Architectural notations interoperability using the DUALLY framework

Ivano Malavolta
Dipartimento di Informatica, Università dell'Aquila,
Via Vetoio, 67100 L'Aquila,
[ivano.malavolta@di.univaq.it

Abstract

Nowadays different notations for architectural modeling have been proposed, each one focussing on a specific application domain, analysis type, or modeling environment. No effective interoperability is possible to date.

DUALLY *is an automated framework that aims to offer an answer to this need allowing architectural languages and tools interoperability.* **DUALLY** *has been implemented as an Eclipse plugin and it is based on model transformation techniques. This demonstration paper shows* **DUALLY** *by applying its approach to a UML-based notation and an outstanding ADL.*

1. Introduction

A proliferation of architectural notations (i.e. various ADLs and UML-based approaches) can be noticed today; each notation may differ either conceptually, technologically and operationally from the others. Even the adoption of UML for modeling architectures (e.g. [8, 6]) is biased by different concerns: a number of UML profiles and extensions have been proposed for modeling different architectural concerns, increasing even more the proliferation of architectural languages. Furthermore, there is not a unique way to model a software architecture (as already claimed in [3, 9]), testifying that it is also impractical to have a "universal" notation. Moreover, very limited interoperability possibilities among tools and notations exist because of their inherent differences.

These considerations led us to propose **DUALLY**, a framework to create interoperability among ADLs themselves as well as UML. Figure 1 conceptually shows the infrastructure of **DUALLY**.

Let us suppose that an architectural model M1 (conforming to its metamodel MM1) has been developed and that arises the need to model the same architecture using a different notation (whose metamodel is MM2). **DUALLY** provides the infrastructure to automatically obtain a model M2 in the target notation. This is possible because metamodeling experts can define semantic links between the metamodels (i.e. MM1 and MM2) and then **DUALLY** automatically instantiates these semantic links into model-to-model transformations. Therefore architects will use the generated transformations to interchange the notations to represent a specific architecture.

The main advantages **DUALLY** exposes can be summarized as follows: (i) **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 software architects, making **DUALLY** extremely easy to use; (ii) **DUALLY** permits the transformation among both formal ADLs and UML model-based notations; (iii) software architects can continue using familiar architectural notations and tools, and can reuse existing architectural models; (iv) **DUALLY** permits both languages and tools interoperability; (v) the semantic links among two architectural notations are defined once, and reused for each model that will be made; (vi) **DUALLY** is implemented as an Eclipse plug-in, so it is extensible and can easily cooperate with other Eclipse plug-ins.

2 The DUALLY framework

DUALLY works at two abstraction levels: meta-modeling (upper part of Figure 1), and modeling (lower part of Figure 1).

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 set of semantic links so as to relate architectural concepts in MM1 with the corresponding elements in MM2. The semantic links are captured by a weaving model. Weaving models are particular kinds of models containing links among models, meta-models or UML profiles.

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 enable the software architect to automatically translate the M1 specification (written according to MM1) into the corresponding M2 model (conforming to MM2) and viceversa. The generation of the transformations consists in the execution of higher-order transformations that take as input the semantic links between metamodels and produce the model-to-model transformations. The higher order transformations are general, thus the approach of **DUALLY** is metamodel independent, allowing full interoperability between all the architectural notations whose concepts may be defined through a metamodel or a UML profile.

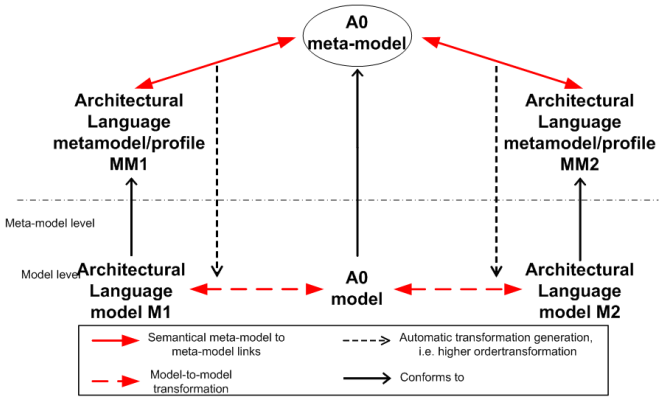


Figure 1. DUALLY Conceptual View

As it can be noticed in Figure 1, the weaving models (and their corresponding generated transformations) relate MM1 to MM2 (as well as M1 to M2) passing through what we refer to as A_0. A_0 is a minimal meta-model representing a semantic core set of architectural elements (e.g. components, connectors, behavior); it provides the infrastructure upon which to construct semantic relations among different ADLs. It is specific to software architectural domain and it acts as a bridge among architectural languages. The main benefit of using A_0 is the implied star architecture in which A_0 is the center of the star while **DUALLY**'s transformation engine is in charge of maintaining the transformation network. Due to space restrictions the why and how of A_0 cannot be discussed here and we refer to [7] and the **DUALLY** website [1]. The A_0 metamodel can be extended if there is the need to relate elements existing in both $MM1$ and $MM2$ which are not contemplated in A_0.

