UNIVERSITI PUTRA MALAYSIA

THE PRACTICE OF UNIT TESTING BY FIVE SOFTWARE DEVELOPMENT COMPANIES IN MALAYSIA

MARZILAH HUSSAIN

FSKTM 2002 10

THE PRACTICE OF UNIT TESTING BY FIVE SOFTWARE DEVELOPMENT COMPANIES IN MALAYSIA

By

MARZILAH HUSSAIN

A Project Paper Submitted in Fulfillment of the Requirement for the Degree of Master of Science (Information Technology) in the Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

October 2002

ABSTRACT

Software testing is regarded as the most important and the most crucial of the final stages in software development. Profound knowledge and experience in software testing is often an asset in a software company to ensure the reliability and quality of any developed project. The core of software testing strategies mainly lies with unit testing, which is a type of testing carried out on single subroutine or module, almost throughout the development process, depending on the development model used by the developer. There are some issues raised in software testing literature, be it in textbooks, journals, or articles, regarding the process of unit testing, its importance, and the problems faced by developers in performing unit tests. However, through the research and readings of the author on the literature, these issues have not been formally discussed in a Malaysian perspective. Hence, this study of the unit testing practice of software development companies operating in Malaysia is carried out. After reviewing the current available literature on unit testing with no specific geographical or timeframe limitation, a study is carried out on five software development companies, within the geographical limit of Malaysia and as of the current time, on their engaged practice of unit testing. It is meant to gauge the importance they place in unit testing, as well as to find out their methods, their processes, and the problems they encounter in unit testing. These findings are then analyzed and compared to the current literature. It is hoped that these case studies will shed more light on the real-world practice of software development, especially of unit testing, in a Malaysian context.

ABSTRAK

Pengujian perisian dianggap sebagai salah satu tahap terpenting dan kritikal di antara tahap-tahap terakhir pembangunan perisian. Pengetahuan mendalam dan pengalaman dalam pengujian perisian acapkali menjadi aset syarikat perisian dalam memastikan reliabiliti dan kualiti setiap perisian yang dibangunkan. Strategi terpenting diantara strategi-strategi pengujian perisian kebanyakannya berlandas pada pengujian unit, iaitu pengujian yang dilakukan ke atas sesuatu sub-rutin atau modul, hampir disepanjang pembangunan perisian bergantung kepada model pembangunan perisian yang dipilih oleh pembangun. Beberapa isu telah dibangkitkan di dalam kesusasteraan pengujian perisian, sama ada di dalam buku-buku rujukan, jurnal-jurnal, dan makalah-makalah, berkaitan proses-proses pengujian, kepentingan pengujian unit, dan masalah-masalah yang dihadapi oleh pembangun dalam melaksanakannya. Walaubagaimanapun, melalui pengkajian dan pembacaan penulis berkaitan kesusasteraan tersebut, isu-isu ini belum pernah dibincangkan secara formal di dalam perspektif Malaysia. Maka dengan itu kajian tentang pengujian unit ini dilaksanakan di Malaysia. Setelah mengulas kesusasteraan terkini yang tercapai tentang pengujian unit tanpa sebarang had geografi dan masa, satu kajian dijalankan ke atas lima syarikat pembangun perisian di Malaysia tentang praktis pengujian unit yang diambil. Tujuannya ialah untuk melihat dan memastikan kepentingan yang diletakkan, metodologi yang dipilih, proses-proses yang diambil, dan masalah-masalah yang dihadapi dalam pengujian unit. Penemuan-penemuan ini dibincang dan dibanding dengan kesusasteraan terkini. Adalah diharapkan kajian-kajian kes ini akan memberi lebih pengetahuan tentang praktis pengujian unit dalam dunia sebenar, terutamanya praktis pengujian unit, dalam konteks Malaysia.

