DEVELOPMENT OF AN AUTOMATED TEST DATA GENERATION AND EXECUTION STRATEGY USING COMBINATORIAL APPROACH

By

MOHAMMAD FADEL JAMIL KLAIB

Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

June 2009

ACKNOWLEDGEMENTS

The work described in this thesis was undertaken under the main supervision of Dr. Kamal Zuhairi Zamli, to whom I am grateful for his support, his interest during all levels of my PhD study, and for his insightful and critical comments in writing the published papers and the following thesis. To say the least, without Dr.Kamal's encouragement and enthusiasm, I will probably would not have gone this far. Also, even though Dr.Kamal is very busy, he took an enormous task of revising my thesis word by word. His efforts are greatly appreciated and will never be forgotten. Thanks again Dr.Kamal.

I also wish to thank my co-supervisor Dr. Nor Ashidi Mat Isa, who gave me his ever devotion and all valuable information which I really require to finish my thesis.

I am also thankful to all my friends in Malaysia, who gave their support and help through many helpful and enjoyable discussions. In particular, I am thankful to all academic staffs in the School of Electrical and Electronic Engineering, USM, and all those persons who have encouraged me to complete my study. Thanks!

I will never forget to be thankful to whom my love will never end, to my father and my mother, to my brothers and sisters, uncles and aunts, they all gave me their lasting encouragement in my studies, so that I could be successful in my life. Dad and mom, thank you for the prayers – this thesis is for both of you.

Finally, I would like to thank my loving wife (Haneen). To my daughter (Felesteen), even with her disturbing cry, she enters the joy to my heart and keeps me going; thanks for being patient all along. I am sorry to have sometimes neglected all of you to pursue my dream.

The work reported here would have not been possible without the e-science fund grant entitled “Development of a Fault Injection Tool to Ensure Dependability of Commercial-off-the-shelf Components (COTs) for Embedded System Applications”, and fellowship support from Universiti Sains Malaysia.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

PEMBANGUNAN PENJANAAN DATA UJIAN DAN STRATEGI PELARIAN AUTOMATIK MENGGUNAKAN PENDEKATAN BERGABUNGAN

ABSTRAK

Untuk memastikan tahap piawaian jaminan kualiti dan keboleharapan sesuatu perisian, pengujian hendaklah dijalankan untuk setiap konfigurasi. Tetapi, masalah letupan konfigurasi tidak memungkinkan pertimbangan keseluruhan terhadap semua nilai data ujian. Kekangan sumber, masalah kos, dan masa untuk dipasarkan yang ketat adalah merupakan antara faktor yang menghalang terhadap pertimbangan keseluruhan itu. Penyelidikan terdahulu menyimpulkan bahawa strategi persampelan berdasarkan interaksi t-cara antara parameter adalah sangat efektif. Berdasarkan kesimpulan ini, terdapat banyak strategi t-cara yang sedia ada telah dihasilkan.

Bidang penyelidikan ini mengalami pertumbuhan yang pesat sejak 10 tahun yang lalu dalam membantu proses perancangan ujian, terutamanya dalam mengurangkan data ujian yang perlu digunakan secara sistematik berdasarkan sesuatu interaksi t-cara yang terpilih. Walaupun terdapat banyak kemajuan, integrasi dan automasi strategi daripada proses perancangan dan pengujian amat tidak dititik beratkan. Dalam praktis sekarang, data ujian yang disampel perlu diekstrak secara manual dan ditukarkan dalam format tertentu sebelum ia boleh dilaksanakan (sama ada oleh penguji sendiri, atau alatan perisian daripada pihak ketiga). Masalah integrasi dan automasi ini amat menyusahkan kerja jurutera pengujian terutamanya apabila modul yang perlu diuji adalah bersaiz besar.

Selain daripada isu berkaitan integrasi dan automasi, perancangan untuk persampelan dan pembinaan data ujian yang paling minima daripada keseluruhan data ujian adalah juga masalah lengkap NP. Oleh yang demikian, tidak mungkin akan ada strategi bagi

menghasilkan data ujian yang optimal untuk setiap kes data ujian. Bagi menyahut cabaran yang digariskan di atas, tesis ini membincangkan rekabentuk, implementasi, dan penilaian, strategi GTWay untuk menerbitkan data ujian t-cara yang optimum. Tidak seperti strategi yang lain, GTWay dapat membantu proses perancangan dan larian data ujian secara automatik (serentak) yang diintegrasikan sebagai sebahagian daripada implementasinya. Keputusan empirikal membuktikan GTWay, dalam banyak keadaan, mengatasi strategi sedia ada dalam aspek penghasilan data ujian yang minima. Julat masa penghasilan ujian data juga adalah berpatutan seiring dengan perancangan dan larian ujian yang diintegrasikan.

