On the Efficiency of Text-based Comparison and Merging of Software Models

Ferenc Attila Somogyi

The importance of model-driven engineering in software development has been steadily increasing in the last few years, as models are more often used in the industry for various purposes. Models can be described and edited via either a graphical or a textual interface. The graphical approach usually makes the model more easily readable, but writing efficiency does not scale well with model complexity. It is the more convenient and common approach to use, especially if the model has to be interpreted by third-party users as well. On the other hand, the textual approach is rarely used compared to the graphical one. It can be harder to read at first (especially for third-party users), but writing efficiency scales well with model complexity. Generally speaking, using a textual representation to describe and edit the model pays off when the model is substantial in size or complexity, especially if there are efficient development tools available for the editing process. The availability of such tools results in the editing process being more similar to traditional source code-based development. During source code-based development, teamwork is supported by version control systems that help manage the different versions of source code files. The same idea can be applied to models and their textual representations in order to facilitate teamwork during model-driven engineering. In previous work [1], we briefly presented a method that can be used to compare and merge the textual representations of software models. We also compared this method to other, already existing model comparing or merging approaches. In addition to the raw texts, the method uses the abstract syntax trees (AST-s) built from the texts as basis of the comparison. This makes the comparison a lot more accurate, as AST-s give more accurate information than the raw texts. The textual representations are also needed though, as discovering some conflicts require textual comparison in addition to comparing the AST-s. The method is not dependent on any particular modeling environment or formal language used to describe the textual representations. This is achieved by demanding certain operations from the parser of the language. Thus, the method can be the foundation of conflict handling in version control systems that support the textual representations of models. The main difference between the presented method and other, specialized model comparing and merging approaches (like EMF Compare) is generality, meaning that most approaches only work with a specific modeling environment. Another difference is the fact that the presented method aims to compare the textual representations instead of the structure of the model. This results in more overall accuracy while still maintaining generality (with certain restrictions). In this paper, we examine the different steps of the method in detail with the focus being on efficiency and performance. These are important considerations if the method is to be used in real version control systems. We also consider and review alternatives to different steps of the method by examining their advantages and disadvantages.

References

[1] F.A. Somogyi. Merging Textual Representations of Software Models, *MultiScience - 2016. microCAD International Multidisciplinary Scientific Conference*, under publish.

[2] EMF Compare. *https://www.eclipse.org/emf/compare/*.