Model transformations on the Preprocessor Metamodel - Graph Transformation approach

László Vidács

There is an important trend in software engineering that focuses on development methods where models are used besides concentrating on source code only. In previous work we designed a metamodel for C/C++ preprocessing (Columbus Schema for C/C++ Preprocessing [1]). The metamodel describes the source code from the preprocessor's point of view and enables the studying the preprocessor constructs in detail.

Refactoring techniques are considered as promising means for software development. There exists extensive literature on refactoring object-oriented programs. Refactoring on C++ code is supported by a variety of tools. Preprocessor means obstacle for these tools. Since preprocessing loses information about macros and different configurations, it is not enough to work on the preprocessed code. Tool developers face the problem of having two different languages: C/C++ and the preprocessor language. There are several research papers about solving this problem, but the aim of these contributions is to implement only the C/C++ refactorings. A recent work of Garrido [2] combines directive usages into the C language. The contribution contains several preprocessor related refactorings, which are not related to the C language itself but to the preprocessing directives. Refactorings made on directives may be not so complicated as language dependent refactorings combined with directives. But they may have important role in refactoring real C/C++ programs. However, refactorings are usually considered from the implementation point of view only, not from a conceptual view.

A conceptual view on preprocessor refactorings on the basis of metamodels and graph transformations allows to express the properties of refactorings more precise. A conceptual view also opens the possibility for viewing refactorings on the semantical level. In this contribution we provide transformations on preprocessor constructs (like adding a new macro parameter) as graph transformations. We use a single pushout approach providing the left-hand side and right-hand side of the transformation. The transformed graph is a directed, labelled and attributed graph. It is a program graph according to the UML class diagram of the metamodel. To be more general we extend the usual notation with the concept of multi-nodes. It is common to use NACs (Negative Application Condition) to prevent application of a graph transformation rule, but we chose to use OCL expressions instead, because in program transformation context the application conditions are as complex as the structural changes in the graph. OCL is appropriate not only to formalize conditions but to check them in a program graph.

To validate this approach we choose the USE system instead of existing graph transformation engines. USE can handle UML metamodels and models, this concept fits to our existing program representation. The graph transformation rule is given as a USE description and converted into basic operations on the model instance. The USE system evaluates OCL expressions to check the application conditions and performs the operations. The time consuming part of a transformation is to find the appropriate place in the graph where it is applicable. In case of refactoring the user has to determine the place of the transformation, in our implementation the place is a parameter of the transformation rule.

Our approach uses the result of graph transformation theory in software (re)engineering. Transformations on preprocessor constructs are not well studied yet. In this paper they are formalized using a higher level representation which also enables condition checking. The metamodel-model approach of the USE system provides a flexible framework to manipulate graphs and to visually check the transformation rules. Besides model to model transformations the reverse engineering tools of the Columbus system give the possibility of source code to source code level transformations.

References

[1] L. Vidács, Á. Beszédes, and F. Rudolf. Columbus Schema for C/C++ Preprocessing, *CSMR*, IEEE Computer Society, Tampere, Finland

[2] A. Garrido. Program Refactoring in the Presence of Preprocessor Directives *Ph.D. thesis, UIUC* 2005