TEST MODELS FOR SOFTWARE TESTING

By

MASNITA AB. GHANI

Thesis Submitted in Fulfillment of the Requirement for the Degree of Master of Science in Faculty of Computer Science and Information Technology

University Putra Malaysia

May, 2005
DECLARATIONS

I declare that the material contained in this project is the end of my own work and that due acknowledgement has been given in the references to ALL sources be they printed, electronic or personal.

SIGNED: ........................................

MASNITA AB.GHANI

DATE: ......................... 9 MAY 2005
This report 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.

Signed: ..........................................

PN. NORHAYATI MOHD ALI.
Project Supervisor,
Department of Information Systems,
Faculty of Computer Science and Information Technology,
University Putra Malaysia.

Date: 9/05/05
Test Models for Software Testing

By

Masnita Ab. Ghani

GS 12594

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

May, 2005
ABSTRACT

This study is to analyze the test models that are used in there company referred as Company A, B and C. This three company located in Klang Valley and each of them has a different background of business. The objective of this study is to identify the test models that are used by company handling software testing activities steps; features are analyzed from the three test models. A refinement test models are proposed in this study with relevant steps and features.
Model Penguji untuk Penguji Perisian

Oleh

Masnita Ab. Ghani
GS12594
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Putra Malaysia

May, 2005
ABSTRAK

ACKNOWLEDGEMENTS

Firstly, I thank to Allah S.W.T for giving me the strength, patience and courage to face all the obstacles in life.

Here, I would like to express my sincere thankfulness and appreciation to my supervisor, Puan Norhayati Mohd. Ali for her willingness to give excellent guidance. Thanks for her concern and invaluable support and guidance throughout the period of my studies that lead to success of the thesis.

Lastly, it is my pleasure to acknowledge the support and understanding given by my beloved husband Mr. Azhar Ahmad, my parents Puan Kamsiah Atan and Encik Ab. Ghani Ahmad and my children Syasya and Muhammad Aiman throughout the period of my study.
TABLE OF CONTENTS

Contents                                                                 Page

Title page                                                                 i
Declarations                                                               ii
Approval Sheet                                                             iii
Abstract                                                                  v
Acknowledgments                                                           viii
Table of Contents                                                          ix
List of Tables                                                             xii
List of Figures                                                           xiii

CHAPTER I: INTRODUCTION

1.0 Introduction                                                           1
1.1 Problem Statement                                                      3
1.2 Project Objectives                                                     3
1.3 Project Scope                                                          3

CHAPTER II: LITERATURE REVIEW

2.0 Introduction to Test Model                                            4
2.1 Bayesian Graphical Model (BGM)                                         4

ix
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>27</td>
</tr>
<tr>
<td>2.2</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>40</td>
</tr>
<tr>
<td>4.3</td>
<td>42</td>
</tr>
<tr>
<td>4.4</td>
<td>47</td>
</tr>
<tr>
<td>4.5</td>
<td>49</td>
</tr>
<tr>
<td>4.6</td>
<td>53</td>
</tr>
<tr>
<td>4.7</td>
<td>54</td>
</tr>
<tr>
<td>4.8</td>
<td>57</td>
</tr>
<tr>
<td>Figure No.</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.1</td>
<td>State Transaction in a State Graph that Represents a FSMs</td>
</tr>
<tr>
<td>2.2</td>
<td>Parallel Between a Classical Validation Paradigm and Software Testing</td>
</tr>
<tr>
<td>2.3</td>
<td>Usage Model as a Directed Graph with Transaction Probabilistic on The Arcs</td>
</tr>
<tr>
<td>3.1</td>
<td>Activities in Methodology</td>
</tr>
<tr>
<td>4.1</td>
<td>Test Team Structure</td>
</tr>
<tr>
<td>4.2</td>
<td>General Steps of Testing in Company A</td>
</tr>
<tr>
<td>4.3</td>
<td>General Steps of Testing in Company B</td>
</tr>
<tr>
<td>4.4</td>
<td>General Steps of Testing in Company C</td>
</tr>
<tr>
<td>4.3</td>
<td>The Activities in The Refinement Test Model</td>
</tr>
</tbody>
</table>
CHAPTER I: INTRODUCTION

1.0 Introduction

The important of software testing and its implications with the respect to software quality cannot be overemphasized. Software testing is a critical element in software life cycle and represents the ultimate review of specification, design and coding [1].

