A Contribution to the Specification of Model Transformations with Metamodel Matching Approach

Karima Berramla, LAPECI Laboratory, University of Oran 1 Ahmed Ben Bella, Oran, Algeria
El Abbassia Deba, LAPECI Laboratory, University of Oran 1 Ahmed Ben Bella, Oran, Algeria
Abou El Hassen Benyamina, LAPECI Laboratory, University of Oran 1 Ahmed Ben Bella, Oran, Algeria
Djilali Benhamamouch, LAPECI Laboratory, University of Oran 1 Ahmed Ben Bella, Oran, Algeria

ABSTRACT

Model-driven engineering (MDE) is a paradigm based on the intensive use of models throughout the life cycle of an application, where model transformation plays an important role. Various model transformation approaches have been proposed, but developers are still faced with the complexity of model transformation specifications. Most of these approaches are based on the specification of transformation rules with a concrete syntax at a low level where the developer must master the transformation language. The question at this level is how to generate a model transformation specification that must be at a very abstract level, independent of any transformation language. This article aims to propose an approach to generate an abstract representation of transformation rules and these are used to produce a source code written in a chosen transformation language. The transformation rules are calculated semi-automatically by using a matching technique on elements of source and target metamodels. This idea is illustrated by different transformation examples.

KEYWORDS

Generation Automatic of Programs, Jaccard Similarity Index, Jaro Winkler Distance, Matching Techniques, Metamodel, Model, Model-Driven Engineering, Model Transformation, Model Transformation Generation

1. INTRODUCTION

Since the software engineering advent, software development has become more and more important in different application domains and evaluates in a fast-growing manner. To face up to this situation, the Object Management Group (OMG) has proposed Model-Driven Engineering (MDE) to solve the problems posed in the software engineering context, such as optimization of development time for computer systems. Consequently, MDE tries to provide an answer to these problems. The model-driven engineering is based on the systematic use of models to automate part of the development process followed by engineers. Also, it’s a domain of research in full emergence that considers models as basic elements in the development of computer systems. Thus, MDE proposes for modeling the applications in a high-level abstraction where it places the models in the heart of design process and generates application code from one or more models. It provides concrete solutions to better control the complexity of systems and it focuses to increase interoperability, reusability and migration of models in heterogeneous Applications.

DOI: 10.4018/IJISMD.2017070101
Thus, MDE is based on two important concepts: metamodeling and model transformation. The first one is based on using a metamodeling language for specifying models to define a system at an abstract level. The second concept permits to create and manipulate models in order to automate and reduce the complexity of the development process. MDE was built around the utilization of standards such as MOF, UML/OCL, XMI where literature on these formalisms are quite abundant.

MDE is a generative engineering technique in which models occupy a very important place and must be as consideration sufficiently precise and rich in order to be interpreted or transformed by machines. The system’s development process can then be seen as a sequence of model transformations partially ordered, each transformation taking one or more input models and generating one or more output models until an executable code. The concept of model transformation has a principal role in MDE where it automates critical operations in the development of complex systems such as “refinements”, “refactoring” and “code generation”. The automation of these operations can increase the reuse of elements in a project to another that causes economies in terms of effort and time. However, this automation can also introduce an additional risk of error because of erroneous model transformations.

To implement a model transformation, we must pass through two steps: the first one concerns the metamodel definitions and the second one presents the specification of transformation rules. Various model transformation approaches have been proposed and adopted but developers are still faced with the complexity of model transformation specifications. Therefore, the implementation of model transformation requires mastery of a transformation language as well as sufficient knowledge of the metamodels space and models on which the transformation is to be applied. For this purpose, approaches have been taken to help the developer to generate model transformations in a more or less automatic way. Most of these approaches are based on the specification of transformation rules with a concrete syntax at a low level (Varró, 2006; Dolques, 2010). In this context, the generation of transformation rules has been proposed as a solution to automate the model transformation. This solution can be divided into two classes (Dolques, 2010), the first class is based on the metamodel alignment and the second is based on the example’s model where each class can be divided too into several sub-classes.

At this level, a question imposes oneself: how to generate a model transformation specification that must be at a very abstract level independent of any transformation language? This paper aims to propose an approach to generate an abstract representation of transformation rules and these latter are used to produce a source code written in a chosen transformation language. The transformation rules are calculated semi-automatically by using a matching technique on elements of source and target metamodels. The matching technique uses two different similarity functions. This idea is illustrated by transformation examples with different types: endogenous and exogenous transformations.

The rest of this paper is structured as follows: basic knowledge about transformation process is presented in section 2. Section 3 introduces the problem statement and discusses the classification of model transformation generation by presenting the related works. Section 4 presents the dimensions of transformation rules generation. In section 5, introduces the importance of matching in MDE context. Section 6 details the proposed approach in general and specifically the generation of transformation rules. Finally, section 7 concludes this paper.

2. BACKGROUND AND MOTIVATION

In this section, we will recall the basic ideas and the fundamental concepts of model transformation to have a good knowledge in MDE Discipline.

In MDE context, model transformation defines the translation of model into another model based on a transformation program written in a specific language. Figure 1 illustrates the idea of model transformation according to Model-Driven Architecture (MDA). In the next, we define the following
Abstract and Concrete Data Type Optimizations at the UML and C/C++ Level for Dynamic Embedded Software
www.igi-global.com/chapter/abstract-concrete-data-type-optimizations/36338?camid=4v1a

Designing Web 2.0 Collaboration Tools to Support Project-Based Learning: An Activity-Oriented Approach
Ronnie Cheung and Doug Vogel (2012). International Journal of Systems and Service-Oriented Engineering (pp. 1-14).
www.igi-global.com/article/designing-web-20-collaboration-tools-to-support-project-based-learning/78915?camid=4v1a