Workpackage 3: Model Engineering

Deliverable D3.1.b: Model Weaving
**Contract Number**: 034081  
**Project Acronym**: MODELPLEX  
**Title**: MODELling solution for comPLEX software systems

**Deliverable N°**: D3.1.b  
**Due Date**: 01/2007  
**Delivery Date**: 03/2007

**Short Description**:  
Report on Model Weaving Prototypes

**Lead Partner**: TU Dresden  
**Made available to**: Public

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Author</th>
<th>Checked by</th>
<th>Internal Approval</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>16/11/06</td>
<td>Steffen Zschaler, TUD</td>
<td>Jendrik Johannes, TUD</td>
<td></td>
<td>Initial Version</td>
</tr>
<tr>
<td>0.2</td>
<td>25/01/07</td>
<td>Steffen Zschaler, TUD</td>
<td>Jendrik Johannes, TUD</td>
<td></td>
<td>Modified outline based on information on precise requirements</td>
</tr>
<tr>
<td>0.3</td>
<td>30/01/07</td>
<td>Jendrik Johannes, TUD</td>
<td>Marcos Didonet Del Fabro, INRIA</td>
<td></td>
<td>Added material on Re- useware</td>
</tr>
<tr>
<td>0.4</td>
<td>31/01/07</td>
<td>Marcos Didonet Del Fabro, INRIA</td>
<td>Jendrik Johannes, TUD</td>
<td></td>
<td>Added material on AMW</td>
</tr>
<tr>
<td>0.5</td>
<td>31/01/07</td>
<td>Jendrik Johannes, TUD</td>
<td>Marcos Didonet Del Fabro, INRIA</td>
<td></td>
<td>Refinements</td>
</tr>
<tr>
<td>0.6</td>
<td>31/01/07</td>
<td>Marcos Didonet Del Fabro, Inria</td>
<td>Jendrik Johannes, TUD</td>
<td></td>
<td>Refinements</td>
</tr>
<tr>
<td>0.7</td>
<td>02/02/07</td>
<td>Jendrik Johannes, TUD</td>
<td>Marcos Didonet Del Fabro, INRIA</td>
<td></td>
<td>Refinements</td>
</tr>
<tr>
<td>0.8</td>
<td>12/02/2007</td>
<td>Steffen Zschaler, TUD</td>
<td>Dimitrios Kolovos, YORK</td>
<td></td>
<td>Updates corresponding to reviews received</td>
</tr>
<tr>
<td>0.9</td>
<td>21/02/2007</td>
<td>Steffen Zschaler, TUD and Marcos Didonet Del Fabro, INRIA</td>
<td>Jan Aagedal, SINTEF</td>
<td></td>
<td>Updates corresponding to reviews received</td>
</tr>
<tr>
<td>0.10</td>
<td>12/03/2007</td>
<td>Jendrik Johannes, TUD and Marcos Didonet Del Fabro, INRIA</td>
<td></td>
<td></td>
<td>Updates corresponding to reviews received</td>
</tr>
<tr>
<td>1.0</td>
<td>15/03/2007</td>
<td>Final Deliverable</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table of Content

1. Executive Summary ......................................................................................................4
2. Introduction ...................................................................................................................4
3. Glossary of Abbreviations ............................................................................................5
4. Reuseware and AMW – A Comparison of Technologies ..............................................5
5. Description of Prototypes ............................................................................................6
   5.1. Reuseware: Using Model Weaving to Separate Computation and Communication 6
      5.1.1. Using Model Weaving to Separate System Aspects .............................................6
      5.1.2. Prototype installation and execution ....................................................................7
   5.2. AMW: Using Model Weaving for Transition Between Meta-Model Versions ......11
      5.2.1. Transition between metamodel versions ..............................................................11
      5.2.2. Tool design ..................................................................................................13
6. Conclusions and Summary .........................................................................................16
7. References ....................................................................................................................16

List of Figures

Fig. 1 A comparison of AMW and Reuseware .................................................................6
Fig. 2 The core model: A model of a file system .............................................................7
Fig. 3 An aspect model: Observer pattern .......................................................................7
Fig. 4 Reuseware Update Site .......................................................................................8
Fig. 5 Creating a new Reuseware Project in two steps ...................................................9
Fig. 6 Models in the components folder and the Weaving Composer in the composers 
   folder .........................................................................................................................10
Fig. 7 Weaving of the observer aspect #1 .....................................................................10
Fig. 8 A woven model in Reuseware ............................................................................11
1. Executive Summary

This document accompanies the prototypes that form deliverable D3.1.b “Model Weaving”. It compares the two basic technologies made available by the participating Parties and proceeds to present the installation and use of the two tool prototypes.