The increasing visibility of software as a system element and the attendant “cost” associated with a software failures are motivating forces for well planned through testing. It is not usual for a software development organization to expand 40 percent effort on testing. In the extreme, testing human-rated software can cost three to five times as much as all other step combined [1]. Software testing is a vital part of the software lifecycle. To understand its role, it is important to understand the definition of software testing

Formal process carried out by the specialized testing team in which a software unit, several integrated units or an entire software package are examined by running the program on a computer. All the associated tests are performed according to approved test procedure on approved test case.

Software testing is defined as 'the execution of a program to find its faults'. Thus, a successful test is one that finds a defect. This sounds simple enough,
but there is much to consider when we want to do software testing. Besides finding faults, we may also be interested in testing performance, safety, fault-tolerance or security [2].

Testing often becomes a question of economics. For projects of a large size, more testing will usually reveal more bugs. The question then becomes when to stop testing, and what is an acceptable level of bugs. This is the question of 'good enough software'. It is important to remember that testing assumes that requirements are already validated.

Testing objectives

Software testing objective can divide into two, where it consist direct objective and indirect objectives. First objective is direct objectives are to identify and reveal as many error as possible in the tested software. Second to bring tested software, after correction of the identified error and retesting to an acceptable level of quality and lastly is to perform the required test efficiently and effectively with budgetary and scheduling limitation.

Second objective is Indirect is to compile a record of software errors for use in error prevention by corrective and preventive actions [1].
1.1 **Problem Statement**

Difficulties of various software testing approach therefore the needs of test model are important.

1.2 **Project Objectives**

The project aim is

- Identify test model that used in software testing.
- Analyze test model by looking at steps and common features.
- Study test model that used in company.
- Propose refinement of test model.

1.3 **Project Scope**

This study will focus on structural test model and functional test model.

- Select three companies in Klang Valley.
- Study three test models applied by each company.
- Identify the approach that used in each test model.
- Identify the steps and common features of test model.
CHAPTER II: LITERATURE REVIEW

2.0 What is Test Model

Test Model is modeling technique or approaches that use in the software testing. The approach that used in Test Model is different from one another. In this paper present three test models. There are Bayesian Graphical Model (BGM), Extended Finite State Machines (EFSMs) and Usage Model. Two of the model Bayesian Graphical Model (BGM) and Usage Model are statistical based model. The rest of the test model is structural and functional based model.

2.1 Bayesian Graphical Model

Bayesian Graphical Model (BGM) is derived from Bayesian statistical methodology, which is characterized by providing a formal framework for the combination of data with the judgments of experts such as software testers [3][4]. BGM present formal mechanisms for the logical structuring of the software testing problem, the probabilistic and statistical treatment of the uncertainties to be addressed, the test design and analysis process, and the implication of test result[3][4].
Test Procedure

The test procedure for the model is as follows: select a number to test. If the test fails then the software is modified and retested. The BGM for the software is adjusted to reflect information about the nature of the software failure and the belief about likely success in fixing the problem without new faults.

For test which are successful, the probabilistically propagate the implication of the success across the BGM. This reduces the current probability of software failure for many of the various nodes on the model and particularly for those nodes most strongly connected to the node where we have observed a test pass. Then need to choose a further test.

The fixing and retesting each time when the fault occurred and update the probabilities for successful test until exceed test resources or have reach the point where the probability that the software is reliable is sufficiently high that there is no need for further testing. This criterion may be refined if there are several different types of potential faults, to terminate with low probability for faults with major consequences but to tolerate a higher probability for faults with minor consequences.

The BGM approach provides probabilities assessments of the reliability of the software being tested before and during the testing process. As such, these
assessments provide a natural approach. There are two criteria for test suite. First, to judge the software acceptable if all tests are successful, then choose the test suite which maximizes the conditional probability of software acceptability, given success for each test, subject to any constraints. We may access the value of such a termination probability before carrying out the test suite thus may judge a priori whether the resources are sufficient to test the software to required level of confidence.

Second, each time the fault is found, there may need for regression testing. Typically prefer to find most of the fault as early as possible in the testing sequence. The test set have to select to optimize a termination probability, then sequence the test so that, at each stage, have to chose the subject to any practical constraints, the test with maximum probability of finding a fault given that each previous test has been successful.

BGM approach is straight forward to design tests to take account of different levels of fault consequence and show how regression testing and test-retest cycle can be accommodated and resolved.
How to create BGMs for Software Testing?