DUALLY is developed in the context of the ATLAS Model Management Architecture (AMMA) [2]. More specifically, it is available as a plugin of the Eclipse platform that extends the ATLAS model weaver (AMW) [4]. Models and meta-models are integrated into the same AMMA platform and, since AMMA is built on top of Eclipse, they are automatically integrated with several modeling technologies, such as Ecore and UML2. A high-level overview of the technologies we used is represented in Figure 3. Both meta-models and models (also weaving models via AMW's specific editor) are expressed via XMI, this allows users to define meta-models and models through any editor that exports models in the XMI format and to import it into **DUALLY** in a straightforward way. In particular, UML profiles can be realized using any UML tool which exports in UML2 [1], the EMF-based implementation of the UML 2.0 meta-model for the Eclipse platform. The transformation engine is based on ATL transformations [5], so that it is fully compatible with the other components of the AMMA platform used in the context of **DUALLY**.

[1]The home page of the DUALLY project is http://dually.di.univaq.it while the source code can be found in http://sourceforge.net/projects/dually, released under the GNU General Public License (GPL).

References

[1] UML2 project Web site, http://www.eclipse.org/uml2/.

[2] J. Bézivin, F. Jouault, P. Rosenthal, and P. Valduriez. Modeling in the large and modeling in the small. In *Lecture Notes in Computer Science, Volume 3599, Pages 33 46*, Aug 2005.

[3] E. M. Dashofy, A. van der Hoek, and R. N. Taylor. An infrastructure for the rapid development of xml-based architecture description languages. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 266–276, New York, NY, USA, 2002. ACM Press.

[4] Didonet Del Fabro M., Bézivin J., Jouault F. and Breton E. and Gueltas G. AMW: a generic model weaver. In *Proc. of 1re Journe sur l'Ingnierie Dirige par les Modles, Paris, France. pp 105-114*, 2005.

[5] F. Jouault and I. Kurtev. Transforming Models with ATL. In *Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005, Jamaica, pp 128-138*, 2006.

[6] P. Kruchten. Architectural Blueprints - The "4+1" View Model of Software Architecture. *IEEE Software*, 12(6):42–50, November 1995.

[7] I. Malavolta, H. Muccini, P. Pelliccione, and D. A. Tamburri. Providing Architectural Languages and Tools Interoperability through Model Transformation Technologies. Technical report, TR 001-2009, University of L'Aquila, Computer Science Department. Available at the **DUALLY** site, 2009.

[8] N. Medvidovic, D. S. Rosenblum, D. F. Redmiles, and J. E. Robbins. Modeling Software Architectures in the Unified Modeling Language. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(1), January 2002.

[9] N. Medvidovic and R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, 26(1), January 2000.

Appendix: how the demonstration will be carried out

The demonstration will be carried out using two projectors to provide both a technical and a practical perspectives in parallel. In the following, a description of the demonstration is given steps by step:

1. *Software Architectures and Architecture Description Languages*
   We will start the demonstration by giving a short introduction to software architectures and the existent languages and tools to describe software architectures. A table summarizing existing approaches will be presented with the aim of providing a snapshot of the state of the art and of the practice in this area.

2. *Model transformation techniques and the AMMA platform*
   As the audience might not be familiar with model transformation techniques and with the AMMA platform, their characteristics will be introduced and shown in details.

3. *The* **D**U**AL**L**Y** *framework*
   The parts that compose the **D**U**AL**L**Y** framework will be conceptually shown on projector 1 and practically shown on projector 2 (see Figure 2). By means of projector 2 we will illustrate how the weaving model and woven meta-models are represented within the **D**U**AL**L**Y**'s editor.

4. *Putting in practice* **D**U**AL**L**Y**
   In this presentation step we show how the conceptual features of **D**U**AL**L**Y** are applied to a real case study, explaining the typical usage session of our framework from the point of view of a software architect. Figure 4 depicts the modeling technologies we used to develop the case study. We selected a UML profile for software architectures (see Figure 5 and [7]) and the DARWIN/LTSA (see Figure 7 for the DARWIN/LTSA metamodel) modeling and analysis environment. We choose these notations (i) because DARWIN/LTSA is one of the most outstanding ADLs in literature and (ii) for the presentations sake since the two notations are not so different and thus it is straightforward to understand all the steps of the **D**U**AL**L**Y** process.

   In order to show the transformations from *UMLCC* to DARWIN/LTSA we selected the software architecture of a multi-tier environment capable of maintaining a fail-safe, client-server like communication within a safe and secure environment such as a military vessel: the Integrated Environment for Communication on Ship (IECS)[2] (see Figure 10 for the static description of IECS-MS architecture using the *UMLCC* profile). The case study's specification comes from a project developed within Selex Communications, a company mainly operating in the naval communication domain.

   Figure 9 and Figure 8 show the models produced by the transformations generated by **D**U**AL**L**Y**; they conform to the A_0 and the DARWIN/LTSA metamodels, respectively. In these figures we included also the interface of the programs we used to develop or import the models to show that **D**U**AL**L**Y** provides also tools interoperability.

5. **D**U**AL**L**Y***zation guidelines*
   In this step we aim to show to the audience how to **D**U**AL**L**Y**ze new ADLs. This will be shown by explaining step-by-step the **D**U**AL**L**Y**zation process.

