Semantics of Metamodeling

Formal Semantics of Metamodeling Frameworks

Ethan Jackson

ISIS, Vanderbilt University

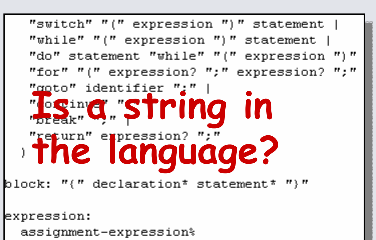
Semantic Anchoring Infrastructure

Kai Chen

ISIS, Vanderbilt University

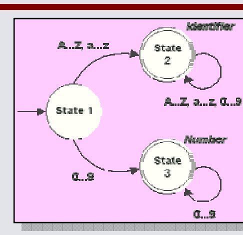
Chess Review
May 11, 2005
Berkeley, CA

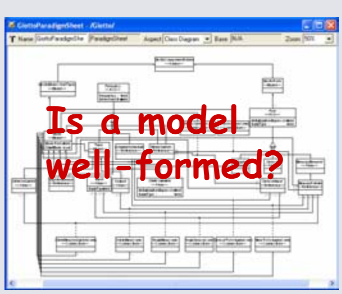
Metamodels As Formal Objects



Pumping Lemma

If it is accepted by the DFA





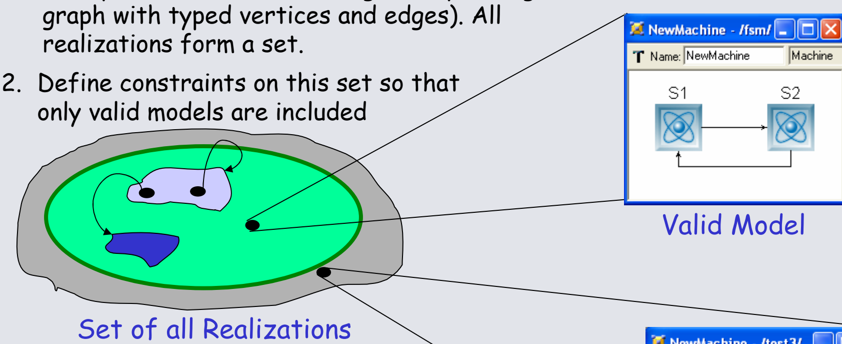
?

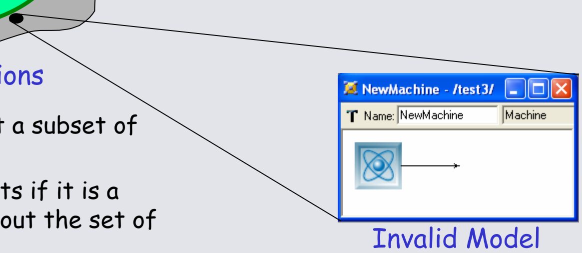
?

If Graph structure, type structure, containment, aspects, etc… are valid

Denotational Structural Semantics

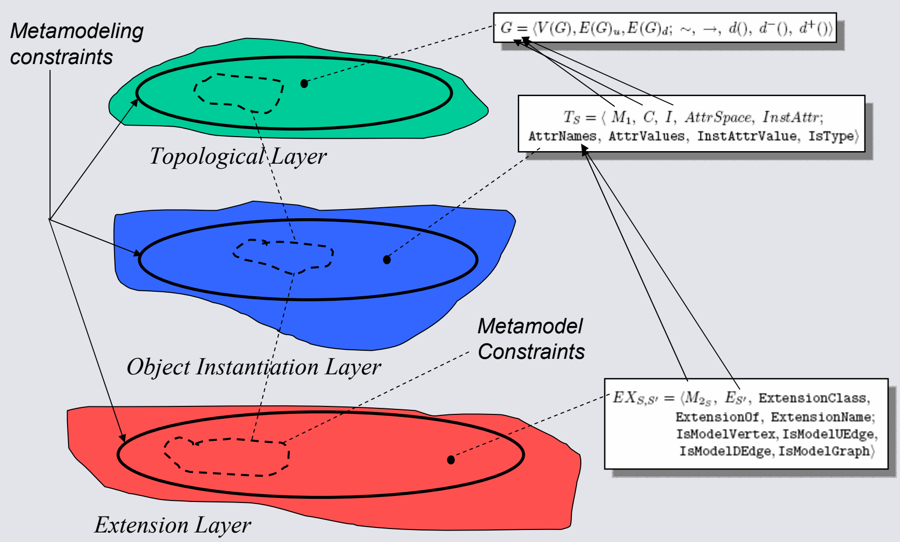
1. Define an algebraic structure that is rich enough to express all of the modeling concepts. (e.g. a graph with typed vertices and edges). All realizations form a set.
2. Define constraints on this set so that only valid models are included





