Quality-Driven Software Architecture Model Transformation

Mari Matinlassi
VTT Technical Research Centre of Finland, P.O. Box 1100, 90571 Oulu, Finland
Mari.Matinlassi@vtt.fi

1. Motivation

In order to achieve productive software development with high quality software products, skilled people and effective tools are needed. The concept of quality-driven software architecture model transformation (QAMT) is concerned with the latter, tools. The ultimate goal of QAMT is to multiply the productivity of individual software modellers with advanced modelling tools. Advanced modelling tools can be used to do most of the work on behalf of the developer. However, QAMT is not a tool, but rather a technique for transforming architecture. Transforming architectural models to new models according to defined quality requirements (i.e. manual QAMT) is likely to be a common routine for developers in the software industry. In an ideal state, QAMT would be fully automated in a tool, guiding the software (PF) architect in his/her work, thus making the work faster and easier.

2. Example

Assumptions: (1) software (product family, PF) architecture is available as a platform independent model [1] in the CASE tool. The existing model is called source model and its structure is designed by respecting certain quality attributes. (2a) At least one of the attributes or requirements for the architecture model has been changed or modified, and therefore an architectural model transformation is required. Or, (2b) there is a need to derive a new architecture for another PF member with different quality requirements.

2.1. Where in the model is transformation needed?

The architect evaluates the source model against the quality requirements set for the target model. This evaluation step in the process will be semi-automatic. The architect may, for example, search for all the design patterns or architectural patterns utilized in the source model. The search may also be constrained with search parameters (e.g. find patterns promoting extensibility in structural diagrams).

As the result of this step, the architect identifies the potential parts of the architecture that require transformation in order to meet the requirements set for the target model.

2.2. What should the target model be like?

The architect identifies the potential candidates for target model architecture. Similarly to searching the model in the previous step, the architect may search the solution database directly, for instance, for the following:

- Are there any allocation patterns available that support modifiability?
- What pattern(s) would be suitable for the problems of extensible architecture?
- Do the patterns found in the model allow being transformed into something else?

This way, the tool will guide the architect in the decision-making process, enabling an easy, semi-automatic way of trying various approaches to the problem.

The architect then makes the decision about transforming the model. Here, the architect selects the pattern or patterns requiring to be transformed while also making the decision about suitable target patterns. Although the decisions are guided by the information available in the tool, the architect is, in the end, responsible for the final decision.

2.3. Transforming the architecture

After the architect has made the decisions, the QAMT will be performed automatically by pushing a button. The tool will apply the transformation rules automatically. In addition to employing transformation rules, the tool uses source model data and target model data in the automation of the transformation.

3. Results and challenges

In order to automate QAMT we need:

A Stylebase - Knowledge on existing solutions to architectural problems (i.e. architectural patterns and design patterns) captured in a way that they can be understood both by the tool and the developers.
A Rulebase - Knowledge on transformations, including the assumptions, limitations and exact rules for performing transformations in a model. The assumptions and limitations are designed for guiding people in the decision process, whereas the transformation rules are for the tool.

A tool - An advanced CASE tool combining the software (PF) architecture model, stylebase and rulebase with a comfortable user interface.

3.1. Stylebase

The function of the stylebase is to describe architectural styles/patterns and design patterns explicitly enough to be utilized by the tool [2]. Currently, the data included in the stylebase consists of: pattern name, reference, abstraction level, diagram, purpose, quality attribute, component type, component role, connector type, data topology, control topology, guide and figure. Currently, a total of 11 architectural patterns/styles are described in the stylebase.

The task of developing a stylebase including all the relevant information required in transformation is not a simple one. For example, the quality attribute is an essential parameter in the stylebase, while it is also a highly ambiguous one. Furthermore, there is no explicit list available on quality attributes nor any explicit knowledge about which attribute(s) are promoted by the different styles. Therefore, the quality support information used in this study is only one possible solution to the problem.

3.2. Rulebase

Transformation rules are the units in which transformations are defined. A rule is responsible for transforming a particular selection of the source model to corresponding target model elements. In the rulebase, two kinds of rules are needed. First, there are rules for constraining transformations, e.g., transformation may be allowed only between patterns at the same level of abstraction and with the same purpose. These rules guide the user when designing the target model. Second, there are more specific transformation rules, e.g., concerning how to transform pattern A to pattern B. The specific rules need to be defined carefully in order to make them best serve the tool in executing transformations. Currently, the rulebase includes transformation rules for one experimental transformation: layers to blackboard.

3.3. Tool extension

First, an evaluation of suitable modelling tools was done [3]. Then, a tool prototype (Q-Tra) was designed and implemented on top of Telelogic Tau/Developer [4]. The architect uses Tau for modelling and Q-Tra for searching the model, searching the stylebase and executing the transformation. The first implementation trial of the stylebase was carried out on a set of objects in a linked list, after which the stylebase was implemented with My-SQL database. Q-Tra combines the information attached in the components of the model and the information captured in the stylebase. It searches the model for components matching the search parameters and provides the search results to the user. Similarly, Q-Tra provides an application form window for stylebase management. The rulebase is implemented in a similar way as the stylebase.

Our first attempts at integrating QAMT and the current CASE tools during the last months have not been easy due to the evolvement of the modelling language (UML) and the tool vendors' divergent representations of the standard.

4. Conclusion and future research

The experiences made with the tool prototype have shown that QAMT is a viable concept. The development of the tool prototype has given us a lot of practical insight into developing the transformation itself.

Future research topics on the subject may include extending the transformation technique for model dynamics in addition to model structure, and adapting the technique for transforming more detailed modelling patterns, i.e. design patterns.

The technique could also be utilized for the definition of several different transformations, which would enable an abstraction of a generic model of QAMT. Finally, if a viable standard transformation description language emerges, the generic model could be described using the QVT approach (Queries/Views/Transformations). A generic, standard model, which is independent of the transformation implementation platform, would enable easier automation of transformation. This would enhance the role of the tool in the transformation and thus bring the transformation closer to MDA.

References