Ain Shams University
Faculty of Computer and Information Sciences
Information Systems Department

OPTIMIZATION OF VALIDATION AND VERIFICATION FOR SOFTWARE QUALITY IMPROVEMENT

Thesis dedicated for the degree of Master in computer and Information sciences
Submitted to the Department of Information Systems
Faculty of Computer and Information Sciences
Ain Shams University – Cairo

By

Pakinam Boghdady Nour-El-Din Boghdady
B.Sc. in Computer and Information Sciences (2007)
Ain Shams University – Cairo

Under the supervision of

Prof. Dr. Mohamed Fahmy Tolba
Scientific Computing Department
Faculty of Computer and Information Sciences
X-Vice President of Ain Shams University – Cairo

Prof. Dr. Mohamed Hashem Abdel-Aziz
Information Systems Department
Faculty of Computer and Information Sciences
Vice Dean for Students affairs-Ain Shams University – Cairo

Dr. Nagwa Lotfy Badr
Head of Information Systems Department
Faculty of Computer and Information Sciences
Ain Shams University – Cairo

Cairo 2012

ACKNOWLEDGEMENT

I acknowledge my deep gratitude to ALLAH for all gifts and blessings in my life. And for helping me to complete this work on a level that I hope will please the reader. Special thanks are due to my supervisors; Prof. Dr. Mohamed Fahmy Tolba, Prof. Dr. Mohamed Hashem and Dr. Nagwa Badr at the Faculty of Computer and Information Sciences, Ain Shams University for their brilliant guidance, encouragement, constructive feedback, resourceful ideas and suggestions that helped in enhancing my thesis.

Special thanks go to Prof. Dr. Ahmed Mohamed Hamad, the dean of the Faculty of Informatics and Computer Science, The British University in Egypt who continuously supported me, supplied me with useful resources and followed my progress either directly or indirectly helping me to finish this work in the moments I thought it never will.

Finally with all my appreciation and love that no words can express, I dedicate this Master's to my beloved family members for their love and support, my dearest parents and my brother Nour. They provided me with love, prayers, blessings, and care all the way through the work. They are behind any success in my life.

ABSTRACT

The validation and verification processes are an essential and integral part of the software development process. Many Software Quality Assurance SQA standards have set guidelines for these processes. Software testing can be of two types white box or black box testing. The white box offers verification of the software under test while the black box testing offers its validation at several levels. In software development practice, software testing in general accounts for as much as 50% of the total development effort. The complexity of software projects is increasing rapidly and in turn both cost and time of the testing process have become a major proportion of the software development process. It is therefore imperative to reduce the cost and improve the effectiveness of software testing by automating the testing process. The testing effort is divided into three parts: test case generation, test execution, and test evaluation. Test case generation is the core of any testing process and automating it saves much time and effort as well as reduces the number of errors and faults. In the past decades, a great amount of research effort has been spent on automation of software test. Research on the automation of test cases generation is also approaching maturity in the sense that the gap between theory and practice is narrowing.

The success of the software validation and verification does not only depend on good thinking of test cases or good preparation of the testing environment, procedures and criteria but depends much greater on the proper development of requirements as well as managing these requirements continuously and trying to create a rigid set of test cases that is documented and validated once and used over and over again. Efficient development of the software requirements can lead us to proper understanding of the customer requirements thus proper identification of functionalities.

Test cases, which should be run on the system under test to check that it meets the customers required functionalities at the end of the project, if determined, developed, and validated against the customer needs in the requirements development phase, then this will lead to an increase in the productivity and hence an improvement in the software quality as well when the real testing time comes. Therefore, if the test cases are generated at an early stage when all the customers' requirements are well understood and documented then it will be easy to have test cases that fully cover them.

Besides, when the actual testing time comes the test cases will be ready for execution. Test cases are always developed manually by the QA team (from SRS, Design document, old manuals, MOM...) but the generation of test cases automatically can save us too much time leading to an increase in the Software productivity. The test cases can be generated automatically based on different models like UML diagrams, Finite state machines, Prototypes and others. The testing process that uses the test cases generated from a model is called Model-Based Testing (MBT). The MBT approach depends on three key elements: The model describing the software behavior/structure, the test-generation algorithm (testing criteria) and the tools providing infrastructure support for testing.

This thesis improves the quality of the validation and verification processes by enhancing the performance and efficiency of the testing process through automating the test cases generation process. The improvement of the validation and verification processes plays a tremendous role in improving the overall quality of the software. It proposes an XML-based automated approach for generating test cases from activity diagrams. The proposed model presents an architecture that creates from activity diagrams a special table and a directed flow graph that are used in conjunction with the category partitioning method to generate a final reduced set of test cases. The proposed model applies the cyclomatic complexity metric in a proposed coverage checking process which validates the generated test paths during the generation process to ensure meeting a hybrid coverage criterion. The generated test cases can be sent to any requirements management tool to be traced against the requirements. Moreover, any redundant or infeasible test cases causing activities conflicts are automatically eliminated. The final generated set of test cases is suitable for unit, integration, system as well as acceptance testing. The proposed model is automated and supported by a prototype tool called Test Cases Automatic Generator (TCAG) that is applied to around 87 different case studies in different domains. Experimental evaluation is presented to prove that the model saves time, space and consequently cost, improving by that the overall performance of the testing process. The generated test cases are executed using any automatic capture replay tool.

