TEST DATA GENERATION METHOD FOR DYNAMIC – STRUCTURAL TESTING IN AUTOMATIC PROGRAMMING ASSESSMENT

MD. SHAHADATH SARKER

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

UNIVERSITI UTARA MALAYSIA

2016

Permission to Use

In presenting this thesis in full fulfillment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the University Library may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by my supervisor(s) or, in their absence, by the Dean. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok
Kedah Darul Aman
Malaysia

Declaration

I declare that this thesis is my own work and has not previously been submitted in any form for another degree or diploma at any other university or other institute of tertiary education. Information derived from the published and unpublished works of others have been acknowledged in the text and a list of references is given.

Md. Shahadath sarker
2016

Abstrak

Penaksiran Pengaturcaraan Automatik atau dikenali sebagai APA telah diketahui sebagai suatu keadah yang berkesan dalam membantu para pensyarah untuk melaksanakan penaksiran dan penggredan tugasan pengaturcaraan pelajar. Untuk melaksanakan pengujian dinamik dalam APA, adalah menjadi suatu keperluan untuk menyediakan set data ujian melalui proses penjanaan data ujian yang bersistematik. Sekiranya memfokus kepada bidang pengujian perisian, pelbagai kaedah untuk mengautomasikan penjanaan data ujian telah dicadangkan.Walaubagaimanapun, kaedah-kaedah ini jarang diguna pakai di dalam kajian semasa APA. Terdapat kajian awalan yang cuba mengintegrasikan APA dan penjanaan data ujian, tetapi masih terdapat jurang dari segi menerbitkan dan menjana data ujian untuk pegujian dinamik-berstruktur. Untuk mengatasi jurang ini, kajian ini mencadangkan suatu kaedah penjanaan data ujian untuk melaksanakan pengujian dinamik-berstruktur (atau dikenali sebagai DyStruc-TDG).DyStruc-TDG direalisasikan sebagai alatan fizikal yang bertindak sebagai penjana data ujian untuk menyokong fungsian APA.Dapatan daripada ekperimen kawalan yang dilaksana berdasarkan reka bentuk *one-group pre-test* dan *post-test* mendapati bahawa DyStruc-TDG memperbaiki kriteria kecukupan data ujian kebolehpercayaan (atau pengujian positif) dalam penaksiran pengaturcaraan.Kaedah yang dicadangkan ini adalah dijangkakan dapat membantu para pensyarah kursus pengaturcaraan awalan untuk menerbitkan dan menjana data ujian dan kes ujian untuk melaksanakan penaksiran pengatucaraan automatik untuk pengujian dinamik-berstruktur tanpa memerlukan pengetahuan khusus dalam reka bentuk kes ujian.Dengan mengguna-pakai kaedah ini sebagai sebahagian APA, beban para penyarah secara tidak langsung dapat dikurangkan secara berkesan oleh kerana penaksiran tipikal yang manual senantiasa cenderung kepada ralat dan penyebab kepada ketidakseragaman.

Kata kunci: penjanaan data ujian, Penaksiran Pengaturcaraan Automatik, pengujian dinamik, pengujian berstruktur, path coverage, Modified Condition/Decision Coverage.

Abstract

Automatic Programming Assessment or so-called APA has being known as a significant method in assisting lecturers to perform automated assessment and grading on students' programming assignments. Having to execute a dynamic testing in APA, it is necessary to prepare a set of test data through a systematic test data generation process. Particularly focusing on the software testing research area, various automated methods for test data generation have been proposed. However, they are rarely being utilized in recent studies of APA. There have been limited early attempts to integrate APA and test data generation, but there is still a lack of research in deriving and generating test data for dynamic structural testing. To bridge the gap this study proposes a method of test data generation for dynamic structural testing (or is called DyStruc-TDG). DyStruc-TDG is realized as a tangible deliverable that acts as a test data generator to support APA. The findings from conducted controlled experiment that is based on one-group pre-test and post-test design depict that DyStruc-TDG improves the criteria of reliability (or called positive testing) of test data adequacy in programming assessments. The proposed method is expectantly to assist the lecturers who teach introductory programming courses to derive and generate test data and test cases to perform automatic programming assessment regardless of having a particular knowledge of test cases design in conducting a structural testing. By utilizing this method as part of APA, the lecturers' workload can be reduced effectively since the typical manual assessments are always prone to errors and leading to inconsistency.

