Abstract

Mutation Testing is an important technique for evaluating the quality of a test suite. The technique involves inserting small changes to the source code of a program to produce a set of altered programs, each of which is called a “mutant”. The test suite in question is considered incomplete when it could not detect the changes made in those mutants. As a result, the findings from mutation testing may lead to a better, improved test suite. The fundamental idea of mutation testing is to imitate the malfunction that may occur in the program, such as using incorrect mathematical operators, incorrect variable names and incorrect reference types of the object.

This research studies eight mutation operators for OCL (Object Constraint Language). These operators are used to create mutants from OCL specifications in the subset of the UML (Unified Modeling Language) standard. These operators are examined in two aspects: (1) the assessment of test coverage of the test suite according to the mutation operators and (2) the performance of the eight mutation operators when applied in a real world specification. Results from the case study used here indicates that the operators can produce mutants effectively where 67% of which are meaningful.