ACKNOWLEDGEMENT

The development of this work is made successful with the contribution and support from many people, either directly or indirectly. Firstly, the author would like to thank her project supervisor, Puan Norhayati Mohd. Ali for her guidance, support and help through out the work. Secondly, special thanks sincerely to her special friends, Syafril Aznan Mohamad Zubir, Norizan Nasir, and Nurul Amilin, who never gave her support and help in completing this project paper. Thirdly, thanks to Mr. Malik who introduced her to several of those people engaged in unit testing from the five companies studied here. The author really appreciates their help in providing her invaluable information regarding unit testing. Lastly, the author would like to thank all her family members, immediate and extended, for their support during the process of completing this study.

APPROVAL

This project paper is submitted to the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia and was accepted as fulfillment of the requirement for the degree of Master of Science

.....

Puan Norhayati Mohd. Ali

Project Supervisor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

Date: .....

LIST OF FIGURES

Page

LIST OF TABLES

Page

TABLE OF CONTENTS

Page

CHAPTER ONE

INTRODUCTION

Software testing has been regarded as the most important and the most critical component of the software engineering process. It is done right after the design and coding process before the validation and maintenance stage of the Software Development Life-Cycle (SDLC). The main objective of software testing is to discover the errors and bugs of the software that has been developed. A system is extensively tested in order to ensure its reliability. Moreover, the testing will determine whether the software fulfill the user requirements or not. Therefore, software testing plays a vital role in software development.

There are various types of testing in software testing strategy, namely unit testing, integration testing, validation testing, and system testing. This testing strategy may be viewed in the context of The Spiral (Pressman, 2001). All these types of testing should be performed in a sequence in order to conduct test properly and thoroughly. Unit testing begins as the first step of these testing types. It tests a single module or component of software design with the intention of finding errors. Unit testing is conducted to ensure that each piece of code that has been written performs the function that it is designed to do. All single units and modules that have been tested and free from any error will be integrated during the integration process.

This paper discusses about some identified issues of unit testing as theoretically mentioned in books, journals, articles, and other resources. The issues are such as, for instance, unit testing process, its importance, some of its typical problems, and other

related issues and discussion. Therefore, by referring on these issues, the author will describe some objectives and scope of this paper in order to find current knowledge of unit testing practice in five software development companies operating in Malaysia.

1.1 Problem Statement

Unit testing is a part of testing strategies that is practiced by software development companies with the goal of error detection. Unit testing is the earliest type of testing, where the next level consists of integration testing. As a part of software testing strategy, almost all companies engage in unit testing. All the information on unit testing recorded in textbooks, articles, and journals were written by people who are involved in unit testing and those who did some research on the testing, and other resources. However, the information and discussion on unit testing practice in software development companies in Malaysia is not widely spread out.

As a third-world country still considered under development, Malaysia is one, among those countries, that has better future in information technology and computerized system implementation. Almost all companies or organization in Malaysia has engaged computerized system using high levels of Information Technology (IT). This IT awareness among Malaysian people and its organizations led to the widespread of software companies that provide services and facilities to support the implementation of IT. These companies can be categorized into large, medium, and small scale. Each of them are involved in software development, where testing has become the critical issue in order to qualify the products and applications that have been developed.

Among the various stages of software testing, namely unit testing, integration testing, system testing, and acceptance testing, unit testing is the core because it is done at

the earliest stage in testing strategy compared to others. It tests software units and modules before they are integrated.

Concerning the issue of unit testing, this paper tries to find out the practice of unit testing by five software development companies operating in Malaysia, and investigate whether they practice unit testing as theoretically mentioned in textbooks, journals, articles and other resources.

1.2 Objectives

Most, if not all literature about unit testing written in printed documents such as book, articles, and others are theoretical in nature. If they were written based on practical studies, the studies were not done in Malaysia. Thus, this paper serves to achieve its main objectives through a study of unit testing practice by five software development companies in Malaysia. These objectives are:

1. To review the knowledge about unit testing. This discusses about the concept of unit testing, its processes, as well as its importance and the problems faced by testers.

2. To find out how these five companies selected at random practice the process of unit testing and to what extent unit testing is practiced in these companies.

3. To discover other related issue of unit testing such as the benefits of unit testing achieved by these companies and the problems they face.

4. To compare the current practice of unit testing in these randomly selected five companies with the theoretical knowledge on unit testing outlined in books

and other resources, and also with the steps of unit testing activities according to IEEE Standard for Software Unit Testing (1008 – 1987).

