Automatic Pattern Proposition in Transformation Life Cycle

Mahsa Sadat Panahandeh, MDSE Research Group, Department of Software Engineering, University of Isfahan, Isfahan, Iran
Bahman Zamani, MDSE Research Group, Department of Software Engineering, University of Isfahan, Isfahan, Iran

ABSTRACT
Transformation patterns optimize transformations and improve their internal structure. This paper presents an automatic method for pattern proposition in transformation life cycles. Our approach combines the transformation engineering by the idea of transformation patterns and proposes appropriate patterns in the design and implementation phases of transformation life cycle. We evaluate the impact of proposed patterns on generated transformations based on several metrics, automatically. The proposed patterns in the design phase are based on the structure of transformation, and implementation patterns are proposed according to the result of the evaluation. Our method is presented as a semi-automatic process for using the transformation patterns in transformation life cycles. The results of applying suggested patterns include decreasing the complexity, memory usage, execution time, as well as increasing the quality, efficiency, and modularity.

KEYWORDS
Model Driven Engineering, Pattern Proposition, Transformation Life Cycle, Transformation Pattern

INTRODUCTION
A pattern is an idea that has been useful in one practical context and will probably be useful in others (Fowler, 1997). Transformation patterns identify the correct and reusable solution in transformation implementation and optimize the transformations, too. Similar to patterns such as design and analysis patterns in software engineering, transformation patterns can be used in transformation engineering to improve the quality and efficiency of transformations. Transformation engineering means applying the sound principles of engineering in the implementation of a model transformation (Guerra et al., 2013). Several life cycles are presented in recent research (Bézivin et al., 2003; Siikarla et al., 2008; Lano & Kolahdouz-Rahimi, 2010; Bollati et al., 2013; Guerra et al., 2013), for building the transformation based on engineering methods. These life cycles, that two of the best ones (Bollati et al., 2013; Guerra et al., 2013) are explained in this paper, create a transformation using a methodology that includes several steps, spanning from the specification of transformation to the implementation. However, we can use design, architectural, or specification patterns that are called transformation patterns in different phases of these life cycles, in the same way the software patterns use in the life
cycles of software engineering. One important part in using a pattern is recognition of the appropriate pattern based on the structure of the problem. The main goal of this paper is automatic proposition of applicable patterns in the transformation life cycle. The transformation patterns presented in the past (e.g., (Bezivin et al., 2003) or (Iacob et al., 2008)), have introduced only the specification of several patterns. Some of these patterns are formally described in a few papers, but these are not usable in an automatic manner. In fact, developers must use them manually, based on their understanding of the pattern. In addition, these presented patterns are not applied to transformation life cycles; they are just related to one phase of the transformation implementation, e.g., transformation specification or transformation architecture.

In this paper, we present a practical method for automatic pattern proposition in transformation life cycle. The method automatically suggests proper patterns in two phases, i.e., design and implementation phases, of transformation life cycle to the developer of transformation. Proposition of design patterns is based on the structure of transformations while implementation patterns are conducted from the result of applying the other patterns. At each stage, in addition to the proposing the implementation patterns, evaluation of applying suggested patterns can be shown to the developer. As it is demonstrated in the evaluation section, this evaluation is measured based on several metrics, automatically. Evaluation of our method has shown that, for using the patterns in transformation life cycles, it improves the quality of transformation specification and design, modularity, simplicity of transformation, efficiency, as well as decreasing redundancy, duplication, complexity, execution time and memory usage (in many cases). This paper is organized as follows. Next section is dedicated to the related work. Then we explain about the necessity of using the transformation life cycle and introduce two transformation life cycles that are the most complete life cycles which are used in this paper. Subsequently, a transformation pattern is introduced and two patterns as two samples are presented in this section. At this time, we give an overview of our method for pattern proposition in transformation life cycles. Description of our method is structured in two subsections. First, we elucidate the pattern proposition part as the core of our method, and second subsection outlines evaluation part of our method. In this paper, we use a case study to clarify our method. Finally, in the conclusion section we conclude by discussing future work.

RELATED WORK

Transformation patterns have been studied and used by several researchers previously. Most of these researches focus on the introduction of problems in transformation domain and solve them by presenting several patterns. In (Bezivin et al., 2003), authors profess that patterns are a good way of capitalizing experience to increase the reusability. This paper recommends two patterns for ATL\(^1\) transformation language. One of these patterns explains specification of the transformations. This pattern suggests an additional input metamodel that contains auxiliary variables of a transformation. This metamodel covers recurrent hard-coding in ATL language that caused by unmatched elements of the transformation within the source metamodel. Another pattern is used in the implementation of transformations. This pattern is about matching multiple elements in ATL rules. Multiple matching is supported in the new version of ATL, so this pattern is not applicable. Patterns presented in (Bezivin et al., 2003) are only introduced and no manner is presented for using these patterns in the transformation languages.

In (Duddy et al., 2003), one declarative transformation language is presented based on the concepts of patterns and requirements of OMG RFP\(^2\). This pattern based transformation language contains transformation patterns, their relations and transformation rules. The introduced patterns are used to match source elements of transformations and present a template for target elements of transformations. So, presented patterns are used as constraints frame on the source and target elements that must be satisfied by a transformation. These patterns are the main section of the transformation specification format in the presented transformation language and are not applicable to any transformation languages.
Think Social Capital Before You Think Knowledge Transfer
www.igi-global.com/article/think-social-capital-before-you/2685?camid=4v1a

Virtual Communities of Practice: A Mechanism for Efficient Knowledge Retrieval in MNCs
www.igi-global.com/chapter/virtual-communities-practice/6961?camid=4v1a