Towards User-Centric Concrete Model Transformation

Iman Avazpour
Centre for Computing and Engineering Software and Systems, Swinburne University of Technology, Hawthorn, VIC 3101, Australia
Email: iavazpour@swin.edu.au

Abstract—Model transformations are the key part of Model Driven Engineering (MDE). However, their specification is not user-friendly, due to excessive use of metamodels and textual representation of transformation languages. This paper briefly describes the approach and tool under development for user-centric transformation generation using concrete model visualisations.

I. INTRODUCTION

Model transformation is a key part of Model Driven Engineering (MDE) which involves transforming models of different abstraction levels or based on different metamodels to each other, while preserving their intent. The process of generating such transformations starts with correspondence specification, i.e. users need to define which elements of the Left Hand Side (LHS - the source) and the Right Hand Side (RHS - the target) models correspond. These correspondence specifications are usually defined on metamodels of the participating LHS and RHS models, using textual (and in some cases graphical) representation of transformation languages. The way correspondences are specified creates a pragmatic barrier for many users (average modellers), since metamodels are not user-friendly artefacts and may get fairly complex [1], [4] and also, textual representations are often hard to maintain.

The cognitive effects of visual representations have been discussed previously [5]. Based on this, this research aims to provide a more human-centric view of correspondence specification by utilising user-defined, concrete model visualisations (as graphs, charts, diagrams, source code, etc.). Users then specify correspondences between these concrete visualisations using drag and drop of visual elements and transformation functions, and the actual transformation code is generated behind the scene. We also seek information retrieval techniques to provide recommendations to users as to which model elements on the both sides of the transformation are more likely to match, making it easier to comprehend large-scale models.

Previous research on trying to specify model transformation on concrete representations has mostly focused on specific domains [2]. Some techniques used concrete syntax in conjunction with abstract syntax (metamodels) to improve understandability of the abstract notation for the average users [1], [8], and others have been shifted towards by-example transformations [6], [7]. Unlike these techniques, the approach presented here is not limited to a certain domain and regardless of what abstraction level the input models are, the user is capable of defining visualisations for desired LHS and RHS models. Once the visualisation is defined, the system extracts its abstraction and analyses models for possible and likely correspondences so that users have the opportunity to choose from those correspondences or discard them.

II. CONVERT (CONcrete VISual ASSistEd TRANSFORMATION FRAMEWORK)

CONcrete VISual ASSistEd Transformation (CONVERT) is the framework and tool being developed for concrete model transformation. CONVERT contains an integrated collection of techniques to support specification and generation of model transformations in a more user-centric manner. Figure 1 depicts simple architecture of CONVERT. Solid arrows in the figure indicate user interaction, while dashed arrows indicate automatic procedures. Thick arrows in the middle demonstrate the direction of transformation which is model to visualisation, visualisation to visualisation and back. The following paragraphs describe parts of the system briefly through a transformation generation scenario.

In order to generate a transformation using CONVERT, users are required to provide example(s) of Source and Target models to specify the correspondences between visual elements and model context. The rendering of visual elements are predefined in the system. Therefore, this interaction will result in generation of transformation code to transform source and target examples to their corresponding in-house representation models (i.e. Visualisation). Figure 2 contains screenshots of CONVERT where visualisation of an input model (in this case an XML file containing sales records) is being generated in the form of a BarChart (a) and the final visualisation is ren-
A generic metamodel (Abstraction) is re-engineered automatically by the system using examples. Transformation Code generator (TCG) is responsible for generating the transformation code required and uses the correspondences and the re-engineered abstraction as inputs.

Once Visualisations are defined (e.g. in form of Charts, Source code, Boxes and lines, etc.), correspondence specification between visualisations will be performed with simple drag and drop of visual elements. The framework enables this by considering each visual element as a partial model which incorporates its own abstraction, forward and reverse transformation rule templates. These templates take shape as the user drags and drops elements. Figure 2 part (c) shows an example where a visualisation of a class diagram and java source code have been rendered. As an example, an operation in the class diagram can be dragged and dropped on a function in java code (showed by dashed red line) and their contained elements (showed in popups) can be dragged accordingly (depicted by solid green lines) to form a complete transformation rule. TCG records these correspondences and generates the required transformation code. Since the visual elements incorporate their own abstraction data, transformation code generation between visual elements in TCG follows the same routine as the transformation code generation for model to visualisation (there, the abstraction was reverse engineered from input models). Therefore, the defined correspondences and the abstraction are provided as inputs to TCG for transforming model examples-to-visualisation and visualisation-to-visualisation and back.

III. NEXT STEPS

CONVERT is capable of rendering a specific model with multiple visualisations (e.g. to visualise part of a system with charts and other parts with boxes and lines). For generating such visualisations, the user can utilise multiple source/target examples. Variety of visualisation elements including boxes, lines, code, etc. are defined in the framework. However, adding new elements requires their rendering to be programmed which reduces the flexibility of the system. A new version of the framework is being developed which benefits previous research ([3]) to improve framework’s flexibility. The same drag and drop concepts will be adopted to generate arbitrary visual elements to be used in visualisation of source and target model examples. At the same time, the scalability of visualisation to more complex model rendering is also being investigated.

A complete roundtrip transformation (model to visualisation, visualisation to visualisation and back) can be generated using CONVERT. However, it is limited to one-to-one element correspondences. Methods are being investigated to integrate transformation functions to handle many-to-many transformations. The future release of CONVERT will be capable of generating XML representation of transformation rules which will then be convertible to transformation languages of choice using the TCG.

A two stage user study will be conducted to validate the approach. Controlled experiment with a group of students, where a group of students familiar with basics of modelling will be briefed on transformation generation using CONVERT and one current framework (ATLAS and ATL, ALTOVA MapForce, etc.), and a more formal validation by practitioners.

IV. CONCLUSION

This research is towards providing an approach where the users have the freedom of using the visualisation of their choice for developing model transformations. At this stage, the introduced framework (CONVERT) is very limited and more capabilities will be integrated to be able to handle complex transformation tasks.

REFERENCES