Keywords: test data generation, Automatic Programming Assessment, dynamic testing, structural testing, path coverage, Modified Condition/Decision Coverage.

Acknowledgement

Firstly, it is my great pleasure to thank Allah (swt) that has given me opportunities to complete this thesis. A billion thanks those, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior, and acts throughout the study period.

I express my sincere gratitude to my supervisor Dr.Rohaida Binti Romli for her continuous support, cooperation, valuable suggestions and motivation for completing the thesis and she exchanged her interesting ideas, thoughts and made this thesis easy and accurate.

Lastly, I would like to thank all of my friends for their encouragement and valuable suggestions during my study period.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

APA	Automatic Programming Assessment
ATDG	Automatic Test Data Generation
DyStruc-TDG	Dynamic Structural- Test Data Generation
MC/DC	Modified Condition/Decision Coverage
OOAD	Object Oriented Analysis and Design
UML	Unified Modeling Language
UUM	Universiti Utara Malaysia

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Learning computer programming languages has become essential for students who pursue their study in Information Technology, Computer Science and Software Engineering disciplines. Computer introductory programming courses are commonly offered for the first year degree students who pursue their study in these fields (Truong *et al.*, 2005). Effective and good programming skills are necessary for students in order to be a master in programming. Students can be skilled in programming only through practices (Lahtinen *et al.,* 2005). Computer programming courses are normally designed with full of practical besides theory. The goal of practical course is to develop student's basic understanding of programming principles and writing basic source code. Therefore, students are given many programming exercises as take home assignments or hands on practice in the class in order to develop student's programming understanding and skill (Rohaida *et al*., 2010). If the assessment of programming exercises is done by manually for a large number of students it leads to workload to lecturers and assessing manually is really difficult task which cannot ensure the consistency and accuracy of the marking scheme (Rohaida *et al*., 2010). Therefore, the concept of Automatic Programming Assessment (APA) has become very important to assess students program for grading and providing feedback (Saikkonen *et al*., 2001). Besides, APA can improve students marking assessment in terms of consistency and thoroughness testing (Gupta *et al*., 2012).

According to Jackson (1996) APA is founded on software testing technique. The programming assessment normally involves the measuring of the program quality. In order to achieve program quality the program should be tested. Hence, through the software testing technique the quality of the program can be measured (Rohaida *et al*., 2010). Software testing is a method for locating, measuring, and disclosing errors that occurred in a program (Latiu, 2012). Software testing can be categorized into two types: static analysis and dynamic testing, in which static testing is a test that does not involve in the execution of the program (Zin *et al*., 1994). On the other hand, dynamic testing requires a program execution with test data (Chu *et al.,* 1997). Test data is data which is developed as input in order to perform testing for any software program (Korel, 1990).

The contents of the thesis is for internal user only

REFERENCES

Bertolino, A., & Marchetti, E. (2005). A brief essay on software testing.*Software Engineering, 3rd edn. Development process*, *1*, 393-411.

Blumenstein, M., Green, S., Nguyen, a., & Muthukkumarasamy, V. (2004). GAME: a Generic Automated Marking Environment for programming assessment. International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004., 212–216 Vol.1.

Burnstein, I. (2003). *Practical Software Testing*, Springer-Verlag, New York.

Cheng, Z., Monahan, R., & Mooney, A. (2011). nExaminer: A semi-automated computer programming assignment assessment framework for Moodle.

