Developing a fully automatic bridge between UML profiles and EMF metamodels ⋆

Ivano Malavolta, Henry Muccini, Marco Sebastiani

University of L'Aquila, Dipartimento di Informatica
ivano.malavolta,henry.muccini,marco.sebastiani@univaq.it

Marco ►*io non ce l'ho un indirizzo che si chiama marco.sebastiani@univaq.it...*◄

Abstract. Ivano ►*abstract*◄

1 Introduction

Ivano ►*introduzione*◄

2 Motivational background

Ivano ►*1 pagina descrizione della situazione attuale SENZA il nostro bridge*◄ Ivano ►*spiegare il motivo per cui ci serve il nostro bridge*◄

In the last decade many languages for specifying and analyzing software architecture have been proposed. Architectural Description Languages (ADLs) are used to describe software architecture in terms of components, their interconnections and their behaviors. Nowadays basically two approaches appear and evolve. The first is the UML approach, which is largely adopted in every industrial contexts. The second approach is based on DSLs (Domain Specific Languages). It aims at representing each domain with a specific metamodel. The UML approach uses profiles to model domain specific concepts. A UML profile describes the semantics of systems or applications which is not supported by the UML metamodel. DSLs, on the other hand, represent each domain with a specific metamodel. Both approaches are adopted to tailor the language in system design to a specific need, and are currently used by an increasing number of communities. The volume of artifacts of both approaches is rapidly increasing too. Whatever the future evolution of MDE in either branch, we need interoperability between these approaches: to produce an UML profiled model from a model conforming to a DSL metamodel and vice versa. At the moment, when

In previous researches the problem of interoperability and integration has already been addressed. In Abouzahra et al. [**?**], the integration process starts with an UML profile and one has to define the mappings between the DSL metamodel and the UML profile elements. The model transformations between the DSL and UML models are automatically derived from these mappings. Another approach has been proposed by Manuel Wimmer in this [**?**]. Wimmer assumes that there's no available profile yet, and

⋆ This work is partly supported by the Italian PRIN d-ASAP project.

he proposes a Bridge Generator component which generates transformations and the profile by means of an explicit, and manually build, mapping model. The principal benefit of this approach is the reduction of error having a single source of information (the mapping model) and automated repetitive tasks by means of the transformation.

Starting from this point we propose as a solution a model-driven engineering tool to build these bridges. The system we propose connects the UML approach to the DSL one, as the Abouzahra partially does, by means of several automated transformations, as in Wimmer approach. Differently from the previous approaches, our bridge automatically generates a metamodel from an existing profile and then it automatically generates transformations which convert a UML model which has applied a profile to the corresponding model conforming to the brand new generated metamodel.

3 The automatic bridge

Ivano ►*descrizione completa del bridge senza dettagli*◄

Marco ►*da inserire immagini, e completare l'ultimo capoverso, quello che dipende dallimmagine. Per il resto, salvo altre considerazioni "SOLO" da revisionare*◄

The concept of bridge is used to denote the capability of interoperability between the UML approach and the DSL one. When a corresponding bridge is available for a metamodel and a corresponding profile, it is possible to produce automatically an UML profiled model from a model conforming to the metamodel in question and conversely. In our project we implement a bridge between the metamodels, either between models. The resulting artifacts of the bridge are: a metamodel which represents the profile we are about to translate, automatically generated transformations which bridge a UML model conforming the profile in question and a model conforming to the brand new generated metamodel.

Marco ►*io ci metterei la figura che rappresenta schematicamente il bridge, quella che descrive quali sono gli input, qual e il bridge ad alto livello e cosa restituisce. Se va bene la inserisco e chiudo questo paragrafo descrivendola come di seguito*◄ **Marco** ►*senza figura questa parte la cancello –¿*◄

In the below figure, source models are white, representing the input artifacts to be transformed, target models are red, to denote that theyre automatically generated. The yellow shapes denote the bridge engine. As it can be noticed, the bridging procedure is split in two phases: the first one, the yellow ellipse in figure, works at the metamodeling level by generating a metamodel which is the representation of the profile (and the UML metamodel); the second phase, the yellow arrow in figure, works at the modeling level by automatically generating transformations from a UML model which has applied a profile to a model which is conform to the brand new generated metamodel.