Apart from show-casing different basic technologies for model weaving, the two prototypes also cover different applications of model weaving, namely separation of the concerns of communication and functionality, and model migration.

2. Introduction

This document accompanies the prototypes that form deliverable D3.1.b “Model Weaving”. There will be releases of further instances of D3.1.b in months 11 (full report) and 17 (extended prototypes) of the MODELPLEX project. This document presents initial model-weaving prototypes, to be delivered in month 5.

Model weaving is about merging different models, so that the information contained in one model is added to information from another model in potentially many places throughout that model. In other words, information of one model crosscuts information represented in another model. Cross-cutting here means that different aspects of the modelled system interact with each other. When such aspects are modelled separately and are combined through model weaving, crosscuts are made explicit (and are potentially materialised in the resulting model). An exact definition of the term model weaving and how it relates to the notion of Aspect Oriented Modelling (AOM) is not completely agreed on at the moment. A clear definition, as a first proposal for a standardized definition of the term, will be one of the contributions of the month 11 deliverable.

Model weaving can be used for a multitude of purposes and can be realised in different ways. Consequently, we have decided to develop more than one prototype. In this way, MODELPLEX’ potential solution space is not narrowed down prematurely. Instead, we can identify and pursue the best technology and application for the problems posed by the case studies defined in WP1. In this document, we report on the following two prototypes:

1. Using model weaving to separate the concerns of computation and communication. This prototype has been implemented based on the Reuseware Composition Framework from TUD (http://reuseware.org/ecoreweaving).

2. Using model weaving at the meta-model level. We present one scenario that handles the transition of models between different versions of meta-models. This prototype has been implemented based on the AMW (ATLAS Model Weaver) toolkit from INRIA (http://www.eclipse.org/gmt/amw).

This report is structured as follows. In Section 4, we more closely inspect the differences and commonalities between the two technologies used. In Section 5 the actual prototypes are presented. We conclude with an outlook on further development of the prototypes and further directions of work.
3. Glossary of Abbreviations

- AMW – ATLAS Model Weaver
- ATL – ATLAS Transformation Language
- ATLAS – Atlantic Data Systems
- EMF – Eclipse Modelling Framework
- Ecore – Metamodelling Language of EMF
- WP – Work package
- TUD – Technische Universität Dresden
- INRIA – Institut National de Recherche en Informatique et en Automatique

4. Reuseware and AMW – A Comparison of Technologies

Reuseware and AMW both implement very basic technologies that can be used to implement model weaving. While Reuseware focuses on model composition—that is, merging two models to create a new, combined model—AMW’s focus lies on model transformation, leading to some differences in their respective approach to model weaving. Fig. 1 shows the two tools applied to model weaving. AMW uses a weaving model to describe the model elements to be related in the weaving process. This weaving model is external to the models to be woven, those models do not need to be modified or prepared to be useable for model weaving. The weaving model is then translated into a transformation recipe that can be executed by the ATL engine producing the woven model. This transformation takes the two input models conforming to metamodels A and B, and executes the desired operation. In contrast, Reuseware expects models that have been specifically produced to be composed. In particular, the models must provide a so-called composition interface, exposing model elements that can be accessed and manipulated during weaving. Elements of the composition interface of the two models can then be provided as parameters to composition operators provided by, or built on, the Reuseware Composition Framework. Executing the composition operator produces a woven model.