6. *Future work*
   We will carry out the future work on **D**U**AL**L**Y** in two directions: (i) to use **D**U**AL**L**Y** in other contexts, different from software architectures and (ii) to extend the tool itself. As we said above, the transformation engine of **D**U**AL**L**Y** is metamodel independent, so we can use it in other contexts. For example, we are evaluating to use **D**U**AL**L**Y** with Klaper as pivot metamodel; Klaper is a kernel interchange language with a focus on performance and reliability attributes. The **D**U**AL**L**Y** tool can be extended in different ways: (i) we are investigating on how to semi-automatically generate also the weaving models, so that all the process of "duallyzation" becomes automatic; (ii) if the target model is modified **D**U**AL**L**Y** will provide the means by which all the modifications made in the target model will be automatically traced back to the source model; (iii) at the moment is up to the metamodeling expert to create weaving models that will produce valid model transformations, we will provide a technique to verify the quality of the generated transformations.

7. *References*
   Finally, the web site of the **D**U**AL**L**Y** will be shown to allow the audience to know where they can download the framework and the case studies as well as to find more details about the tool that is under construction and the release dates.

Depending on the time availability, some of the aforementioned items may be shortened or deleted.

[2] D. Colangelo, D. Compare, P. Inverardi, and P. Pelliccione. Reducing Software Architecture Models Complexity: A Slicing and Abstraction Approach. In Formal Techniques for Networked and Distributed Systems - FORTE 2006. pp.243258, 2006.

