

DUALLY Use case	Ivano Malavolta ivanomalavolta@yahoo.it Department of Computer Science University of L'Aquila L'Aquila - Italy
Semi-automatic interoperability between Darwin and Acme within the DUALLY framework	
User Guide	18/05/2008

Introduction

Many architectural languages have been proposed in the last fifteen years, each one with the chief aim of becoming the ideal language for specifying software architectures. What is evident nowadays, instead, is that architectural languages are defined by stakeholder concerns. Capturing all such concerns within a single, narrowly focused notation is impossible. At the same time it is also impractical to define and use a "universal" notation, such as [UML](#). As a result, many domain specific notations for architectural modeling have been proposed, each one focussing on a specific application domain, analysis type, or modeling environment.

These considerations led us to propose **DUALLY** a framework to create interoperability among ADLs themselves as well as UML. As conceptually shown in the figure below, **DUALLY** permits to transform concepts of an architectural model M1 into the semantically equivalent concepts in the architectural model M2. Each M_i conforms to its MM_i meta-model or UML profile. Every MM_i can be a meta-model of an architectural language. Therefore, **DUALLY** works at two abstraction levels: meta-modeling (upper part of the figure), and modeling (lower part of the figure).

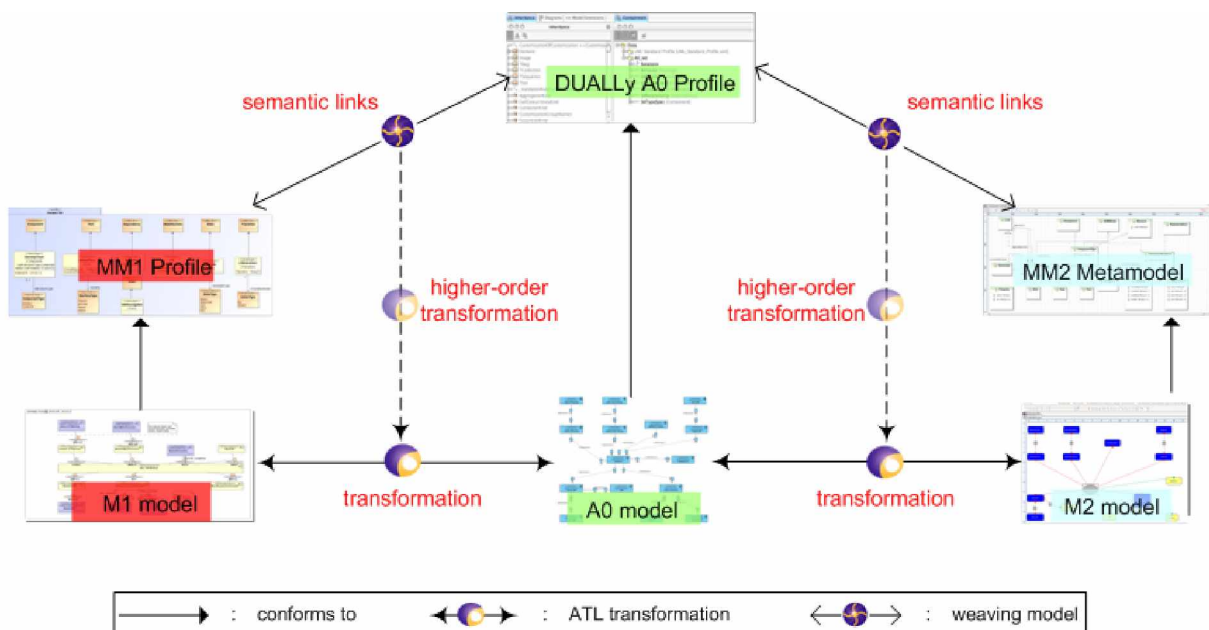


Figura 1 - Conceptual overview of DUALLY

At the meta-modeling level, model driven engineers provide a specification of the architectural language in terms of its meta-model or UML profile. They then define a meta-transformation so as to semantically relate architectural concepts in MM1 with the equivalent elements in MM2.

At the modeling level, software architects specify the SA using their preferred ADL or UML-based notation. **DUALLY** allows the automatic generation of model-to-model transformations which permit the software architect to automatically translate the M1 specification (written according to MM1) into the corresponding M2 model (conforming to MM2) and viceversa.

As it can be noticed in the figure, the meta-transformation (and its corresponding generated transformation) relates MM1 to MM2 (as well as M1 to M2) passing through what we refer to as A0. A0 represents a semantic core set of modeling elements, providing the infrastructure upon which to construct semantic relations among different ADLs. It acts as a bridge among the different architectural languages to be related together.