The author hopes that by describing all the objectives, it will result in a clear understanding of the purpose of this paper.

1.3 Scope

In order to meet the purpose, the scope of this research is described. Research and study activities were done in randomly selected five software development companies in Malaysia that practice unit testing. In-depth understanding of unit testing, on how the process is run, the importance, the arising problems of unit testing and other related information are discovered. The author hopes that from the research and observation, the practice of unit testing in five software development companies operating in Malaysia can be methodologically gauged and analyzed.

CHAPTER TWO

LITERATURE REVIEW

The explosion of information technology and widespread usage of software all around the world has led people to be increasingly dependent on software as components of any system. However, system failure is another matter of real concern. The concern mostly affects mission-critical software such as those used in the fields such as aviation control, critical communication, and other mission-critical application. Developing a software or application and testing it will challenge even the most established company. The key phase of testing is unit testing that plays a major role in testing efforts and it is the core testing strategy.

2.1 Unit Testing Concepts

Unit testing, literally, is a testing of units or modules. According to Fiedler (1989), unit testing is the first formal test activity performed in the SDLC and it occurs during the implementation phase after each software unit is completed. A software unit can be one module, a group of modules, or a subsystem, and depending on the architecture of the system, it is generally a part of a larger system.

Unit tests are typically designed to test software units, and they form the foundation of upon which the system tests are built. Since software units and unit tests are fundamental entities, unit testing is critical in ensuring the final quality of completed systems (Fiedler, 1989).

Unit test verifies that a small chunk of code that typically is a particular path through a method or function. It typically does not test application-level functionality, because this will be done at integration, acceptance, functional, performance, and other type of testing (Thomas & Hunt, 2002).

Unit testing begins when the design start and continues until the design is stable. It is iterative with design where test results validate the proposed design or lead to modifications. Unit testing is a classic type of testing, usually a "white box" test of an individual module (Dichter, 1993).

2.2 The Process of Unit Testing

Pfleeger (1998) mentioned that unit testing follows exactly the step of software testing. Firstly, the code is examined by reading through it, trying to spot algorithm, data, and syntax faults. Secondly, the code is compiled and remaining syntax faults are eliminated. Lastly, test cases are developed to show that the input is properly converted into the desired output.

According to R. Venkant Rajendran, Director of Deccanet Ltd. on his paper titled "*White Paper on Unit Testing*", unit testing must employ a number of effective testing techniques, which are functional testing, structural testing, and heuristic or intuitive techniques. All these techniques are used to catch the defects that occur in software that can be classified as omissions, surprise, and wrong implementation.

Several guidelines should be followed to ensure the effectiveness of unit testing done by many developers from various types of companies. Different company will perform different steps of unit testing accordance the resources allocated for that activity. John Robbins suggests that there are a few techniques that make unit testing more

effective (Robbins, 1999). One of the techniques consists of four distinct areas that can be divided into prerequisites, before coding, during coding, and after coding:

1. **Prerequisites**

   This first prerequisite consists of version control and bug tracking system. These two pieces of software are important to make unit testing an effective effort.

2. **Before Coding**

   At this point, two ideas should be kept in mind. The first is to start writing unit tests as soon as the code is written because they need to be developed in parallel. Second, to think about the techniques to apply in order to test something that is not yet written.

3. **During Coding**

   Unit tests should be run all the time during coding. Whenever the code is changed or a new code is added, unit tests must be ran again to check whether anything is broken.

4. **After Coding**

   Code coverage is the key to the most effective unit testing. It means executing as many lines of code as possible. A line not executed is a line waiting to rash.

Unit tests are self-explanatory but there is a need to document any key assumptions to ease others when they are running the tests and need references to understand the significance and parameters of the test without wasting time. It is necessary to have Quality Assurance (QA) staff to test the product as a whole and to sign-

off the quality as a whole, while unit tester tests a unit and ensure the quality of that unit. This will result in a higher quality product (Robbins, 1999).

