Pattern-based Re-Engineering of Software Systems

Abstract

Software has to be constantly re-engineered due to changes in its environment and to fulfill new requirements. Thus its maintainability is very important. This thesis introduces an approach to support re-engineering activities in (object oriented) software systems by assessing and improving its maintainability based on software patterns.

An existing approach for the automated recognition and rating of design pattern instances in source code is extended to also recognize design defects, i.e. instances of bad smells and antipatterns. In order to improve recognized design defects, this thesis develops a graphical language for the specification of program transformations.

Usually, the goal of such a transformation is to improve the structure of a software, leaving its observable behavior unchanged (refactoring). In order to show that certain aspects of behavior are not changed by a transformation, this thesis develops an automatic verification of transformations to fulfill user defineable criteria. Such criteria allow to specify structural properties of a program, that have to be preserved or must not be created by a transformation. The verification tries to prove the criteria to be inductive structural invariants across different versions of arbitrary programs that may be created by the execution of complete transformations. The specification language has been especially designed to facilitate complex program transformations as well as their automatic verification.

If a transformation has the potential to violate a criterion, the verification systematically computes counter examples showing problematic executions and presents them to the re-engineer to help him correct the specifications.