Using Domain-Specific Modeling for Design and Verification of Cyber Physical Systems

17 March, 2016
Juha-Pekka Tolvanen
jpt@metacase.com
Motivation

- Domain expert vs. verification expert
  - Domain experts are not necessarily familiar with development and testing tools
  - Developers and verification experts don’t always master the domain knowledge

- Domain-specific languages enable to use directly the domain concepts for both development and V&V
  - Common vocabulary to enable feedback and communication
  - Higher level languages improve productivity
  - Automated transformations improve quality
Separate vs. common language

Domain knowledge (subject matter expert)

Technical knowledge (developers)

Development

Requirements

Features

Functions

Implementation

Testing Verification Validation
Domain-Specific Modeling (DSM)

- Use of concepts from the problem domain
  - Already familiar => no need to learn new
  - Have known semantics
- Having a special focus
  - Use concepts that are relevant for the task: development, testing, verification, validation
- Use concrete syntax that enables communication and collaboration close to domain’s natural representation
  - Not a cryptic programming/scripting language
- DSM is applied in particular for automating repetitive development efforts*, but less in testing and V&V

* See references on EADS, NSN, Nokia, Panasonic, Polar, Elektrobit, USAF
Example: Industrial Process Plant
Example Specification

Closed loop, Heat transfer, Liquid circulating (CHL)

May include:
- System Requirements Tree
- System Requirements
- Component Requirements
- Interface Requirements
How to define a language for a given domain: steps

1. Identify abstractions
   - Concepts and how they work together
2. Specify the metamodel
   - Language concepts and their rules
3. Create the notation
   - Representation of models
4. Define the generators
   - Various outputs and analysis of the models

- The process is iterative: try solution with examples
  - Define part of language, model with it, define more...
Roles for language definition & use

Experts define languages & generators

Team models with domain concepts & generate code, tests...
Step 1) Identify abstractions from domain terminology

- Detailed information specifying functional & physical characteristics of a component of a system, plant or facility (e.g. pump)
Design with domain-concepts & abstractions

Step 2) Specify the metamodel
Domain terminology and visualization: Valves
Step 3) Create the notation: Example on Valve
Step 4) Define the generators: PLC code for Valve actions

```c
/* generates entry and exit actions */
newline

do -("action") {
  do -("transition") { _gName(); }
  id; :Valve position; '();
}
newline
```
Developing the system with DSM
Generating the code from models

PLC code
How to test a cooling system?

1. Design time
   - Domain-Specific Modeling Language already captures several rules of the system
   - Language prevents errors already in the design stage

2. Testing and V&V
   - DSL can capture aspects related to testing and V&V
   - Same language concepts used for both development as well as for testing
How to test a cooling system?

- As components (e.g., pumps, values, heat exchanger) wear out, new components are substituted
  - Common for original requirements or design to not exist
  - May not know how current facility implementation deviates from original design or requirements
- Concern: newly substituted component can create potential operational or safety issues such as:
  - Temperature: Produce too much heat?
  - Pressure: Incorrect input/output pressure?
  - Flow rates: Conflicting flow rates in the configuration?
  - Control logic errors...
  - Instrument configuration...
Example: Cooling in process plant*

* M. Blackburn, P. Denno, Virtual Design and Verification of Cyber-Physical Systems
Specifying properties of components
Both structure and behavior

• Same objects: different views used to formalize different aspects of the system
• Languages integrated: can share objects used in different diagram types

Behavioral constraint: if valve is closed then pump should be closed else if value is open then pump can be open
Toolchain and stakeholder roles

- Conceptual representation of DSM and formal methods toolchain

<table>
<thead>
<tr>
<th>Role:</th>
<th>DSM Modeler</th>
<th>Application Engineer/Test Engineer</th>
</tr>
</thead>
</table>

1) Metamodel Dev.  
2) Diagram Dev.  
3) Generators/Transformation  
4) Analysis & Test Gen.  
- Simulation  
- Documentation  
- SCR/TTM

Formal methods analysis  
Test vector generation  
Requirement-to-test traceability  
Generator(s)

- b) Application Models  
- c) Generator Template(s)
DSM integrates with formal methods

MetaEdit+ -based DSM for Process Facility Design

T-VEC Tabular Modeler (TTM) and T-VEC Vector Generation System (VGS)

- Model transformation
- Formal methods analysis
  - Theorem proving
  - Property checking
- Test vector generation
- Test driver generation
- Requirement-to-test traceability
Potential Flow rate Issue (seeded defect)
Analysis identifies unsatisfiable constraints

TTM Table t_150

Hyperlink from analysis report to model defect

Failed Pre-Condition Relation: cv_equal_to
Exact SS File Location: t_150.SS
Input Domain At Error:
- t_V1_VAR.flow_rate: Valve_type_flow_rateDataType [0 .. 1200]
- Hx1.flow_rate: Heat_Exchanger_type_flow_rateDataType [11000 .. 11000]
Experiences from the CHL case

Model-based approach to:
- Analyze requirements and create design specifications
- Generate implementation code, deployment, docs etc.
- Generate tests and provide traceability of tests to corresponding requirements

Benefits
- Improved system validation
- Ability to better trace rationale
- Improved systems engineering
**Why DSM: emerging and enabling technology for V&V**

- Provides relevant and intuitive graphical abstraction for specific domain or related subdomains
- Allows for rich semantics required for formal analysis and test generation
  - Necessary for V&V effectiveness and efficiency
- DSM tooling allows multiple views to be integrated
  - Model transformation often built into the tools
  - Integrates formal analysis and test generation tools
  - Formal methods hidden behind the scenes
  - Model languages are evolvable
Summary

- DSM provides differing views of system designs across multiple disciplines
  - Approach enables collaboration between domain experts and developers, development and verification
  - Leverage and integrate new analysis and model-based automation (e.g., simulation, synthesis, generation, etc.)

- Future challenges – increase in modeling & analysis tools
  - Apply further views and multi-model consistency (e.g. in automotive)
  - Integrate with various testing tools, each focusing on different aspects
Thank you!
Questions, please?

For references on examples and cases contact:
Juha-Pekka Tolvanen, jpt@metacase.com
www.metacase.com
References

- USAF, ICSE, http://dl.acm.org/citation.cfm?id=227842