DEVELOPMENT OF AN AUTOMATED TEST DATA GENERATION AND EXECUTION STRATEGY USING COMBINATORIAL APPROACH

ABSTRACT

To ensure acceptable level of quality and reliability of a typical software product, it is desirable to test every possible combination of input data under various configurations. Due to combinatorial explosion problem, considering all exhaustive testing is practically impossible. Resource constraints, costing factors as well as strict time-to-market deadlines are amongst the main factors that inhibit such consideration. Earlier work suggests that sampling strategy (i.e. based on t-way parameter interaction) can be effective. As a result, many helpful t-way sampling strategies have been developed in the literature.

Much useful advancement has been achieved in the last 10 years particularly to facilitate the test planning process, that is, in terms of systematically minimizing the test data to be considered for testing (i.e. based on some t-way parameter interactions). Despite such a significant progress, the integration and automation of the strategies from the planning process to execution appears to be lacking. In the current practice, the sampled test data need to be manually extracted and converted to some acceptable format before they can be executed (e.g. by a human tester, a code driver or a third party execution tool). This lack of integration and automation between test planning and execution can potentially burden the test engineers especially if the software module to be tested is significantly large.

Apart from integration and automation issues, strategizing to sample and construct minimum test set from the exhaustive test space is also a NP complete problem (i.e. nondeterministic polynomial). As such, it is often unlikely that efficient strategy exists that can always generate optimal test set. Motivated by such challenges, this

paper discusses the design, implementation, and validation of an efficient strategy, called GTWay. GTWay, unlike other strategies, supports both t-way test generation and automated (concurrent) execution integrated within the strategy itself. Empirical evidences demonstrate that GTWay, for some cases, outperforms other strategies in terms of the number of generated test data. The test generation time is also within reasonable value considering the fact that some overhead is required to permit the integration between test generation and execution.

CHAPTER 1

INTRODUCTION

Computing technology has gone a long way since the first Babbage computer. Today, many chores that were once manual have been taken over by computers. Factories use computers to control manufacturing equipments. Electronics manufacturing use computers to test everything from microelectronics to circuit card assemblies.

Software is one of the major components that drive the functionality and automation of computers. Here, software can be viewed as a collection of written program, functions, and procedures that enable the user to accomplish the task at hand. From washing machine controllers, mobile phone applications to sophisticated airplane control systems, software is becoming an indispensable part of our lives.

Imagine the world without software. For instance, our household washing machine may still be bulky as the controls may be composed of all mechanical switches. Similarly, our hand phone without software may have too limited capabilities to be useful. As these two examples illustrate, software (whenever possible) are becoming increasingly popular replacement for its hardware counter parts.

Our growing dependency on software can be attributed to a number of factors. Unlike hardware, software does not wear out. Thus, the use of software can help to control maintenance costs. Additionally, software is also malleable and can be easily customized as the need arises.

Nevertheless, the fact that software is malleable and can be easily customized can also be a burden. Here, testing is often sought for to ensure quality (i.e. whether or not the software is reliable and meets its specification). In the next section to come, this chapter will highlight an overview of software testing and the problem statement in order to set the scene of the work undertaken in this research work. Additionally, this chapter also highlights the roadmap of the thesis.

1.1 Overview of Software Testing

Covering as much as 40% to 50% of the development costs, software testing is an integral part of software engineering lifecycle. In a nut shell, software testing can be viewed as the process of executing a program with the intent to find error (Myers, 2004). Putting the overall picture as far as the overall software engineering product lifecycle is concerned, software testing can be viewed as the following (see Figure 1-1).

