

CHALLENGES FOR ADVANCED DOMAIN-SPECIFIC MODELING FRAMEWORKS

István RÁTH
Advisor: Dániel VARRÓ

I. Introduction

In this paper, I identify main challenges for domain specific modeling frameworks. *Arbitrary abstract to concrete syntax mapping* enables the toolsmith to create visual representation to complex logical models with the possibility to determine what detail to show and what to hide in a flexible way. *Interactive model simulation* based on a model transformation architecture allows for the design time analysis and precise fine tuning of behavioral aspects.

The main goal of language engineering is to integrate the following aspects into a coherent framework: (i) *abstract syntax*, which specifies how logical model elements are stored and represented in the modeling environment; (ii) *concrete syntax*, which specifies the graphical representation of models; (iii) *well-formedness rules*, to describe static and language-specific constraints; (iv) *dynamic (operational) semantics* which describe the behavioral attributes of the system-under-design; and (v) *transformations* which provide formal and declarative support for creating generative bridges between various modeling domains.

Many state-of-the-art tools, such as MetaEdit [1], Microsoft's DSL Tools [2], openArchitectureWare [3] or the upcoming Graphical Modeling Framework [4] solve the traditional problems of custom modeling (the first three aspects) in an elegant fashion. However, there are multiple key areas where existing DSM frameworks typically fall short. The problem of *abstract to concrete syntax mapping* arises from the fact that abstract syntax models tend to be too complicated and difficult to handle for humans. *Interactive model simulation* is another important language engineering aspect which has been lacking support in traditional modeling tools.

With the *ViatraDSM Framework*, our main goal was to push the limits of tool integration further, in order to ease and speed up the construction of rich domain-specific visual languages. The ViatraDSM framework is tightly integrated with the VIATRA2 [5] engine, and relies on its modelspace containment and model transformation facilities to provide support for the evaluation of *well-formedness constraints*, *model simulation* and *interdomain transformations*.

II. Arbitrary abstract syntax to concrete syntax mapping

In constructing domain-specific visual languages, the clear separation of the logical modeling layer and the visualization layer is important. Abstract syntax models tend to be quite complicated, especially if they are tailored to the needs of the code generator, rather than the user. Therefore, the visualization layer needs to be independent, because that way the language engineer is free to construct diagrams that are easy to handle and do not confuse domain experts.

It is important to note the difference between the *model level* and the *metamodel level* separation of logical models and diagrams. On the *model level*, most tools separate the two layers by giving the user (almost) complete control over what parts of the logical model are visualized. The *metamodel level* separation, enables the language engineer to define visualization independently. Logical and diagram metamodels are constructed separately. This means that the GUI-driven editing only affects the diagram models directly, and the required changes to the logical model are generated by a mapping interface.

Our approach to map logical and diagram models, *bidirectional mapping*, makes use of the fact that the user can only apply a limited and well-defined set of modifications to the diagram through the graphical interface. Therefore, this approach directly maps the editing actions of the user to the logical model.

III. Interactive model simulation

Since the ViatraDSM framework is built on the VIATRA2 model transformation infrastructure, interactive model simulation can be implemented more easily. Ongoing research focuses on the following areas:

- Guided simulation using a *step-based execution*. This can be implemented with parametrized, single-step transformation rules. In that case, the toolsmith constructs a model transformation which executes a single transformation step. The framework runs these transformation steps, with support for user interaction at non-deterministic choice points.
- *Debug-style simulation* with support for more precise control over the transformation process. This can be regarded as an evolutionary continuation of the step-based execution approach. From the point of view of the ViatraDSM framework, the most challenging aspect of this approach is to provide a domain-specific interface to the debugger.
- General *trace representation* to preserve the different states of the model during simulation. Tracing is an important aspect of model simulation, because it provides useful feedback on simulation runs. With a trace, the various states of the system and the non-deterministic choices can be regenerated at a later time.

This research topic is strongly connected to those of debug-style simulation, because traces should ideally have a domain-specific interface as well.

IV. Conclusion

To sum up, current research is active in the following areas:

1. Arbitrary model-to-diagram and diagram-to-model mapping based on a declarative specification, to facilitate the metamodel level separation of diagrams and logical models, and provide a flexible and transparent bidirectional mapping between them.
2. Design time interactive model simulation and constraint evaluation using transparent model transformation techniques, with support for the generation of traces. A domain-specific interface to the generic transformation engine is in the planning stage.

References

- [1] Metacase, “MetaEdit+,” URL: <http://www.metacase.com/mep/>.
- [2] Microsoft, “DSL Tools,” URL: <http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/default.aspx>.
- [3] M. V. Bernd Kolb, “openarchitectureware and eclipse,” URL: <http://architectureware.sourceforge.net/data/oawEclipse.pdf>.
- [4] The Eclipse Project, “Graphical Modeling Framework,” URL: <http://www.eclipse.org/gmf>.
- [5] *VIATRA2 Framework*, An Eclipse GMT Subproject, URL: <http://www.eclipse.org/gmt/>.