DUALLY is automated and implemented as an [Eclipse](#) plugin. MM_i meta-models are expressed in [Ecore](#). MM_i profiles can be realized using any UML tool which exports in [UML2](#), the EMF-based implementation of the UML 2.0 meta-model for the Eclipse platform. A0 is represented as a UML profile. Transformations among meta-models/profiles and model-to-model transformations are implemented in the context of the [AMMA](#) platform. Model-to-model transformations are then automatically instantiated, thus providing the possibility to automatically reflect modifications made on a model designed with one extension to one or even all of the other extensions.

The main advantages **DUALLY** exposes can be summarized as follows:

- **DUALLY** works at two abstraction levels, providing a clear separation between model driven experts (the technical stakeholder) and software architects (the final users). The model transformation engine is completely hidden to a software architect, making **DUALLY** extremely easy to use.
- **DUALLY** permits the transformation among formal ADLs and UML model-based notations and viceversa.
- Software architects can continue using familiar architectural notations and tools, and reusing existing architectural models.
- **DUALLY** permits both languages and tools interoperability.
- The meta-transformations among two architectural notations are defined once, and reused for each model that will be made.

Further details on **DUALLY** may be found in the technical report referenced in the last part of this guide.

Installation

This use case requires Eclipse 3.3.1, EMF 2.3.1, ATL 2.0.0, AMW 2.0.0, AM3 1.0.0, UML2 2.1.0, **DUALLY** 1.0

An Eclipse bundle for Windows platforms containing **DUALLY** as well as all the required plugins is available here: <http://dually.di.univaq.it/downloads.html>.

Download and unzip the use case example, it contains an ATL project.

The ATL project structure should look like this:

DUALLY_DarwinAcme_useCase/

- metamodels
 - ◆ ACME.ecore **Acme metamodel**
- models
 - ◆ In
 - § DarwinLTSA.profile.uml
 - § IECS_Darwin.uml **IECS DarwinLTSA-profiled model**
 - § UML_Standard_Profile.MagicDraw.DSL_Customization.profile.uml
 - § UML_Standard_Profile.MagicDraw_Profile.profile.uml
 - § UML_Standard_Profile.UML_Standard_Profile.profile.uml
 - § UML_Standard_Profile.Validation_Profile.profile.uml
 - ◆ out
 - § IECS_A0.uml **IECS A0-profiled model**
 - § IECS_ACME.ecore **IECS ecore model**
 - § IECS.acme **IECS textual specification in ACME**
- profiles
 - ◆ A0
 - § A0.profile.uml **A0 UML profile**
 - ◆ DarwinLTSA
 - § DarwinLTSA.profile.uml **Darwin/LTSA UML profile**
- transformations
 - ◆ ACME_A0_DUALLY-Right2Left.asm
 - ◆ ACME_A0_DUALLY-Right2Left.atl **A0 to ACME transformation**
 - ◆ ACME_A0_DUALLY-Right2Left.ecore
 - ◆ ACMEspecificationExtractor.asm
 - ◆ ACMEspecificationExtractor.atl **ACME to textual specification trans.**
 - ◆ ACMEspecificationExtractor.launch
 - ◆ Darwin_A0_DUALLY-Left2Right.asm
 - ◆ Darwin_A0_DUALLY-Left2Right.atl **Darwin to A0 transformation**
 - ◆ Darwin_A0_DUALLY-Left2Right.ecore
 - ◆ UML2Copy.asm
 - ◆ UML2Copy.atl **UML2 copying transformation**
 - ◆ useCase_A0_ACME.launch
 - ◆ useCase_Darwin_A0.launch
- weavingModels
 - ◆ ACME_A0.amw **Weaving model between ACME and A0**
 - ◆ ACME_A0.amw.prop
 - ◆ Darwin_A0.amw **Weaving model between Darwin and A0**
 - ◆ Darwin_A0.amw.prop

Execution

Among the notorious ADLs we selected the DARWIN/LTSA (see Figure 2 for the DARWIN/LTSA UML profile) modeling and analysis environment and the ACME ADL (see Figure 3 for a simplified version of the ACME metamodel). This use case shows how to create semantic links between metamodels and UML profiles and how this links can guide the automatic

generation of transformations at the modeling level. Moreover, Figure 4 shows the A0 profile, containing a core set of architectural elements; it is embedded into the DUALLY tool and serves as a bridge between Darwin and Acme. The how and why of A0 is detailed in the Technical Report referenced at the end of this document. The main benefit of using A0 is the implied star architecture in which A0 is the center of the star.

