

# Analysis of Model Transformations Workshop Summary

Jürgen Dingel  
Queen's University  
Kingston, ON, Canada  
dingel@cs.queensu.ca

Levi Lúcio  
McGill University  
Montreal, QC, Canada  
levi@cs.mcgill.ca

Hans Vangheluwe  
McGill University and  
University of Antwerp  
Montreal, QC, Canada and  
Antwerp, Belgium  
hv@cs.mcgill.ca

Dániel Varró  
Budapest University of  
Technology and Economics  
Budapest, Hungary  
varro@mit.bme.hu

## 1. INTRODUCTION

To facilitate the processing and manipulation of models, a lot of research has gone into developing languages, standards, and tools to support model transformations – a quick search on the internet produces more than 30 different transformation languages that have been proposed in the literature or implemented in open-source or commercial tools. The growing adoption of these languages and the growing size and complexity of the model transformations developed require a better understanding of how all activities in the model transformation lifecycle can be better supported.

The AMT (Analysis of Model Transformations) workshop aims to address this issue by providing a forum in which the analysis of model transformations to support the development, quality assurance, maintenance, and evolution of model transformations is studied. The adoption of existing analysis techniques and tools developed, e.g., in the context of general-purpose programming languages and source code transformation are of particular interest, but also the identification of analysis challenges and solutions specific to model transformations or certain classes of model transformation languages.

This year we organised the first edition of AMT, having accepted 8 papers by authors from 12 countries. The program was organised in three sessions, namely: *intents of model transformations and model transformation chains*; *model transformation testing and debugging*; and *quality of model transformations*. In an additional session Dániel Varró discussed the certification of model transformations.

The last one and a half hours of the workshop were dedicated to the discussion of topics of interest to the audience. Four main topics were identified: *the relation between the analysis of model transformations and program verification*; *the debugging of model transformations*; *requirements*

*for model transformations*; and finally *the certification of model transformations*. The audience and workshop organisers formed four groups to discuss these topics, each group being composed of three to five members. In the remaining of this document we present the conclusions reached by each of those groups.

## 2. THE RELATION BETWEEN THE ANALYSIS OF MODEL TRANSFORMATIONS AND PROGRAM VERIFICATION

The first question raised during the discussion concerned the applicability of program verification methods to the analysis of model transformations. A first informed opinion from the members of the group was that the analysis of model transformations cannot in general reuse program verification techniques. This is due to the fact that the analysis of model transformations is typically (but not exclusively) concerned with properties of the relations between those transformations' input and output models, whereas program verification is rather concerned (but not exclusively) with properties of those programs' executions (e.g. that certain execution conditions are never violated or that they are always reached). As such, and given the difference between the relational and imperative computing paradigms used respectively in model transformations and programs it seems that: albeit some program verification techniques may be reused for the analysis of model transformations, specific techniques for the analysis of model transformations must be developed.

Given that a member of the discussion group works for Microsoft Research, the discussion diverged slightly on how the analysis of model transformation can help in the certification of software. It is the case that in current development practices at Microsoft Research certain data (or model) translators are implemented using simple rewriting rules. Because such rules can be seen as model transformations, the analysis of model transformations might be used to certify those parts in a software development process. There are seemingly many such opportunities of applying simple model transformation analysis techniques to parts of software development practices. This might mean that model transformation analysis techniques might contribute

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AMT '12, Oct 01 - October 05 2012, Innsbruck, Austria  
Copyright 2012 ACM 978-1-4503-1803-7/12/10 ...\$15.00.

in a non intrusive way to the certification of the software currently being built. It was also mentioned that software engineers should be the ones to identify the usefulness of such approaches within their software development work.