Unit testing is done by developers that originally wrote the code with the aim to prove that the unit works as designed. A unit does not tile with other module except after being tested. All of the path through the module and the interface should be tested with all valid forms of data passed to it (Telles & Hsieh, 2001). According to Jeff Canna (2001), in order to ensure that a particular method of a class successfully performs a set of specific tasks and the expected output is produced, there are some guidelines that need to be followed. These guidelines are based on the lightweight Extreme Programming methods.

1. **Write the unit test before writing code for class it tests**

   Write the units tests first before writing the code that they will test. The class to be tested does not yet exists but pretend that it exists. During writing unit test, the tester may find many syntax errors in the unit test code but should stay with it. Then run the unit test and fix the errors in the unit test. Run the test code again until it passes. The test code is done when the unit test pass itself.

2. **Capture code comments in unit test**

   During writing of the unit test, there is a need to comment some behavior in the code of the unit test so it will be easier for the tester in the future to understand why the test code is written in such ways.

3. **Test all public method of the class**

   However, methods with very straightforward functionality, for example, getter and setter methods, don't need unit tests unless they do their getting and setting in some unique way.

4. **Put each test case in the same package as the class it is testing**

   This type of organization allows each unit test to call methods and reference variables that have access modifiers of package or protected members.

5. **Avoid using domain-specific objects in unit tests**

   Domain objects are specific objects to an application. For example, a spreadsheet application might have a register object if the class already has the domain object. It is fine to use these objects to test. However, do not tie the objects to the class throughout the tests because all wrapped together, code reuse will be time consuming.

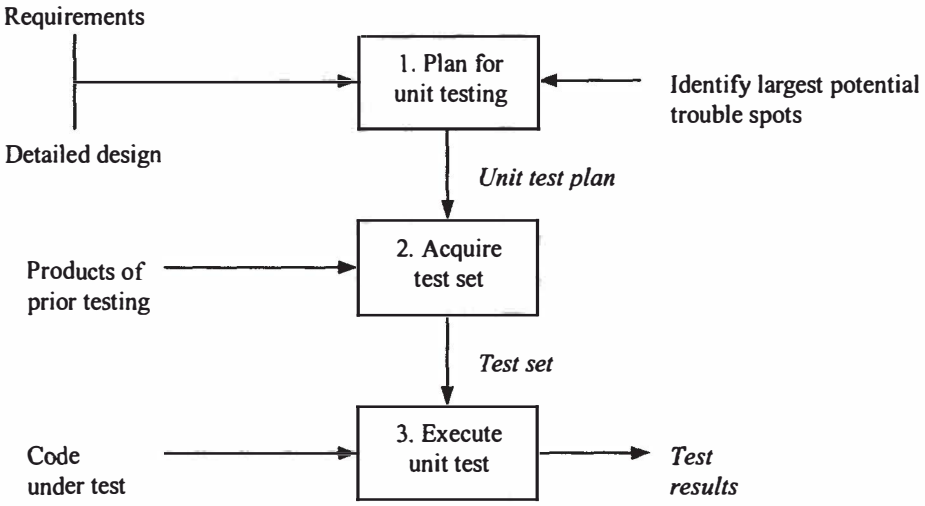
2.2.1 A Typical Unit Testing Road Map

Figure 2.1 below, which is based on IEEE standard 1008 – 1987, shows a typical road map for unit testing (Braude, 2001).

1. The inputs to the test planning process consist of the requirements and the detailed design. Recall that each requirement corresponds to well-defined code (a function, where possible). The detailed design typically consists of additional classes and methods. These are important to the quality of the application, and they must be tested to the same degree as the individual requirements. The output of test planning process is a unit test plan.

2. The next step is acquisition of the input and output data associated with each test. Some of these data may have already been acquired for prior testing (e.g., previous iterations, previous versions of the product, or prior releases). The product of this step is called the test set.