The process of developing a BGM corresponding to software system that needs to be tested is generally as follows:

1. A list of Software Action (transaction) is prepared.
2. Software action are sequence (if required)
3. Related SAs are connected to take account that different SAs may not fully independent.
4. The input spaces are partitioned
5. SAs are converted into BGMs
6. Conditional probabilities on the arcs of the BGMs are elicited
7. Probabilities of nodes without arcs feeding into them are elicited
8. Prior probabilities for observed node are assessed
9. relevant in cases where one test actually tests several software actions

Step 6 - 7 allow the expert knowledge to be taken account. Step 9 is relevant in cases where one test actually tests several software actions.

Step 3 and 4 are essential in this method, there are three situation of interest suppose that are two SAs A and B forming parts of two transactions. A and B is common if A and B are the same piece of code so that test of A with a given set of inputs necessary also tests B if the details are identical. A and B is related if before you carry out a test of A, you expect the test information about
the reliability of B. A and B are independent if you believe the test of A cannot
give you information about the reliability of B.

Software Action may be related for various, for example they may share some
code or use similar algorithms or codes may supplied by the software supplier.
In this case, the actions remain distinct, but related. The input spaces for
different software actions will often different, so the entire input domain must
be partitioned into sub domain which share the same software actions.

Using the BGMs for Testing.

Mapping of Domain Nodes to Tests

Any tests that carry out will result in observation of subsets of the domains
nodes the various graphical models constructed to represent the software-testing
problem. It is important to map the test to the domain nodes. Each test is likely
to test several aspects of the problem. One simple way of forming the mapping
is to list all possible domain nodes, list all possible tests and form matrix
showing which test result in observation.

This is straightforward when there already exists a given test suites. The focus
will be on test design. To avoid unnecessary duplication the mapping of domain
nodes to test is mapping to potential test and must be carried out at the SAs
level example the stage reached before partitioning the input space. Which
partition is then observed by a given test is choosable and becomes one of the features addressed by test design.

**Computation**

All the result required by the approach are computationally straightforward using any package capable of performing the basic algorithms of Bayesian graphical Modeling such as Netica™ (Norsys Software Corporation, Vancouver Canada) and HUGIN Expert A/S Aalborg, Denmark, which provide inter alia, libraries of C routines providing BGM tools.

**Preposterior Analysis Assuming Existing Test Suites**

Preposterior analysis of a given test suite provides an assessment of the posterior reliability of the software, assuming that it passes the given test suite and is the key to accessing the efficiency of a particular test suite. Before running the tests, we can examine the implications for the model of all tests passing. It is straightforward to compute these implications for the various reliability measures we employ. Similarly, it is straightforward to explore the implications of test failures and alternative sequences of the proposed test suite so that we may use such assessment to compare and select between different potential test suites.
Simplifying and Sequencing Test Suites

Given a test suite, can calculate the implication of any subset of tests and any sequences of tests. This enables a study of overlap of tests. It is easy to sequence a given test suite; for example for early selection of tests which test software area with high remaining probability of failure.

Design and Analysis of Test Additional to a Test Suite

To use the outputs of the BGM to identify software areas which would contain substantial unreliability even if all of the tests in the test suites were to be successful and so to design extra tests to tackle these areas.

Design of New Test Suites

A simple approach is to choose tests in order to reduce the probability of at least one fault remaining in the software being tested by considering all possible test in turn and assessing the implication for the model assuming a successful test for each test. This will produce one or more tests with the biggest gain in the chosen performance measure. This test is selected as the first test to be run. This process will be repeated until the desired criteria will achieve. This algorithm should produce one which near optimal.

Test design must take account of constraints such as sequencing. It is simple to update the BGM using composite test which does take account of
sequencing and in principle easy to take account of batch constraint for test
design. A further consideration in efficient test design is that it is useful to be
able to sequence tests according to some criterion.

**Posterior Sensitivity Analysis**

This is important because it helps to inform decisions as to when the software
will be ready for release, given specific test suites. A simple approach based on
one-off changes to root nodes is a follow

1. Specify and calibrate the model.
2. Apply the given test suite to the model and calculate the desired
   posterior summaries assuming that all test are successful
3. Leaving all other root nodes unchanged for each root node in the model
   in turn, we take the probability of at least one fault and alter it on an
   appropriate scale.
4. Then apply the given test suite to the altered model and calculate the
   desired posterior summaries.
5. The differences between the posterior summaries for the initial and
   altered models measure the sensitivity of our conclusions to the
   probability specification for the altered root node.