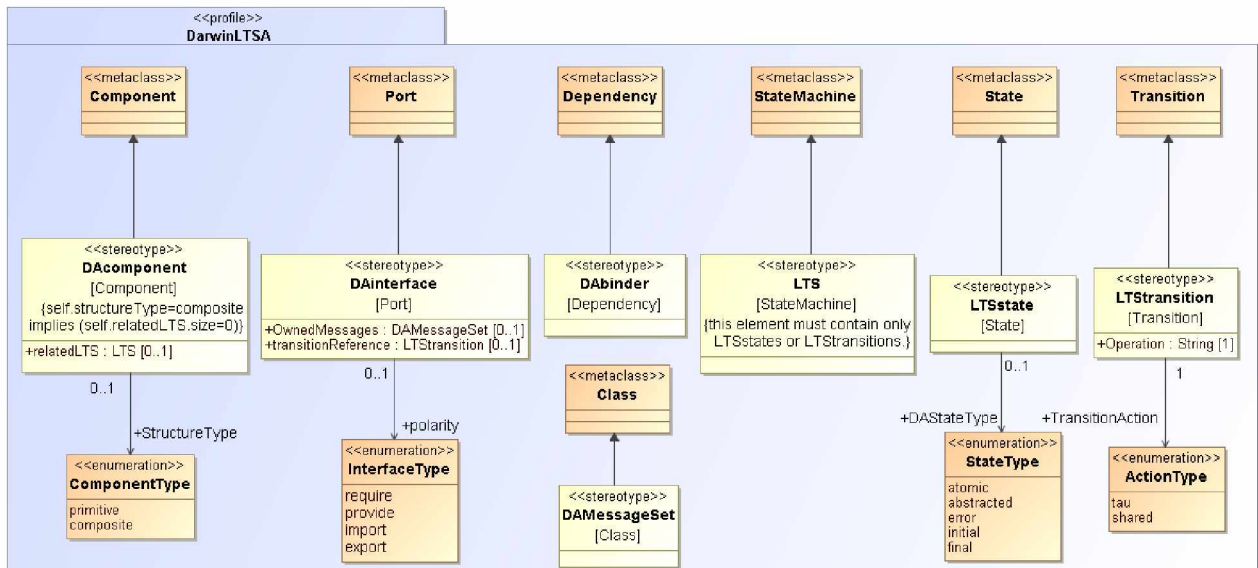


Figure 2 - Darwin/LTSA UML profile

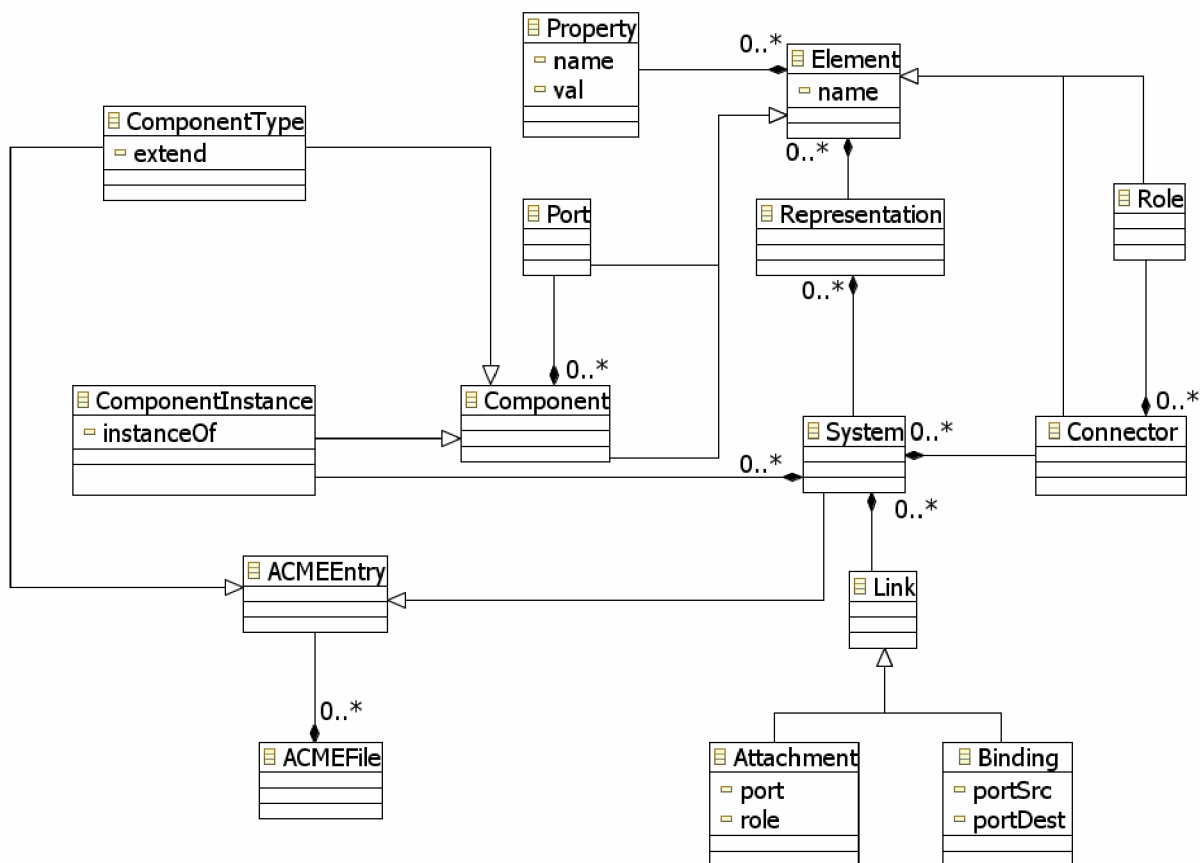


Figure 3 - ACME simplified metamodel

We will describe the process of “DUALLYzation” of both Darwin/LTSA and ACME notations now through the following steps:

- 1) Develop the UML profile containing Darwin/LTSA elements. For example we developed it using the MagicDraw UML modeling tool (see Figure 2) and import it into the Eclipse platform;
- 2) Develop the simplified metamodel of Acme. We did it graphically, through the Ecore graphical editor (see Figure 3). It is already into the Eclipse platform;
- 3) Create the weaving model between Darwin/LTSA profile and A0. While creating such weaving model, you have to specify the DUALLYWeavingMetamodel as weaving metamodel, the DUALLYWeavingPanelExtension as weaving model panel and DUALLYWovenPanelExtension as the left and right metamodel panels. The user interface should look like in Figure 6.