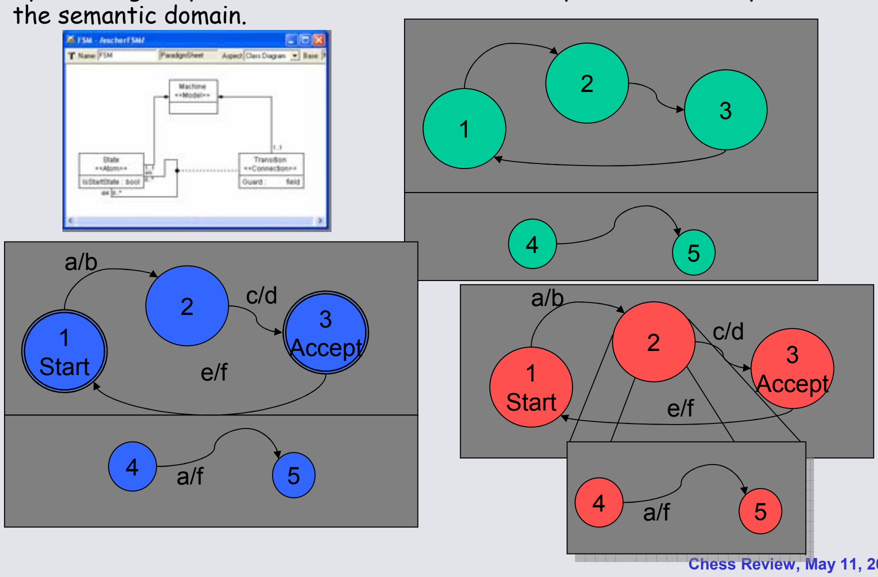
3. A metamodel carves out a subset of this valid space.
4. A meta-metamodel exists if it is a metamodel that carves out the set of all metamodels

Tailoring the Semantics



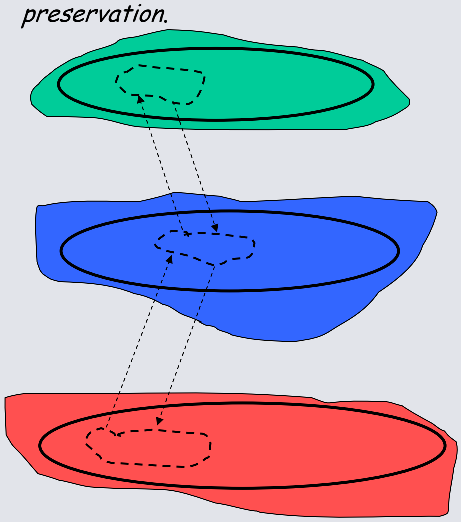
The Layered Approach

By choosing a layer, one chooses the set of concepts that naturally suite the semantic domain.



Layer Preservation

Layers progressively adds information. This is stated formally as *layer preservation*.



Downcasting

(2.94) $\texttt{DOWNCAST}_{j,k} : \mathbb{M}_{j_S} \mapsto \mathbb{M}_{k_S},\ j > k > 1$

(2.95) $\texttt{DOWNCAST}_{2,1}(m) = m.M_1$
$\texttt{DOWNCAST}_{3,2}(m) = m.M_{2_S}$

Upcasting

(2.96) $\texttt{UPCAST}_{k,j} : \mathbb{M}_{k_S} \mapsto \mathbb{M}_{j_S},\ j > k > 1$

(2.97) $(\texttt{UPCAST}_{1,2}(m) \in \mathbb{M}_{2_S}) \wedge (\texttt{UPCAST}_{1,2}(m).M_1 = m)$
$(\texttt{UPCAST}_{2,3}(m) \in \mathbb{M}_{3_{S,S'}}) \wedge (\texttt{UPCAST}_{2,3}(m).M_{2_S} = m)$

