A Formal Framework to Unified Metamodel for Consistent Transformation

Jagadeeswaran Thangaraj[1,2] and Senthilkumaran Ulaganathan[2]

[1] Maynooth University, Maynooth, Ireland
[2] VIT University, Vellore, TN, India
Jagadeeswaran.Thangaraj@mu.ie

Abstract

We propose the development of a formal framework to unified metamodel within which models of class artefacts can be shared between different phases in a model-oriented environment. Model transformation approaches use different metamodels to represent source and target model of the system. This paper investigates for a unified metamodel when they share set of core representations in different phases and checks the possibilities for multidirectional transformation for code generation, upgradation and migration purposes. We envisage that the construction of an institutional framework for unified metamodel will not only increase the reusability of class artefacts but also provide a foundation for the consistent, independent and multidirectional transformation.

1 Introduction

Recently, autonomous systems development approaches get more attention. These help the developers to build software systems from requirements phase through maintenance. A model-driven development based technique is one of the approaches to software development. These play a role in both forward and reverse engineering of development part, such as, software design through implementation. In software design, developers model a system's design according to the client's requirements and they generate code using this design using some transformation techniques. In some cases, they transfer code implementation to design through reverse engineering for upgrading or verifying the correctness of the system they have developed, or transferring into other languages for further developments. The objective of this approach is to increase productivity, reduce time consumption and be cost effective.

2 Background

Object Managements Group's Model Driven Architecture (MDA) [8] relies on a set of concepts being models, modelling languages, metamodels and model transformations. A model provides a full description and overview of the system to be developed. A metamodel defines the abstract syntax of models and the interrelationships between model elements. In MDA, metamodels play a key role. Metamodeling is a technique for constructing properties of the model in a certain domain. The metamodel defines the general structure and characteristics

of real world phenomena. Therefore, a model must conform to this protocol of the metamodel similarly to how a grammar conforms for a programming language. Model transformation is a mechanism for transforming one model into another model, based on some transformation rules or programs. A transformation engine performs the transformation using corresponding program which reads one source model conforming to a source metamodel and writes a target model conforming to a target metamodel.

2.1 A Unified Metamodel

Recently, many researchers are working towards developing a unified representation to generate code from design when sharing core elements in related formalisms [3][2][9]. We have introduced a unified metamodel which has unified properties of source and target metamodels of design to implementation. In model transformation, source metamodel is different from target metamodel. Unified metamodel is based on the intersection of both source (USE [6]) and target (Spec# [7]) representations. The unified metamodel has same values for related properties for source and target metamodel as shown in Fig.1. Our approach also provides two main applications in model transformation such as: Model reusability and Multidirectional transformation.

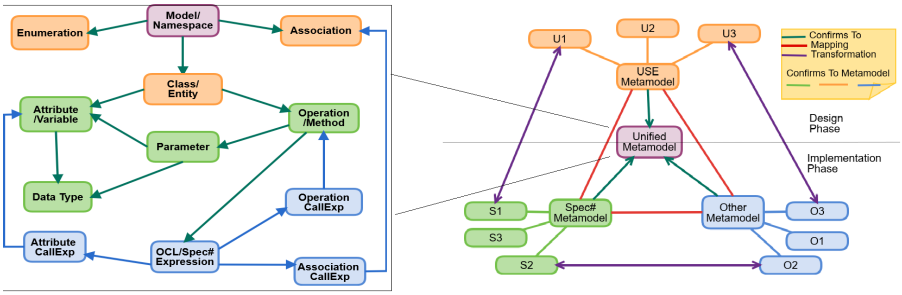

Fig. 1. Model transformation using unified metamodel

3 Our proposal: An Institution for Unified Metamodel

The question that this proposal addresses is: how to develop a formal framework to unified metamodel that allows software developers to reliably re-use models between different software representations in a model-oriented environment. The research aim of this proposal is to establish a formal framework of a unified metamodel within which these artefacts can be shared between programs and models in a model-oriented environment in order to avoid lost of constructs. Such a framework should provide a precise, formal definition of how the class artefact is constructed from unified metamodel. The framework should also be able to provide for the heterogeneity in the formalisms used, since both the software representation and the derived artefact may be based on different semantic or

models. The theory of institutions has been identified as a mechanism by which these goals can be achieved. Institutions have been defined for many logics and formalisms, including programming-related formalisms[1]. Institutions allow you to build up specifications in a formalism-independent manner using some basic constructs from category theory. Specifically, we propose to investigate this in the context of class invariants, building on our research strengths; these two artefacts are *ownership types* and *expose* blocks. These artefacts represent vital elements of the software verification process.

4 Significance

The models are defined in terms of their associated modelling languages, defined as metamodels which are themselves (typically) defined in terms of some meta metamodelling language. These metamodels also provide the basis for defining transformations between models, given a suitable transformation language. It is still easy for meaning to become lost or blurred when unification of metamodels in a model-oriented environment. One relatively simple example of a model transformation is reverse-engineering a USE class specification from a piece of Spec# code. This research focuses primarily on reasoning about ownership types [4] that are specified in terms of the concrete data and on the complications that arise due to unification. However, formally relating ownership types across multiple model representations requires a shared understanding of the model semantics. While significant advances have been made, supplying a formal framework for model-oriented development does still need more concentration in order to support consistent, independent and multidirectional transformation.

In conclusion, we have successfully defined unified metamodel for USE and Spec# and proved the required properties to reuse models and multidirectional transformation [5]. Our current task is that of implementation of institutional representation for unified metamodel. A significant future challenge is the integration of proofs for model instances generated by institutional morphisms of unified metamodel in corresponding formalisms.

References

1. D. Sanella and A. Tarlecki, "Foundations of Algebraic Specification and Formal Software Development". Springer, 2012.
2. Fayoumi, Amjad and Kavakli, Evangelia and Loucopoulos, Pericles, "Towards a Unified Meta-Model for Goal Oriented Modelling", In EMCIS 2015.
3. Huseyin Ergin, Eugene Syriani, "A Unified Template for Model Transformation Design Patterns", In Patterns in Model Engineering, co-located with STAF 2015.
4. Jagadeeswaran.T, Senthilkumaran.U, "Introducing Ownership Types to Specification Design". In IJSER DOI:https://dx.doi.org/10.14299/ijser.2017.08, 2017.
5. J.Thangaraj, S.Ulaganathan, "Towards Unified Metamodel for Multidirectional Transformation", In Journal of Engineering and Applied Sciences, In press.
6. Martin Gogolla, Fabian Büttner, Mark Richters, "USE: A UML-based specification environment for validating UML and OCL", Science of Computer Programming 07
7. Mike Barnett, Rustan Leino, Wolfram Schulte, "The Spec# programming system: An overview", In CASSIS 2004: Springer, 2004.
8. OMG: Object Management Group:MDA Guide, version 2.0. ormsc/2014-06-01, http://www.omg.org/mda/ 2014.
9. S. Sepúlveda, C. Cares and C. Cachero, "Towards a unified feature metamodel: A systematic comparison of feature languages", In CISTI 2012, Madrid, pp.1-7, 2012.