



Project acronym: MODELPLEX

Project full title: MODELLing solution for comPLEX software systems

Contract n° 034081

## **Workpackage 3: Model Engineering**

### **Deliverable D3.1.b: Model Weaving**



**Information Society**  
Technologies

Project co-funded by the European Community under the "Information Society Technology" Programme

**Contract Number:** 034081  
**Project Acronym:** MODELPLEX  
**Title:** MODELLing solution for comPLEX software systems

**Deliverable N°:** D3.1.b  
**Due Date:** 10/2008  
**Delivery Date:** 10/2008

**Short Description:**  
Prototype on Model Weaving

**Lead Partner:** Technische Universität Dresden (TUD)  
**Made available to:** Public

Rev	Date	Author	Checked by	Internal Approval	Description
0.1	02/10/08	Jendrik Johannes, TUD			Initial version
0.2	08/10/08	Jendrik Johannes, TUD	Sebastian Cech, TUD		Completed document for internal review
0.9	28/10/08	Jendrik Johannes, TUD		<b>1<sup>st</sup> Internal Reviewer:</b> Miguel Angel Fernandez, TUD <b>2<sup>nd</sup> Internal Reviewer:</b> Tom Ritter, FHG	Updates corresponding to reviews received
1.0	28/10/08	Final Deliverable			

## Table of Content

1. Executive Summary .....	5
2. Introduction.....	5
3. Glossary of Abbreviations and Terms.....	5
4. Installation and Requirements .....	6
5. New Features .....	6
5.1. Example: Decomposing Level 0 CIM Models .....	7
5.2. Using Reuseware: Composition Programs as Level 1 CIM Models .....	7
5.3. New Features: Composition Program Customization.....	8
6. Conclusion and Next Steps.....	10
7. References .....	10
Appendix A: Initial CIM Level 1 Composition System.....	11
Appendix B: Initial CIM Level 1 Customization .....	12

## List of Figures

Figure 1 – Installing Reuseware and dependencies through the update manager (Europa) .....	6
Figure 2 – A CIM Level 0 model decomposed into fragments.....	7
Figure 3 – A composition program composing CIM Level 0 model fragments .....	8
Figure 4 – Selecting the <i>CIM Level1 Model</i> template when creating a composition program .....	9
Figure 5 – A CIM Level 1 model: A composition program with customization .....	10

## 1. Executive Summary

This report accompanies the model weaving prototype (Reuseware Composition Framework) as month 26 (M26) incarnation of MODELPLEX Deliverable D3.1.b “Model Weaving”. It explains how to acquire and run the prototype and demonstrates its usage on examples. The prototype is an extension of the version delivered in month 18. The new features are explained and demonstrated on examples based on the TID case study.

## 2. Introduction

This document describes the model weaving prototype that is the main artefact of this deliverable. The prototype is part of the Reuseware Composition Framework<sup>1</sup>, developed at TUD. In this document we show an application of the prototype in the TID case study.

With the Reuseware Composition Framework, *composition systems*<sup>2</sup> can be modelled for arbitrary languages. These systems can then be used to weave and compose *model fragments*<sup>3</sup>. This document demonstrates the new features of the prototype. It neither discusses nor introduces details of the concepts behind the composition framework. Consult the M14 report about those details, as well as [1] in which the results of this work have been published. In addition, this document describes plans for further development of the prototype.

The remainder of this document is structured as follows. Section 3 contains a glossary of abbreviations. How to install Reuseware (the prototype) is explained in Section 4. Section 5 demonstrates the usage of Reuseware for the TID case study and describes new features that were introduced into the prototypes based on the case study’s requirements. Section 6 concludes and outlines the next steps in the prototype’s development.

## 3. Glossary of Abbreviations and Terms

- CIM – Common Information Model
- CIM Level 0 – Model conforming to the original CIM metamodel
- CIM Level 1 – Model abstracting from CIM Level 0 models
- TID – Telefónica Investigación y Desarrollo
- TUD – Technische Universität Dresden
- WP – Work Package

---

<sup>1</sup> <http://reuseware.org>

<sup>2</sup> A composition system (as defined in M18 prototype document) defines which models are reusable units and at what places these models can be extended during composition.