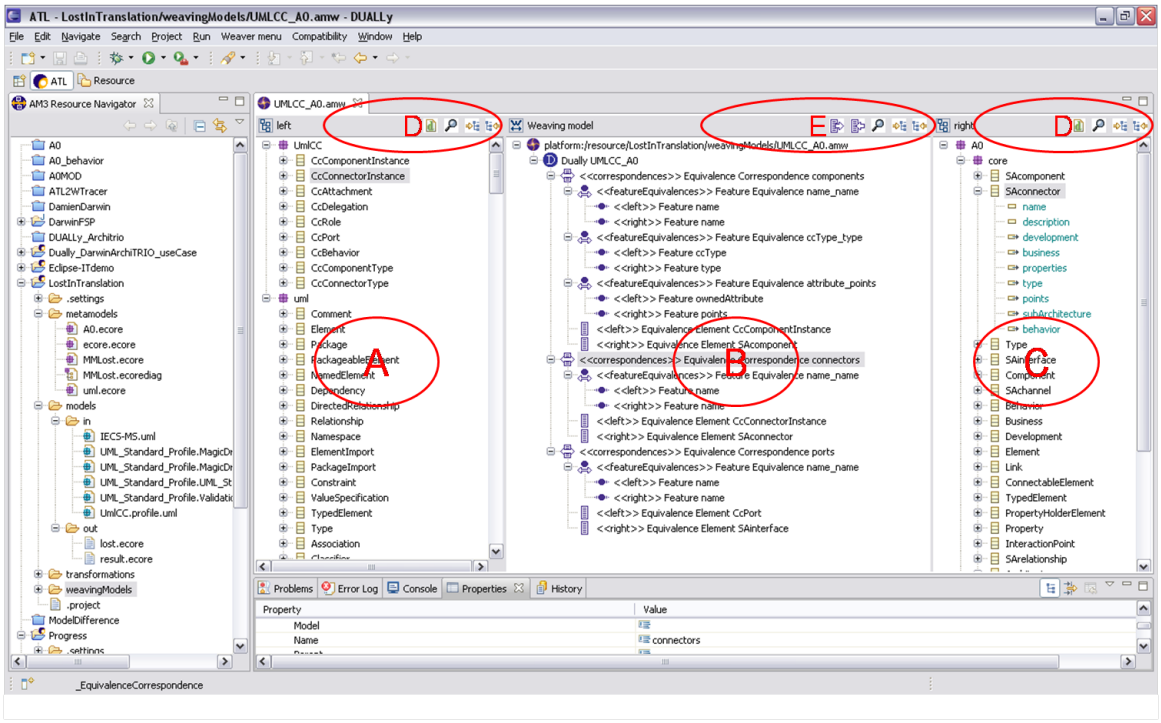


Figure 2. Graphical interface of DUALLY

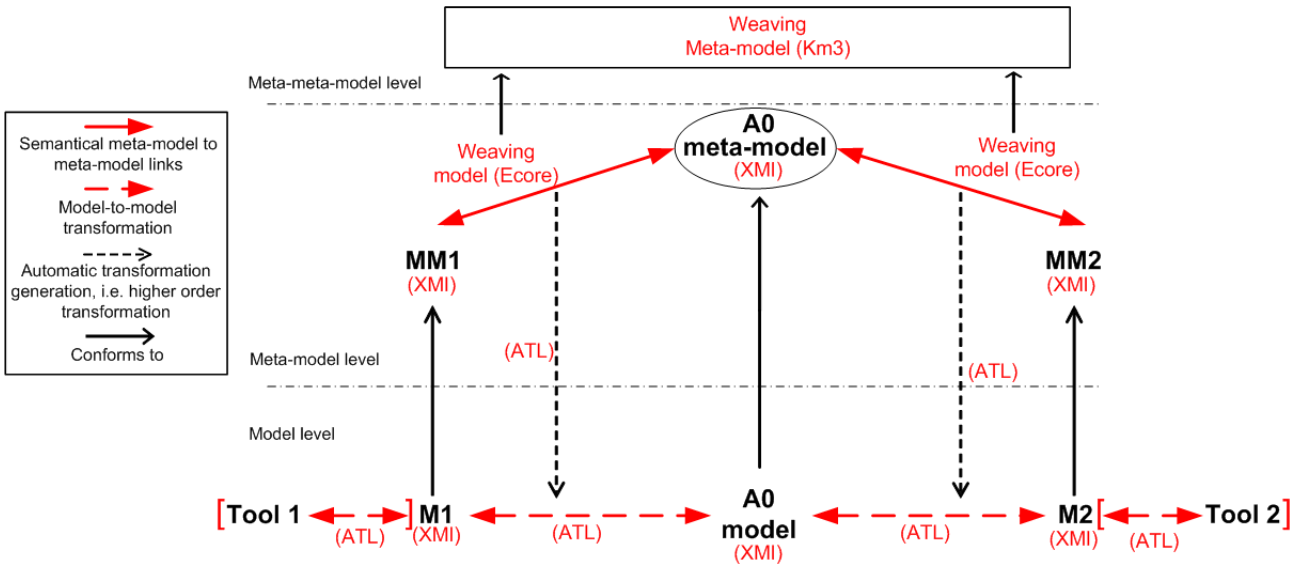


Figure 3. Modeling technologies used in the context of DUALLY.

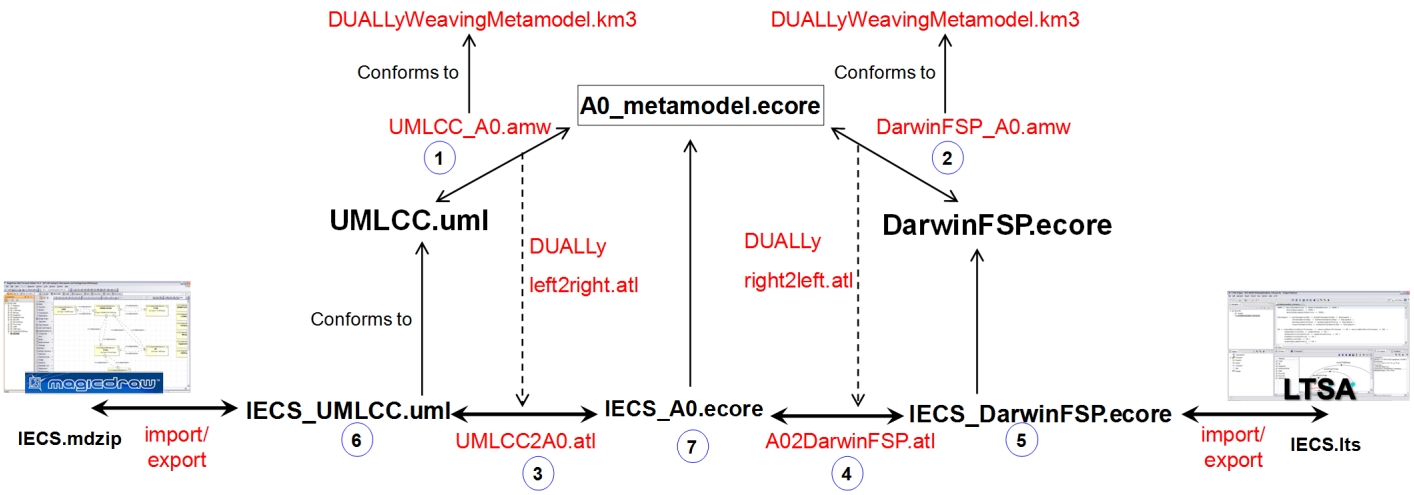


Figure 4. DUALLY instantiated

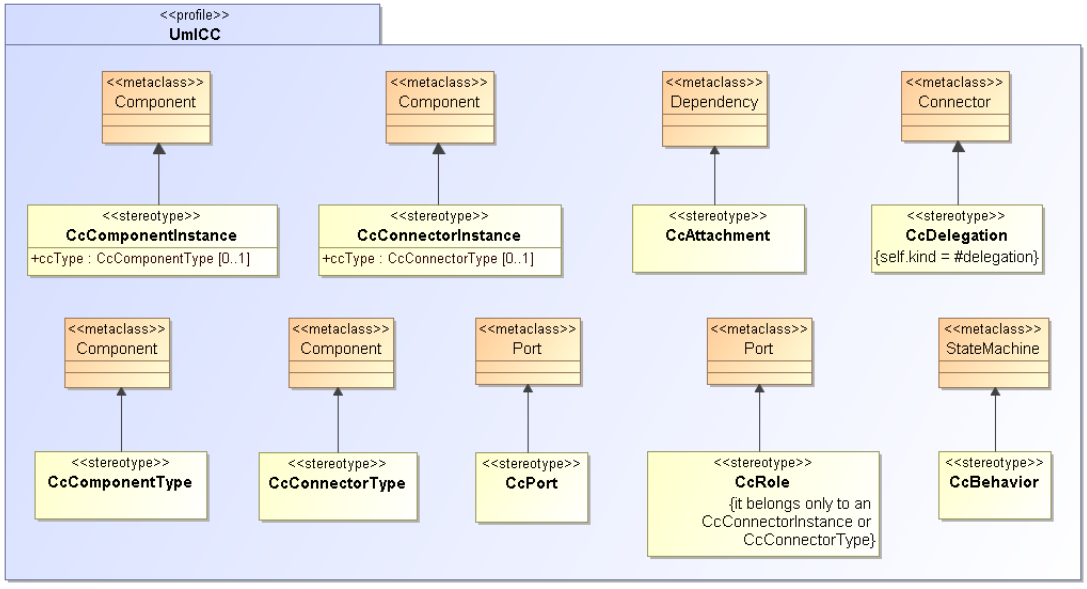


Figure 5. *UMLCC* profile.