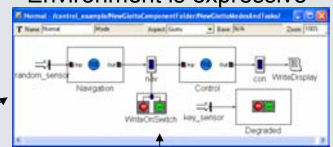
Layer Preserving

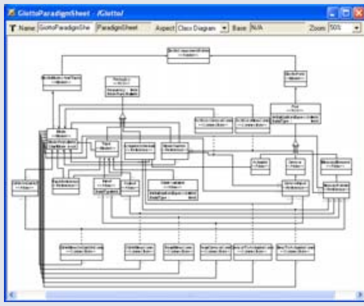
Definition 2.45. Let $k \geq 1$ and $M = \{M_k, M_{k+1}, \ldots, M_j\}$ be a set where the n^{th} element is a set M_n such that $M_n \subseteq \mathbb{M}_{n_S}$. The set M is *layer preserving* if:

(2.98) $\forall_{(k \leq i < j)}\ M_k = \texttt{DOWNCAST}_{k+1,k}(M_{k+1})$

Applications

Environment is expressive

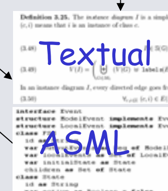




Automatically generated

Trivial bijective mapping

Automatically generated



Additional semantics that utilize the formal structures, **Proofs**

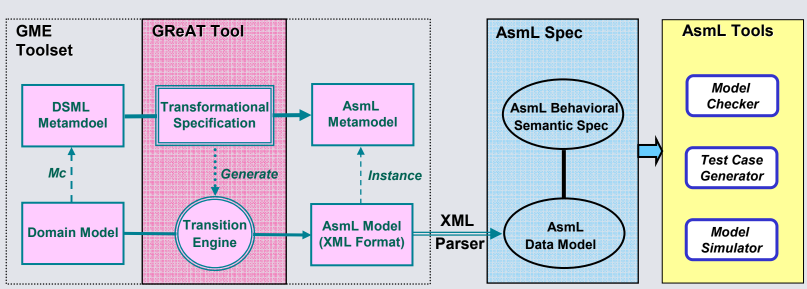
Giotto Metamodel is a ***formal*** object that simultaneously describes a modeling environment and a structural semantics.

The semantics of the metamodeling framework need not be reproduced, but just referenced.

MoC is descriptive, yet “baggage-free” because it builds off of the formal definition of the metamodel.

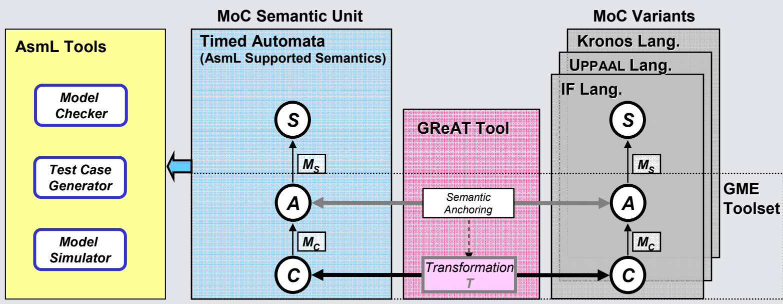
Semantic Anchoring Infrastructure

- Semantic Unit
  - A well-defined DSML that captures the semantics of a particular model of computation.
- Semantic Anchoring
  - Define the semantics of a DSML through the transformational specification to a semantic unit.