<sup>3</sup> Model fragments (as defined in M14 report) are reusable models that may contain variation points.

## 4. Installation and Requirements

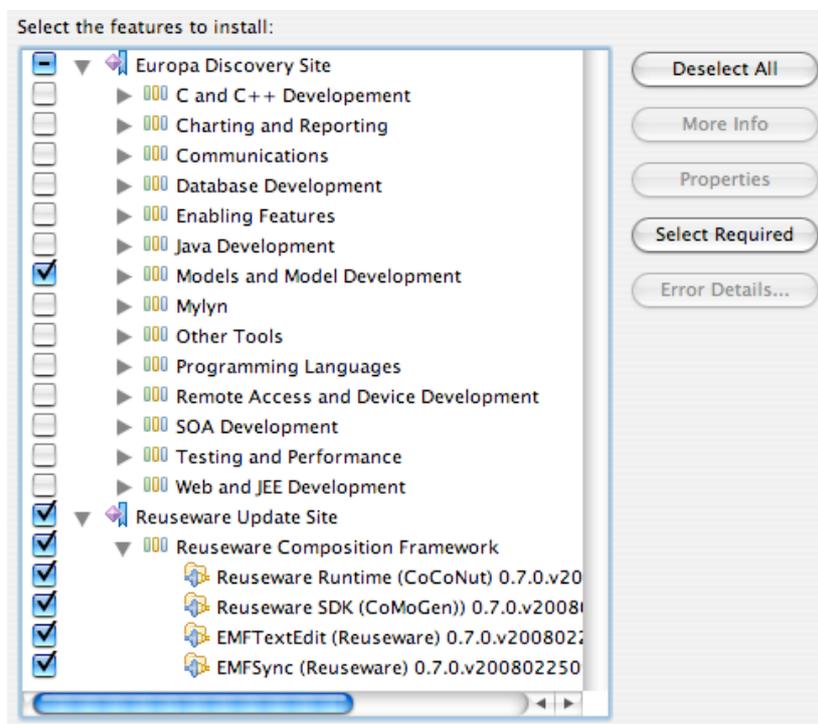
The Reuseware Composition Framework consists of a set of Eclipse plug-ins that require an Eclipse Platform installation<sup>4</sup> of version 3.3 or higher. The framework can be installed within the Eclipse platform using the update manager (*Help > Software Updates > Find and Install...*).

To install the framework, create a new remote site with the following properties:

Name: **Reuseware Update Site**

URL: **http://reuseware.org/update**

**Installation for Eclipse 3.3 (Europa)** Select the *Reuseware Update Site* and the *Europa Discovery Site* and search for updates. Select all components from the *Reuseware Update Site* and press *Select Required* to automatically resolve dependencies you might not have installed yet (cf. Figure 1). Reuseware depends on several components from the “Models and Model Development” branch of the Europa Discovery Site.



**Figure 1 – Installing Reuseware and dependencies through the update manager (Europa)**

**Installation for Eclipse 3.4 (Ganymede)** Select all components from the *Reuseware Update Site* and press *Install*. If additional dependencies need to be installed, Eclipse will do that automatically.