3. Finally, the tests are executed.



Source from Braude, Eric J., 2001

Figure 2.1 Road Map for Unit Testing

The three phases of unit testing process mentioned above (Figure 2.1) can be expanded into eight activities. These activities of unit testing based on IEEE Standard for Software Unit Testing (1008 – 1987) are as follows:

1. Plan the general approach, resources, and schedule.

2. Determine requirements-based characteristics to be tested.

3. Refine the general plan.

4. Design the set of tests.

5. Implement the redefined plan and design.

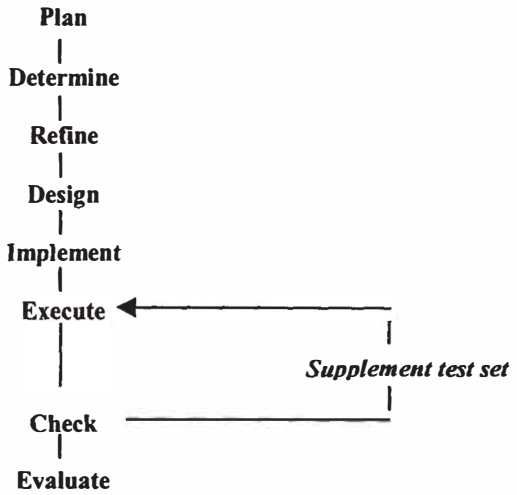
6. Execute the test procedures.

7. Check for termination.

8. Evaluate the test effort and unit.

When more than one unit is to be unit tested (for example, all those associated with a software project), the Plan activity should address the total set of test units and should not be repeated for each test unit. The other activities must be performed at least once for each unit.

Under normal conditions, these activities are sequentially initiated except for the Execute and Check cycle as illustrated in Figure 2.2. When performing any of the activities except Plan, improper performance of a preceding activity or external events (for example, schedule, requirements, or design changes) may result in the need to redo one or more of the preceding activities and then return to the one being performed.



Source from IEEE Xplore

Figure 2.2 Unit Testing Activities

2.3 Planning Unit Testing

The numbers of potential unit to test are usually very large. A systematic approach is needed. The goal is to detect as many errors as possible at a level as serious as possible (Braude, 2001). There are six steps involved in the planning for unit testing.

2.3.1 Decide on the Philosophy of Unit Testing

The first issue is to identify what are "units". A unit can be a whole application, a part of the application, or a function of an application. It depends on the type of application and the approach to its development. For example, in an object-oriented development, unit tests usually test the methods of each class, then the classes of each package, and then the package as a whole (Braude, 2001).

The second issue is to identify who will be doing the units tests. Testing a unit is ideally planned and performed by someone other than the developer of the application (Braude, 2001). Sometimes, they are planned and performed by the QA organization. Commonly, the individual engineers are responsible.

2.3.2 Decide How Unit Tests Will Be Documented

Unit test documentation consists of test procedures, input data, the code that executes the test, and the output data. The unit test documentation can be package with the code itself or included in separate documents.

The advantage of packaging it together is to make it convenient for the tester to understand the tests and the results. The disadvantage is the resulting bloat in the size of the source code.

The executions of unit tests are done by test drivers and utilities and these tools are documented for future use.

2.3.3 Determine the Extent of Unit Testing

The extent of testing has to be defined in a conscious manner since it is impossible to test "everything". For example, if a banking application consists of withdrawals, deposits and queries, unit testing should specify that every method should be tested with an equal amount of legal, boundary, and illegal data. With consideration on the critical aspect of withdrawal and deposit operations, perhaps the methods can be tested three times as extensively as the query method.

The limit of what constitutes unit testing have to be defined. The architect must specify when the testing process should end. In exact, unit should be tested for a fixed time, or until the first three failures are obtained. If no limits are set, it will be impossible for tester to conclude when to stop testing and declare a unit as erratic.

The theme of unit testing is to perform tests that are most likely to expose errors. The testers should prioritize tests in accordance with their likelihood to produce defects thus they can optimize the time spent testing. The tests to be prioritized depend on the unit under test. For example, codes that executes in loops are major source of errors compared to boundaries and interfaces.

2.3.4 Decide How and Where to Get the Test Input

The architect should decide what tools are used to generate test inputs. The testers can use automated tools that feeds randomly created input data, or they can test the in a manual way by inputting the data themselves, or a hybrid of automated and manual