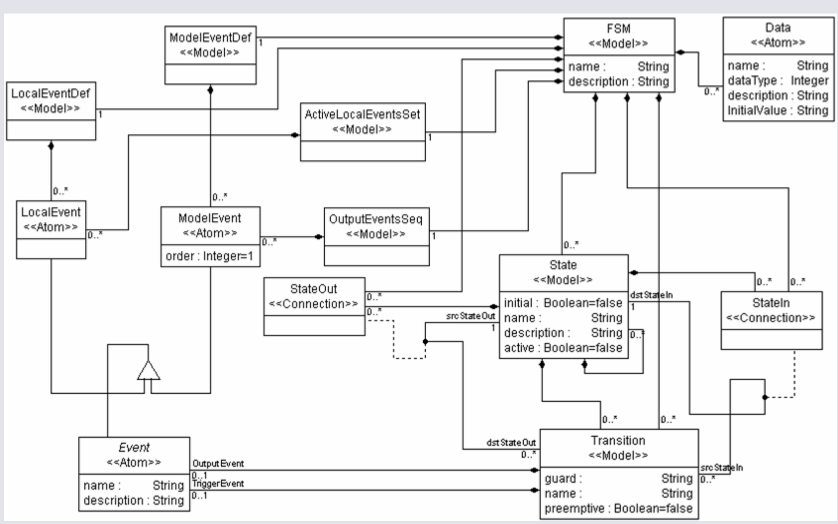


A Semantic Unit for Timed Automata Based Modeling Language

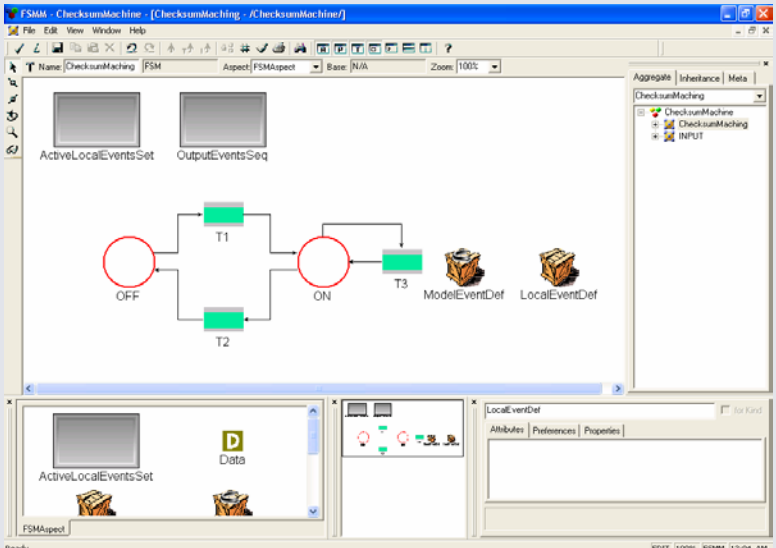
- Common semantic domain for varied timed automata based modeling languages.
  - Guard
  - Priority
  - Synchronization



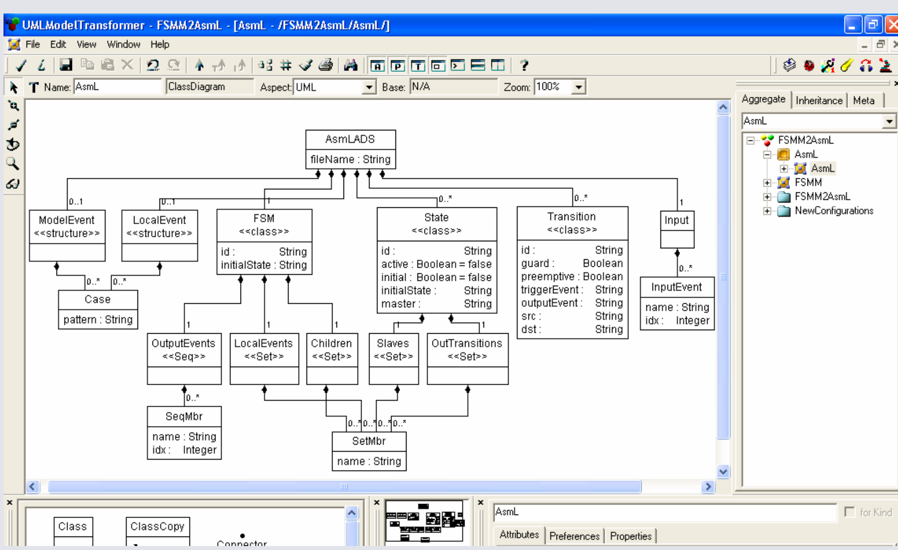
FSM Metamodel



FSM Model



Metamodel for AsmL Abstract Data Model



AsmL Abstract Data Model

FSM - Microsoft Visual C# .NET [design] - FSM.asml

```
<asml>
interface Event
structure ModelEvent implements Event
structure LocalEvent implements Event
class FSM
  id as String
  var outputEvents as Seq of ModelEvent
  var localEvents as Set of LocalEvent
  var initialState as State
  children as Set of State
class State
  id as String
  var active as Boolean = false
  var initial as Boolean
  var initialState as State?
  var master as State?
  var slaves as Set of State
  var outTransitions as Set of Transition
class Transition
  id as String
  var guard as Boolean
  var preemptive as Boolean
  var triggerEvent as Event?
  var outputEvent as Event?
  src as State
```

