Model-Based Design

Edited by Janos Sztipanovits,
Presented by Gabor Karsai
ISIS, Vanderbilt University

Objectives

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

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

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

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

Research Agenda

Baseline:
- Modeling of Abstract Syntax using UML class diagrams and OCL as meta language;
- Meta-programmable Graphical Model Editor (GME)

Research Agenda:
1. Using the Meta-Object Facility (OMG) as the meta-metamodel
2. Specification of DSML semantics via meta-modelling
3. Foundation for composing DSML-s
4. New Semantic Domains
5. Extension of meta-programmable tools
Semantics via Meta-Modeling: Baseline

Meta-modeling language with well-defined semantics

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

New Direction: Precise Meta-Modeling

Abstract syntax modeling with MOF

Concrete syntax modeling

Ongoing Work: (Poster: Compositional Metamodelling for DSMLs)

- Changing from UML-CD/OCL to MOF in abstract syntax modeling
- Developing set-valued semantics for MOF using TLA+ as meta-language
- Developing \( \text{MOF}^\text{DSML} \rightarrow \text{TLA}^\text{DSML} \) translator
Semantics via Translation: Baseline

Modeling language with well-defined semantics

Semantics via Translation: Baseline

Synchronous Dataflow (SDF)

Hierarchical Signal Flow (HSF)

New Direction: Translational Specification of DSML Semantics

Ongoing Work: (Poster: Compositional Metamodelling for DSMLs)

• TLA+ specification of the semantics of selected models of computations (reference models)
• Specifying DSML semantics via specifying mapping between the abstract syntax of DSML and the reference models
• Developing modeling and translation tools
New Semantic Domains: HyVisual – Hybrid System Modeling Tool Based on Ptolemy II

HyVisual is a DSML designed for hybrid system modeling, specialized from Ptolemy II by combining its CT and FSM domains.

Major Applications of Meta Modeling: Metropolis Meta Model (Formal Model)

- How are designs represented in Metropolis?

  Metropolis “meta-model”: a language + modeling mechanisms
  - represents all key ingredients: function, architecture, refinement, platforms
  - parser and API to browse designs, interact with tools
Metropolis Meta-Model

- Must describe objects at different levels of abstraction
  - Do not commit to the semantics of any particular model of computation
- Define a set of “building blocks”
  - specifications with many useful Models of Computations can be described using the building blocks
  - Processes, communication media and schedulers separate computation, communication and coordination

Core Metropolis “Meta Model”

- Processes
  - Concurrent Computation, sequential execution within each process
  - Execution within a process can be suspended with the AWAIT statement
- Interfaces
  - The types of Ports, bundle function declarations
- Media
  - Model Communication by providing definitions of Interface Functions
  - Passive, cannot run in their own threads
- Netlists
  - STRUCTURE and CONCURRENCY
- Quantity Managers
  - Performance modeling
- Constraints
  - Specification & Synchronization
- Formal Semantics
  - Operational Semantics based on Action Automata
2. Model Transformations

Research Agenda

Baseline:
- Meta-models of the Abstract Syntax of the source and target DSML-s;
- Tools helping the procedural implementation of translators

Research Agenda:
1. Meta Generators: Specification and generation of model transformations using graph rewriting technology
2. Embeddable generators
3. Generative extensions to modeling languages
Model Transformation Using Meta-Models and Graph Rewriting

Karsai/Agrawal, ISIS

Component Specialization

Model of Computation semantics defines communication, flow of control

Schedule:
- fire Gaussian0
- fire Ramp1
- fire Sine2
- fire AddSubtract5
- fire SequenceScope10

Java actor definitions are parsed and then specialized for their context.

abstract syntax tree

specializer

target code

specialize for
- data types
- parameter values
- scheduling
- by
- token unboxing
- inlining
- partial evaluation
- dead code elimination

cheese code
3. Model Synthesis

Example design flow:

Challenge:
Compose plant and controller models using library components with alternatives so as to satisfy design constraints

Research Agenda

Baseline:
- Constraint-based design space exploration coupled to a dataflow language

Research Agenda:
1. Pattern-based model synthesis
2. Models of Computations as design patterns
3. Design constraints and design patterns for multiple Models of Computations
Pattern-based Model Synthesis

Main Design Flow

Simulink Behavioral Models And Model Components

DESERT Model Synthesis Flow

Component Abstraction \( (T_d) \) → Design Space Modeling \( (M_d) \) → Design Space Encoding \( (T_e) \)

Component Reconstruction → Design Decoding → Design Space Pruning

Using meta-modeling and model transformations we made the constraint based design space exploration tool suite composable with different design flows.

Example: Use of Model Translation in the DESERT Tool Chain

Simulink® Metamodel

Component Abstraction \( (T_d) \)

Abstracted Simulink Model Components