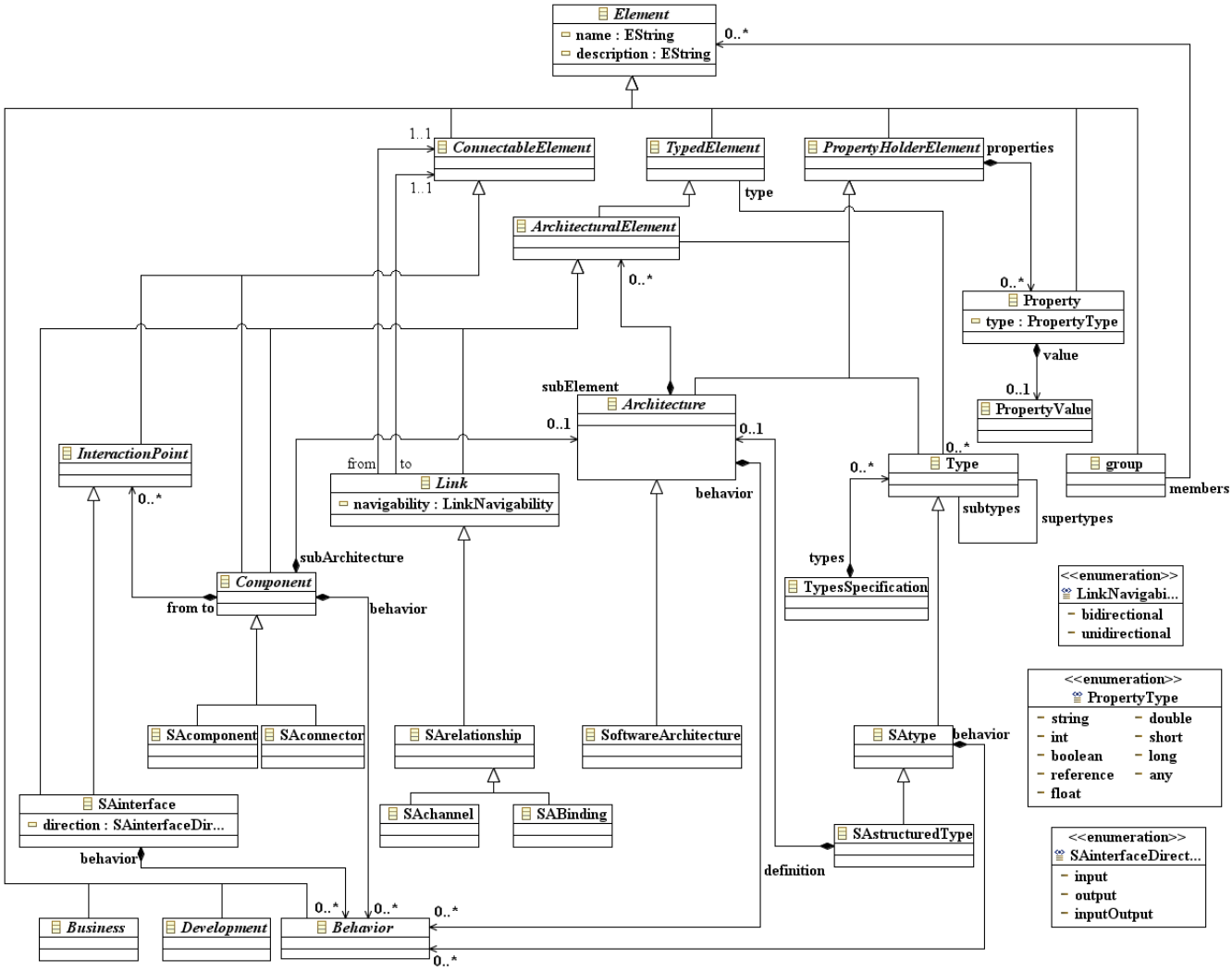


Figure 6. A_0 **metamodel.**

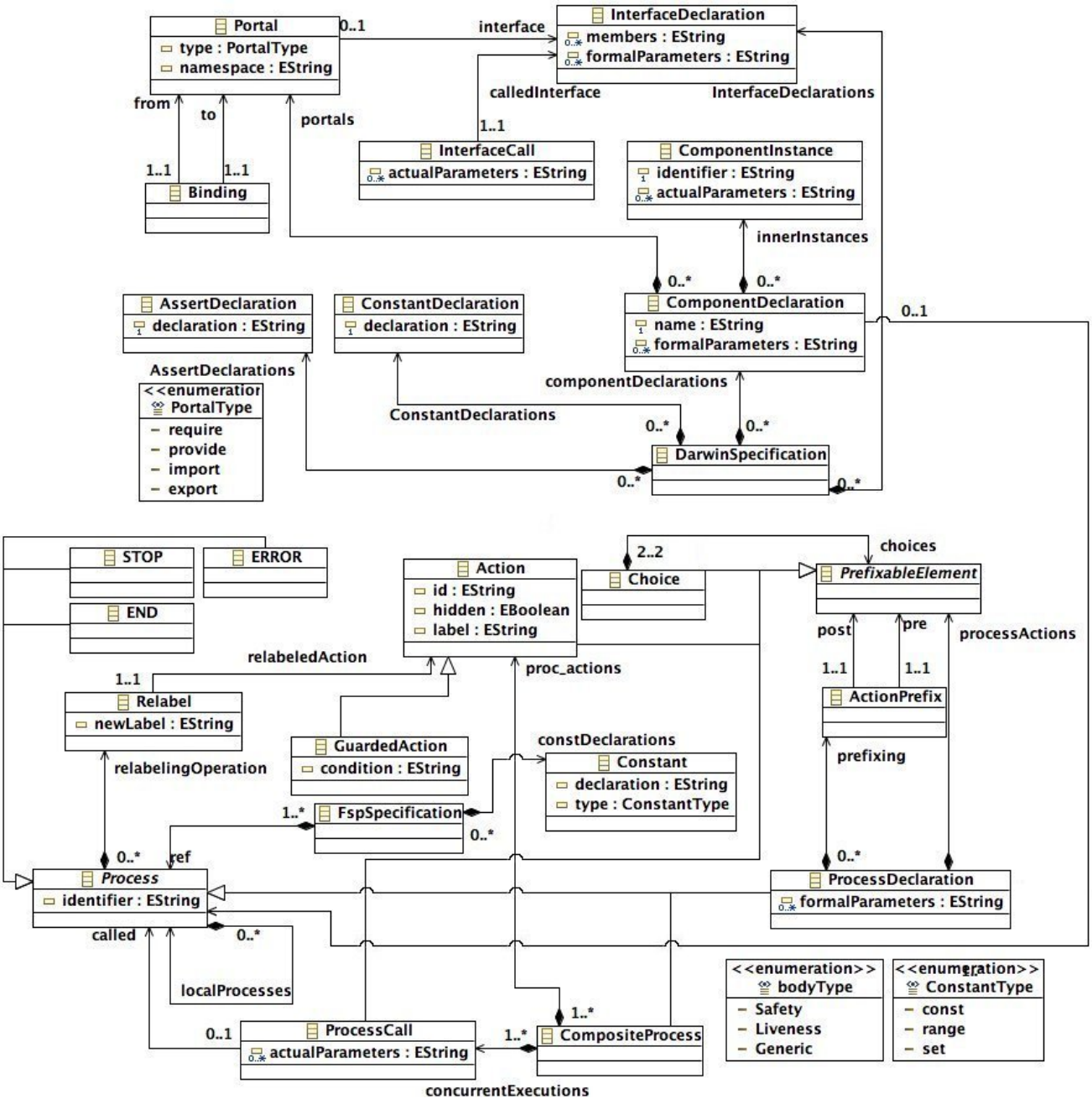
