Teaching Software Testing Methods Based on Diversity Principles*

Zhenyu Chen, Jinyu Zhang, and Bin Luo
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
Software Institute, Nanjing University, Nanjing 210093, China
zychen@software.nju.edu.cn

Abstract

Software testing is the primary approach to support software quality assurance. Many novel software testing methods have been proposed to achieve various tasks in recent years. It is a challenge to teach these new testing methods and classical testing methods within limited time. This paper reports our work in progress on the new teaching approach to software testing methods based on diversity principles.

1. Introduction

Software testing is an essential and key activity in the lifecycle of software development. Software testing remains the primary way to support software quality assurance, despite some alternative approaches, such as code inspection, formal verification, etc., can also be used. Software testing has been widely adopted in industry. Nowadays software quality becomes the dominant success criterion in the software industry. In the past decades, a lot of advanced techniques, automation tools, and effective processes of software testing have been proposed and developed by researchers and practitioners. Apart from these achievements, the testers' skill, enthusiasm and commitment still play key roles in a successful test process [1]. Therefore, software testing becomes an essential and important subject in the teaching program of software engineering [2].

As claimed by A. Bertolino in [1], "*Education must be continuing, to keep the pace with the advances in testing technology*". Many advanced software testing methods, such as adaptive random testing [3], cluster test selection [4], similarity-based test selection [5], etc. have been proposed in recent years. Although these new testing methods are proposed to achieve various targets, they share a common characteristic: diversity of test cases. Actually, diversity is not a new idea in software testing. It is implicit in many traditional testing methods, such as domain testing [6], coverage testing [7], combination testing [8], etc. It is necessary, interesting and challenging to teach various software testing methods based on diversity principles. This paper reports our work in progress on the new teaching approach to diversity principles in software testing.

2. Challenge and solution

The "Software Testing and Quality" is a core curriculum for undergraduate students in Software Institute of Nanjing University. The "Software Testing and Quality" focuses on teaching various software testing methods and their applications in practice. The duration of the curriculum is only 12 weeks and 3 hours per week. For the curriculum as for other related software testing curricula [9], it must be carefully designed so that students could gain an in-depth understanding of various software testing methods and skillfully use these methods in practice. Many software testing methods, both practical and theoretical ones, will be taught in

* This work was partially supported by the Project of Software Engineering Specialty Construction (2007-2201~2203), the Project of Software Engineering Teaching Team (2010-118), and the Project of Software Engineer Education Innovation (2008-042) by the Ministry of Education of China.

978-1-4577-0348-5/11/$26.00 ©

12 weeks. The first challenge for students is to learn the basic concepts and techniques of many testing methods within limited time.

Students who advance the state of the art can advance the state of practice in the future. Although most of our students will become software practitioners, the novel testing methods from academic will be taught. The successful stories in academic may happen in industry some years later. Students are required to implement these testing methods in practice skillfully, so that they are capable to transfer these methods into industry in the future. The second challenge for students is to implement these testing methods in practice by overcoming the gaps between academic and industry [10].

In order to empower the students' potential, we introduce the diversity principles of software testing methods in the teaching program. A unified framework of software testing methods based on diversity principles will be introduced in the "Software Testing and Quality". Such a unified framework is beneficial to understand the fundamental principles in testing methods. The students can master more quickly testing methods based on diversity principles. Furthermore, students can distinguish these testing methods easily in the unified framework, such that they can gain in-depth understanding of the limitations and the possibilities offered by these available testing methods. This is an essential knowledge to transfer novel testing methods into various application scenarios in industry.

3. Software testing method

The "Software Testing and Quality" includes many software testing methods. We give a brief introduction to six testing methods, three of which are classical ones (DT, CvT, and CmT) and the other three are new ones (ART, CTS, STS).

1. **Domain Testing (DT)**. DT is to partition the input domain into subdomains and then select some representative data from each subdomain for testing [6]. These subdomians are called equivalence classes. DT has several names, such as equivalence class testing, partition testing, etc.
2. **Coverage Testing (CvT)**. CvT produces test cases with measuring the execution of entities, such as statement, branch, etc. CvT could be extended to control-flow coverage, data-flow coverage, graph coverage, etc [7].
3. **Combination Testing (CmT)**. CmT is to identify test cases by combining values of the different test input parameters based on some combinatorial strategies [8].
4. **Adaptive Random Testing (ART)**. ART is to maximize the distance of next test case from a candidate set, such that test cases can realize "even spreading" [3].
5. **Cluster Test Selection (CTS)**. CTS is to partition test cases into different clusters based on their execution profiles. Then a subset of test cases will be sampled from clusters [4].
6. **Similarity-based Test Selection (STS)**. STS maximizes the distances among the selected test cases using a (dis)similarity measure between pairs of abstract test cases [5].