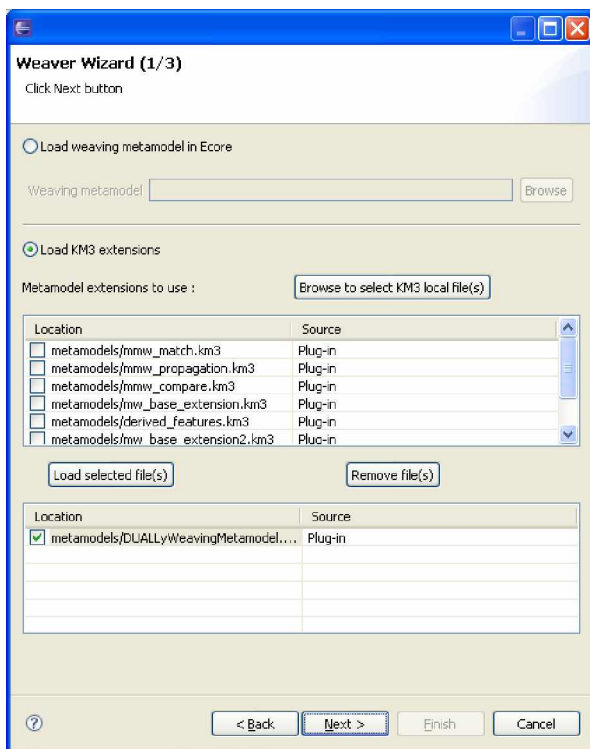


Figure 4 - Choose of the DUALLY weaving metamodel extension

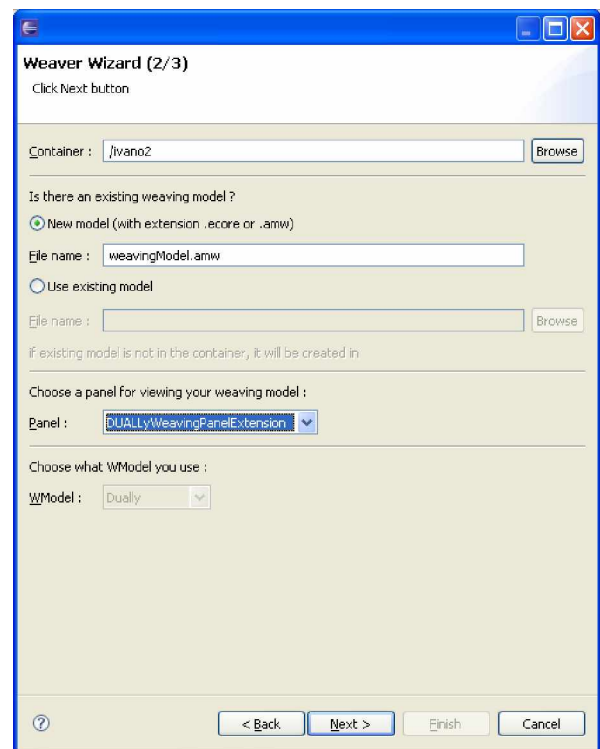


Figure 5 - Choose of the DUALLY weaving panel

- 4) Create the weaving model between ACME metamodel and A0. This step is similar to the previous one. Please refer to Figures 7 and 8 to have a conceptual idea of the semantic links that we created between A0 and Darwin/LTSA and Acme, respectively.
- 5) Press the button on the toolbar over the weaving model panel with the desired direction to automatically generate the transformations at the modeling level. Now you have a new ecore file in the same directory of the weaving model; you have just to extract the ATL specification from that file using the AM3 specific menu.