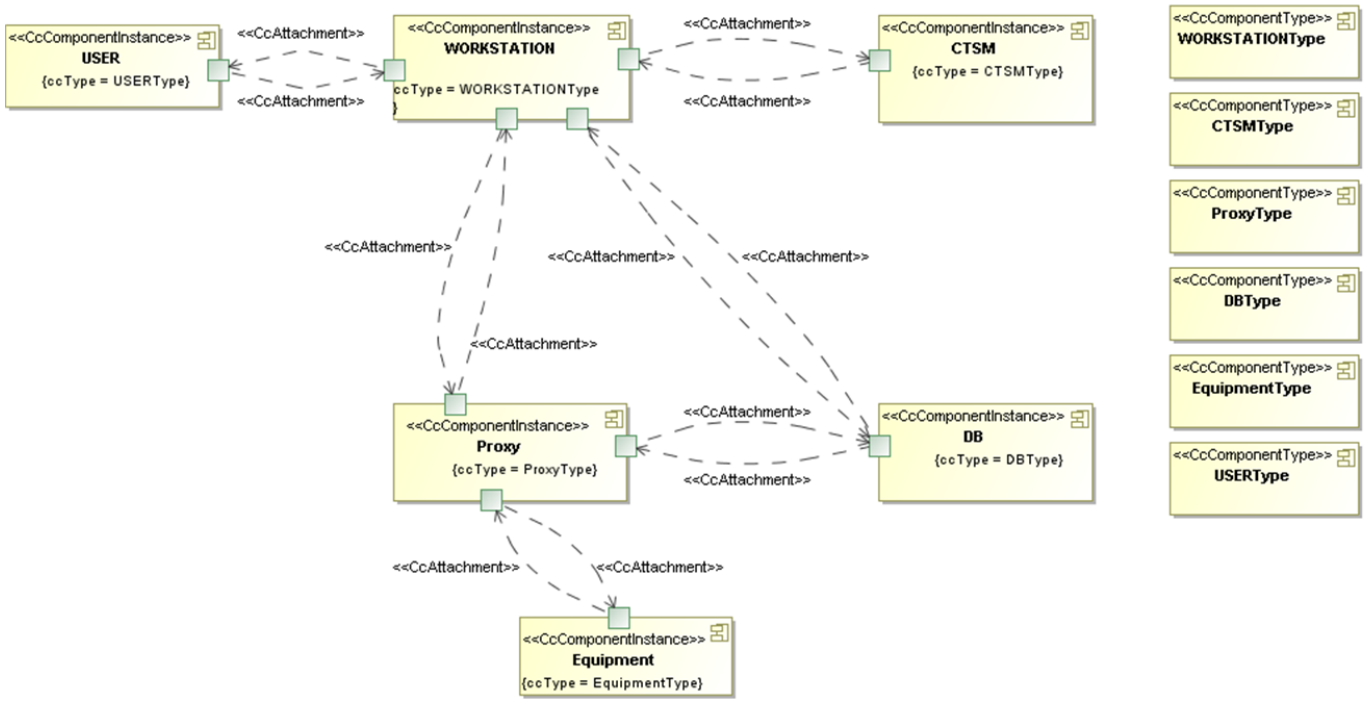
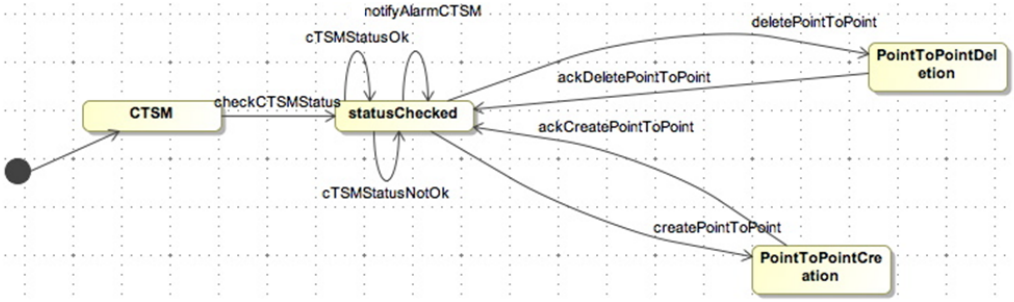


