Model-based design focuses on the formal representation, composition, and manipulation of models during the design process.
Domain Specific Modeling Languages (DSML)

\[ L = \langle C, A, S, M_S, M_C \rangle \]

- **Abstract Syntax** \( A \)
- **Concrete Syntax** \( C \)
- **Parsing**
- **Semantic Domain** \( S \)
- **Mathematical abstraction for specifying the meaning of models**

**DSML-s are the foundations for model-based design**

Tool Chain Example

- **Common Semantic Domain: Hybrid Automata**
- **Abstract Syntax: Meta-Models**
- **Domain Models and Tool Interchange Formats**

**Vehicle Control Platform Tool Chain**

- **Simulink**
- **StateFlow**
- **ECSL-DP**
- **GME**
- **Observation**
- **DESERT**
- **PTOLEMY**

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 Metamodelling

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

Metamodelling Languages

Changing from GME/Meta 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.

META-MODEL

DOMAIN-MODEL

DSML: StateFlow

GME, UDM & GREAT
Completed tool suite, available through the ESCHER Repository

Chess Review, November 18, 2004 9
Chess Review, November 18, 2004 10
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. Translational Specification of Behavioral Semantics

\[
\text{MDSML}_i, \text{SU} \rightarrow \text{MDSML}_i, \text{SU}
\]

- 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

Metamodeling and Model Transformations Use Cases

2. Semantic Anchoring of DSML-s

\[
\text{MDSML}_i \rightarrow \text{MDSML}_i, \text{SU}
\]

Semantic "Units"
More On Semantic Anchoring

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

Example: HSML (Ptolemy II) → FSM

![Diagram showing semantic anchoring from HSML to FSM]

- MOF
- Set-Valued SD
- FSM MetaModel
- FSM Model
- FSM Data Model
- FSM Interpreter
- Syntax conversion
- Expressed in
- Instance of
- Set-Valued
- Abstract State Machine SD

Chess Review, November 18, 2004  16
3. Semantic Integration of Tools

Semantic Unit (Common Semantic Domain)

Modeling Tool

DSML-1

Semantic Unit

Analysis Tool

DSML-2

Transformation

T = T₁ ∘ T₂

Metamodeling and Model Transformation Use Cases

xGiotto and Metropolis

xGiotto functional description

MDSML₁, SU

Mapping 1

Platform2

Metropolis Description

Mapping 2

Platform Selection

Mapping 3

Platform3

Metropolis Description

Simulator

Platform Estimation

xGiotto Code Verification

Code for the Target platform

WCET Estimator

Schedulability Results
Metamodelling, Model Transformation and Analysis

4. xGiotto and Metropolis

xGiotto Functional Modeling Tool

Metropolis Platform Architecture Modeling

Common Semantic Domain Simulator

Common Semantic Domain: Petri Nets

QSS

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

Component Structure

Interface Theory
(Henzinger et. al.)

Common Semantic Domain

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

Main Design Flow
Design Space Modeler

Behavioral Models And Model Components

Component Abstraction (T_A)
Design Space Modeling (M_J)
Design Space Encoding (T_E)
Component Reconstruction

DESERT Model Synthesis Flow

We have extended the Design Space Exploration Tool with a Mozart-based constraint engine.
Metamodel for a Subset of AsmL

Data Structures
AsmL data Structure

- Event & FSM class

```plaintext
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

```plaintext
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

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

Chess Review, November 18, 2004
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)
• Get instantaneous transition

```plaintext
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
```

• The semantic mapping specification for FML consists of a sequence of mapping rules.
A Hierarchical FSM Model

Output XML file