論 文 の 内 容 の 要 旨

論文題目　Automatic Verification and Testing for Software with Multiple Versions
-- Improving Software Quality in the Era of Multiple Versions --

（複数バージョンのあるソフトウェアの自動検証・検査
-- 複数バージョン管理時代のソフトウェアの品質向上 --）

氏　名　馬　雷 (Ma Lei)

Nowadays, software becomes one of the most important mediums to improve the labor force. It is widely used to manage data, control vehicle engines, schedule network communication, manipulate medical equipment. It affects almost every aspect of our society and our daily life. Software defects and failures, however, can cause severe tragedy and loss. Ensuring their quality is therefore very important.

The verification and testing of software systems to improve the software quality are two major significant activities in the software development cycle. However, while their significance is widely known, it is also known that occupy large part of cost in software development and maintenance. According to Lehman's software evolution laws, a software system has to be continuously changed to increase its functionalities, to fix bugs, to and adapt for new requirements over its lifecycle. Such changes are released as a series of updated software versions that share many commonalities among multiple versions. As the widely adoption of revision control system, Software Product Lines, and *clone-and-own* techniques, more and more similar version variants are produced.

This brings new challenges for conventional verification and testing techniques, which are only able to analyze one single version. Separate verification and testing for each individual version would cause many redundancies in analyzing the same code. In other words, the results from separate analysis processes are also difficult to be shared among multiple versions. Sharing common knowledge is important to improve the overall analysis results and achieve better quality assurance guarantee such as code coverage and bug detection capability. Therefore, there is a strong demand to create and design novel verification and testing techniques for multiple versions, improving the overall quality guarantee for all versions efficiently. However, the simultaneous analysis of multiple versions is inherently challenging, especially for managed languages like Java and C#, and platforms designed to handle a single version.

In this thesis, we first propose a general concept and framework *project centralization* to manage

multiple versions. Project centralization shares commonalities of multiple versions as much as possible while preserving the behaviors (run-time semantics) of each version. We formalize the version conflict problem and propose a graph representation for all versions of products under analysis. Based on this representation and graph formalization, we transform the project centralization problem into a graph-coloring problem where existing solutions can be applied to calculate both the optimal and near optimal solutions. We implement different existing techniques and compare their effectiveness for project centralization. We can conclude that our proposed heuristic algorithm is both efficient and effective to calculate the near optimal solution, compared to the optimal solution. Based on this framework, we perform consecutive of studies, implement many software verification and testing tool chains, and show that our tools are useful for improving software quality and correctness of general multiple version software.

Distributed systems are typical complicated software examples that are operated with multiple versions. They involve multiple concurrently running peers, which communicates each other over network. The multiple peers also use different versions of a software application. Each peer also runs a set of threads, which brings an additional dimension of concurrency, making the verification even more challenging. By extending project centralization, we further propose *process centralization* techniques for such challenging software quality insurance fields, which simulate the running of multiple processes by a single centralized process with the equivalent run-time behavior. With combined project centralization and process centralization approaches, we have successfully applied our tools to verify some practical distributed applications with multiple versions, demonstrating the effectiveness of our technique in revealing bugs that are unable to be detected by using existing techniques. We have also demonstrated that the centralized application runs more efficiently, compared original distributed applications, where each process runs on a different peer.

In additional to verification, testing is also a major approach for detecting software defect and improving software quality. Among various testing techniques, automatic testing techniques like random testing are proved to be useful in finding bugs and improving software quality. Conventional techniques suffer from low code coverage and allow testing only one single product. We further propose novel program analysis enhanced testing approaches and centralization based approaches for multiple versions. We propose fully automatic enhanced automatic testing techniques by adopting domain knowledge of software under testing. Our technique automatically performs static program analysis on the software under test to extract the software properties and information like constants and method side effects. Our tool also performs dynamic phase program analysis, which analyzes the feedback results when executing the generated test cases. Our design and implementation outperform the current most advanced fully automatic random testing tool Randoop after many strict and thorough evaluations on more than 30 widely used benchmarks and by researchers on their developed collection products inside Software Competence Center Hagenberg (SCCH). The evaluation results demonstrate the usefulness of our tool in

both improving code coverage and bug detection, which further improves the software quality.

To perform automatic testing for multiple versions, we refine our project centralization to the method level. Our approach is able to share the common code at the method level, which is accurate enough for automatic test case generation for multiple versions. We design and implement the novel testing strategies to test multiple software versions simultaneously. Our technique reduces the redundancies in analyzing the same code while sharing the common testing results among all versions. The experiments on real-world benchmarks demonstrate that the reuse of analysis results can improve the overall testing performance, achieving higher coverage more efficiently compared to separately test each individual version in a conventional way.

In summary, this thesis focuses on the issues to improve software quality, specifically on proposing novel techniques for the management, verification and testing multiple versions (with version conflicts) in the era of many coexisted version variants. We summarize the main achievements of our proposed concepts, techniques and implementations from our consecutive studies on the management, verification and testing to improve software quality for multiple versions.

Our work successfully makes steps further and solves important problems for multiple version related analysis: (1) manage multiple version variants, sharing common code while preserving the behavior of each version. (2) simulate the runtime behavior of multiple concurrently running processes by a single centralized process with the equivalent run-time behavior. (3) The verification of distributed application with multiple version coexisted and running on multiple peers, detecting unknown bugs that conventional techniques cannot. (3) propose domain knowledge enhanced fully automatic testing techniques by using a combined static and dynamic program analysis techniques. (4) design novel testing approach for multiple versions, sharing testing results while reducing testing redundancies.

Although our work may just represent a small piece of the whole iceberg: the verification and testing research for multiple versions, where many other problems are still needed to be solved, we demonstrate that the multiple version management and analysis are very promising in sharing commonalities, improving the analysis performance while reducing analysis redundancies. As more and more problems are caused by the multiple versions, more novel fundamental and practical techniques and solutions are needed from research community, which is expected to be one of the most important issues and research topics in software quality insurance in the near future. This thesis provides the foundations and software tool chains for further research studies on multiple versions.