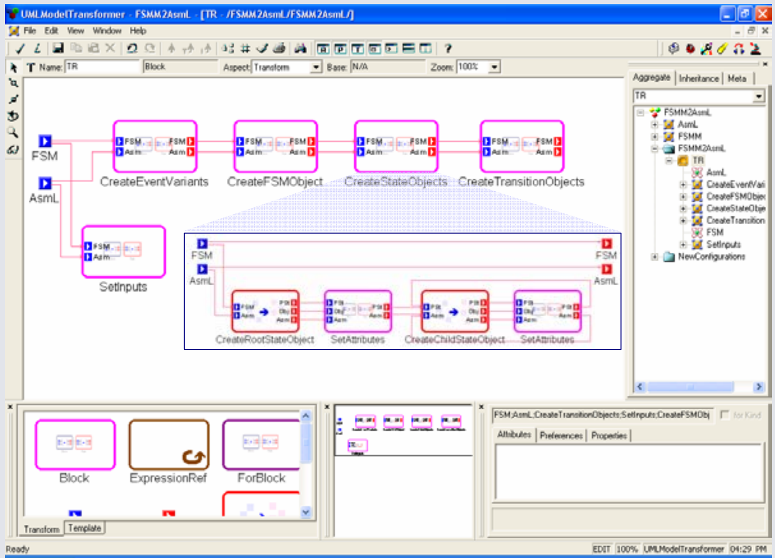

AsmL Behavioral Semantic Specifications

FSM - Microsoft Visual C# .NET [design] - FSM.asml

```
invocateSlaves (fsm as FSM, s as State, e as Event) =
  require isHierarchicalState (s) and s.initialState <> null
  step
    let ids as State = getActiveSlave (fsm, s, e)
  step
    let pt as Transition? = getPreemptiveTrasition (fsm, ids, e)
  step
    if pt <> null then
      doTransition (fsm, ids, pt)
    else
      step
        if isHierarchicalState (ids) then
          invocateSlaves (fsm, ids, e)
      step
        let npt as Transition? = getNonpreemptiveTranstion (fsm, ids, e)
      step
        if npt <> null then
          doTransition (fsm, ids, npt)

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)
```

