A Generic Executable framework for Model-Driven Engineering GEMDE

Eclipse Embedded Day 2011

Adrián Noguero
adrian.noguero@tecnalia.com
• Why GEMDE?
• Internal architecture
• Configuring the tool
  – Defining correct models
  – Defining model exploitation strategies
  – Creating your development stages
• Extending GEMDE
• Establishing a Model-Driven Engineering process is a tough task!
  – Too many *languages*…
  – Too many *editors*…
  – Too many *transformation engines*…
  – Too many *transformations*…
  – Too many *disconnected tools*…

• How can we manage all these issues coherently?
Why GEMDE?

• GEMDE is **generic** a toolkit, built on Eclipse, for the implementation of a **Model-Driven Engineering** process
  – It can be used independently from the model/code editors

• **Integrates tools** for:
  – **Model validation**. Define and check the rules that define a correct model
  – **Model transformation**. The processes that exploit the models (e.g. for code generation)
  – Linking with **third party tools**. Connect design models with analysis tools in early stages.

• Define an **executable methodology** easily
  – Step by step execution
  – Automated generation of models
• Defining the **correctness** of the models is vital for any MDE process
  – Reduces errors in model exploitation phases
  – Clarifies the semantics

• OCL enables the definition of constraints, however…
  – OCL is very complex
  – Not easily reusable / manageable
GEMDE provides a user-friendly interface to OCL. Experts may use the OCL editor.
Constraints are stored in a **repository** and are organized in validation packages.

Select the constraints that will be included in the validation package.
Configuring GEMDE – Defining correct models

Validation packages are executed against individual model elements

GEMDE enables exporting constraints and validations as plug-in extensions to be used in your MDE processes
Models are typically exploited through

- **Transformations**: code/docs generation, model2model transformations…
- **Tools**: schedulability analysis (Symta/S), performance analysis (PEPA)…

Too many tools

- Transformation engines: ATL, Acceleo, JET, Xtend2, Xpand, MOFScript…
- Thousands of analysis tool types: schedulability, fault trees, safety cases, energy consumption…
GEMDE provides a unified interfaces for all transformation engines and tools.
Configuring GEMDE – Model exploitation strategies

Transformations are managed from a central repository

New transformations can be easily defined inside the tool
• GEMDE allows the user to define a step-by-step MDE methodology using Eclipse Cheatsheets.

- Define tasks to ease the adoption.
- Enable methodology execution.

Methods might be related to some tools, customizing the required input. GEMDE manages this using the concept of an "active method."
Configuring GEMDE – Defining development stages

Cheatsheet enable both definition and execution of the MDE process!

GEMDE services are available for methodology execution!
• GEMDE is a **framework tool** with many extension points
  – Statically contribute with OCL / Java constraints & validations
  – Statically contribute with transformations
  – Contribute new model transformation engines
  – Contribute with new interfaces to third party tools
  – Create new methods
• Contributions use the plug-in mechanisms of Eclipse
THANK YOU FOR YOUR ATTENTION!

Questions?