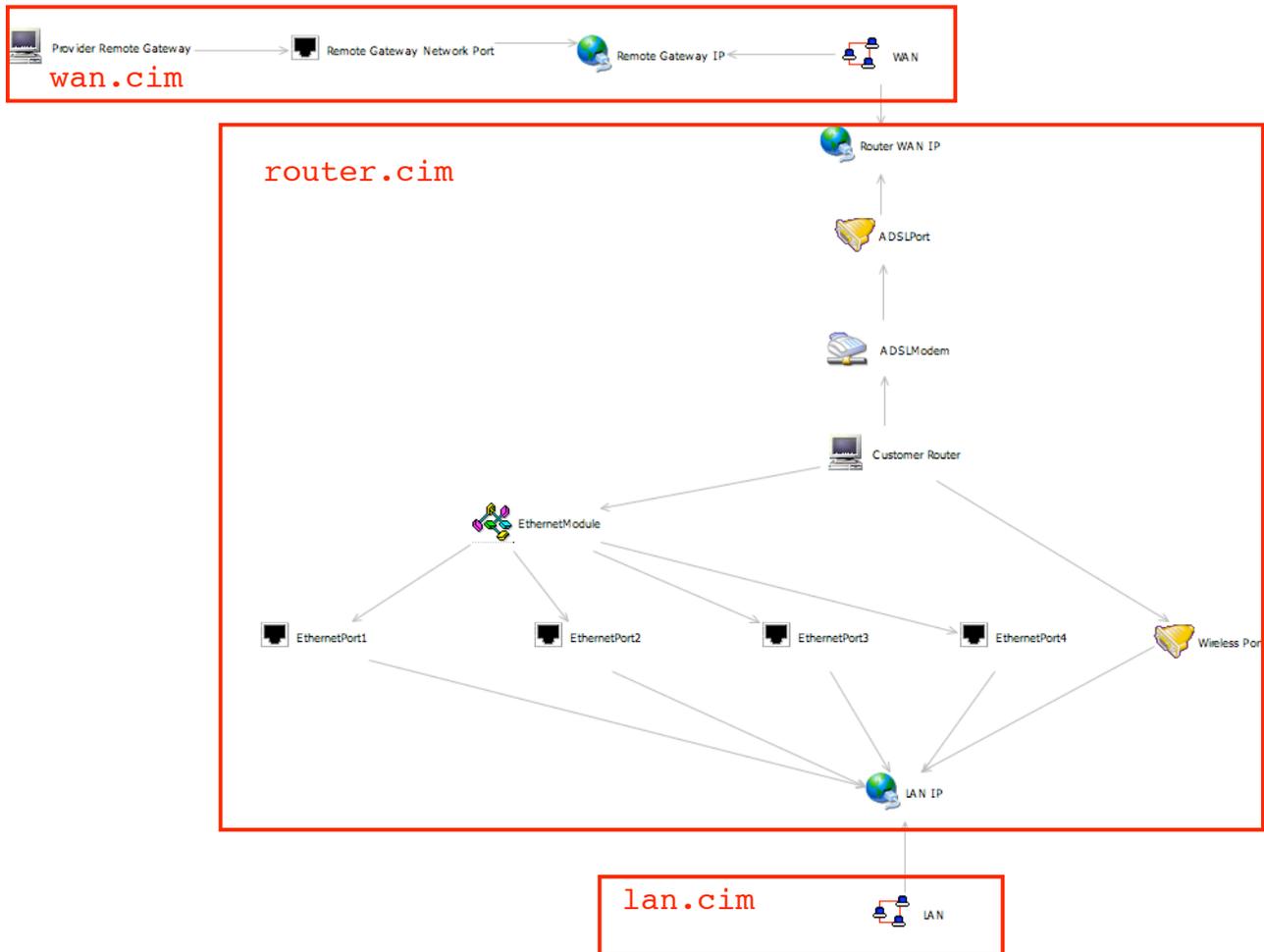
## 5. New Features

This section introduces the new features of Reuseware that support customizing the prototype for specific use cases. As an example, we use models from the TID case study. The example was developed in collaboration with TID and drove the development of the new features. This section first introduces the example (cf. Section 5.1) it then shows how Reuseware is used for the example and which no features are desired (cf. Section 5.2) it concludes by demonstrating the new features for customization (cf. Section 5.3).

<sup>4</sup> <http://www.eclipse.org/downloads>

## 5.1. Example: Decomposing Level 0 CIM Models

The CIM Level 0 models are modelled using the CIM DSL editor developed within MODELPLEX by Xactium for TID.<sup>5</sup> Such models describe network configurations in all details. The TID case study proposes (Requirement 224 in [2]) to abstract the modelling to higher levels (Level 1, Level 2, ...) with more abstract concepts. On Level 0, for instance, an *EthernetPort*, is a single model element, while on Level 1, a *Router* (containing several Ethernet ports) is a single model element.



