="FSM_Example" />
      <AttributeVal _id="id7ed" idx="3" name="initialState" value="S1" />
      <SequenceVal _id="id7d4" idx="1" name="outputEvents" />
      <SetVal _id="id7df" idx="2" name="localEvents" />
    - <SetVal _id="id7fd" idx="4" name="children">
        <SetMbr _id="id808" name="S1" />
        <SetMbr _id="id80a" name="S2" />
        <SetMbr _id="id80b" name="S3" />
      </SetVal>
    </AsmLObject>
  + <AsmLObject _id="id82b" idx="7" name="S1" count="8" className="State">
  + <AsmLObject _id="id82c" idx="8" name="S2" count="8" className="State">
  - <AsmLObject _id="id82d" idx="9" name="S3" count="8" className="State">
      <AttributeVal _id="id845" idx="3" name="inital" value="false" />
      <AttributeVal _id="id846" idx="2" name="active" value="false" />
      <AttributeVal _id="id847" idx="0" name="name" value="S3" />
      <AttributeVal _id="id855" idx="4" name="initialState" value="S31" />
      <AttributeVal _id="id865" idx="5" name="master" value="null" />
    - <SetVal _id="id875" idx="6" name="slaves">
        <SetMbr _id="id880" name="S32" />
        <SetMbr _id="id881" name="S31" />
      </SetVal>
    - <SetVal _id="id892" idx="7" name="outTransitions">
        <SetMbr _id="id8a4" name="T3" />
      </SetVal>
    </AsmLObject>
```

Semantic Anchoring Architecture of DSML-s