Although the two tools are based on different technology, both provide weaving capabilities in that they allow to crosscut one model with another one, producing a woven model. At the present time, it is not clear what are the precise advantages or disadvantages of the two technologies when compared to each other. This means, that it is not presently clear, from which technology MODELPLEX will benefit the most. It would, therefore, be unwise at the current time to reduce the space of potential solutions by focusing on one tool only. Hence, we have built exemplary, initial prototypes based on each technology. It should be judged based on input from the case studies which technology to use for which use case.
5. Description of Prototypes

In this section the two prototypes are described in detail on one example each. That includes descriptions of how to install the prototypes and run the examples.

Both prototypes are built on top of the Eclipse Platform, which needs to be installed first. If you have already a copy of the Eclipse Platform (version 3.2.x) running on your machine you can use it. Another copy can be acquired from http://www.eclipse.org/downloads/. Different downloads for different operating systems (Windows, Linux or OS X) are available. Usually you will get the correct version for your machine offered at the download site.

After downloading, simply extract the archive to a folder of your choice and run the Eclipse Platform by clicking on the Eclipse icon in the folder.

5.1. Reuseware: Using Model Weaving to Separate Computation and Communication

This section describes the model-weaving prototype based on the Reuseware Composition Framework developed and maintained at TUD. After introducing an example that can be executed by the prototype, the section continues with a tutorial-like structure: It describes how to install the Reuseware Composition Framework and the prototype itself. The former is a prerequisite for this prototype and hence needs to be installed as well. After installation, we will demonstrate how the model weaving example can be executed using the prototype.

5.1.1. Using Model Weaving to Separate System Aspects

The example utilizes the modelling language Ecore provided by the Eclipse Modelling Framework [1]. We chose this modelling language for an initial example because it has a manageable size and good tool support inside the Eclipse Platform already. However, the Reuseware Composition Framework can be used to easily create similar weaving systems for other modelling languages.
The basic idea is to design a system in which the relationship between different objects is modelled in a core model, while the communication between the objects is separately modelled in an aspect model. Three main advantages are gained by this approach:

- It leads to a better understanding of the overall system, since its model is decomposed in two different models describing separate concerns of the system.
- Communication between classes can be modelled in different ways by creating different aspect models. These aspect models can be easily exchanged as different alternatives in the weaving.
- An aspect model, once defined, can be reused in combination with different core models or at different positions within one core model.

The concrete example we present models a file system. The core model presented below models the structure of a file system that consists of a central class (FileSystem), files, folders and views on the system (e.g., a table of content of a folder like known from conventional operating systems). While some structure of the system is described in the core, no communication between different elements is modelled.

![Fig. 2 The core model: A model of a file system](image)

To model communication, we utilize two design patterns: a simplified version of the commonly known observer and mediator patterns [2]. We refer to these two models (the observer model is shown below) as aspect models.

![Fig. 3 An aspect model: Observer pattern](image)

The complete example files can be obtained from: [http://reuseware.org/examples/ecoreAOM.zip](http://reuseware.org/examples/ecoreAOM.zip). In the following, we walk through the complete example in the prototype.

### 5.1.2. Prototype installation and execution

In the following, we first describe how to obtain the prototype and the Reuseware Composition Framework it is based on. After that, we show how to execute the example presented above in the installed prototype.
5.1.2.1. Installing the Reuseware Composition Framework and the Prototype

Inside the running Eclipse Platform the Eclipse Update Manager can be used to install the Reuseware Composition Framework and the model weaving prototype based on it. To launch the Update Manager, select Help > Software Updates > Find and Install...

Select Search for new features to install in the upcoming dialogue. You get presented a list of update sites to which the Reuseware Update Site needs to be added. Do so by selecting New Remote Site... and fill the form as shown below.

![Reuseware Update Site](image)

Fig. 4 Reuseware Update Site

Select the Reuseware Update Site and the Callisto Discovery Site and press Finish. The latter needs to be contacted to obtain the Eclipse Modelling Framework that is used by the Reuseware Composition Framework. The update sites are now contacted (make sure you are connected to the internet).

What you now see is a list of available updates on the selected sites. Select the following:

- Callisto Discovery Site > Models and Model Development > Eclipse Modeling Framework (EMF)
- Reuseware Update Site > Reuseware Composition Framework
- Reuseware Update Site > Generated Composition Systems > Ecore based composition system

The latter includes the model weaving prototype.