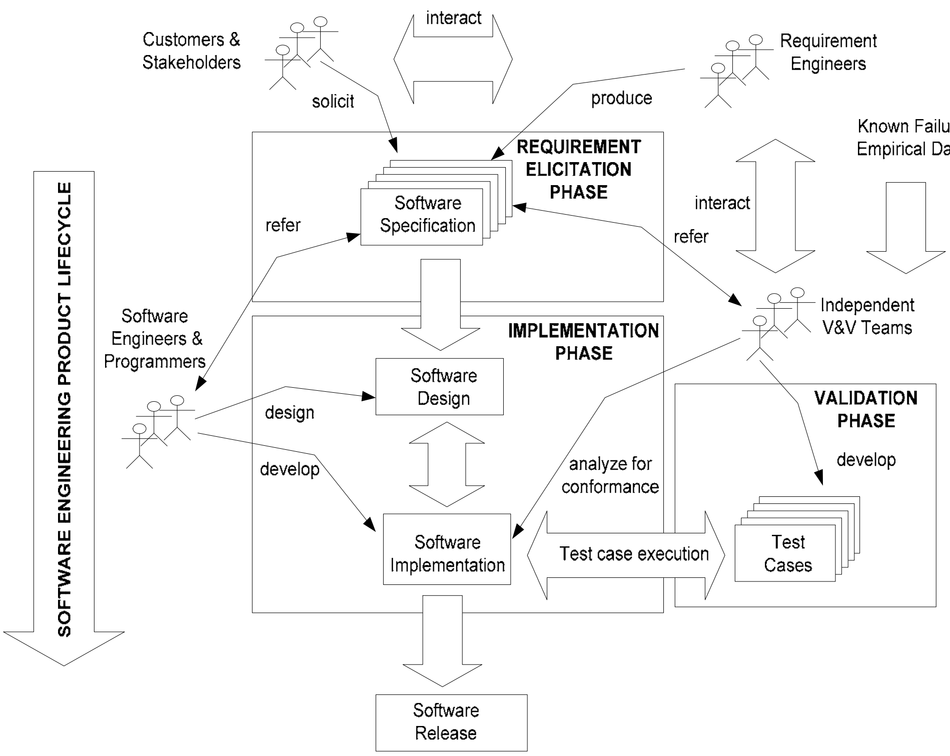

Figure 1-1 Software Engineering Product Lifecycle

Referring to Figure 1-1, software engineering product lifecycle starts with the requirement elicitation phase. Here, the customers and stakeholders interact with the requirement engineers to produce the software specifications. Based on the specifications, software engineers and programmers collaborate to produce software design and implementations. This activity occurs in the implementation phase. Software testing falls under the validation phase which may occur in parallel with the requirement elicitation phase and implementation phase. The independent verification and validation (V&V) team needs to consult the requirement engineers for software specification. Based on the software specification, the V&V team produces the test cases to be executed against the software implementation. If the execution results satisfy the requirement specification, then the software is ready to be released, otherwise, some additional work need to be done to the design and implementation until conformance is achieved.

As seen above, the purpose of testing is not to prove anything, rather to reduce the perceived risk of not working to an acceptable value. The key challenges in software testing are not only dependent on the actual execution of the test cases but also the production of quality test cases.

1.2 Problem Statements

Covering as much as 40 to 50 percent of the development costs and resources (Beizer, 1990, Kaner *et al.*, 1999, Pan, 1999), testing can be considered as one of the most important activities in product development for both software and hardware (Bryce *et al.*, 2005, Tsui and Karam, 2007). In order to ensure accepted quality and reliability, many combinations of possible input parameters, hardware/software environments, and system conditions are tested and verified against for conformance based on system's specification (Cohen *et al.*, 2007a, Cohen *et al.*, 2007b).

Lack of testing can lead to disastrous consequences including loss of data, fortunes, and even lives. For instance, consider the accident that occurred during the European Space Agency's launching of Ariane 5 in 1996. Investigation by independent researchers from Massachusetts Institute of Technology reveals that the disaster is caused by the mismatch of the hardware and software component faults (Lions, 1996). The component erroneously puts a 64 bit floating point number in to a 16 bit space, causing overflow error. This overflow error affected the rocket's alignment function, and hence, causing the rocket to veer off course and eventually exploded a mere 37 seconds after lift off.

Despite its importance, exhaustive testing is impossible due to the fact that the number of test cases can be exorbitantly large (Chaudhuri and Zhu, 1992, Copeland, 2004, Roper, 2002) even for simple software and hardware products. Consider a hardware product with 20 on/off switches. To test all possible combination would require 2^{20} = 1,048,576 test cases. If the time required for one test case is 5 minutes, then it would take nearly 10 years for a complete test.

