

# oAW 4 Introduction and Overview

Markus Voelter, [voelter@acm.org](mailto:voelter@acm.org), [www.voelter.de](http://www.voelter.de)



## **Table of Contents**

<b>INTRODUCTION.....</b>	<b>2</b>
<b>CORE FEATURES.....</b>	<b>2</b>
<b>IDE FEATURES.....</b>	<b>4</b>
<b>GUIDE TO THE DOCUMENTATION.....</b>	<b>5</b>
<b>GETTING SUPPORT.....</b>	<b>6</b>

## Introduction

openArchitectureWare (oAW) is a suite of tools and components to assist in model driven software development. It is built upon a modular MDA/MDD generator framework implemented in Java(TM) and supports arbitrary import (design) formats, meta models, and output (code) formats. Supporting tools (such as editors and model browsers) are based on the Eclipse platform.

openArchitectureWare is a “tool for building MDS/MDA tools”. At the core there is a workflow engine allowing the definition of transformation workflows as well as a number of prebuilt workflow components that can be used for reading and instantiating models, checking them for constraint violations, transforming them into other models and then finally, for generating code.

## Core Features

The following is a comprehensive list of the current features of openArchitectureWare.

- The workflow engine precisely controls the generator's workflow, as specified in an XML workflow definition.
- With a suitable instantiator, oAW can read any model. Currently, we provide out-of-the-box support for EMF , various UML tools (MagicDraw, Poseidon, Enterprise Architect, Rose, XDE, Innovator, ...), textual models (using JavaCC or antlr parsers as frontends), XML, Visio as well as pure::variants variant configuration models.
- Can generate any kind of output, using a powerful template language called Xpand that supports template polymorphism, template aspects and many other advanced features necessary for building non-trivial code generators such as optional typing of model elements and sorting and filtering of element sets.
- Explicit domain metamodel: the metamodel of the problem domain is represented as Java classes. Metamodels can be built using either Eclipse EMF or oAW's own metamodel generator that creates those Java classes from metamodels rendered in UML. The generator already comes with the most important UML metaclasses, which can be reused to build UML profiles.
- Various ways of checking model constraints: a semi-declarative checking using Java APIs, a proprietary declarative constraints checking language as well as optional OCL integration based on the Kent OCL framework.
- Support for multiple models: you can read several models as part of one generator run. These models can use different concrete syntaxes (one could be UML, another XML and a third one Visio). The generator then “weaves” these models together to

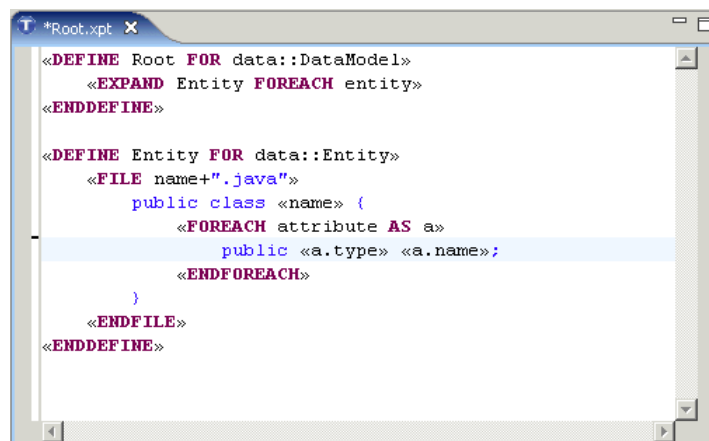
form a comprehensive, all-encompassing model. Constraints can be checked over model boundaries; references among models can be established.

- Model-to-model transformations can be implemented using the Wombat language, a textual, functional transformation language. There are two very important properties of this language: first, it has a very concise and powerful syntax. Second, it can transform between the various metamodels, *e.g.*, you can transform a model defined with the oAW-classic metamodel into an EMF model! Alternatively you can integrate third party transformation languages such as ATL. An Adapter is provided.
- The Recipe Framework allows you to define validation rules for artefacts created outside of the generator (such as manually written subclasses). During code generation, these rules can be instantiated; later, the Eclipse IDE will read these checks and verify them. This will help to guide developers beyond the modelling/generation stage.
- Complete modular design: You can enhance or extend any of the framework components as you see fit

## IDE Features

openArchitectureWare comes with a number of Eclipse Plugins that help make development more efficient. Here are a couple of screenshots.

The following one shows the Xpand template editor. In addition to syntax highlighting, it provides metamodel-aware code completion facilities as well as static error checking.



```
«DEFINE Root FOR data::DataModel»
  «EXPAND Entity FOREACH entity»
«ENDDFINE»

«DEFINE Entity FOR data::Entity»
  «FILE name+".java"»
  public class «name» {
    «FOREACH attribute AS a»
      public «a.type» «a.name»;
    «ENDFOREACH»
  }
  «ENDFILE»
«ENDDFINE»
```

You can also define metamodel extensions, *i.e.*, properties on metaclasses, that are available in the template language. They are defined externally to the metamodel implementation classes.