Note: Select also the following elements to obtain a graphical editor to visualize Ecore models.

- Enabling Features > Batik
- Graphical Editors and Frameworks > Graphical Editing Framework
- Models and Model Development > Graphical Modelling Framework

Continue with the installation by following the instructions. When the installation is finished, you are asked to restart the Eclipse Platform. The installation of the Reuseware based model weaving prototype is now finished. We will now show how the example can be executed using the prototype.

5.1.2.2. Executing a Model Weaving

The first step to execute the weaving in the prototype is to create a new Reuseware Project inside the Eclipse Platform. Do so by selecting File > New > Project... and select Reuseware > Reuseware Project in the upcoming dialog (Fig. 6 on the left). Configure the project by giving it a name. You may leave the predefined folder names (Fig. 6 on the right).

The structure of the created project is the generic Reuseware Project structure. The first folder of interest is the components folder. We regard core model as well as aspect models as components in the system. Thus, all models have to be placed in the components folder. You should be able to simply drag and drop the model files (*.ecore) from the extracted example archive into the folder in the running Eclipse environment.
The composition interface is, in this case, formed by all classes in a model. The Reuseware Composition Framework implicitly makes all classes part of the composition interface, so no special markup is required. Other applications of the Reuseware Composition Framework may require the composition interface to be defined explicitly.

Fig. 5 Creating a new Reuseware Project in two steps

As the next step, we have to describe a composition operator that defines how one class is woven into another. Composition operators are a general concept in Reuseware. They know how to perform a certain composition task. In the case of this example, the required composition operator needs to be able to weave classes from the aspect model into classes of the core model. The composition operator can be reused in other (Ecore-based) examples. Thus, it is part of the prototype and not bound to this concrete example. The Ecore weaving composition operator (*ecore-Weaver.bc*) has to be placed into the *composers* folder.
The last step before the weaving can actually be executed is to specify where an aspect model is woven into a core model. Such a specification is called a composition program in Reuseware. Composition programs are placed into the compositionProgram folder. Three different weavings are included in the example. They are specified in composition programs of the type *.eweaving. Here, we discuss only one weaving: ObserverWeavingFS.eweaving, in which the communication aspect Observer should be woven into the core model such that it realizes communication between the FileSystem class and the FSFolder and FSFile classes.

First, the core and the aspect are loaded. Then, the Ecore weave operator is executed (weave keyword). As arguments, the weaver takes several pairs of positions. The first position of each pair is pointing at several classes in the core model. The second always points at a class in the aspect model (CoreModelClasses <- AspectModelClass). Classes are identified by their names in the models. Regular expressions can be used to address a set of classes. In this example, the Observer class from the aspect is woven into the FileSystem core class. The Subject class from the aspect is woven into all core classes that have names starting with “FS”.
Two more weavings are provided with the example: MediatorWeavingFS.eweaving used Mediator instead of Observer for the communication. This is done by loading a different aspect model and changing the position parameter in the weaver execution. As mentioned, an aspect model can be used at multiple points in the same core model. In the third weaving (ObserverWeaving-View.eweaving), the observer aspect is reused to express communication between folders/files and the different view classes. The difference to the first weaving is that the view classes (“*.View”) are addressed for weaving instead of FileSystem.

Since the prototype makes use of the automatic incremental build mechanism of the Eclipse Platform, the woven models automatically appear in the out folder; given, that all core models, aspect models, and weaving definitions are correct. The output of the first weaving is displayed in Fig. 8.

Note: To obtain a class diagram representation for a woven *.ecore model in the output folder, right click on the file and choose Initialize ecore_diagram diagram file.

5.2. AMW: Using Model Weaving for Transition Between Meta-Model Versions

This section describes the AMW (ATLAS Model Weaver) prototype for model weaving. AMW provides a base weaving metamodel that enables the creation of links between model elements [4] [5]. The metamodel may be extended to add different semantics. Weaving models can be used in numerous application scenarios, such as metamodel comparison, traceability, model matching, model annotation, tool interoperability.

