Empirical Assessment of Advantages and Disadvantages of Model Transformation Languages

Stefan Höppner, Institute of Software Engineering and Programming Languages, Ulm University, Germany
Matthias Tichy, Institute of Software Engineering and Programming Languages, Ulm University, Germany

1 Context

Model-driven software engineering envisages the use of model transformations to evolve models. Model transformation languages, are touted with many benefits over general-purpose languages. However, most of these claims have neither been substantiated nor investigated thoroughly. Moreover, they frequently lack the contextual information required to critically assess their merit or build meaningful empirical studies around them.

2 Objective

The main objective of our work is to aggregate all necessary data to set up proper evaluation and use this data to asses the most prevalent claims about model transformation languages empirically. We aim to provide evidence on whether those claims withstand rigorous empirical scrutiny. We further want to provide a foundation of data upon which more empirical evaluation can be built.

3 Method

To address our objectives, we employ several research methodologies. Initially, we use a structured literature review to determine the state of research and what claims about quality attributes of MTLs are propagated in literature. The results serve as a basis for conducting semi-structured interviews to collect qualitative data on relevant factors and co-founding factors pertaining to the claims discussed. We quantify the identified effects using structural equation modelling and an online survey. Finally, we use repository mining and design science to collect and prepare artefacts. The artefacts are used in two separate case studies to empirically evaluate several MTL quality attributes based on the previously identified factors.

4 Results & Contributions

Using the previouslz described methodologies we provide results and contributions five areas which build on each other. In the following we give an overview over each contribution area.

4.1 Claims and Evidence in Literature

Literature associates MTLs with 15 different quality attributes [1]. Most quality attributes are associated positively and negatively with MTLs highlighting the intricate balance between useful abstractions and drawbacks thereof. Moreover, we believe that a portion of this variance can be attributed to the diverse range of languages, each with their own unique tradeoffs between accentuating capabilities and accepting drawbacks.

These results are contrasted by a bleak state of evidence for claims made about the quality attributes.

More than 70% of all claims found in our literature review lack any form of substantiation. Much of the evidence that is given stems from examples that demonstrate the claimed advantage or disadvantage in a single case. We were only able to identify four studies that use empirical methods. Moreover, only one of these studies focused on model transformation languages as the central artefact of evaluation.

Another concerning observation is, that citations are often used to reference other works that make similar claims rather than literature that provides proper substantiation.

4.2 Factors for Advantages and Disadvantages

We identified six main influence factors, namely *GPL Capabilities*, *MTL Capabilities*, *Tooling*, *Choice of Language*, *Skills* and *Use Case* [6]. Each factor comprises several sub-factors.

The influences of all factors are split into two groups, direct influences and context influences. We found that *GPL Capabilities*, *MTL Capabilities* and *Tooling* have a direct influence on perceived quality attributes of MTLs and *Choice of Language*, *Skills* and *Use Case* define context that moderates the type and strength of influence of the other factors.

The effectiveness of MTLs depends on their capabilities, which can vary based on the language in question and whether the abstractions provided are useful for the specific case, i.e., bidirectional support is only useful for bidirectional transformation cases. Additionally, the developer's skill plays a crucial role too. If they are unable to effectively utilise the capabilities of a language due to inexperience or lack of knowledge, none of the advantages might apply. Finally, the tooling available for MTLs can either enhance or impede their capabilities. This again depends on the language and use case as tooling may or may not exist.

4.3 Quantification of Influence Weights of Factors

Contrary to hypothesis formulated based on our previous results, moderation effects are nearly as nuanced as the direct influences of MTL Capabilities. The size of meta-models, for example, moderates the influence on *Comprehensibility* and *Ease of Writing*. But the strength of this moderation differs immensely between different MTL Capabilities. Overall, we found the transformation size to be the most important moderating factor.