Choy, M., Nazir, U., Poon, C. K., & Yu, Y. T. (2005). Experiences in using an automated system for improving students' learning of computer programming. In *Advances in Web-Based Learning–ICWL 2005* (pp. 267-272). Springer Berlin Heidelberg.

Chu, H. D. (1997). *An evaluation scheme of software testing techniques* (pp. 259-262). Springer US.

Clarke, L. a. (1976). A System to Generate Test Data and Symbolically Execute Programs. IEEE Transactions on Software Engineering, SE-2(3), 215–222.

Clarke, L. A. (1976). A system to generate test data and symbolically execute programs. *Software Engineering, IEEE Transactions on*, (3), 215-222.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 319-340.

Edvardsson, J. (1999, October). A survey on automatic test data generation. In *Proceedings of the 2nd Conference on Computer Science and Engineering* (pp. 21-28).

Edvardsson, J. (1999, October). A survey on automatic test data generation. In *Proceedings of the 2nd Conference on Computer Science and Engineering*(pp. 21-28).

Foong, O. M., Tran, Q. T., Yong, S. P., & Rais, H. M. (2014, June). Swarm inspired test case generation for online C++ programming assessment. In*Computer and Information Sciences (ICCOINS), 2014 International Conference on* (pp. 1-5). IEEE.

Fraenkel, J. R., & Wallen, N. E. (1993). *How to design and evaluate research in education* (Vol. 7). New York: McGraw-Hill.

Ghani, K., & Clark, J. A. (2009, September). Automatic test data generation for multiple condition and MCDC coverage. In *Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on* (pp. 152-157). IEEE.

Ghani, K., & Clark, J. A. (2009, September). Automatic test data generation for multiple condition and MCDC coverage. In *Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on* (pp. 152-157). IEEE.

Goodenough, J. B., & Gerhart, S. L. (1975). Toward a theory of test data selection. *Software Engineering, IEEE Transactions on*, (2), 156-173.

Guo, M., Chai, T., & Qian, K. (2010, April). Design of Online Runtime and Testing Environment for Instant Java Programming Assessment. In*Information Technology: New Generations (ITNG), 2010 Seventh International Conference on* (pp. 1102-1106). IEEE.

Gupta, N., Mathur, A. P., & Soffa, M. L. (1998). Automated test data generation using an iterative relaxation method. *ACM SIGSOFT Software Engineering Notes*, *23*(6), 231-244.

Gupta, S., & Dubey, S. K. (2012). Automatic Assessment of Programming assignment, 452003, 315–323.

Hakulinen, L., & Malmi, L. (2014, June). QR code programming tasks with automated assessment. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 177-182). ACM.

Hayhurst, K. J., Veerhusen, D. S., Chilenski, J. J., & Rierson, L. K. (2001). A practical tutorial on modified condition/decision coverage.

Ihantola, P. (2006). Automatic test data generation for programming exercises with symbolic execution and Java PathFinder. *Master's thesis, Helsinki University of Technology, Departement of Theoretical Computer Science*.

IPL Information Processing Ltd. (197a). Designing Unit Test Cases. Available http://www.ipl.com/pdf p0823.pdf. Retrieved on: 20 Feb 2009

Jackson, D. (2000). A semi-automated approach to online assessment. *ACM SIGCSE Bulletin*, *32*(3), 164-167.

Jackson, D., & Usher, M. (1997). Grading student programs using ASSYST. ACM SIGCSE Bulletin, 29(1), 335–339.

James, R., Ivar, J., & Grady, B. (1999). The unified modeling language reference manual. *Reading: Addison Wesley*.

Koomen, T., & Pol, M. (1999). *Test process improvement: a practical step-by-step guide to structured testing*. Addison-Wesley Longman Publishing Co., Inc.

Korel, B. (1990). Automated software test data generation. *Software Engineering, IEEE Transactions on*, *16*(8), 870-879.

