DUALLY: a framework for Architectural Languages and Tools Interoperability

Ivano Malavolta, Henry Muccini, Patrizio Pelliccione
Dipartimento di Informatica, Università dell'Aquila,
Via Vetoio, 67100 L'Aquila,
[ivano.malavolta,muccini,pellicci]@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 two outstanding architectural description languages.*

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 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, e.g., in [**?**]), 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 plugin, so it is extensible and can easily cooperate with other Eclipse plugins.

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.

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 ac-

cording 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 **DUALLY** is metamodel independent.

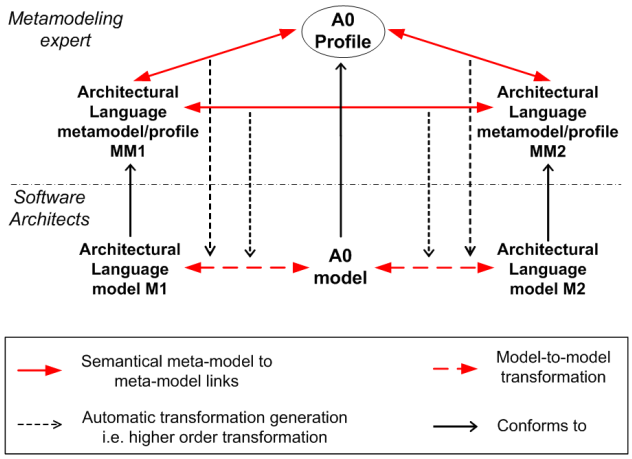

Figure 1. DUALLY Conceptual View

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 UML profile and it represents 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. Interested readers may refer to [**?**] and to the **DUALLY** website [1] for more details. The direct arrow between $MM1$ and $MM2$ signifies that it is also possible to define a direct weaving model between them if there is the need to relate elements existing in both the meta-models/profiles which are not contemplated in the A_0 profile. The quality of the generated transformations is discussed in [**?**].

DUALLY is developed in the context of the ATLAS Model Management Architecture (AMMA) [**?**]. More specifically, it is available as a plugin of the Eclipse platform that extends the ATLAS model weaver (AMW) [**?**] (see Figure 2 for a screen capture of the tool). 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. 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, the EMF-based implementation of the UML 2.0 meta-model for the Eclipse platform. The transformation engine is based on ATL transformations [**?**], so that it is fully compatible with the other components of the AMMA platform used in the context of **DUALLY**.

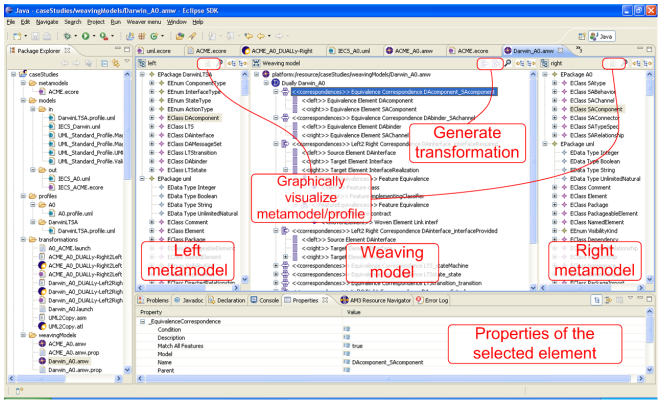

Figure 2. Graphical interface of DUALLY

Acknowledgments

The work is partially supported by ARTDECO (Adaptive infRasTructure for DECentralized Organizations), an Italian FIRB (Fondo per gli Investimenti della Ricerca di Base) Project.

References

[1] 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.

[2] 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.

[3] 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.

[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).

[4] I. 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 **D**U**AL**LY site, 2008.

[5] 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.