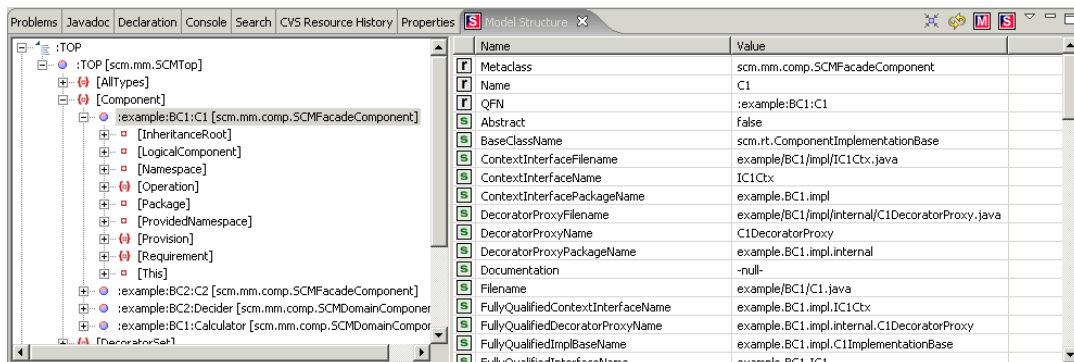
```

Extensions.ext
import org::openarchitectureware::meta::uml;
import org::openarchitectureware::meta::uml::classifier;
import org::openarchitectureware::core::meta::core;
import scm::mm::base;

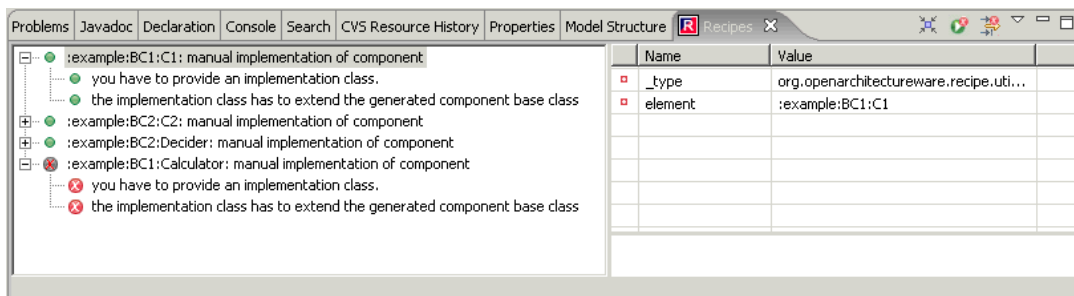
JavaTypeName(Element element) :
    PrimitiveType.isInstance(element) ?
        ((PrimitiveType)element).Name
    :
        ((SCMClass)element).FullyQualifiedName;
/**
 * Documentation
 */
UpperCaseName(Element e) :
    ((String)e.Name).toFirstUpper();

```

A model structure view helps you view and understand models that have been parsed from UML or other sources. For EMF based models this is not necessary, since the EMF-supplied editors provide exactly the same functionality.



The checks verified by the recipe framework can be rendered nicely in an Eclipse view. Changing something in the manually written source code will update the checks in real time.



In subsequent versions, more editors (such as editors for workflow files, wombat transformation scripts and constraint checks) will be added.

## Guide to the Documentation

Please consider the following tips when reading the documentation:

- The first thing you should read is the installation documentation. This will help you set up openArchitectureWare correctly.
- Then you should read the workflow documentation. All aspects of oAW are controlled by the workflow file. Understanding how it works is crucial.
- Assuming you want to migrate projects from oAW 3 or otherwise need to use the *Classic* mode of oAW 4, please see the *MigrationAndOAWClassic* documentation. It shows how to work without EMF and use the “old style” openArchitectureWare. Note that you should be familiar with oAW 3 before reading this documentation; if you’re not, you should read the docs for the oAW 3 exampleWorkspace first.
- If you want to work from EMF models, take a look at the *emfExample* documentation. It shows you how to generate code from EMF models.
- You can then selectively read the other examples including the *uml2Example*, the *oclExample*, the *atlExample* and the *wombatExample*. Note that you really should read the preceding documents in this listing!
- Finally, you should use the various reference docs to understand details about the various aspects of oAW 4.

## Getting Support

To get support, you should go to [www.openArchitectureWare.org](http://www.openArchitectureWare.org). In the community section of the web site, specifically in the forums, you can ask all kinds of questions. Professional support is also available – again, see [www.openArchitectureWare.org](http://www.openArchitectureWare.org).

## About our Sponsors

**itemis GmbH & Co. KG** is an independent IT service company with an emphasis on consulting, coaching, and software development. Every single item's expert provides many years of project experience and widespread knowledge about all object oriented and component based software development issues - especially in the field of model driven software development.

**b+m** is the founder of the openArchitectureWare project. The software was originally developed within the scope of many successful projects. b+m opened the software to the community in late 2003. All of the paradigms of Model-Driven Software Development including Product Line Engineering and not only the generator framework have become a key concept for product and customer specific development at b+m. b+m customers can make use of long time experience and substantial know-how in that field. Located at the company headquarters in Melsdorf/Kiel and at its subsidiaries in Berlin, Cottbus, Hamburg, Hanover and Kiel the b+m staff of 205 provides practical solutions for customized business applications, business process optimization and comprehensive architecture, project and quality management.

**oose Innovative Informatik GmbH** offers coaching, consulting and training in all areas of software engineering. The main focus of their activities are software architecture, requirements engineering and project-management. oose has first-hand information and experience, because its staff takes part actively with others in current trends, standards and innovations. Our staff support this and pass their know-how regularly on by writing and publishing books or speaking at conferences, etc. Within the OMG oose collaborates actively on the specifications of the UML and the SysML.

**MID Enterprise Software Solutions GmbH** is a leading supplier of optimized tool environments for standardsbased and model-centric software development as well as business process modeling. This includes professional tool consulting and tool components to build a complete tool environment using the best techniques and tool modules available - Architectural and Operational Excellence. With innovatorAOX, MID provides a holistic standard tool environment for object- and function-oriented software development as well as business process and data modeling to help its customers establish highly efficient processes and tool environments for software production, The unique and seamless integration of business process modeling into the development process ensures an unprecedented level of convergence of business requirements and implemented IT systems. Project members from all departments speak the same language and all requirements are clearly described.