**Figure 2 – A CIM Level 0 model decomposed into fragments**

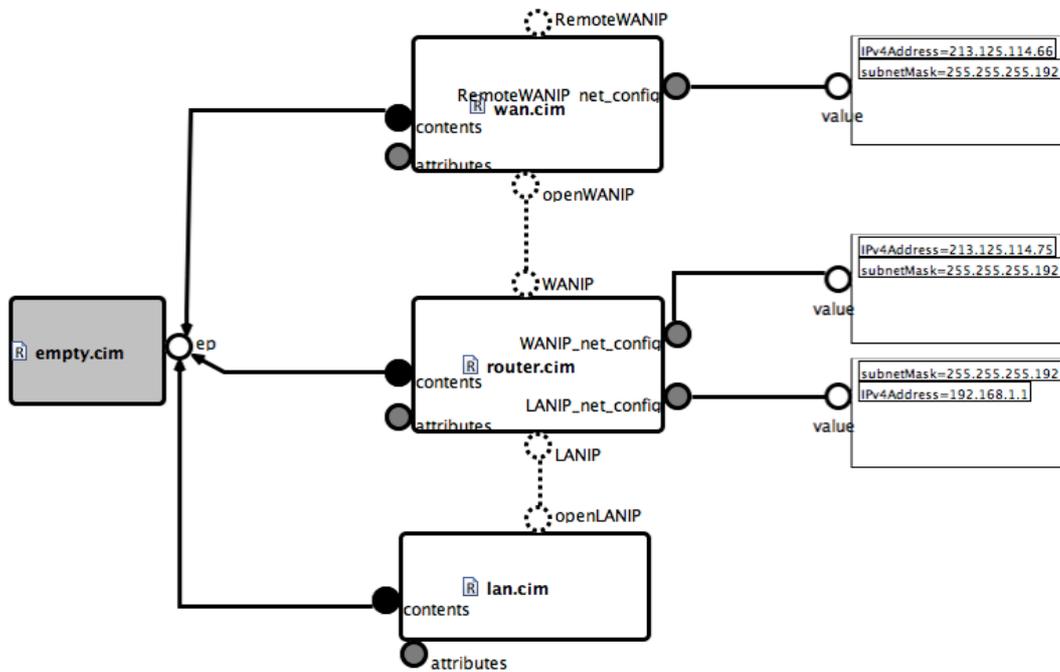
Experimenting with Level 0 models and Reuseware, we discovered that concepts of higher abstraction levels can be expressed by Level 0 models. That is, a collection of Level 0 model elements can represent one Level 1 element. For example, a *Router* (on Level 1) can be described by a Level 0 model fragment containing several *EthernetPorts* and other elements.

Consider the example in Figure 2, modelled in the Level 0 DSL. The three boxes indicate that the model can be decomposed into three fragments: *wan.cim*, *router.cim* and *lan.cim*. These model fragments represent Level 1 concepts.

## 5.2. Using Reuseware: Composition Programs as Level 1 CIM Models

Using the mentioned fragments and an appropriate composition system (cf. Appendix A), we can define a composition program that constructs the original Level 0 model (cf. Figure 2).

<sup>5</sup> The editor used in this example is preliminary; a new version being developed.



**Figure 3 – A composition program composing CIM Level 0 model fragments**

The composition program is shown in Figure 3. It combines all three fragments (*wan.cim*, *router.cim*, *lan.cim*) into an empty core Level 0 model (*empty.cim*) using contribution composition links<sup>6</sup>. Additionally, configuration composition links<sup>7</sup> define relations between fragments. The three boxes on the right represent *value definitions*. A value definition is a new feature in Reuseware composition programs that can be used to extend models with String values (and not model elements). This is appropriate to set attributes of model elements. Here, some attributes of certain elements inside the fragments are set from the outside.

The composition program can be seen as a Level 1 model: The fragments (except the *empty.cim*) represent single Level 1 elements. The composition links represent relations between them. The value definitions represent attribute settings of the elements. In comparison to a model expressed in a conventional modelling language, there is no specific syntax (i.e., different visualizations – shapes, icons, etc. – for different elements) and the empty core and the links to it are shown as well as some unused ports.