The insights gained from our quantitative work [4] help to provide clear suggestions on further actions for language developers, researchers & developers to take. For further language development, we suggest to focus on the development of transformation specific reuse mechanisms. This is because of the surprisingly high importance of reuse mechanisms for several MTL quality attributes. We also believe that such features provide a unique selling point for MTLs.

For further empirical evaluation, we suggest investigating the cost of reimplementing MTL abstractions in general purpose languages. Most prominently the cost of manually handling traces. This also includes an assessment of how much tracing is required in real-world use cases to allow for a proper cost-benefit analysis.

4.4 The Suitability of ATL for Expressing Model Transformations

By quantitatively investigating ATL transformations we draw a number of conclusions on the suitability of the language for writing model transformations [2]. First, over half of the complexity of a transformation resides within bindings, meaning that over half of the effort spent is spent on assigning values to the output model. This leads us to draw the a similar conclusion as Hebig et al. [3]. Conditioning on types, as ATL does it, provides a well suited abstraction for model transformation development.

We also found, that the majority of bindings in ATL map one attribute of an input model element to one attribute of an output model element. This suggests that the main effort in writing ATL transformations stems from defining how the output should look like.

Overall our results show, that ATL provides several useful abstractions and shifts the focus of transformation development onto the definition of transformation logic. Only little complexity resides in describing how the transformation should be executed or how elements should be selected. Therefore, the expressiveness of ATL for the investigated transformations is high.

4.5 A Historical Perspective on ATL Versus Java Based on Complexity and Size

When comparing the complexity of the transformations written in Java SE14 and Java SE5 [5], we found that both the WMC and lines of code are greatly reduced in Java SE14. However, no significant changes in the number of required words exist. We attribute this to newer language features in Java that reduce the amount of explicit control flow one has to write. These features enable a more functional, data flow driven style of coding which leads to fewer, but much wider lines of code.

Surprisingly the distribution of complexity over the different transformation aspects reveals only slight improvements in Java SE14. In both language versions large portions of the code are overhead produced by manually implementing tracing, model traversal and setup.

We can also again highlight the overhead entailed when using Java. While in ATL over half of all code is used for defining bindings, in Java it is only about 25% as much.

Overall our results show, that new language features in Java SE14 now enable a style of writing transformations with significantly less cyclomatic complexity. At the same time, it is still impossible to hide those transformation aspects that ATL abstracts away from properly.

5 Conclusion

Our work contributes much needed systematisation and empirical ground work to the body of knowledge on model transformation languages.

We demonstrate that empirical evaluation of model transformation languages is feasible and necessary. Efforts to provide more empirical substance need to be undergone, and lacklustre language capabilities and tooling need to be improved. The results of this thesis can provide a basis for these further actions.

References

[1] S. Götz, M. Tichy, and R. Groner. “Claimed advantages and disadvantages of (dedicated) model transformation languages: a systematic literature review”. *Software and Systems Modeling* 20.2 (2021), pp. 469–503.

[2] S. Götz, M. Tichy, and T. Kehrer. “Dedicated Model Transformation Languages vs. General-purpose Languages: A Historical Perspective on ATL vs. Java”. *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,* INSTICC. SciTePress, 2021, pp. 122–135.

[3] R. Hebig et al. “Model Transformation Languages Under a Magnifying Glass: A Controlled Experiment with Xtend, ATL, and QVT”. *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* ESEC/FSE 2018. 2018.

[4] S. Höppner and M. Tichy. “Traceability and reuse mechanisms, the most important properties of model transformation languages”. *Empirical Software Engineering* 29.2 (2024), pp. 1–55.

[5] S. Höppner, M. Tichy, and T. Kehrer. *Contrasting Dedicated Model Transformation Languages vs. General Purpose Languages: A Historical Perspective on ATL vs. Java based on Complexity and Size: Supplementary Materials.* 2021.

[6] S. Höppner et al. “Advantages and disadvantages of (dedicated) model transformation languages”. *Empirical Software Engineering* 27.6 (2022), p. 159.