The same argument is applicable for any software system. As illustration, consider the option dialog in Microsoft Excel software (see Figure 1-2). Even if only View tab option is considered, there are already 20 possible configurations to be tested. With the exception of Gridlines colour which takes 56 possible values, each configuration can take two values (i.e. checked or unchecked). Here, there are 2^{20}x56 (i.e. 58,720,256) combinations of test cases to be evaluated. Using the same calculation as the previous example, it would require nearly 559 years for a complete test of the View tab option.

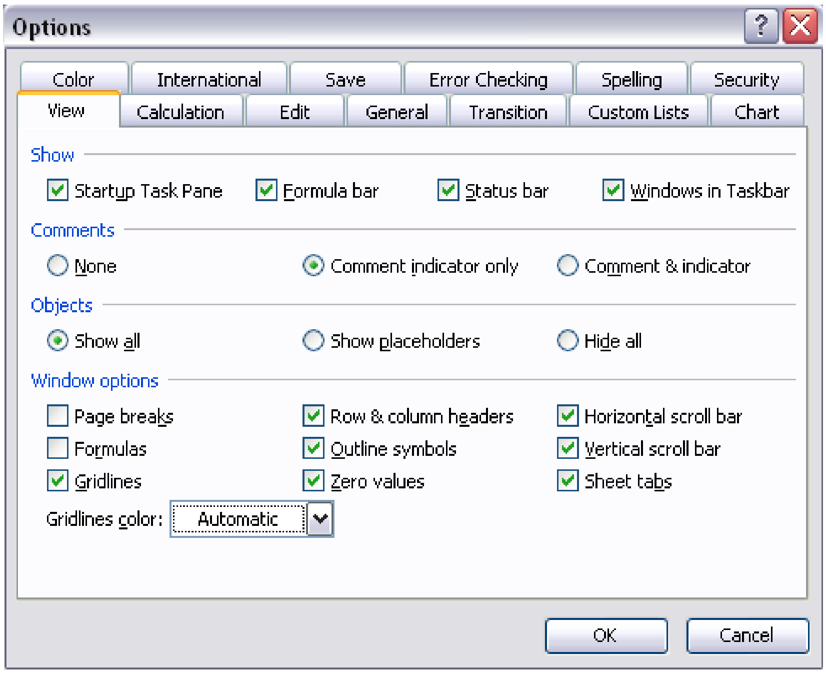

Figure 1-2 Microsoft Excel View Tab Options

The above mentioned examples highlighted the common combinatorial explosion problem in software testing. Given limited time and resources, the research questions are:

- What are the smaller optimum sets of (sampled) test data to be considered for testing?
- How can one decide (i.e. the strategy) on which combination of data values to choose over large combinatorial data sets?
- Will the test coverage be significantly affected by using lesser combinatorial data sets?

Combinatorial explosion problem (Cohen et al., 1997, Cohen et al., 2006b, Colbourn et al., 2004, Tai and Lei, 2002) poses one of the biggest challenges in modern computer science due to the fact that it often kills traditional approaches to analysis, verification, monitoring and control. A number of techniques have been explored in the past to address the combinatorial explosion problem. Undoubtedly, parallel testing (e.g. (ITL/NIST, 2008, Younis et al., 2009)) can be employed to reduce the time required for performing the tests. Nevertheless, as software and hardware are getting more complex than ever, parallel testing approach becomes immensely expensive due to the need for faster and higher capability processors along state-of-the-art computer hardware. Apart from parallel testing, systematic random testing could also be another option (Antony, 2003, Duran and Ntafos, 1984, Schroeder et al., 2004, Tseng et al., 2001). However, systematic random testing (e.g. (Ammann and Offutt, 1994)) tends to dwell on unfair distribution of test cases.

Earlier work (e.g. (Bryce and Colbourn, 2006, Dalal *et al.*, 1999, Kuhn and Okum, 2006, Kuhn and Reilly, 2002, Kuhn *et al.*, 2004, Yan and Zhang, 2008)) suggests that from empirical observation, the number of input variables involved in software and hardware failures is relatively small (i.e. in the order of 3 to 6), in some classes of system. If t or fewer variables are known to interact and cause fault (Ellims *et al.*, 2008b), test data can be generated on some t-way combinations (i.e. resulting into a smaller set of test data for consideration).