4. Diversity Principle

Software testing, in the simplest term, is to execute software under test and observe the results to validate whether it behaves as intended and identify potential faults. In order to improve the effectiveness of testing, we should exercise different behaviors of software as far as possible. That is so-called diversity.

A natural object of diversity is the input domain of software. An important assumption on this type of methods is that diversified input data often leads to diversified behaviors [3]. There exist many different perspectives to achieve diversity on the input domain. Hence it results in different software testing methods. Domain testing is a stratified sampling strategy, such that test cases can evenly spread in the equivalent classes of input domain [6]. Combination testing produces test cases, such that they can evenly cover the combinations of input parameters [8]. Adaptive random testing (ART) is a more general idea. It produces test cases to evenly spread in the input space. Such diversity is based on some common distances, such as Euclidean distance [3].

The object of diversity can also be derived from the source code. Statement coverage testing is to generate or select test cases, such that they can cover different statements as many as possible. The idea of other coverage testing methods is similar [7], but it may derive different objects of diversity, such as branch, def-use pair, etc. A more powerful object of diversity is the profiles of test cases, such as function call profile, method call stack, etc. The diversity of profiles can also be measured by some common distances. Cluster test selection (CTS) selects test cases by cluster analysis and sampling strategies [4]. Different from CTS, Similarity-based test selection (STS) uses abstract profiles from a model instead of real profiles from a program. STS selects test cases in the style of ART, such that the similarity of test cases is minimized [5].

Table 1. Summary of diversity principles

Method	Object	Metric
Domain Testing	Input Domain	Binary
Coverage Testing	Control-Flow, Data-Flow, etc.	Binary
Combination Testing	Input Domain	Binary
Adaptive Random Testing	Input Domain	Distance
Cluster Test Selection	Profile	Distance
Similarity-based Test Selection	Abstract Profile	Distance

We summarize the diversity principles of the six software testing methods in Table 1. Three new ones (ART, CTS, STS) adopt distance as a metric of diversity, such that they could be more extendibility. However, these three new software testing methods often require more complex techniques, such as genetic algorithms, machine learning, etc.

5. Teaching approach

The students before taking the "Software Testing and Quality" already have some basic knowledge on software testing, such as black-box testing, unit testing, etc. Hence the "Software Testing and Quality" can be considered as an advanced course on software testing. Two main objectives of this course are:

1. In-depth understanding of various software testing methods;
2. Skillful implementation of these software testing methods in practice.

In order to achieve these two main objectives, we utilize the teaching approach: *lecture-tutorial-project-panel*. This teaching process will be repeated for each testing method.

1. Prepare a formal lecture to present the basic concepts, techniques of software testing methods. Teaching should be supported by some related books or papers.

2. Develop a tutorial using some small examples (such as the triangle program) to reinforce the concept of the method and illustrate the effectiveness. The bugs could be created manually by students or teaching assistants.
3. Develop a project to reinforce the contents of the lecture and tutorial through practice. The programs should be middle size (such as some programs in SIR website [11]), such that there are some chances to illustrate the limitations of testing methods. A lot of bugs could be generated by some mutation tools to evaluate testing methods [12].
4. Organize a panel discussion with students. The students should discuss (a) the challenges and experiences to implement the testing method, (b) the limitations on the effectiveness and efficiency of the testing method, (c) the potential solution to address the encountered problems.

Since the software testing methods will be taught one by one, the former testing method should facilitate introduction to the latter testing method. Domain testing is one of the most widely used software testing methods. The concept of domain testing is easy to explain, although students may appear knowledgeable long before they achieve competence [13, 14]. Therefore, it is a good choice to teach domain testing as the first software testing method in this course.

In the first panel discussion, we will focus on the limitations of domain testing. One of the challenges for students is how to obtain the suitable equivalent classes. This challenge encourages students to use another simple testing method: random testing. And then it is natural to extend adaptive random testing for diversity. On the other hand, it also makes students consider how to use source code if the specification is absent. Then coverage testing is introduced. Even if some students can successfully get suitable equivalent classes in their projects, they may face the combinatorial explosion problem for many input variables. Then combination testing is introduced.

In the panel discussion on coverage testing, we will present some concepts and techniques of regression testing. There are many regression testing methods based on different coverage criteria. The fault detection capability will be used to evaluate these testing methods. To improve the effectiveness, some machine learning techniques, such as cluster analysis, can be used in test selection. These techniques can utilize more rich execution information, such as function call profile, method call stack, etc. Then cluster test selection is introduced.

