Model Integrated Design of Embedded MPSoC

July 6, 2004
MPSOC’04

Dr. Janos Sztipanovits

Institute for Software Integrated Systems (ISIS)
Vanderbilt University
Overview

◆ MDA
  - An emerging paradigm for software development

◆ Domain-Specific MDA: MIC
  - From DSL-s to DSML-s
  - The role of transformations

◆ Example

◆ Challenges and opportunities
  - Tool Integration
  - Semantic Foundation

◆ Conclusions
Model-Driven Architecture
A Paradigm for Software Development

History:
- OMA (CORBA) → UML → MDA:
  - Object composition
  - Models for software
  - Models in development

Key points:
1. Models are not accidental but essential to system development
2. Models are expressed in modeling languages
3. Models are built, analyzed and transformed during development that lead to executables

Source: Jean Bézivin: "From Object Composition to Model Transformation with the MDA" TOOLS USA, August 2001, Santa Barbara
Model-Driven Architecture
The Role of Modeling Languages

Development process:
• Platform Independent Model:
  – UML
  – extension with UML profiles or metamodeling
• Platform Specific Model:
  – UML
  – extension with UML profiles or metamodeling

Key points:
1. 4-layer metamodeling architecture
2. MOF is the metalanguage
3. Not specific to CORBA

4-Layer Meta-Model Language Architecture:

- Meta-Modeling Language:
  - MOF
- Meta-Models
- Domain-Specific Modeling Language (DSML)
  - UML, ESML...
- Domain Models
- Application System
**Model-Driven Architecture**

**The Role of Transformations**

**Development process:**
- **Platform Independent Model**: a view of the system from a platform independent viewpoint
- **Platform Specific Model**: a view of the system from a platform-specific viewpoint

**Key points:**
1. Most relevant issue: platform-independence (CORBA/EJB/.NET)
2. Additional (P/S) information is used to map PIM into a PSM
3. Transformations are *models*

Source: MDA Guide V 1.0.1 (www.omg.org)
Model-Driven Architecture
Model Transformation Variants

Marking:

Metamodel-based:

Type-based:

Pattern-based:

Model merge:

Source: MDA Guide V 1.0.1 (www.omg.org)
**Development starts with domain engineering**
Understand and capture domain concepts and invariants
Develop Domain-Specific Modeling Language(s) (DSML) that capture
- Domain concepts and relationships
- Domain invariants as well formedness rules

**Systems are constructed from domain-specific models**
Analysis and generation are core activities

**Domain-Specific Tool Suites play a key role in the development process**
- Modeling environment(s) for DSML-s
- Model transformation tools
- Model analysis tools
- Model-based generators
MIC Components

Metamodelling

Metamodels

Domain-specific modeling

Model-Model Transformations

Tool Integration

Domain models

Translation

Synthesis & Generation

Analysis

Execution
MIC Approach to Metamodelling

**DSML for Signal Processing: SF**

Describes the structure of a design

**Metamodel for SF:**

Specifies the invariants in all designs (Domain architecture)
DSML-s are **affordable** only if tools are still reusable.

“Meta-programmable” tools:
1. Model repositories
2. (Visual) modeling environments
3. Model transformation tools

**Common underlying theme:**

*Tools are configured through metamodels*

<table>
<thead>
<tr>
<th>Metaprogrammable Tool</th>
<th>Metamodel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model repository</td>
<td>Schema, consistency/integrity rules</td>
</tr>
<tr>
<td>Modeling editor</td>
<td>Abstract and concrete syntax of DSML</td>
</tr>
<tr>
<td></td>
<td>Static semantics</td>
</tr>
<tr>
<td>Model transformation</td>
<td>Abstract syntax of source and target</td>
</tr>
<tr>
<td></td>
<td>Model transformation process</td>
</tr>
</tbody>
</table>
MIC Approach to Model Transformations

**Meta-level:** Translator design

**Implementation:** Execution

- Input Models
- Target Models
- Input Interface
- Output Interface
- UDM
- GRE

**Physical interface**

**Input abstract syntax**

**Output abstract syntax**

**Formal, explicit, and precise model of the transformations**
Model Transformation Tool Chain

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

See Karsai
http://www.isis.vanderbilt.edu

GME

Meta-Programmable Modeling Tool

Meta-Programmable Transformation Tool

Code Generator

C/G

GRE

DEBUG

Source Models

Target/Executable Models

Target Platform

MetaModel of Source

MetaModel of Domain-to-Target Mapping

MetaModel of Target

Meta-models

Meta-programmable tools

Generated tool

Models and applications

Janos Sztipanovits
Example: Simplified Automotive Design Flow
**Opportunities: Composable Tool Chains**

**Automotive Experiment**
- **Modeling**
  - Simulink/Stateflow
  - Matlab/Simulink
  - R-T Workshop
- **Controller Synthesis**
  - Checkmate
  - Ptolemy
- **System Analysis**
  - WindView
  - Checkmate
  - ARIES
- **Code Synthesis**
  - MPC555/OSEK
  - PENTIUM
- **Validation**
  - Design Feedback
  - Valid Code/Model
- **Target Analysis**
  - Valid Model
  - Design Feedback
- **Platform**
  - Integrated Model
  - Valid Model
  - Test Vectors

**Avionics Experiment**
- **Comp/Platf Modeling**
  - UML/Rose
  - Manual
  - ESML/GME
- **Component Implement.**
  - Honeywell
  - CMU
  - TimeWeaver
- **System Modeling**
  - ARIES
  - SWRI/ASC
  - ESML/GME
- **Component Integration**
  - AIRES
  - SWRI/ASC
  - ESML/GME
- **Validation**
  - ARIES
  - SWRI/ASC
  - ESML/GME
- **Target Analysis**
  - PENTIUM/TAO/BOLD-STROKE

**BACKPLANE**
- **Open Tool Integration Framework**
  - Metagenerators
  - Metamodel Composition & Validation
  - Metamodelling
  - GrEAT
  - GME/Meta
  - UML-CD/OCL

- Janos Sztopanovits

- **Integrated Physical/Computational Modeling and Analysis**
- **Model-based Generators**
- **Hybrid System Analysis**
- **Customizable (metaprogrammable) modeling tools and generators**
- **Open tool integration framework; configurable design flow and composable design tool chains**
Solution: Open Tool Integration Framework (OTIF)

RFP is Discussed at 
MIC PSIG
OMG

Share models using Publish/Subscribe Metaphor

Status:
- Completed, tested in several tool chains
- Protocols in OMG/CORBA
- CORBA as a transport layer
- Integration with ECLIPSE

http://www.isis.vanderbilt.edu/Projects/WOTIF/default.html
Challenges: Semantic Anchoring of DSML-s
Conclusion

- The hard problem of building complex MPSoC embedded systems is the integrated design of physical and computational components.
- Domain-specific modeling languages and model transformations are key technologies for future progress.
- Model-Integrated Computing evolves to be a mature technology for the development of complex applications.