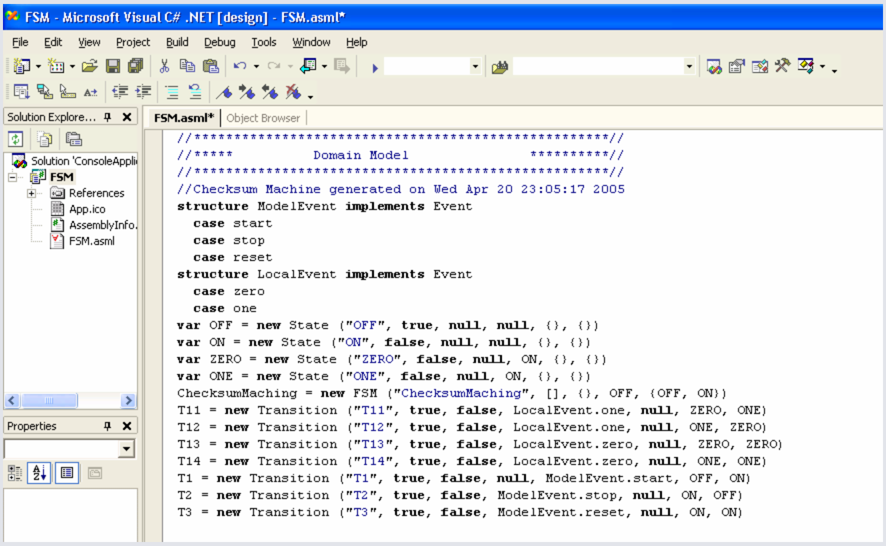
Transformational Specifications



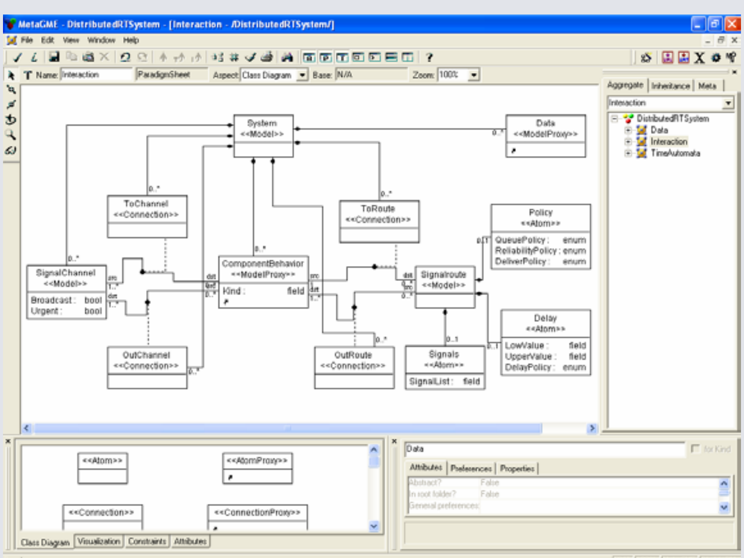
AsmL Data Model in XML Format

```
<AsmLADS _id="id988" fileName="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="UDM\AsmL.xsd">
- <FSM id="ChecksumMaching" _id="id9d5" initialState="OFF">
  + <Children _id="id9f4">
    <LocalEvents _id="id9e5" />
    <OutputEvents _id="id9e0" />
  </FSM>
+ <Input _id="id98b">
+ <LocalEvent _id="id9bf">
+ <ModelEvent _id="id9ad">
- <State id="OFF" _id="ida17" active="false" master="" initial="true" initialState="">
  + <OutTransitions _id="ida5b">
    <Slaves _id="ida3d" />
  </State>
+ <State id="ON" _id="ida18" active="false" master="" initial="false" initialState="ZERO">
+ <State id="ZERO" _id="ida74" active="false" master="ON" initial="false" initialState="">
+ <State id="ONE" _id="ida75" active="false" master="ON" initial="false" initialState="">
  <Transition id="T11" _id="idade" dst="ONE" src="ZERO" guard="true" preemptive="false" outputEvent=""
    triggerEvent="LocalEvent.one" />
  <Transition id="T12" _id="idadf" dst="ZERO" src="ONE" guard="true" preemptive="false" outputEvent=""
    triggerEvent="LocalEvent.one" />
  <Transition id="T13" _id="idae0" dst="ZERO" src="ZERO" guard="true" preemptive="false" outputEvent=""
    triggerEvent="LocalEvent.zero" />
  <Transition id="T14" _id="idae1" dst="ONE" src="ONE" guard="true" preemptive="false" outputEvent=""
    triggerEvent="LocalEvent.zero" />
  <Transition id="T1" _id="idb0e" dst="ON" src="OFF" guard="true" preemptive="false" outputEvent="ModelEvent.start"
    triggerEvent="" />
  <Transition id="T2" _id="idb0f" dst="OFF" src="ON" guard="true" preemptive="false" outputEvent="" triggerEvent="ModelEvent.stop" />
  <Transition id="T3" _id="idb10" dst="ON" src="ON" guard="true" preemptive="false" outputEvent=""
    triggerEvent="ModelEvent.reset" />
</AsmLADS>
```

AsmL Data Model

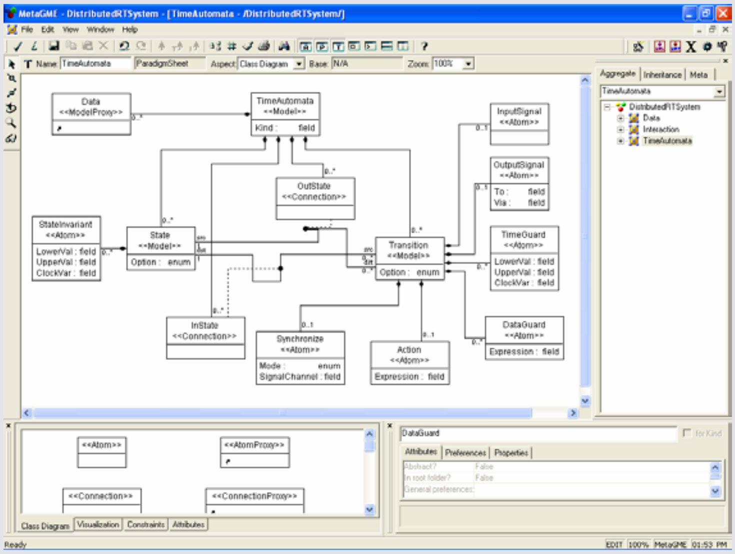


Distributed Real-time System (1)



