Model Transformation Languages under a Magnifying Glass: A Controlled Experiment with Xtend, ATL, and QVT

Regina Hebig[1], Christoph Seidl[2], Thorsten Berger[3], John Kook Pedersen[4], Andrzej Wąsowski[5]

Abstract: In Model-Driven Software Development, models are processed automatically to support the creation, build, and execution of systems. A large variety of dedicated model-transformation languages exists, promising to efficiently realize the automated processing of models. To investigate the actual benefit of using such specialized languages, we performed a large-scale controlled experiment in which 78 subjects solved 231 individual tasks using three languages. The experiment sheds light on commonalities and differences between model transformation languages (ATL, QVT-O) and on benefits of using them in common development tasks (comprehension, change, and creation) against a modern general-purpose language (Xtend). The results of our experiment show no statistically significant benefit of using a dedicated transformation language over a modern general-purpose language. However, we were able to identify several aspects of transformation programming where domain-specific transformation languages do appear to help, including copying objects, context identification, and conditioning the computation on types.

Keywords: Model Transformation Languages; Experiment; Xtend; ATL; QVT

1 Introduction

In Model-Driven Software Development (MDSD), models are processed automatically to support creation, build and execution of systems. Transformations are, among others, used to compute views on models, to validate models, to refactor models as well as to interpret or otherwise execute models. We are specifically concerned with model-to-model (M2M) transformations, i.e., programs transforming instances of a source model to instances of a target model (structured data to structured data). Respective M2M transformation languages come with an implicit promise to be easier to use and more efficient for specifying transformations than general-purpose programming languages (GPLs). In this line of work, we investigated whether this promise holds.

[1] Chalmers | University of Gothenburg, Sweden, regina.hebig@cse.gu.se
[2] Technische Universität Braunschweig, Germany, c.seidl@tu-braunschweig.de
[3] Chalmers | University of Gothenburg, Sweden, thorsten.berger@chalmers.se
[4] IT University of Copenhagen, jkpe@itu.dk
[5] IT University of Copenhagen, wasowski@itu.dk

doi:10.18420/se2019-25

2 Method

We performed a pre-study in collaboration with a Danish industrial partner to investigate suitability of model transformation technology in the context of data aggregation. The results lead to the design of the subsequent experiment, where we had participants perform tasks to comprehend, change and create transformations for typical M2M scenarios. We selected ATL and QVT-O as M2M languages as well as Xtend as imperative GPL.

The experiment was performed in three runs at Chalmers | University of Gothenburg and Technische Universität Braunschweig. Each run of the experiment was performed in a lecture room where participants had enough space to work. Each participant had to use one of the languages, which we assigned to them arbitrarily beforehand, to perform the given tasks. We distributed task sheets in such a way that participants sitting next to each other would solve different tasks (i.e., different languages and/or different M2M scenario). The tasks had to be solved on paper, which eliminates the factor of familiarity with the programming environment. In total, we recruited 78 graduate students who participated in the experiment voluntarily.

3 Results

Analyzing solutions of 231 individual tasks, we found that:

- Handling multi-valued features (collections), recursion, and designing logical branching conditions are among the most difficult skills to master.
- Even well qualified subjects struggle to evolve transformations optimally, producing changes of widely diverse sizes.
- Implicit object creation/structure copying, support for type-driven computation and explicit computation context do appear to reduce the amount of errors.

Furthermore, the results of our experiment show no statistically significant benefit of using a dedicated M2M language over a modern GPL. However, we were able to identify several aspects of transformation programming where domain-specific transformation languages do appear to help, including copying objects, context identification, and conditioning the computation on types. Please refer to our paper for details on the experiment setup, conduction, results and conclusions [He18].

References

[He18] Hebig, Regina; Seidl, Christoph; Berger, Thorsten; Kook Pedersen, John; Wąsowski, Andrzej: Model Transformation Languages Under a Magnifying Glass - A Controlled Experiment with Xtend, ATL, and QVT. In: Proceedings of the 26th Symposium on the Foundations of Software Engineering (FSE). FSE'18, 2018.