At this point the first phase is concluded; indeed we have finished all the work at the metamodeling level; now we are going to create a model conforming to the Darwin/LTSA UML profile and we will execute the two generated transformations to obtain a final Acme specification. Let us specify that the first phase is executed only once for each metamodel/profile to “DUALLYze”, while the software architects may use the generated transformations N times, depending on their needs.

The model of this use case consists of the software architecture of a multi-tier environment capable of maintaining a failsafe, client-server like communication within a safe and secure environment such as a military vessel: the Integrated Environment for Communication on Ship (IECS) (see Figure 9 for the static description of IECS-MS architecture using Darwin/LTSA profile). The case study includes also a behavioral description of the IECS-MS architecture, but for the sake of simplicity we leave the details out of this document.

The case study's specification comes from a project developed within Selex Communications, a company mainly operating in the naval communication domain.

Now we execute first the Left2Right transformation generated from the weaving model between Darwin/LTSA and A0 to obtain a model conforming to the A0 profile (see Figure 10, you can notice that we imported the generated model into another modeling tool, Visual Paradigm, proving that we achieve also the tool interoperability) and then we execute the Right2Left transformation generated from the weaving model between Acme and A0 producing an Ecore model conforming to the Acme metamodel.

Since the AcmeStudio tool allows the import of only textual Acme specification we have to execute another transformation that takes as input the Ecore Acme model and produces the same model in the textual specification; Figure 11 shows the result of this transformation imported into AcmeStudio.

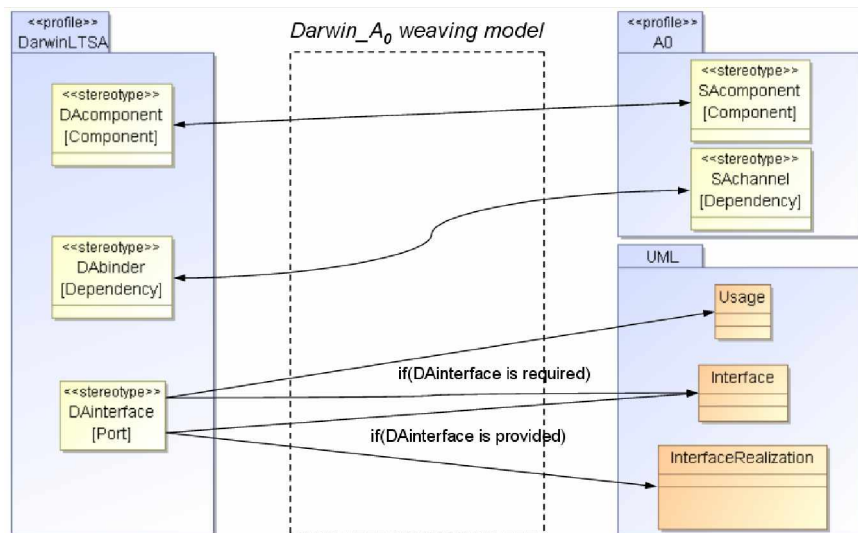


Figura 6 - Weaving model between Darwin/LTSA and A0.

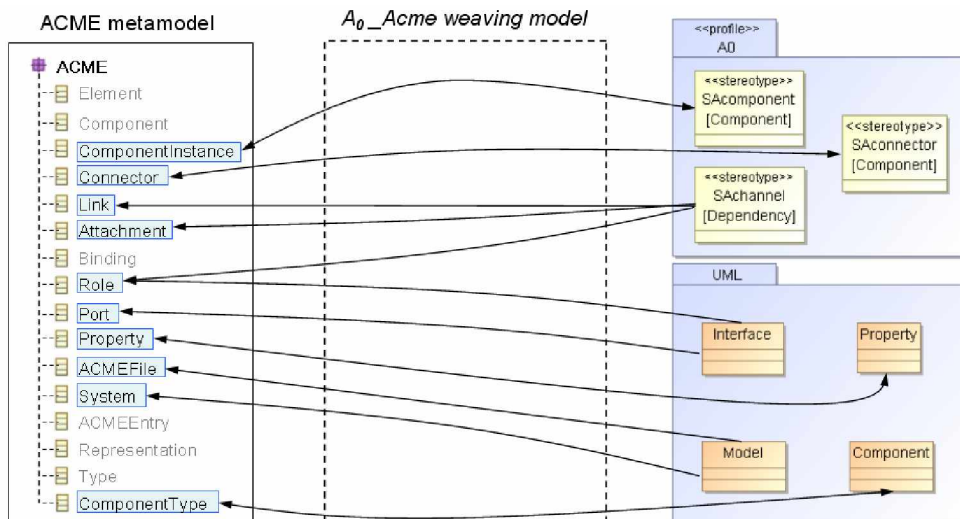


Figura 7 - Weaving model between Acme and A0.