Therefore, the following new features are required:

1. Define different syntax for different fragments in a composition program.
2. Hide uninteresting fragments, ports and composition links.
3. Automate the definition of hidden fragments and composition links.

We extended Reuseware with a customization mechanism that supports these three requirements. The mechanism is explained in the next section by applying it to the above example.

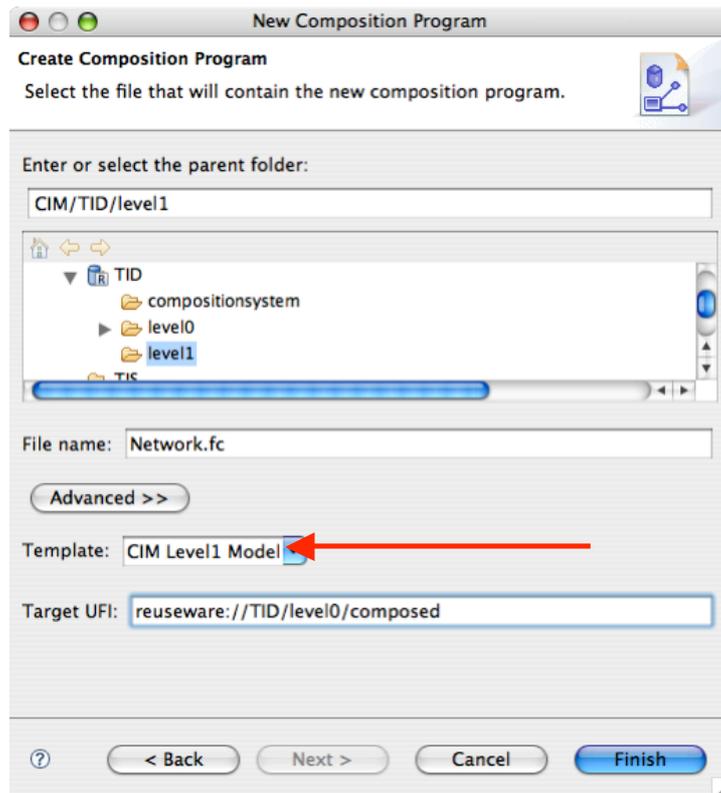
### 5.3. New Features: Composition Program Customization

To implement the above requirements, we defined extension points in Reuseware (using the Eclipse plug-in mechanism). The following extensions are supported:

<sup>6</sup> contribution composition link (as defined in M14 report): copies elements

<sup>7</sup> configuration composition link (as defined in M14 report): changes references between elements

1. *Composition Program Designs* enable specification of custom figures for fragments, ports and composition links. For each figure, a condition over the fragment's interface is given (e.g., does port X exist) to define for which fragments the figure should be used (requirement 1). It is also possible to define that an element is hidden (requirement 2).
2. *Composition Program Templates* define fragments that will be automatically created in a composition program. A template can be selected when a composition program is created with the Creation Wizard (cf. Figure 4). When additional fragments are added to such composition programs, Reuseware automatically connect ports of the newly added with matching ports of the hidden fragments (Requirement 3).



**Figure 4 – Selecting the *CIM Level1 Model* template when creating a composition program**

We defined extensions for an initial CIM Level 1 customization of Reuseware. The customization definition can be found in Appendix B. Basically, we defined different figures and icons for the different fragments through a Composition Program Design extension. Additionally, we defined a template that adds, but hides, the empty core fragment to each composition program based on the template (cf. Figure 4).

Using the customization, the composition program from Figure 2 looks as shown in Figure 5. There are several advantages in defining CIM Level 1 models with composition programs over defining them with a separate CIM Level 1 DSL:

- The prototype, in particular the composition program editor and fragment repository, can be directly reused to define and manage the model fragments, and no new tools need to be developed from scratch.
- The Level 1 concepts (and concepts of other abstraction levels) can intuitively be defined using concepts of Level 0. This makes it easier and more flexible for domain experts (in this case TID) to define new abstractions themselves.
- The semantics of the Level 1 models (and models of other abstraction levels) are given through the composition. Reuseware can compose the Level 1 models to one monolithic

Level 0 models without additional effort. Through this, all tooling (e.g., model transformations and code generators) developed for CIM Level 0 in MODELPLEX can be applied for all CIM models, regardless of their abstraction level.