From code-based testing to model-based testing, similarity-based test selection will be introduced. Similarity-based test selection will be compared with adaptive random testing and cluster test selection, respectively. The key perspective of comparison is the object of diversity. They are input domain, profile and abstract profile for ART, CTS and STS, respectively. It can lead to sufficient discussion on this topic.

In the late of this course, there are two directions to promote students. The first one is "*implementation*". Even if students gain in-depth understanding of these testing methods, It is still a challenge to implement them for programs with complex data structure or new development technologies. These challenges can inspire students to integrate more development techniques and knowledge. This can empower students' potential to transfer them into industry in the future. The second one is "*improvement*". ART, CTS, STS are new testing methods proposed in recent years. They are far from perfect. There are many topics on improvement of effectiveness and efficiency. This may lead students to do some research on new software testing methods.

6. Evaluation approach

As described in the previous sections, teaching testing methods based on diversity principles seems promising. These new testing methods could provide more chances to students. However, it is a challenge to evaluate the teaching approach comprehensively, so that other educators would like to adopt it [15].

Our teaching could be divided into two phases: (1) Classical testing methods (DT, CvT, CmT). (2) New testing methods (ART, CTS, STS). All students will be randomly divided into two groups. One group will submit their projects in the late of each phase. The other group will submit their projects at the end. For the second group, they have chances to improve previous projects when they learn new testing methods. However, they also have a risk of confusion if many testing methods are taught in an inappropriate way. The projects for DT, CvT and CmT of two groups will be compared to evaluate whether the teaching approach based on diversity principles is effective.

7. Conclusion and future work

This paper reports our work in progress on the new teaching approaches to software testing methods based on diversity principles. We hope that the new teaching approaches can help student gain in-depth understanding and skillful implementation on various software testing methods. This achievement may depend on many factors. In the next step, we will focus on the design of tutorials and projects in detail. Our evaluation approach is preliminary. More methods, such as questionnaire, assignment, etc., will be introduced to evaluate our testing approach comprehensively in the future.

References

[1] A. Bertolino, "*Software Testing Research: Achievements, Challenges, Dreams*", FOSE 2007, pp. 85-103.

[2] T. Lethbridge, J. L. Díaz-Herrera, R. J. LeBlanc, and J. B. Thompson, "*Improving Software Practice through Education: Challenges and Future Trends*", FOSE 2007, pp.12-28.

[3] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse, "*Adaptive Random Testing: the ART of Test Case Diversity*", Journal of Systems and Software, 2010, 83(1): 60-66.

[4] C. Zhang, Z. Y. Chen, Z. H. Zhao, S. L. Yan, J. Y. Zhang, and B. W. Xu, "*An Improved Regression Test Selection Technique by Clustering Execution Profiles*, QSIC 2010, pp.171-179.

[5] H. Hemmati, L. Briand, A. Arcuri, and S. Ali. "*An Enhanced Test Case Selection Approach for Model-Based Testing: An Industrial Case Study*", ACM SIGSOFT FSE, 2010, pp.267-276.

[6] L. J. White, and E. I. Cohen, "*A Domain Strategy for Computer Program Testing*", IEEE Transactions on Software Engineering, 1980, 6(3):247–257.

[7] P. Ammann, and J. Offutt. Introduction to Software Testing. Cambridge University Press, 2008.

[8] M. Grindal, J. Offutt, and S. F. Andler, "*Combination Testing Strategies: a Survey*", Software Testing, Verification and Reliability. 2005, 15(3): 167-199.

[9] H. Liu, F. C. Kuo, and T.Y. Chen, "*Teaching an End-User Testing Methodology*", CSEE&T 2010, pp. 81-88.

[10] R. L. Glass, R. Collard, A. Bertolino, J. Bach, and C. Kaner, "*Software Testing and Industry Needs*", IEEE Software, 2006, 23(4): 55-57.

[11] Software-artifact Infrastructure Repository. http://sir.unl.edu/, University of Nebraska

[12] Y. Jia, and M. Harman, "*An Analysis and Survey of the Development of Mutation Testing*", IEEE Transactions of Software Engineering, 2010.

[13] C. Kaner, and S. Padmanabhan, "*Practice and Transfer of Learning in the Teaching of Software Testing*, CSEE&T 2007, pp. 157-166.

[14] C. Kaner, "*Teaching Domain Testing: A Status Report*", CSEE&T 2004, pp. 112-117.

[15] E. O. Navarro, and A. Hoek, "*Comprehensive Evaluation of an Educational Software Engineering Simulation Environment*", CSEE&T 2007, pp. 195-202.