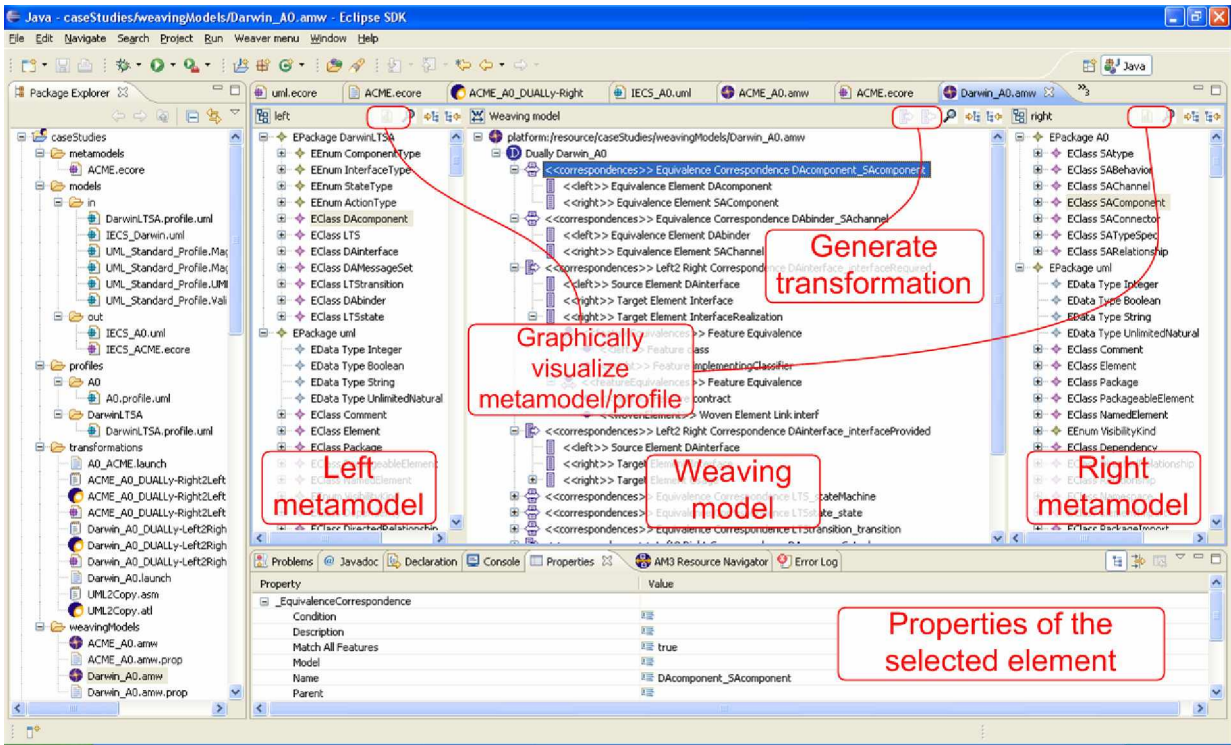


Figure 8 - User interface of DUALLY weaving models editor

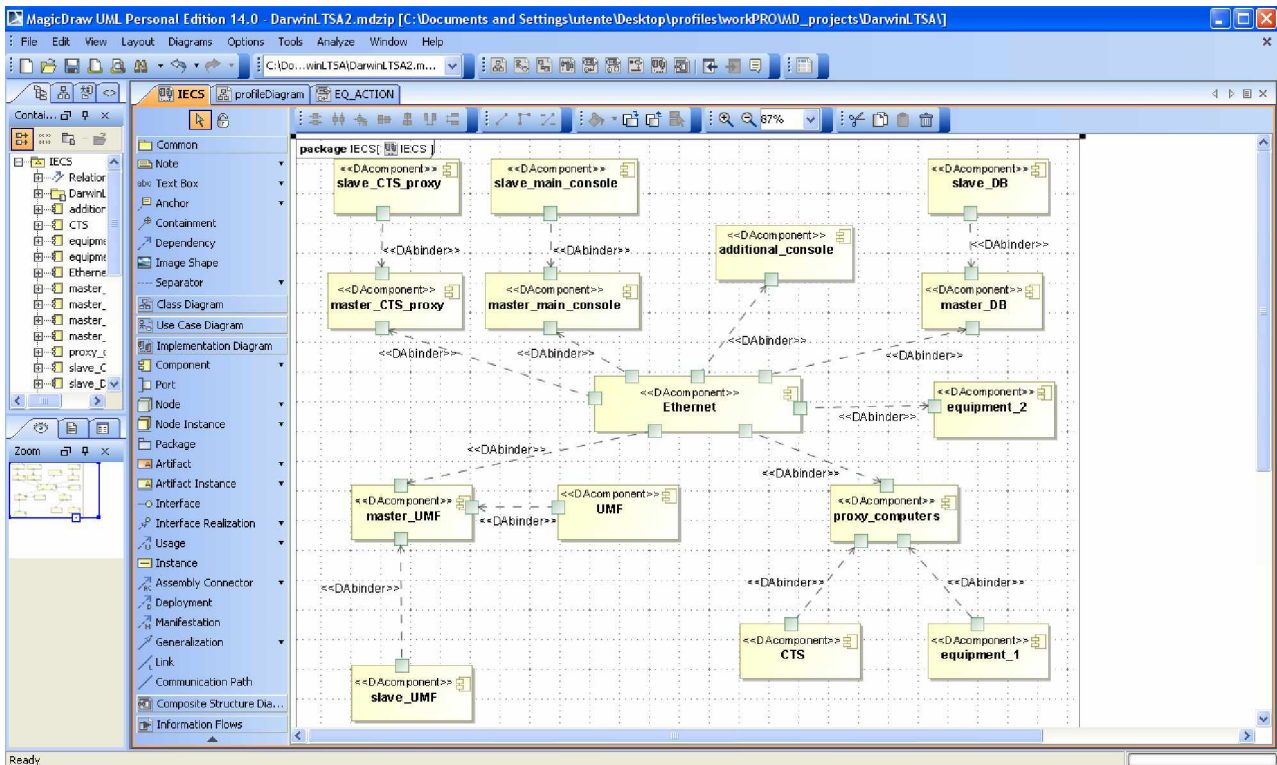


Figure 9 - Static description of IECS-MS architecture conforming to Darwin/LTSA profile.