- Abstract Syntax (1)
- Component Interactions

Distributed Real-time System (2)



- Abstract Syntax (2)
- Component Behaviors

Distributed Real-time System (3)

- AsmL Abstract Data Structure

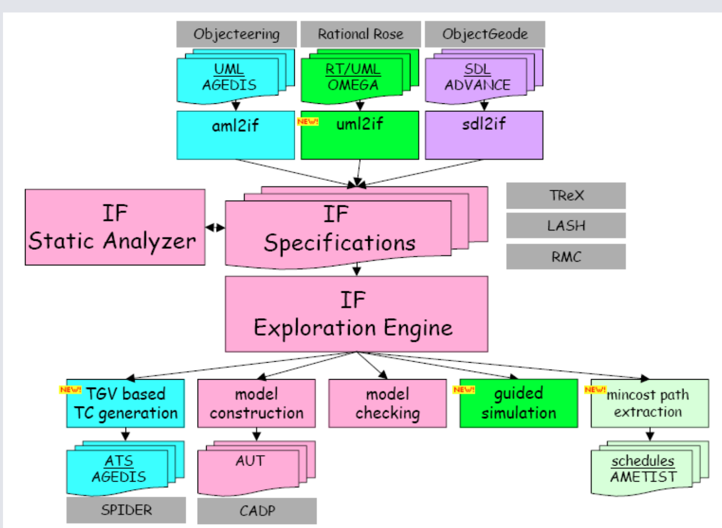
```
class State
  id     as String
  option as STATEOPTION
class Transition
  id     as String
  option as TRANSITIONOPTION
abstract class TimeAutomata
  var currentState                  as State? = null
  abstract property states          as Set of State
  abstract property transitions     as Set of Transition
  abstract property localClocks     as Set of Clock
  abstract property outTransitions as Map of <State, Set of Transition>
  abstract property srcState        as Map of <Transition, State>
  abstract property dstState        as Map of <Transition, State>
  abstract property syns     as Map of <Transition, (SignalChannel, SYNMODE)>
abstract class DRTSystem
  abstract property timeAutomatas  as Set of TimeAutomata
  abstract property signalChannels as Set of SignalChannel
  abstract property signalRouters  as Set of SignalRouter
  var activeAutomatas               as Set of TimeAutomata = {}
```

Distributed Real-time System (4)

- AsmL Behavior Semantics Specification

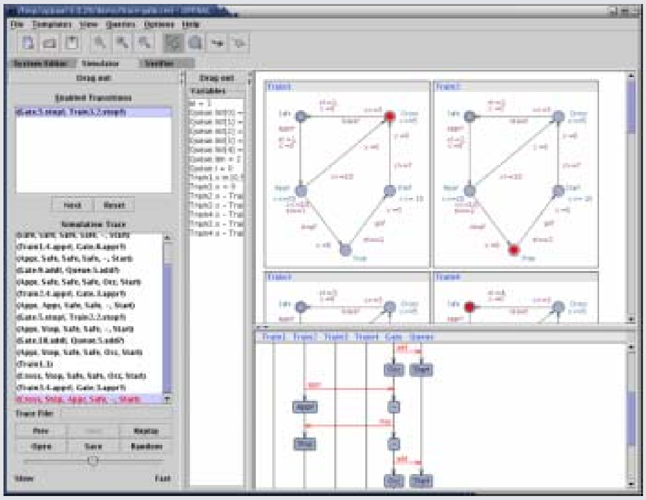
```
//In the current clock time, whether the time guard of an transition is true
  IsTimeGuardTrueNow (t as Transition) as Boolean
    require t in me.transitions
    step return TimeGuard (t)
//In the next clock time, whether the time guard of an transition is true
  IsTimeGuardTureNext (t as Transition) as Boolean
    require t in me.transitions
    step
      forall c in globalClocks
        c.Go ()
      forall c in me.localClocks
        c.Go ()
    step
      let next = IsTimeGuardTrueNow (t)
    step
      forall c in globalClocks
        c.Back ()
      forall c in localClocks
        c.Back ()
    step return next
```


IF



- Asynchronous Component Interaction
- Simulation
- Verification

UPPAAL



- Synchronous Component Interaction
- Simulation
- Verification

The End

Questions?