Marco ►*¡–senza figura questa parte la cancello*◄

The resulting metamodel can be considered a union of the UML metamodel (copied in the target one by means of an ad-hoc trasformation superomposed to our one) and the metamodel described in the profile model. We propose a further optimization to our tool which lets the developper to reduce the size of the UML metamodel imported in the target one.

Marco ►*metto due righe dicendo cosa si dira nel seguito del paragrafo?*◄

In the following we will show the bridge behaviour at the metamodeling level and at the modeling one. In the end we will show the procedure we propose to slice the obtained metamodel.

3.1 The bridge at the metamodeling level

Ivano ►*livello meta con esempietto*◄

At the metamodeling level the bridge takes as input the UML metamodel and the profile. As shown in the previous figure

Marco ►*nel primo paragrafo*◄

, it generates a metamodel by coping the UML metamodel in ECORE formalism, and by adding futher constructs which represent the elements in the profile. The figure

Marco ►*inserire riferimento*◄

represents a UML profile translated into an EMF metamodel. All the UML metamodel elements are in the target metamodel, all the profiled elements (stereotypes and further data types) are represented in the target metamodel. The extension mechanism of a stereotype and its metaclass is represented by means of a binding mechanism. Every super class has a reference to the eventual stereotype.

Marco ►*immagine usata nella presentazione, quella della slide Approccio finale, tra profilo e metamodello target*◄

In the figure

Marco ►*inserire riferimento*◄

we show the behaviour of our approach. We have a stereotype *MyStereotype2* which extends two metaclasses. By means of the superimposed transformation we copy the UML metamodel, and we enrich it by adding extension attributes to every metaclasses which has been extended in the profile. The bridge, instead, creates a brand new element bounded with a composition relationship with the metaclasses it extends. In this scenario we have *MyStereotype1* linked to the *Class* metaclass and *MyStereotype2* linked both to *Class* and to *Port*. As you can see in the fgure the classes of the UML metamodel are extended in the target metamodel since they own their attributes plus an extension extra-attribute which binds them to each stereotype.

3.2 The bridge at the modeling level

Ivano ►*livello modello con esempietto*◄

At the modeling level the bridge automatically generates transformations which bridge a profiled model in a model conforming to the brand new generated metamodel.

Marco ►*immagine usata nella presentazione, quella della slide Generazione di regole di trasformazione, che mostra il codice generato nelle tre fasi. La posso modificare rendendola unimmagine unica e colorando le 3 fasi di generazione*◄

As shown in the above figure, the transformation generates a transformation at the modeling level in three phases: in the first one we generate rules to transofrm UML metaclasses into EMF ones; in the second phase we generate new rules to transform stereotypes applications into metaclassesinstances representing the stereotype in the target metamodel; in the last phase we generate imperative code, inserted at the end of

every rules generated in the fist phase, to trigger the rules obtained in the second phase. The result is a transformation able to bridge a profiled model in a model conforming to the corresponding brand new genarated metamodel.

Marco ►*immagine usata nella presentazione, quella della slide Applicazione sul caso di studio, tra modello profilato e modello target / oppure quella utilizzata nella tesi, capitolo 5.1.1 (ovviamente la rifaccio usando magic draw)?*◄

By executing this brand new transformation we complete the bridge from UML profiling to EMF metamodeling. As example we use the same scenario of the previous paragraph - modeling level. The model element **Marco** ►*... bla bla bla*◄

4 Slicing the obtained metamodel

Ivano ►*slicing*◄

5 Implementation of the bridge

Ivano ►*dettagli sulle tecnologie usate, tipo ATL, EMF, ecc.*◄
Ivano ►*dettagli tecnici dell'implementazione*◄

6 Case study

Ivano ►*case study su SysML*◄

7 Related work

Ivano ►*descrizione di lavori simili in letteratura e come si differenziano da noi... bezivin e wimmer differenze col nostro*◄ **Marco** ►*un paragrafo ogni approccio. bezivin e wimmer differenze col nostro, EMF umluitl.java (solo livello meta)*◄

8 Conclusions

Ivano ►*conclusioni*◄