論 文 の 内 容 の 要 旨

論文題目　Efficient Mutation Analysis for Industrial Software
（産業用ソフトウェアに対する効率的なミューテーション解析）

氏　名　徳本 晋

Industrial software is developed and maintained by many people with different experience and skills within a limited time and cost. It is known that the lack of experience and skills of developers greatly affects the number of defects in the software. In addition, because of the rapid turnover of developers in companies, there are many legacy systems that do not have quality assurance and cannot be identified by whoever created them. Due to the lack of experience and skills of the developers, they may make changes to the parts of the software that are not the root cause of the bugs, and as a result, if the bugs are not fully fixed, multiple defects may occur. Similarly, if the cause of a bug in a legacy system is located in a place that cannot be changed, it must be dealt with by making changes in a place that can be changed, but again, if the bug is not fully fixed, multiple intertwined defects will occur. If these defects are not detected before the release of the software, there is a possibility that significant social damage will occur. One of the best known techniques for detecting complex defects is mutation analysis. It is a method to measure the ability of a given test to detect bugs that are artificially embedded by mutating the elements of the program (mutation). Mutation analysis is not only used to measure the bug detectability of a test; it also has a wide range of applications, such as high-precision fault localization and automated program repair. Higher order mutation analysis, which mutates multiple locations simultaneously, is a technique that has greater potential for detecting complex defects than ordinary mutation analysis. However, higher order mutation analysis generates a large number of mutants (mutated programs), and all of them need to be compiled and tested, which is a problem that results in a very long execution time. Therefore, higher order mutation analysis

has not been widely used in industry, despite the fact that strong applications have been devised. We propose, implement, and evaluate a high-speed higher order mutation analysis method and a mutant optimization technique to improve execution costs to enable the application of higher order mutation analysis to industrial software.

For the high-speed higher order mutation analysis method, we use four techniques to achieve improved execution time for mutation analysis: metamutation, virtual machines (VMs) for mutation, runtime mutation application, and high-order stream split execution. First, to avoid losing the information in the source code to the intermediate representation by, e.g., compile-time optimization, the program elements are replaced by a function called the metamutation function, which indicates the mutation position and type before compilation. This enables the mutation of intermediate expressions to match the mutation to the source code. Second, by running it on a VM, we reduce the overhead by starting a process one time instead of running it per test. Third, rather than rewriting the source code for each mutant, we use the mutant information for each execution state to translate the instructions into mutated ones when executing the intermediate expression. This technique can shorten the compilation time because it only needs to be compiled once. Fourth, while preserving the execution state on the VM, at the time of execution of each instruction, it branches into a state that executes the original instruction and a state that executes the mutated instruction. This makes it possible to shorten the test execution time. We conducted comparative experiments, which indicate that our method is significantly superior to an existing tool, an existing technique (mutation schema generation), and no-split-stream execution in higher order mutation.

For the mutant optimization, we analyze the limits of the reduction of mutants with- out loss of reliability. Existing methods remain challenged in terms of excessive mutant reduction and errors in the mutation score after reduction. The results of evaluation using open source software (OSS) show that the greedy mutant selection method reduces the execution time by approximately 40%, although the reduction is inferior to that of the existing method. To evaluate the impact of the reduction of excess mutants, we measured the mutation score for the test in which the bug detectability was artificially reduced, and the discrepancy in the mutation score of the proposed method was less than that of the existing method.

Leveraging the high-speed mutation analysis foundation, we improve the efficiency of the fault localization technique as an application of mutation

analysis. Fault localization is a technique to reduce the cost of debugging by ranking candidate fault causes based on test results and test execution information. Among the several fault localization techniques, mutation-based fault localization (MBFL) can localize faults with high accuracy but has the problem of high execution cost. Meanwhile, in mutation analysis, it is known that the statement deletion mutation operator has less bias in mutation points and is as effective as using all mutation operators even when used alone. Therefore, we implemented MBFL using only the statement deletion mutation operator (SDL-MBFL) and evaluated the localization of the software used in the actual product and nine actual faults. As a result of the evaluation, SDL-MBFL found more faults than existing methods in the higher ranks of 100 or more.

The overall evaluation of this study in terms of the application of mutation analysis to bug detectability measurement is that it contributes to methods to improve speed and accuracy, and has an advantage over state-of-the-art techniques, especially in higher order mutation. In terms of the application of mutation analysis to fault localization, this work is superior to the state-of-the-art in granularity and reliability. These contributions will greatly reduce the cost of performing higher order mutation and enable the detection of complex defects in industrial software.