بسم الله الرحمن الرحيم

B ١٧٢٢٣

AN EMPIRICAL STUDY OF USING PROGRAM SLICING IN SOFTWARE TESTING AND MAINTENANCE

A THESIS

Presented to Computer Science Department, Faculty of Science, Minia University.

Submitted In Partial Fulfillment For The Award Of The Degree Of Master Of Science (Computer Science)

BY

Tarek Abd-El Hafeez Abd-El Rahman

B. Sc. Mathematics and Computer Science (1997)
Faculty of Science, Minia University

SUPERVISORS

Dr. Moheb Ramzy Girgis

Associate Professor
in the Computer Science Department,
Faculty of Science, Minia University.

Dr. Ahmed Aly Ahmed Radwan

Associate Professor
in the Computer Science Department,
Faculty of Science, Minia University.

Minia University

Faculty of Science

2005

ACKNOWLEDGMENTS

In the name of Allah,
The Beneficent, The Merciful

First and foremost thanks to ***GOD*** for everything ***HE*** gave me.

I would like to specially thank my supervisor, ***Dr. Moheb Ramzy Girgis***, Associate Professor of Computer Science, Computer Science Department, Faculty of Science, Minia University, for suggesting and planning the point of research, supervising the work, reading the manuscript and his valuable advice. He also gave me the initial impetus to look into the problem of software testing and helped me develop many ideas presented in this research.

Acknowledgment is gratefully made to ***Dr. Ahmed Aly Ahmed Radwan***, Associate Professor of Computer Science, Computer Science Department, Faculty of Science, Minia University, for supervising my research and his continuous support and encouragement. He has opened many doors for me that have turned out to be important in my life. He also provided the most excellent help and supervision I could wish for.

My thanks go to all the members of the Department of computer science and the Department of Mathematics, faculty of science, Minia University.

If there is one person I am most indebted to for everything I have been able to achieve, he is my father, who fostered my love for learning and who despite adversities, supported me in all my academic endeavors. I would also like to thank my wife who has always had words of encouragement for me, and who has patiently tolerated my absence during the many long hours I spent preparing this research .

Tarek Abd-El Hafeez Abd-El Rahman.

PREFACE

PREFACE

Software testing is a practical way of obtaining increased confidence in software. The ultimate goal of program testing would be guaranteeing that the program is correct. Program testing consists of generating a set of test cases according to some testing strategy and then checking the outputs produced by the test cases against the expected results.

Object Oriented (OO) systems have been promoted for ease of design, coding and re-use. Much research effort has been focused on design methods and process metrics yet little has been expended on testing and maintenance aspects. It appears that OO is promoted for its ease of design and use implying that testing and maintenance are easier or cheaper than in traditional, structural procedural systems.

It is assumed that OO system testing is no different from traditional system testing. What is apparent from the literature is the fractured approach to testing OO systems and the lack of methodology. There is also no formalism for measuring the adequacy of the test suite in respect of its fault finding abilities. This, however, is similar to traditional systems.

Software will undoubtedly undergo change after it is delivered to the customer. Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment (e.g. a change required because of a new operating system or peripheral device), or because the customer requires functional or performance enhancements. Software maintenance focuses on "change " that is associated with error correction, adaptations required as the software's environment evolves, and enhancements brought about by changing customer requirements.

Program slicing, introduced by Mark Weiser in 1984, is a method for automatically decomposing programs by analyzing their data flow and control flow.

A program slice consists of the parts of a program that (potentially) affect the values computed at some point of interest, referred to as a slicing criterion. Typically, a slicing criterion consists of a pair (line-number; variable). The task of computing program slices is called ***program slicing***.

This thesis presented a study of using program slicing in testing and maintenance of OO programs. We presented an OO representation for OO programs, which views the program elements as objects. This representation is a systematic way to analyze OO programs and collect all the needed information about their elements. Based on this representation, we introduced an approach to compute three types of slices on OO programs, namely, object slicing, data member slicing, and function/method slicing, which employed the slicing techniques for conventional programs. Also, we considered the problem of slicing OO programs in the presence of pointers and recursion.

We have built a system that computes any of the three types of slices on any given program according to the user request. We used this system in testing and maintenance of OO programs. Our system performs data flow analysis on the given program and generates test data automatically (if necessary) to satisfy certain data flow testing criterion.

Our system allows the user to perform data flow testing on the whole program or a slice only to save time and effort. When the user chooses to test the slice only, the program slice enables the user to focus on the program statements pertinent to the affected data flow components. Using the slice reduces the test case generation effort since only the input variables on the slice are considered.

In our system we have employed a program execution based method that uses well-established mathematical techniques to automatically generate test data. The method applies the traditional relaxation technique used in numerical analysis to obtain an exact solution of an equation by iterative improvement of an approximate solution. The results obtained from this method for test data generation are very promising. It provides a practical solution to automated test data generation problem.

It is more efficient than existing program execution base approaches as it requires fewer program executions.

We have conducted four types of experiments using the system. The aim of the first experiment was to evaluate the error-exposing ability of the control flow and data flow testing criteria employed in the developed slicing testing system. The results of this experiment showed that the system is very effective in discovering the errors that may occur in OO programs.

The aim of the second experiment was to demonstrate how the use of the slicing technique in testing can reduce the testing effort. This experiment showed that the use of slicing in testing cause a substantial reduction in the data flow components that need to be tested, which reduces the number of test cases required to cover these components, and in turn reduces the effort needed to generate these test cases.

The aim of the third experiment was to evaluate the effectiveness of the automatic test data generation technique employed in the developed testing system. The result of this experiment showed that the automatically generated test data can cover more control flow components (program edges) and data flow components (definition-use associations of variables) of the program being tested than the manual test data, which reduces the number of test runs required to cover all such program components, and in turn reduces the testing effort.

The aim of the last experiment was to demonstrate how program slicing can be used in maintenance of OO programs. We showed that only the modified part of the program needs to be tested which eliminates the need for regression testing, and showed also that our system has overcome the limitation of Gallagher and Lyle maintenance' technique that restricts the number of new statements to be inserted in a program slice to 10. Our system allows the insertion of any number of new statements, which gives the software maintainer more flexibility in modifying programs.

CONTENTS

CONTENTS