In this document we present the Model Weaver prototype and how it may be used for metamodel comparison and model transition. A set of different possible use cases of AMW is found in the Use cases section of the AMW web site (http://www.eclipse.org/gmt/amw/usecases/). AMW is available for download as an Eclipse GMT component (http://www.eclipse.org/gmt/amw/).

5.2.1. Transition between metamodel versions

Consider two versions of the same metamodel, MMA (v1) and MMA (v2). The different versions have been developed by two different persons, based on a formal specification. The objective is to transform the models conforming to MMA (v1) into the models conforming to MMA (v2). In other words, it is necessary to perform the transition between metamodel versions.

The AMW plug-in is used to produce a weaving model between these two metamodels. This weaving model contains equivalence links between the metamodel elements. A screenshot of a weaving model between two versions of a same metamodel is illustrated in Fig. 10. MMA (v1) is illustrated in the left panel, MMA (v2) in the right panel, and the weaving model is in the middle panel. The selected element in the middle panel represents an equality link between MtcApplication, which is present in both versions. Only equivalence or equality links are created (e.g., there is no...
element linked with the Project class in MMa (v2). This example uses two Scade metamodels (the metamodels are not fully described in this document because they contain approximately 200 elements). Scade is a standard for development of embedded software for the Avionic industry. A complete description of this application scenario is available at http://www.eclipse.org/gmt/amm/usecases/compar/). The weaving model is created automatically after the execution of a set of matching transformations.

Matching transformations are transformations that execute different heuristic methods that analyze the properties of the metamodel elements [3]. This information is used to assign a similarity value between every pair of elements of both metamodels. For instance, a high similarity value is assigned to elements that have the same name and type (e.g., the similarity between two MtcApplication elements is set to (1) one). A high similarity value means that the elements have a good probability to be equivalent. This enables to create weaving models semi-automatically, saving valuable time in the transition process, especially if the two metamodels have several model elements.

The weaving model conforms to a weaving metamodel that is an extension of a core weaving metamodel, which defines elements that support basic link management. The metamodel extension is illustrated in Fig. 11.

```plaintext
class Equivalent extends WLink {
    attribute similarity : Double;
    -- @subsets
    reference left container: ReferredElement;
    -- @subsets
    reference right container: ReferredElement;
}
class ReferenceEqual extends Equal {
}
class ElementEqual extends Equal {
}
class NotEquivalent extends WLink {
    -- @subsets
    reference left container: ReferredElement;
    -- @subsets
    reference right container: ReferredElement;
}
class AttributeEqual extends Equal {
}
```

Fig. 11 Weaving metamodel extension

The metamodel extension contains an Equivalent link. This element has a similarity attribute that saves the similarity estimation between the elements of the two metamodels (MMa (v1) and MMa (v2)). This link is extended by other links, depending on the type of elements that are being compared, for example AttributeEqual or ReferenceEqual. The Equivalent links that have similarity value lower than a given threshold are transformed into NotEquivalent links.

D3.1.b Model Weaving
This weaving model is interpreted to produce a transformation in ATL. This process is implemented using higher-order transformations, i.e., transformations that take a weaving model as input and that produces a model transformation as output. The transformation is generated in a single step, i.e., this process is also automatic. This output transformation is used to execute the transformation of models conforming to \( MMa \) (\( v1 \)) into the models conforming to \( MMa \) (\( v2 \)).

**5.2.2. Tool design**

The design of AMW relies on three notions: metamodel extensions, Eclipse Plugins and the Eclipse Modelling Framework (EMF) platform [1] for manipulating models. AMW is an extensible workbench that defines different extension points to contribute to the main editor (as shown in Fig. 10). It is responsible for controlling the interaction between the different plugged components. The main idea of the implementation is to have a simple user interface of the weaving plugin that may be partially generated, without having to build a specific tool for each weaving metamodel.

The plugin handles a core weaving metamodel that can be extended incrementally. It is initially composed of three components: a left metamodel, the weaving model and a right metamodel. The standard functionalities are themselves added as an extension to the base plugin.