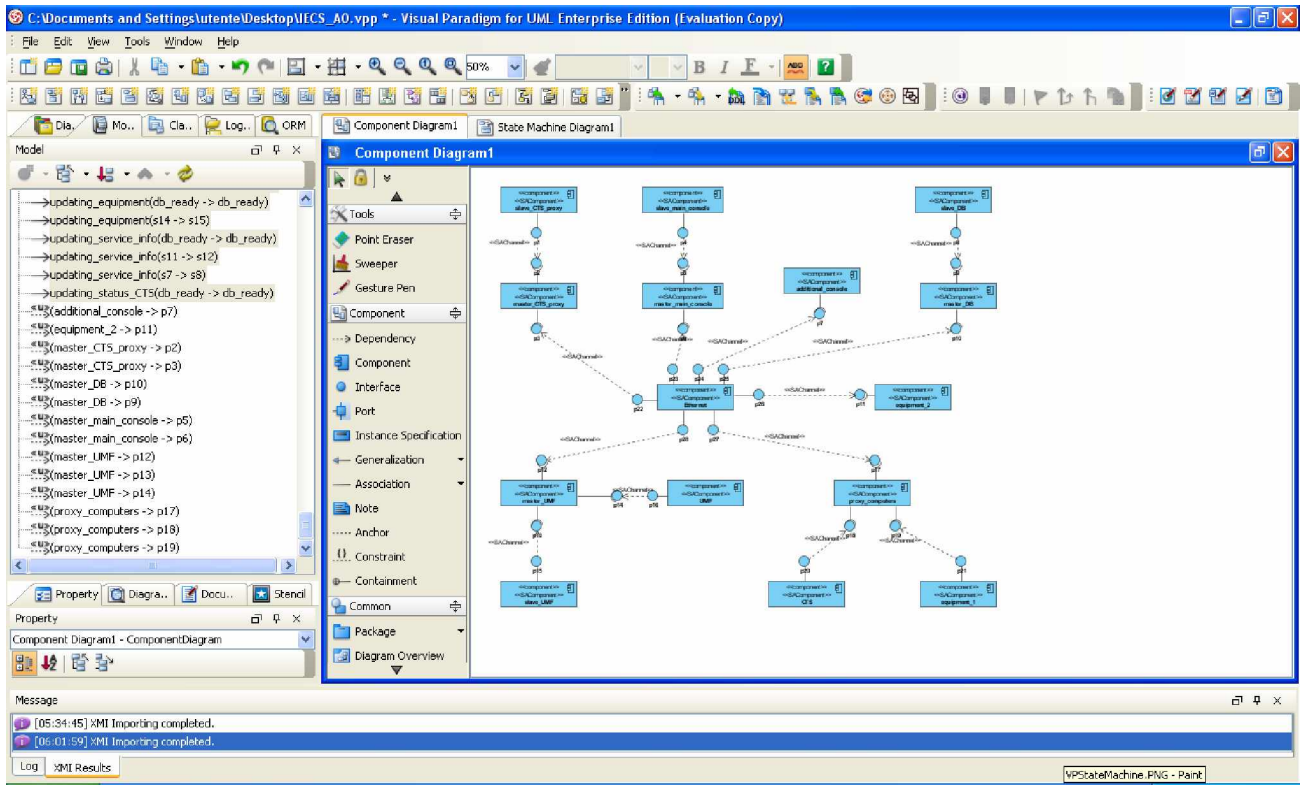


Figure 10- Static description of IECS-MS architecture conforming to A0 profile.

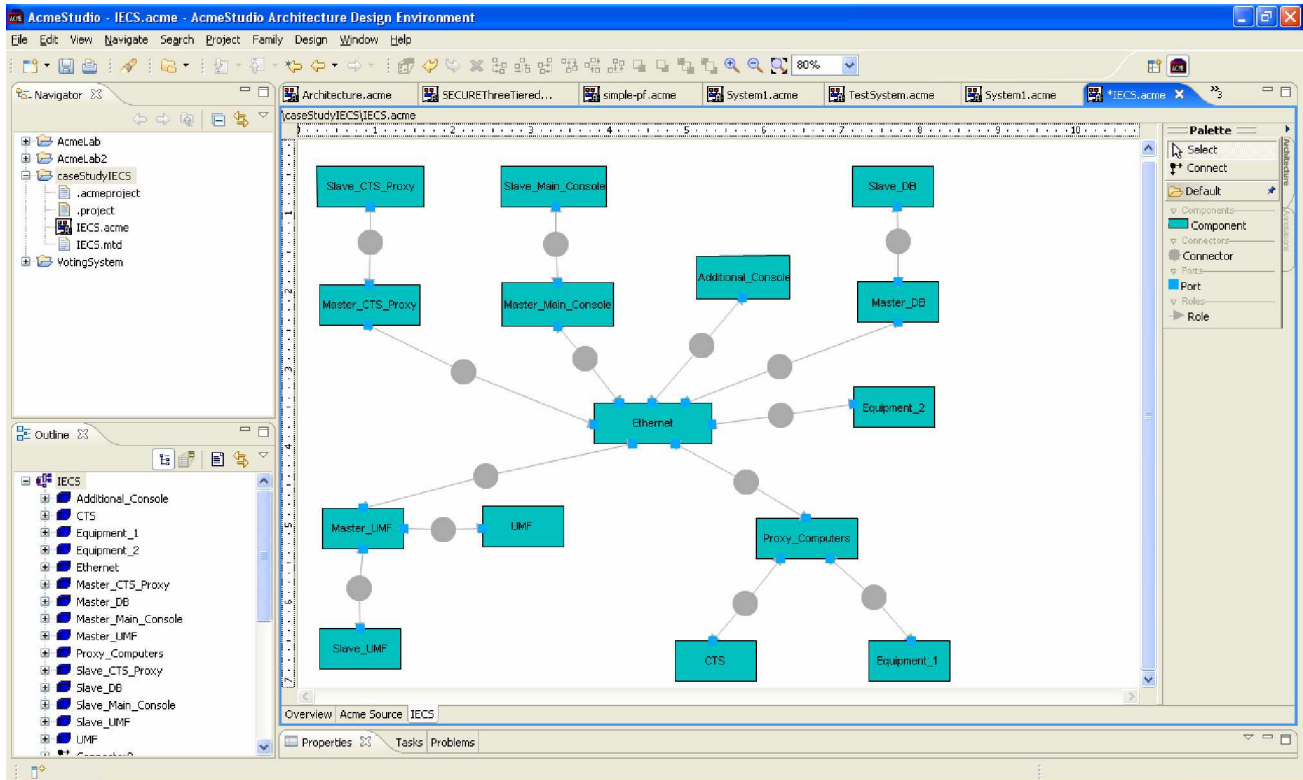


Figure 11 - Static description of IECS-MS architecture using AcmeStudio.

References

- <http://dually.di.univaq.it>, home page of the DUALLY project. The source code can be found in <http://sourceforge.net/projects/dually>, released under the GNU General Public License (GPL).
- Malavolta, H. Muccini, P. Pelliccione, and D. A. Tamburri. **Providing Architectural Languages and Tools Interoperability through Model Transformation Technologies.** Technical report, TR 004-2008, University of L'Aquila, Computer Science Department. Available at the DUALLY site, 2008.