As will be seen in Chapter 2, a number of useful strategies have been reported to facilitate the test planning process, that is, in terms of systematically minimizing the test data to be considered for testing (i.e. based on some t-way parameter interactions). However, the integration and automation of the existing strategies from the planning process to execution appears to be lacking. In the current practice, the t-way sampled test data need to be manually extracted and converted to some acceptable format before they can be executed (e.g. by a human tester (Binder, 2000, Dustin *et al.*, 1999, Fewster and Graham, 1999), a code driver or a third party execution tool (Li and Wu, 2004)). This lack of integration and automation between test planning and execution can potentially burden the test engineers especially if the software module to be tested is significantly large.

In addition to integration and automation issues, strategizing to sample and construct minimum test set from the exhaustive test space is also a NP complete problem (Shiba *et al.*, 2004, Tai and Lei, 2002). As such, it is often unlikely that efficient strategy exists that can always generate optimal test set. Motivated by such challenges, this research work is devoted to investigate an optimum strategy, called GTWay, for systematic t-way test data generation (and reduction). Unlike earlier work, GTWay supports both the test planning process and the automated (concurrent) execution integrated within the strategy itself. In short, using t-way strategy is useful to systematically detect faults in a particular software system is the main hypothesis on this thesis.

1.3 Thesis Aim and Objectives

The main aim of this research is to develop and evaluate a general t-way test data generation and execution strategy, called GTWay, for software configuration testing.

The main objectives of the work undertaken were:

i. To develop and implement the GTWay strategy as a prototype implementation tool.

ii. To investigate automatic execution, when actual values are used, as part of the GTWay strategy.

iii. To investigate and compare the performance of GTWay strategy in terms of test size as well as execution time against existing works.

1.4 Thesis Outline

The remainder of this thesis is organised into five chapters as follows.

Chapter 2 presents an overview as well as highlights the main characteristics of t-way strategies. Using the characteristics, a survey of existing t-way strategies is provided including that of a special case for t-way strategies, the pairwise testing. Towards the end of Chapter 2, an analysis of existing work is presented which provides the requirements and justification for the development of GTWay.

Chapter 3 discusses and justifies the detailed algorithms and implementation for GTWay based on the requirements from Chapter 2. Here, issues related to the enabling automated execution are also explained. Additionally, the prototype implementation is also discussed in order to highlight its usage.

In Chapter 4, a detailed account for evaluating GTWay is presented. Here, the correctness of GTWay strategy will be evaluated. Apart from the correctness evaluation, a comparative study on the effectiveness of pairwise testing versus t-way testing will be highlighted using suitable case studies. Additionally, GTWay will also be compared against existing strategies in terms of the number of generated test data as well as execution time both as a pairwise strategy and as a general t-way strategy.

The conclusion of this work is given in Chapter 5, where the achievements, contributions and problems are summarised. Additionally, the main research hypothesis is revisited and the usefulness of GTWay is debated. Conclusions are drawn from the experience gained from this work and the significance of findings along with considerations for future work.

CHAPTER 2

LITERATURE REVIEW

The previous chapter has established the needs for software testing (i.e. for evaluating conformance and ensuring reliability), and highlighted the possible catastrophic aftermaths due software failure (i.e. including fortune and data losses as well as human fatality). In doing so, the previous chapter has also advocated the fact that testing for all combination of parameters, although desirable, is infeasible due to lack of resources as well as strict time-to-market constraints. Thus, systematic strategies are required to reduce the number of test cases by selecting a subset of these combinations for sampling, executing and analyzing.

In this chapter, these systematic strategies will be elaborated based on the t-way interaction of variables. Specifically, this chapter begins by giving an overview of the concept and terminology that will be used throughout this thesis. Next, the main characteristics of the combinatorial strategies will be identified in order to facilitate their survey and analysis. This survey and analysis is then used to provide justification for the development of GTWay, the strategy that is the basis of this thesis. Finally, this chapter closes by providing a short summary.

2.1 Overview

As discussed earlier, the main focus of the work described in this thesis is on the development of systematic test data minimization strategy based on (t-way) parameter interaction testing (or termed *t-way testing*). Here, the parameter interaction can be specified using a variable (t) indicating how strong the interaction is.