New and Noteworthy

openArchitectureWare 4.2

Peter Friese
peter.friese@gentleware.com
http://www.gentleware.com

Bernd Kolb
b.kolb@kolbware.de
http://www.kolbware.de
What is openArchitectureWare?
- oAW Languages
- New: AOP Mechanisms
- New: Debugger
- New: Xtext 2
- New: Improved Help
- New: Hyperlinking
- New: Refactoring
- New: Product Line Engineering
What is openArchitectureWare?

- MDSD generator framework
- MDSD = Model Driven Software Development
- Reads models (arbitrary meta meta models)
  - Ecore
  - MOF
  - ...
- Validates Models
- Transforms models
  - Model to Model (M2M) using Xtend Language
  - Model to Text (M2T) using Xpand Language
- Generator is controlled using Workflow Engine
  - Dependency Injection / Reflection driven
  - Supports custom workflow components
Inside the generator

- Model
- Model instantiator
- Model validator
- Model to Model Trafos
- Code generation templates
- MDSD generator
- Workflow engine
- Generated artifacts
The basis for all oAW languages
OCL-like syntax
Can be mapped to any meta meta model
  Implementations available for
    • Ecore
    • UML2
    • Java Beans
    • many UML-Tools (RSM/RSA, MagicDraw, Enterprise Architect, Poseidon...)
Good support for graph navigation
Some examples:
  • Built in higher order functions like collect, select
  • Built in utility functions like toList, toSet, ...
  • typeSelect for selecting all Elements of a specific metatype
  • Support for collect abbreviation
    • listA.listB.listC -> take all Elements from listA, ask each of the elements for listB and call listC on each of the resulting elements
- XML-based language
- Describe a sequence of steps to process your model
- Uses the Inversion-of-Control (IoC) pattern
- Tool support
- Interface(s) for you to integrate your own workflow components

```xml
<workflow>
    <!-- load model and store it in slot 'model' -->
    <component class="org.eclipse.mwe.emf.Reader">
        <uri value="platform:/resource/${model}" />
        <modelSlot value="model" />
    </component>

    <!-- generate code -->
    <component class="org.openarchitectureware.xpand2.Generator">
        <metaModel idRef="mm"/>
        <expand value="template::Template::main FOR model" />
        <outlet path="${src-gen}" />
    </component>
</workflow>
```
- Annotate “methods” on meta types
- Functional language for model-to-model (M2M) transformations
- Polymorphism (multiple dispatch)
- Language support for creating and caching model
- Good tool support (syntax highlighting, code completion, …)

```java
import metamodel;
getter(Feature this) : 
 "get"+name.toFirstUpper();
setter(Feature this) : 
 "set"+name.toFirstUpper();

import metamodel;
modify(Entity this):
 setName(this.name + "Entity")
 -> features.add(createFeature())
 -> this;
create Feature this createFeature(Entity e):
 setName("PK_" + e.name);
```
Declarative language
- Validate your model
- Report warnings
- Report errors
- Good tool support (syntax highlighting, code completion, ...)
- Extensions
- Template polymorphism
- Language developed for code generation only
- Extensions defined in Xtend can be used
- Good tool support (syntax highlighting, code completion, ...)

```xtext
DEFINE javaClass FOR Entity
  FILE name+.java

  public class «name» {
    FOREACH features AS f
      private «f.type.name» «f.name»;
      public void «f.setter()»(«f.type.name» «f.name») {
        this.<f.name> = «f.name»;
      }
      public «f.type.name» «f.getter()»() {
        return «f.name»;
      }
    }
  }

ENDFILE
ENDDEFINE
```
When creating reusable generators, we want to change generator behavior without changing the generator itself

- E.g. if we use a 3rd party cartridge
- Using AO mechanisms, it is possible to override templates

```
<workflow>
  <cartridge file="workflow/workflow.oaw"/>
  <component
    adviceTarget="generator",
    class="org.openarchitectureware.xpand2.GeneratorAdvice">
    <advices value="templates::Advices"/>
  </component>
</workflow>
```
- Changing the generator templates is not enough. **Transformations** and **metamodel operations** have to be changed as well.
- Thus Xtend supports Aspects

```
extension org::openarchitectureware::util::stdlib::io;

around my::generator::*(*) :
    info('Invoking ' + ctx.name) -> ctx.proceed();
```

```
<component class="org.openarchitectureware.xtend.XtendAdvice">
    <adviceTarget value="extension"/>
    <extensionAdvices value="metamodel::aop"/>
</component>
```
Features:

- Set breakpoints in workflow, transformations (M2M / M2T)
- Inspect execution context
- Inspect variables
- Java debugging
Features:

- Specify language grammar for textual DSL
- Xtext generates
  - The meta model
  - ANTLR 3 based parser
  - Text editor
    - syntax highlighting
    - folding
    - outline
    - content assist
  - Generator project (workflow, template, extensions,...)
- Use the Check Language to validate your models inside an GMF-Editor
  - Support for cyclic validations
- oAW comes with several utilities for PLE
  - Xweave
    - support for weaving several models together using an aspect model
  - Xvar
    - Selectively remove parts from your model or code
  - Support for feature models to configure the overall generator
    - A primitive one for features in text files
    - Integration with pure::variants
Improved documentation

- Welcome Page
- Online Help / PDF
- Video Tutorials
- Cheat Sheets

Creating an oAW generator project

Introduction

This cheat sheet will guide you through the process of creating a new oAW-based MDSD generator.

You will create a new oAW project with a sample metamodel and try it out.

Let's get started!

1. Click to Begin
2. Open the openArchitectureWare perspective
3. Create a new oAW project
4. Running the generator

Integrating Recipes

Transforming Models

This example uses Eclipse EMF as the basis for its EMF support. While not all aspects of EMF are available, 3rd party tools make EMF a good basis for the horizon already. To get a deeper understanding tutorial at:


You can also run the tutorial without completely unzipping.

Installing the pre-built tutorial

You need to have openArchitectureWare 4.2 install at:

- [http://www.eclipse.org/ght/gmt/download](http://www.eclipse.org/ght/gmt/download)

You can also install the code for the tutorial. It can be EMF samples ZIP file. Installing the demo is easy openArchitectureWare preferences (e.g. globally turn off EMF preferences for the demo, access...)
Productivity Features

- Refactoring
- Hyperlink navigation
- Search for references / declarations
Next steps

- oAW is moving into several Eclipse Modeling Projects
  - Workflow -> EMFT Modeling Workflow Engine (MWE)
  - Xpand -> Eclipse Model to Text (M2T)
  - Xtend -> Eclipse Model to Text (M2T)
  - Support for meta meta models -> Eclipse Model to Text (M2T)

- oAW is submitting a proposal for a new modeling project
  - Textual Modeling Framework (TMF)

- Improve the different languages

- Improve tooling and IDE support (SAFARI / IMP?)
- www.openarchitectureware.org
- www.eclipse.org/gmt/oaw
- www.eclipse.org/emft
- www.eclipse.org/m2t