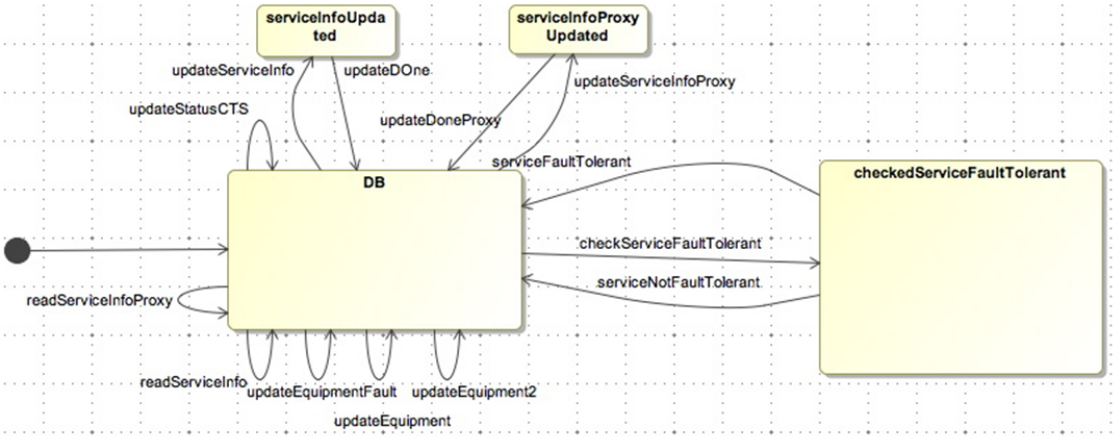
Figure 7. Darwin/LTSA metamodel.



IECS Component Diagram



CTSM State Diagram



DB State Diagram

Figure 8. Static description of IECS-MS architecture using the *UMLCC* profile.