### 3. DEBUGGING OF MODEL TRANSFORMATIONS

When debugging a model transformation, we can distinguish between debugging the model transformation itself and debugging its derived artifacts such as the generated code. In general, debugging a model transformation is easier if the model transformation specification is interpreted directly. There are several analogies between debugging programs and model transformations:

- Inspecting/modifying a program state: the state of an model is probably more complex than the state of a program. In particular, in a model transformation models and graphs are used, rather than plain variables. However, it would seem that the general debugging principle remains the same;
- Stepwise execution: the main question regarding stepwise execution is how fine-grained a single step is. Unlike in debugging programs, we are not interested in single lines of code but rather in, for example, matching an element, creating/deleting an element, or performing an action on the level of the control flow of the model transformation;
- Breakpoint: the first question regarding breakpoints is: where can a breakpoint can be attached in a model transformation? If we consider a breakpoint attached to a model transformation rule, that might mean the execution should be paused when any of the components of a rule is executed (e.g. the *match* rule component, the *apply* rule component, the *application condition* rule component). On the other hand, applying breakpoints to elements of operational model transformation languages (such as Kermeta [1]) is probably more straightforward. It is also imaginable to specify breakpoints without locations but with a condition that triggers the breakpoint, when the condition is satisfied.

### 4. REQUIREMENTS FOR MODEL TRANSFORMATIONS

The discussion centered on the notion of “intent” of a model transformation. This notion was presented at the workshop as *a description of the goal behind the model transformation and the reason for using it*. The group observed that the idea of “intent”, is more related to “early requirements” or “goals” in “goal-oriented” software engineering, than to (formal) requirements specifications. The differentiator is the fact that it is a human/developer who has the intent, whereas specifications seem to be related to artifacts. In light of this, a look at the literature on early requirements and goal-oriented requirements engineering to see where model transformation intents fit in seems warranted. The group then further observed that each model transformation serving a particular intent typically is only a single step in principle bringing the developer closer to a

bigger goal. In that sense, choosing model transformations based on intents and goals is akin to an AI planning activity.

### 5. CERTIFICATION

The certification of model transformation is necessitated when such transformations are used in critical applications. Recently, the output model derived by an automated model transformation developed at TU Berlin was certified as part of a collaborative project, and now it is deployed on a satellite. Certification of a transformation requires to develop justified evidence that the transformation is free of flaws. This frequently involves the combined use of extensive and systematic testing activities and formal methods for the analysis of transformations. Another important aspect is to guarantee end-to-end traceability between the different design artifacts (high-level requirements, low-level requirements, software architecture, source code). Certification aspects of model transformations are being investigated in depth in the CERTIMOT project. Since certification is a complex and time consuming activity, a possible way to proceed is as follows:

- Step 1: certify output models derived by model transformations
- Step 2: certify the use of standalone model transformation programs (plugins) for a given source model
- Step 3: certify standalone model transformation programs for any source model
- Step 4: qualification of the code generator for the transformation plugins

The main conclusion the discussions on this topic was that, instead of addressing the full qualification of a complex model transformation engine, first it can be advantageous to address the certification of individual (standalone) transformation programs derived from a specific set of transformation rules.

### 6. CONCLUSION

The field of analysis of model transformations is in its infancy. However, because of the capability model transformations have to abstractly define complex computations while dealing with domain specificity, explicit model transformations have a role to play in modern software development. If sometimes the computational abstraction granted by model transformation languages allows classical reasoning mechanisms to be applied to model transformations analysis, the richness and diversity of model transformations poses new interesting challenges when compared to program or model verification. AMT has attracted a considerable amount of attention in its first edition, both from established researchers in the field and from the MoDELS audience. As described in the AMT’s group discussion conclusions, the research challenges in the area are varied and scientifically challenging. We expect they should be intensively tackled in the coming years.

### Acknowledgements

We wish to thank all the participants in the group discussions for their ideas that shaped this summary: Moussa Amrani, Fabian Büttner, Ethan Jackson, Eugene Syriani, Claudia Ermel, Stephan Hildebrandt ...

## **7. REFERENCES**

- [1] P.-A. Muller, F. Fleurey, and J.-M. Jézéquel. Weaving Executability into Object-Oriented Meta-Languages. In *MODELS*, pages 264–278, 2005.