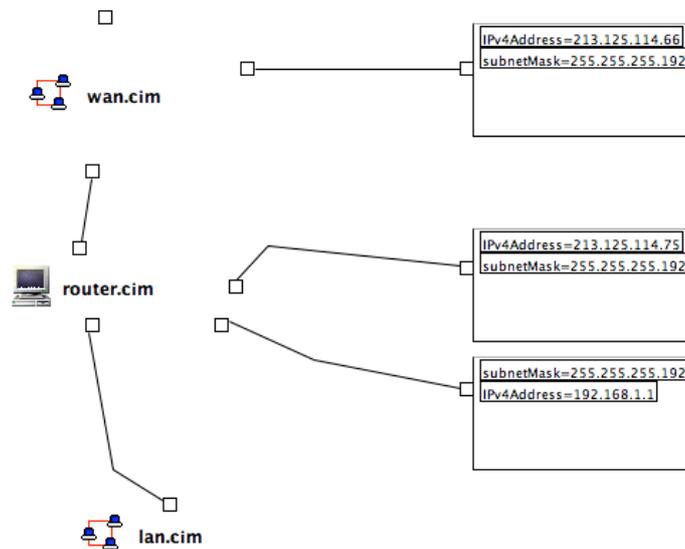


Figure 5 – A CIM Level 1 model: A composition program with customization

## 6. Conclusion and Next Steps

This document presented the new features of the prototype for customization based on the TID case study. The experiences gained by experimenting with the example and the customization mechanism are promising. It did not only show the applicability of Reuseware for the TID case, but also showed some interesting ways of extending Reuseware to make it more flexible and customizable and, with that, a more user-friendly modelling tool. We will therefore continue this work in close collaboration with TID. The next steps are twofold:

1. *Prototype development and concepts:* The new customization mechanisms of Reuseware are applicable for the TID case as shown in this document. To learn more about the general usefulness, we also experimented with examples from the TIS case study. The initial results are promising but also revealed some gaps in the customization mechanism. To aid future enhancement of Reuseware into this direction, we are currently investigating how the concepts behind the customization relate to the model weaving concepts, which are implemented in the Reuseware core (and are described in the M18 report). With the concepts clarified, we will adjust implementation to allow more involved definition of customizations. The goal here is to support *modelling* of customizations, instead of using Eclipse plug-in extension. The results will be presented in the M28 report.
2. *Application in TID case:* We will continue our work on the TID case study in close collaboration with TID. As a next step, we will define a complete Level 1 composition system and customization for Reuseware. Afterwards, we will work on supporting additional abstraction levels. New results will also be presented in the M28 report.

## 7. References

- [1] Heidenreich F., Henriksson, J., Johannes, J., Zschaler, S.: On Language-Independent Model Modularisation. In: Transactions on Aspect-Oriented Development, Special Issue on Aspects and MDE. (to appear) (2008)
  - [2] MODELPLEX IP Deliverable D1.2.c: Updated Requirements Specification. (March 2008)
- D3.1.b Model Weaving

## Appendix A: Initial CIM Level 1 Composition System

The composition system is defined using Reuseware's *Reuse Extension Language*. The language was already used in the M18 prototype deliverable. A reference can also be found at:

