Domain-Specific Modeling Languages for Embedded System Development

MeCoEs, Tampere, Oct 7 2012

Juha-Pekka Tolvanen, Steven Kelly
MetaCase
Contents

- Background to metamodeling
- Industry examples and experiences
- Generating code and beyond
- Summary
Metamodel and model: instance of
1. Enter concepts & their properties
2. Pick rules from templates
3. Draw or import the symbols
4. Specify generators

Language = concepts + rules + notation
Example: Breakfast - a modeling tool for SystemC

- Easy to create virtual platform model
  - Using QEMU for CPU models
  - Easy to connect QEMU to SystemC models
    - Easy to connect interrupt signals to CPU
  - No need of deep knowledge about SystemC TLM2.0
  - Abstraction is LT (Loosely Timed)

- Easy to create SystemC code for high-level synthesis
  - Pin level interface between modules is generated automatically
  - SystemC code for line buffer for image processing is generated
  - Designers can concentrate coding for algorithm

- SystemC model for high-level synthesis can be used on a virtual platform
  - No clock on a virtual platform
  - Virtual platform uses the same SystemC code for a register I/F (address decoder) as high-level synthesis
Breakfast

Virtual Platform

Code generator

SystemC TLM2.0 code including SystemC wrapper for QEMU

TLM (Loosely timed)

Adding code by hand

SystemC

High-level synthesis

RTL

Register map

Contact: Profound
Yasutaka Tsukamoto
tsukamoto@profound-dt.co.jp
Snapshot for an image processing hardware
Sports computer applications
Railway track control system
RTOS configuration
PLC heating application
Automotive infotainment system
Multi-view mobile app development
Smartphone applications
Mobile point-of-sale system
Document engineering
Call processing
Service creation environment
Test vector generation & verification
Printer HW&SW co-development
Language(s) + Code Generator
Experiences from practice

"5-fold productivity increase when compared to standard development methods"

"The quality of the generated code is clearly better, simply because the modeling language rules out errors"

"An increase of at least 750% in productivity, and greatly improved quality in the code and development process"
Productivity increase measured

Comparing to earlier practice (typically compared to coding)
Generation of non-software artifacts

- Single source, multiple targets:
  - Automated build → automating compile and execution
  - Checking completeness and uniformity
  - Simulation data
  - User guides, installation guidelines
  - Configuration
  - Metrics
  - Testing and analysis
  - Material usage
  - Documentation
  - Review reports

Case of Fishing Farm automation system, Preschern, C., et al 2012
Modeling both SW and system

- Combining two or more languages via metamodeling
- Metamodel covers integration rules for
  - Correctness
  - Consistency
  - Completeness
Hardware and software co-design

- Integrating various views with a common metamodel
  - e.g. Y-model
- Main challenge is organizational
  - Changes status quo
Design space exploration

- Performance analysis, error analysis, fault tolerance etc.
  - Integrate with tools like ABSOLUT, SPIN, UPPAAL, etc.
  - Annotate results back to models
Automating testing

- Modeling test cases/test logic and generating test data
Time to develop languages & generators

- **Call processing**: 63 language concepts (XML generator)
- **Touch screen UI applications**: 60 language concepts (C, HTML, build script generators)
- **Voice control application for microcontroller**: 36 language concepts (Assembler generator)
- **Mobile phone applications**: 77 language concepts (Python generator)
- **Automotive infotainment systems**: Java generator for simulation
- **Insurance product specification and management**: 143 language concepts (J2EE generator)
Thank You!

Questions?

USA:
MetaCase
5605 North MacArthur Blvd.
11th Floor, Irving, Texas 75038
Phone (972) 819-2039
Fax (480) 247-5501

International:
MetaCase
Ylistönmäentie 31
FI-40500 Jyväskylä, Finland
Phone +358 14 641 000
Fax +358 420 648 606
For details