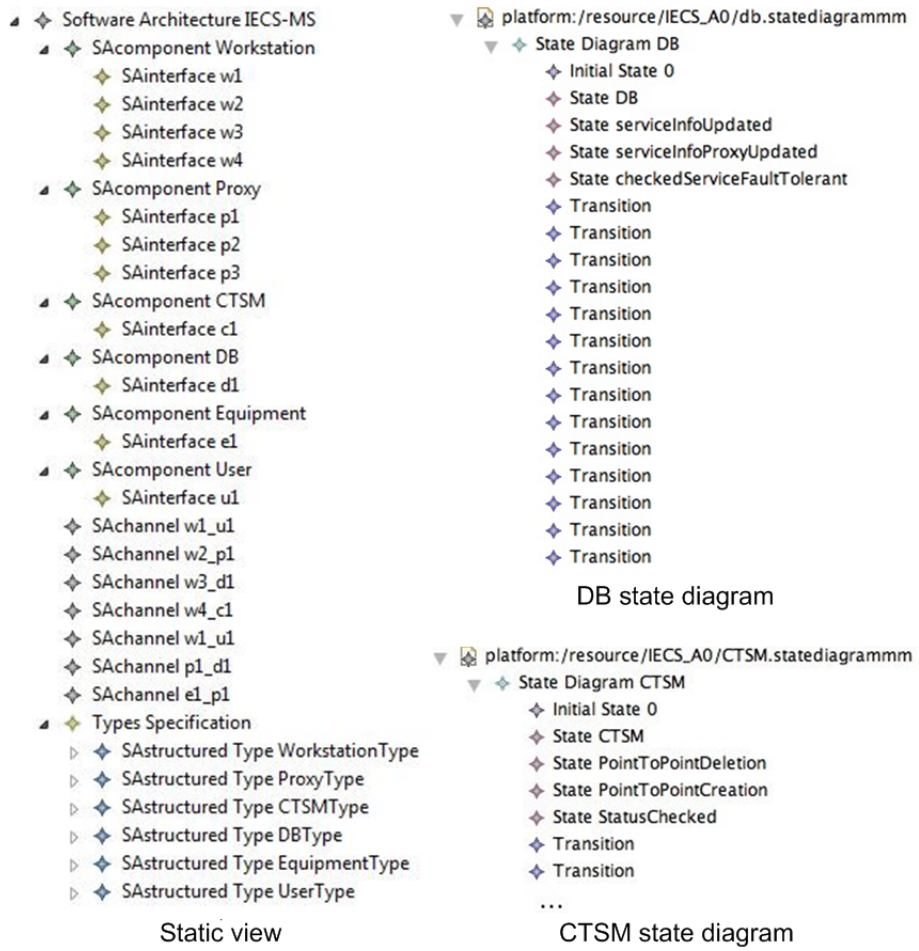


Figure 9. Static description of IECS-MS architecture conforming to the $A0$ metamodel.

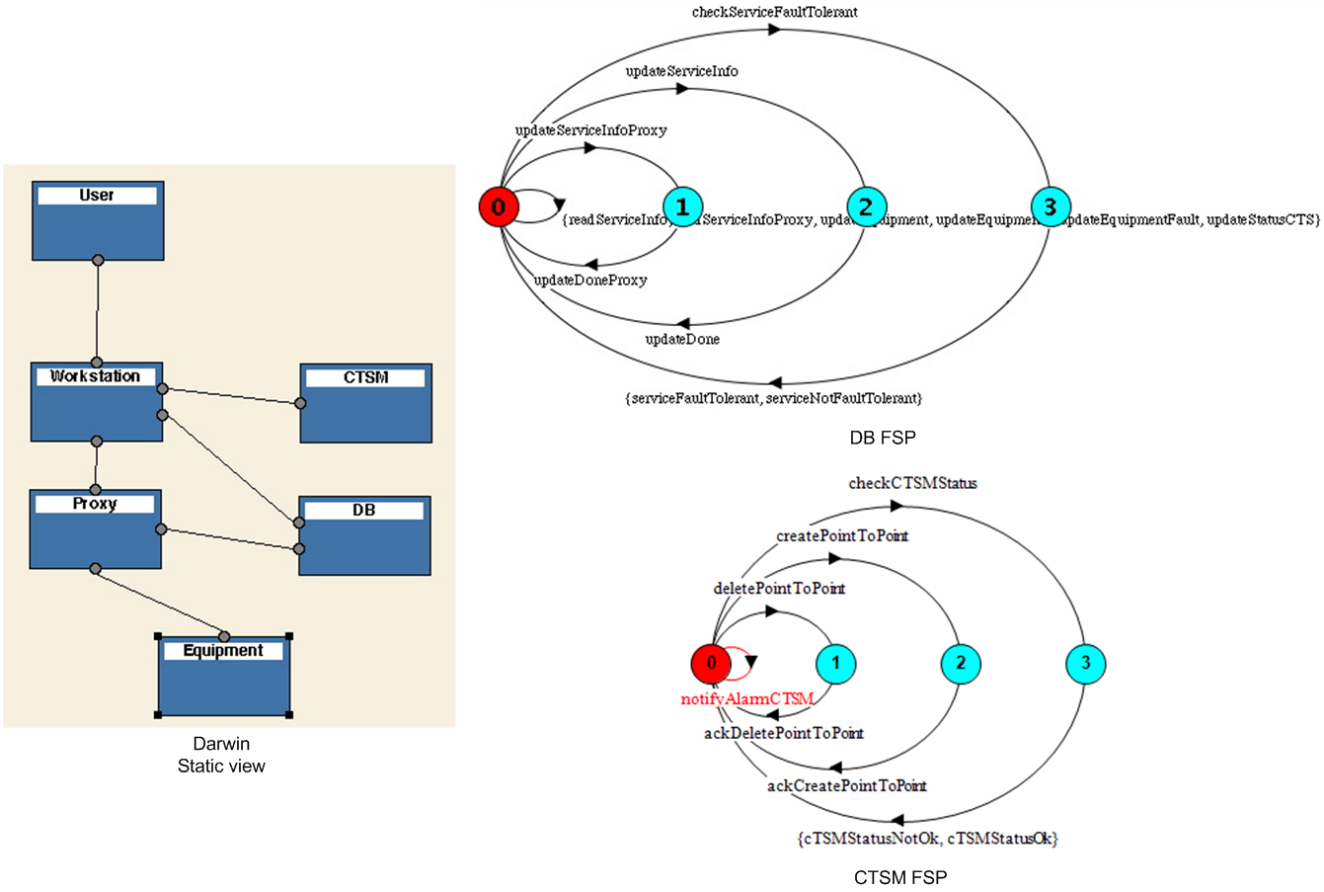


Figure 10. Static description of IECS-MS architecture conforming to Darwin/LTSA.