The screen-shot of Fig. 10 is formed by 3 panels. The left and right panels are tree interfaces extending the standard interface provided by EMF. The weaving model conforms to a weaving metamodel for metamodel comparison. The left and right panels can be exchanged easily, for instance by tree-like panels or by graphical interfaces with boxes to represent elements and lines to represent links. The only constraint is that the components may implement a previously defined interface to return all model elements such that they can be accessed by the weaving component.

The plug-in adapts its interface to handle different weaving metamodels. To this end, we have implemented a standard weaving component using EMF’s reflective API to access model elements and generate user interfaces. It provides standard functionalities for creating and modifying weaving elements, such as a property editor, creation of compositions and references. This provides flexibility to use the prototype for different use cases, since all new metamodel extensions that do not need extra-functionalities can be added without a line of code, just by specifying a metamodel extension.

**5.2.2.1. Installation**

We describe the main requirements and installation instructions to execute the AMW prototype. This information is also available from the AMW download page (http://www.eclipse.org/gmt/amw/download/).

The AMW prototype has five plug-ins:

- Model Weaver Core – `org.eclipse.weaver`: this plug-in is the central plug-in with the base implementation and extension-point definitions.
- Model Weaver Extensions – `org.eclipse.weaver.extension`: contributes to the Weaver Core plugin with the standard left and right panels.
- Matching transformations – `org.eclipse.gmt.weaver.transformation`: provides the transformations that automatically generate weaving models.
The first two plugins must be installed for the standard use. The installation steps are described below:

- Download Eclipse (http://www.eclipse.org). Unzip eclipse .zip file; this will create an eclipse folder with every Eclipse files in it.
- Download EMF (http://www.eclipse.org/emf/). Unzip emf .zip file in the same folder (it also contains an eclipse folder, with the EMF additional files to the Eclipse SDK).
- Download and install the ATL plugins (http://www.eclipse.org/m2m/atl)
- Download and unzip mwplugins.zip into your eclipse/plugin folder, then restart Eclipse.

5.2.2.2. Creating a new weaving model

To create a new weaving model, it is necessary to go into the “File”, “New” menus, and to choose the “Weaving Model” option (see Fig. 12 below). The first page of the wizard is opened.

In this first page it is necessary to choose the metamodel extensions that are used to create the weaving model. In the case of metamodel comparison, we use the base extension, a matching extension, and a comparison extension.

In the second page (see Fig. 13 below), it is necessary to choose the name of the weaving model, and the class that implements the middle panel. There is a standard implementation provided in the plug-in.
In the last page of the wizard, we configure the models that are going to be woven. This interface adapts to the metamodel extension. In the case of metamodel comparison, we must have one left one right metamodel. The metamodels are set up by selecting a metamodel and clicking on “Edit selected metamodel”. Then, the weaving model is correctly configured and can be edited in the main weaving panel (as shown in Section 3.2.1).

After the wizard is called, the weaving model editor is opened with the corresponding model (as illustrated in Figure 10 from the comparison scenario). The tool is ready to create the desired links between the elements of the left and right metamodels.
6. Conclusions and Summary

In this document, we have presented two initial prototypes of model-weaving tools. We have discussed that we are following two parallel technological and application areas in order to offer to MODELPLEX a broad range of choices.

One important question to be studied in the following weeks and months is how the different technologies determine the set of possible applications. To this end, it is planned to apply the tools to common examples and study the benefits and drawbacks of each solution.

Another important direction is application of the technologies to problems posed by the case study providers. WP1 is currently finalizing their initial case study scenarios, which will form the basis for further development in this activity.

Model weaving has already been used in scenarios similar to the ones presented in WP1. For instance, the AMW prototype was used to integrate different views of Systems of Systems [6]. This scenario can be considered as a subset of Systems of Systems Engineering scenario presented by THALES.

In the application scenario presented by SAP (Large Scale Enterprise Business Systems), traceability and composition are part of the modelling challenges and solutions. These are two different use cases for model weaving. Composition and traceability are also needed in the application scenario from Telefónica.

This shows that model weaving is present in several application scenarios. The diversity of application justifies the implementation of different tools (or tools extensions), adapted to specific application scenarios. Finding solutions for requirements of the case study providers will show up needs to improve and extend our approaches and technologies. It will help to enhance and mature the here presented prototypes.

7. References