Korel, B. (1996). Automated test data generation for programs with procedures. ACM SIGSOFT Software Engineering Notes, 21(3), 209–215.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005, June). A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin* (Vol. 37, No. 3, pp. 14-18). ACM.

Latiu, G. I., Cret, O. A., & Vacariu, L. (2012). Automatic Test Data Generation for Software Path Testing Using Evolutionary Algorithms. 2012 Third International Conference on Emerging Intelligent Data and Web Technologies, 1–8.

Luck, M., & Joy, M. (1999). A secure on-line submission system. Software: Practice and Experience, 29(8), 721–740.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., & Silvasti, P. (2004). Visual Algorithm Simulation Exercise System with Automatic Assessment: TRAKLA2. *Informatics in education*, *3*(2), 267-288.

McMinn, P. (2004). Search-based software test data generation: A survey.*Software Testing Verification and Reliability*, *14*(2), 105-156.

Monpratarnchai, S., Fujiwara, S., Katayama, A., & Uehara, T. (2014). Automated testing for Java programs using JPF-based test case generation.*ACM SIGSOFT Software Engineering Notes*, *39*(1), 1-5.

Offutt, J., Liu, S., Abdurazik, A., & Ammann, P. (2003). Generating test data from state-based specifications. Software Testing, Verification and Reliability, 13(1), 25–53.

Pargas, R. P., Harrold, M. J., & Peck, R. R. (1999). Test-data generation using genetic algorithms. *Software Testing Verification and Reliability*, *9*(4), 263-282.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

Rayadurgam, S., & Heimdahl, M. P. E. (2003, December). Generating MC/DC Adequate Test Sequences Through Model Checking. In *SEW* (p. 91).

Romli, R. (2014). Test Data Generation Framework for Automatic Programming Assessment, 84–89.

Romli, R., Sulaiman, S., & Zamli, K. Z. (2010). Automatic Programming Assessment and Test Data Generation, 00(c).

Romli, R., Sulaiman, S., & Zamli, K. Z. (2013). Designing a Test Set for Structural Testing in Automatic Programming Assessment.

Rumbaugh, J. (2003). Object-oriented analysis and design (OOAD).

Saikkonen, R., Malmi, L., & Korhonen, A. (2001, June). Fully automatic assessment of programming exercises. In *ACM Sigcse Bulletin* (Vol. 33, No. 3, pp. 133-136). ACM.

Sekaran, U. (2006). *Research methods for business: A skill building approach*. John Wiley & Sons.

Tillmann, N., & Halleux, J. De. (2008). Pex – White Box Test Generation for . NET, 134–153.

Tillmann, N., De Halleux, J., Xie, T., Gulwani, S., & Bishop, J. (2013, May). Teaching and learning programming and software engineering via interactive gaming. In *Software Engineering (ICSE), 2013 35th International Conference on* (pp. 1117-1126). IEEE.

Truong, N., Bancroft, P., & Roe, P. (2005). Learning to program through the web. ACM SIGCSE Bulletin, 37(3), 9.

Varshney, S., & Mehrotra, M. (2013). Search based software test data generation for structural testing: a perspective. *ACM SIGSOFT Software Engineering Notes*, *38*(4), 1-6.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS quarterly*, 425-478

Watkins, J., & Mills, S. (2010). *Testing IT: an off-the-shelf software testing process*. Cambridge University Press.

Zamli, K. Z., Ashidi, N., Isa, M., Fadel, M., & Klaib, J. (2007). A Tool for Automated Test Data Generation (and Execution) Based on Combinatorial Approach, 19–36.

Zhu, H., Hall, P. A., & May, J. H. (1997). Software unit test coverage and adequacy. *Acm computing surveys (csur)*, *29*(4), 366-427.

Zidoune, W., & Benouhiba, T. (2012). Targeted adequacy criteria for search-based test data generation.2012 International Conference on Information Technology and E-Services, 1-6.