The Impreciseness of UML and Implications for ModelicaML

EOOLT, Paphos
2008-07-08

Jörn Guy Süß, Peter Fritzson, Adrian Pop

Presentation by Peter Fritzson
Outline

• Whole Product Modelling (Software/Hardware) implies Tool Integration
• Why UML is Bad (for Integration)
• Meta-Models and Frameworks
• An Example of Whole Product Modelling
Notations for Whole-Product Modeling

- UML is widely adopted as a software modeling notation
- Modelica is very useful for hardware modeling

- Problem: Size and imprecise semantics of UML
- Possibility: Eclipse offers a minimal and well defined UML-like base platform: EMF (Eclipse Modeling Framework)
UML Suitability Problems

- Model Exchangeability
- Specification Size
- Semantics
- Sub-Languages (OCL, Action Language)
- Incompatible Children (SysML, xtUML)
UML is not Exchangeable
UML Specification Growth (Pages)

- V0.8: 136 pages
- V0.91: 171 pages
- V1.1: 413 pages
- V1.3: 492 pages
- V1.4: 566 pages
- V1.5: 736 pages
- V2.0: 957 pages

Graph showing the growth of UML specification pages across different versions.
What is Metamodelling?

A Software Engineering approach to cheaply build interactive editors for engineering languages that are described in the form of class diagrams using diagrams to specify/code compilers.
UML (Meta-Object Facility) MOF Pyramid

MOF Model

M3 layer
meta-metamodel

M2 layer
metamodels

... models

M1 layer

M0 layer

UML Metamodel

IDL Metamodel

UML Models

IDL Interfaces

[UML1.4.2]
Structure and Consistency Based on MOF

- Classes
- Attributes
- References
- Multiplicities
- Well-formedness Rules

- Eclipse Ecore
  MOF Model Instance
MOF Advantages

- High-level Semantics (Boxes and Lines)
  - Less (visible) technology
  - Easier to understand for client

- Generic Builder Tools (Graphical Editors)

- Large Provided Infrastructure / Framework
  - Reduced cost to build solution
  - Simplified integration between solutions

- Report Generators
  - Produce Work Items
  - Produce Documentation
Defining the MOF Language in MOF
MOF Translation Process

- MOF Graphical Editor
- MOF Textual Editor
- MOF Model
- Manipulation API (Java, C..)
- Model
- Extend to Validate
- Test
- Edit
- Plugin
- Save and Load
- Undo / Redo
- User Interface
MOF Framework Offerings

Eclipse Modeling Framework (EMF): Ecore

Netbeans Metadata Repository (MDR): MOF 1.4

OMG MOF Standard

Manipulation API (Java, C..)

Model

Save and Load

Extend to Validate

Test

Edit

Undo / Redo

Plugin

User Interface
Ongoing Work: ModelicaML – UML Profile for Modelica

• Extension of SysML subset
• Features:
  • Supports Modelica constructs
  • Modelica generic class modeling
  • Modelica syntax in definitions
  • Equation-based modeling
  • Simulation modeling
ModelicaML Diagrams – Overview

ModelicaML Diagram

- **Behavior diagram**
- **Requirement diagram**
- **Structure Diagram**
  - Class diagram
  - Class Internal diagram
  - Package diagram
  - Parametric diagram

- **Activity diagram**
- **Sequence diagram**
- **Equation diagram**
- **State Machine diagram**
- **Use Case diagram**

- New diagram type
- Modified from SysML
- Same as SysML
ModelicaML Class Internal Diagram

- Modelica Connection diagram
  - Better visual comprehension
  - Predefined connector locations

versus

- Class Internal diagram
  - Nested models
  - Top-model parameters and variables
  - Flow direction
  - Other ModelicaML elements
Simulation Diagram
Introduced by ModelicaML

```
<<Parameter>>
force1 = 37000

<<Parameter>>
thrustEndTime = 200

<<Parameter>>
moon.radius = 1.738e6

<<ModelicaModel>>
MoonLanding

Parameters
- Real force1 = 36350
- Real force2 = 1308
- Real thrustEndTime = 210
- Real thrustDecreaseTime = 43.2

Parts
- Rocket apollo(name = "Apollo13", mass(start=1038.358) )
- CelestialBody moon(name = "moon", mass = 7.382e22)

picture

<<requirement>>
Gravity should be gradually increased when the rocket approaches the lunar surface

<<simParameter>>
apollo.altitude[t], [t, 0, 208]

<<simResult>>
apollo.gravity[t], [t, 0, 208]

<<satisfy>>
```

© Peter Fritzson
Saab Bofors Example Application

Executable & Translatable Specifications (Models)

Model Compiler

High-Level Language (Ada95, C, ...)

Compiler

Assembly Code

Assembler

System

Object Code

Processor

Programmable Logic Device

Netlist File

Synthesizer

Placer & Router

Programming File

[SAAB BOFORS]
Future Tool Integration Example
(Saab-Bofors application)
Outlook for ModelicaML

• Continue using Eclipse Modeling Framework as a basis
  • Based on Ecore meta-meta-model

• Make ModelicaML smaller with more well defined semantics
  • Only include semantically well-defined diagrams
  • Remove some UML/SysML constructs with fuzzy semantics
  • Full compilation to Modelica

• Use algorithmic Modelica as (UML) action language?