[http://reuseware.org/index.php/Reuse\\_Extension\\_Language](http://reuseware.org/index.php/Reuse_Extension_Language)

```
reuseextension tidExample
for <example>
{
  Example.elements is Hook if $elements->isEmpty()$ {
    port expr = '$ep'$
  }

  Example.elements is Prototype if $not elements->isEmpty()${
    port expr = '$contents'$
  }
}

reuseextension cimLevel1Core
for <Core>
{
  ProtocolEndpoint is Slot if
    $5 <= elementName.size() and elementName.substring(1, 4) = 'open'${
    port expr = $elementName$
  }

  ProtocolEndpoint is Anchor if
    $5 > elementName.size() or elementName.substring(1, 4) <> 'open' $ {
    port expr = $elementName$
  }
}

reuseextension cimLevel1Collections
for <Collections>
{
  IPConnectivitySubnet has content Prototype {
    point expr = '$type'$
    port expr = '$attributes'$
    value expr = '$Network'$
  }
}

reuseextension cimLevel1ProtocolEndpoints
for <ProtocolEndpoints>
{
  IPProtocolEndpoint.Ipv4Address has content Hook if
    $5 > elementName.size() or elementName.substring(1, 4) <> 'open' $ {
    point expr = '$Ipv4Address'$
    port expr = $elementName.concat('_net_config')$
  }

  IPProtocolEndpoint.subnetMask has content Hook if
    $5 > elementName.size() or elementName.substring(1, 4) <> 'open' $ {
    point expr = '$subnetMask'$
    port expr = $elementName.concat('_net_config')$
  }
}
}
```

```
reuseextension cimLevel1System
for <System>
{
  UnitaryComputerSystem has content Prototype {
    point expr = '$type'$
    port expr = '$attributes'$
    value expr = '$Router'$
  }
}
```

## Appendix B: Initial CIM Level 1 Customization

The customization is defined as an Eclipse plugin in a `plugin.xml` file show below. More information about Eclipse plugins and how they are defined can be found, for example, at:

<http://www.eclipse.org/articles/Article-PDE-does-plugins/PDE-intro.html>

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
  <extension
    point="org.reuseware.coconut.fragment.diagram.cpDesigns">
    <cpDesign
      namespace="reuseware://cim">
      <fragment
        figure="org.reuseware.language.cim.level1.figures.IconNodeFigure"
        height="10"
        hide="false"
        icon="icons/Network.gif"
        width="10">
        <equals
          point="type"
          port="attributes"
          value="Network">
        </equals>
      </fragment>
      <fragment
        figure="org.reuseware.language.cim.level1.figures.IconNodeFigure"
        height="10"
        hide="false"
        icon="icons/Router.gif"
        width="10">
        <equals
          point="type"
          port="attributes"
          value="Router">
        </equals>
      </fragment>
      <fragment
        hide="true">
        <exists
          port="ep">
        </exists>
      </fragment>
    </port>
    figure="org.reuseware.language.cim.level1.figures.SmallPortFigure"
    height="4"
```

```

        hide="false"
        label="false"
        width="4">
    </port>
    <port
        hide="true"
        name="contents"
        priority="10">
    </port>
    <port
        hide="true"
        name="attributes"
        priority="10">
    </port>
    <link
        figure="org.reuseware.language.cim.level1.figures.LinkFigure">
    </link>
</cpDesign>
</extension>
<extension
    point="org.reuseware.coconut.fragment.diagram.cpTemplates">
    <cpTemplate
        baseTargetUFI="reuseware://cim/myStore/composed"
        caption="CIM Level1 Model"
        uFI="reuseware://cim/template/empty.cim">
    </cpTemplate>
</extension>
<extension
    point="org.reuseware.coconut.ui.eclipse.fragmentStores">
    <mapping
        baseUFI="reuseware://cim/template"
        folder="store/">
    </mapping>
    <fragment
        uFI="reuseware://cim/template/empty.cim">
    </fragment>
    <reuseExtension
        file="store/compositionSystem/tidExample.rex">
    </reuseExtension>
    <reuseExtension
        file="store/compositionSystem/tidCore.rex">
    </reuseExtension>
    <reuseExtension
        file="store/compositionSystem/tidCollections.rex">
    </reuseExtension>
    <reuseExtension
        file="store/compositionSystem/tidSystem.rex">
    </reuseExtension>
    <reuseExtension
        file="store/compositionSystem/tidProtocolendpoints.rex">
    </reuseExtension>
</extension>
</plugin>

```