Incremental
Design Pattern Recovery

(in german)

Jörg Niere

Dissertation abstract

During the last decade the size of software systems has increased drastically. To keep the costs small, changes were made only in the implementation of the system without adapting the documentation correspondingly. The maintenance of such systems is extensive through the not available or in part erroneous documentation. Furthermore, changes have unexpected side effects in a multiple way, which increase the costs for the changes additionally. Therefore, it is necessary to recover the documentation from the implementation of the software system.

The analysis of a implementation for the recovery of its documentation has put out that homogeneous implementations exist for recurring problems. These problems with corresponding solutions are known as design patterns and are used among other things to document a software system. The description of a design pattern is informal, which results in a high number of implementation variants. Automatic analyses for the recovery of design pattern instances are not suitable due to the high number of implementation variants and the size of the software systems with a hundred thousand or million lines of source code.

This thesis presents an interactive analysis process for the recovery of design pattern to document a software system. The approach is based on graph transformation rules in combination with fuzzy sets. Each result of the analysis gets a precision value calculated from the accuracy values of a rule. An incremental rule application algorithm produces early relevant results. Especially, during the analysis of large software systems, a reengineer can adapt rules where necessary due to the early results, which reduces the overall analysis time by preventing ineffective analysis. A reengineer can also insert hypotheses and additional information besides the source code.

The presented approach is not limited to the recovery of design patterns, but it can be applied to recover also implementation patterns, deployment patterns, architectural patterns or patterns stemming form pattern languages.