Model-Driven Development of Domain-Specific Applications: Tool Support

Yu Guo    Krzysztof Sierszecki    Christo Angelov
{guo, ksi, angelov}@mci.sdu.dk
Outline

1. Embedded software challenge:
   - Model-Driven Software Development
   - Domain-Specific Languages / Modeling
   - Component-Based Development

2. A brief introduction to COMDES: a domain-specific framework

3. Toolset: a software development environment supporting engineering of COMDES applications

4. Existing platforms supporting tool development

5. Conclusion
Embedded (Control) Systems

- Wide-spread use of embedded systems
  - Billions of microcontrollers are introduced every year
  - The majority of microprocessors are used in embedded applications
- Most of the embedded systems perform control and monitoring functions
- The complexity of software in embedded applications is rapidly increasing
- Remark: the system is statically defined
Embedded Software Development

Problem:
- Language incompatibility between the application and implementation domains
- The implementation is not always consistent with the specification

Solution:
- Domain-specific modeling techniques bridge the application and the implementation domains
- Model-Driven Software Development (MDSD) seems promising to solve the second problem

Diversity of hardware platforms, hard to generate 100% code
- Component-Based Development (CBD) increases the productivity of software development due to the reuse of components
- Requires adequate tool support to reduce the number of defects in code, compared to manual coding, thus improving software quality and dependability
COMDES: Building Embedded Systems Brick by Brick...

- A domain-specific framework providing modeling, analysis and implementation techniques for distributed real-time embedded control systems.
- COMDES is specifically intended for time-critical and safety-critical embedded systems.
- Patterns for pre-validated trusted components – basic, composite, state machine and modal function blocks.
- Design models for embedded systems combining open architecture and predictable behaviour, which are easy to use by application experts.
COMDES - An Overview

- System structure: actors are composed of function blocks
- Functional behavior: state machine, modal function blocks
- Timing behavior: actors are mapped to real-time tasks
- Allocation: actors are allocated to platforms
COMDES Framework

- A solution to the development of software for embedded control systems
COMDES Software Development Process

- The envisioned software development process will cover the main stages of system development, i.e. system software design, analysis, code generation and configuration from prefabricated components.
- System vs. component development

The engineering approach to COMDES system configuration
Application Configuration

- Final executable code consists of data structures (instances) and executable functions (types)
- Each particular instance has a corresponding data structure and is associated by its type with a number of functions
- During application synthesis no executable code is created but only data structures are instantiated, which 'glue' components together – glue codes
COMDES Development Toolset

- The development process requires adequate tool support in order to automate the steps that must be carried out during application development.

- The COMDES Development Toolset is intended to give a solution of the above problem.
COMDES Development Toolset

- The toolset supporting the framework integrates a number of tools, as well as it uses external tools to fulfill its tasks.
Requirements for Development Platforms

- Meta-modeling
- Constraint definition
- Graphical user interface / Editor
- Transformation
- Code generation
- Model reuse / Library
- Extensibility
GME

- Supporting graphic definition of domain-specific modeling languages and generation domain-specific graphic modeling environments
- By Vanderbilt University ISIS
- Pros: GUI, OCL, GReAT
- Cons: GUI customization
Cadena

- An Eclipse-based extensible integrated modeling and development framework for component-based systems
- By SAnToS lab in Kansas State University
- Pros: Model reuse, extensibility
- Cons: GUI, model transformation, code generation
MOFLON

- An integrated, standard-compliant meta-modeling environment
- By Real-Time Systems Lab, Technischen Universität Darmstadt
- Pros: meta-model, OCL, Story Driven Modeling (SDM) for the dynamic semantics
- Cons: GUI, code generation
MetaEdit+

- An environment for developing domain-specific modeling languages and code generators.
- Commercial tool – MetaCase
- Pros: GUI, code generation, support (since 1993)
- Cons: OCL, model transformation, extensibility
Eclipse Platform

- Can provide the technology foundation for a DSL development environment.
- Open source projects, www.eclipse.org
- Pros: There is a broad spectrum of solutions, which offer a variety of capabilities, EMF, GMF, OCL, Java, model transformation, code generation, etc.
- Cons: Some of them are not mature yet, changing frequently
Eclipse Platform

- Can provide the technology foundation for a DSL development environment.
- Open source projects, www.eclipse.org
- Pros: There is a broad spectrum of solutions, capabilities, EMF, GMF, OCL, Java, model transformation, code generation, etc.
- Cons: Some of them are not mature yet, changing frequently

Eclipse Modeling Project and the development of Domain-Specific Languages using a subset of its technologies.
## Survey Summary

<table>
<thead>
<tr>
<th>Metamodeling</th>
<th>GUI</th>
<th>Constraint</th>
<th>Dynamic semantics</th>
<th>M2M</th>
<th>Parser</th>
<th>M2T</th>
<th>Model reuse</th>
<th>Environment extensibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cadena</td>
<td>CALM</td>
<td>Build-in</td>
<td>No</td>
<td>Transformation</td>
<td>No</td>
<td>Cadena core APIs</td>
<td>No</td>
<td>Kind, Type, Instance, Cadena plug-in</td>
</tr>
<tr>
<td>GME</td>
<td>MetaGME</td>
<td>Built-in</td>
<td>OCL</td>
<td>Transformation</td>
<td>GReAT</td>
<td>BON</td>
<td>No</td>
<td>Type, Instance, Add-on mechanism</td>
</tr>
<tr>
<td>MOFLON</td>
<td>MOF2.0</td>
<td>No</td>
<td>OCL</td>
<td>SDM</td>
<td>TGG</td>
<td>Tailored API, Reflective API</td>
<td>No</td>
<td>No, No</td>
</tr>
<tr>
<td>MetaEdit+</td>
<td>GOPPRR</td>
<td>Symbol Editor</td>
<td>Rule templates</td>
<td>Transformation</td>
<td>No</td>
<td>MetaEdit+ API</td>
<td>MERL</td>
<td>No, No</td>
</tr>
<tr>
<td>Eclipse</td>
<td>ECORE</td>
<td>GMF</td>
<td>OCL, Java</td>
<td>Transformation</td>
<td>ATL, QVT</td>
<td>Tailored API, Reflective API, JET, 3rd party</td>
<td>No</td>
<td>Plug-in</td>
</tr>
</tbody>
</table>

28-08-09 7th Nordic Workshop on Model Driven Software Engineering, Tampere, Finland
Component & System Specification

- Meta-models and models are used throughout the whole development

```
Ecore     meta-meta-model
           |
           v
COMDES    meta-model
           |
           v
Repository model
           |
           v
Application model

Analyze
specify
define
reuse

kind  Basic FB

Multiplexer4Float

instance MotorControl

SignalSwitch
```
Experience on the Eclipse

- Component *Kind, Type, Instance*

![Diagram]

- Example: *Basic Function Block*
Toolset in Action
Summary

- COMDES framework provides a domain-specific language for embedded control systems
- A number of existing platforms has been investigated
- An initial prototype of the toolset has been developed on the Eclipse platform
- The engineering of a domain-specific software development environment is not an easy task
- The issues investigated in context of COMDES will hopefully be useful to other research and development projects in the area of software development environments based on models and components.
Thank You ...

Спасибо

謝謝

ありがとう

Tak

Köszönöm

Благодаря

Multumesc

धन्यवाद

Danke

Merci

Так