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

LIST OF ABBREVIATIONS

ADG	Activity Dependency Graph
ADT	Activity Dependency Table
BERT	BEhavioral Regression Testing
BVA	Boundary Value Analysis
CAG	Cyclomatic Activity Graph
CAT	Cyclomatic Activity Table
CCFG	Concurrent Control Flow Graph
CCG	Control Construct Graph
CMMI	Capability Maturity Model Integration
Cowtest	Cow Weighted Test Strategy
CPM	Category Partitioning Method
CRM	Customer Relationship Management
CTL	Computational Tree Logic
CTS	Contracts Transition System
CUTE	Concolic Unit Testing Engine
Dep.	Dependency
DFS	Depth First Search
DOTgEAr	Dataflow-Oriented Test Case Generator
DSE	Dynamic Symbolic Execution
ECCFG	Extended Concurrent Control Flow Graph

EFSM	Extended Finite State Machines
EP	Equivalence Partitioning
FDA	Food and Drug Administration
FM	Feature Model
GA	Genetic Algorithm
GADGET	Genetic Algorithm Data Generation Tool
GUI	Graphical User Interface
GUITAR	GUI Testing framework
IF	Intermediate Format
IFD	Interaction Flow Diagram
IFG	Interaction Flow Graph
IOD	Interaction Overview Diagram
IPS	Instruction Per Second
ISO	International Organization for Standardization
ITM	Intermediate Testable Model
JML	Java Modeling Language
LTL	Linear Temporal Logic
MBT	Model-Based Testing
ML	Maturity Level
OCL	Object Constraint Language
PA	Process Area
ParTeG	Partition Test Generator

PMP	Process Model Profile
QML	Qtronic Modeling Language
RBT	Requirements-Based Testing
RD	Requirements Development
REQM	Requirements Management
RTM	Requirements Traceability Matrix
SAL	Symbolic Analysis Laborator
SAL-ATG	Symbolic Analysis Laborator-Automatic Test Generator
SCOOTER	State COllabOration Testing EnviRonment
SCOTEM	State COllaboration TEst Model
SDG	Sequence Diagram Graph
SEI	Software Engineering Institute
SPL	Software Product Line
SQA	Software Quality Assurance
SRS	Software Requirements Specification
SUT	System Under Test
TCAG	Test Cases Automatic Generator
TPD	Test Paths Details
TS	Technical Solution
TTCN-3	Testing and Test Control Notation version 3
TTM	Test cases Traceability Matrix
UAT	User Acceptance Test

UDG	Use case Dependency Graph
UIT	User Interaction Test
UML	Unified Modeling Language
VAL	Validation
VER	Verification
WATM	Web Application Test Model
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 Overview

The certification is "A written guarantee that a system or component complies with its specified requirements and is acceptable for operational use" [1]. Certification can be applied to organizations or individuals, tools or methods, or systems or products. Certification of organizations aims at assuring that the organization achieves a certain level or proficiency and that they agree to certain standards or criteria. Certification is usually applied to areas such as Software Quality Assurance (SQA).

Validation and Verification processes are the tools of the SQA; thus defined by several standards set by different governmental organizations which are responsible for the certification of different products. For example, the FDA standard [2], IEEE Standards [3,4,5], ISO 9000 certification [6,4] and Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI) Standards [7,4] are set by the Software Engineering Institute (SEI) for managing and guiding the organizations by setting improvement goals and practices for improving them.

The Validation process is defined by the FDA standard as: “Confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled” [2]. And as defined by the IEEE standards: “Confirmation by examination and provisions of objective evidence that the particular requirements for a specific intended use are fulfilled” [3,4,5]. The ISO 9000 standard defines the validation as: “It is a process that uses objective evidence to confirm that the requirements which define an intended use or application have been met. Whenever all requirements have been met, a validated status is achieved. The process of validation can be carried out under realistic use conditions or within a simulated use environment.” [6,4]. The CMMI v 1.3 defines the validation process area as follows:

“The purpose of Validation (VAL) is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment” [7,4]. The Validation (VAL) process area is related to many other process areas like: Requirements Development (RD), Technical Solution (TS), Verification (VER) and Requirements Management (REQM).

The Verification process is defined by the FDA standard as: “Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled” [2]. And as defined by the IEEE standards “Confirmation by examination and provisions of objective evidence that specified requirements have been fulfilled” [3,4,5]. The ISO 9000 standard defines the verification as: “It is the process that uses objective evidence to confirm that the specified requirements have been met. Whenever specified requirements have been met, a verified status is achieved.” [6,4]. The CMMI v 1.3 defines the validation process area as follows: “The purpose of Verification (VER) is to ensure that selected work products meet their specified requirements” [7,4]. Verification (VER) is related to many process areas like: Validation (VAL), Requirements Development (RD) and Requirements Management (REQM).

The V-model [8] explains that the relation between the validation and verification processes is overlapped and that they occur through all the phases of the software development lifecycle. It shows that the development, verification and validation are inherently incremental and they can all move in parallel with different level of detail.

The testing process is the foundation of the validation and verification processes [9,10]. The testing process is composed of three main sub processes: The test case generation, the test case execution and the test cases evaluation, having the test case generation its core sub process. Model-Based Testing (MBT) is one of the most famed approaches used to automate the test case generation sub process.

Model-Based Testing (MBT) refers to the type of testing process that focuses on deriving a test model using different types of formal models like the constraint logic programming, symbolic execution, genetic models, event flow models, Markov chain models and UML models. Those test models are then converted into a concrete set of test cases [11-16]. Those formal models are generally classified into three main categories: requirements models, usage models, and source code dependant models. The requirements models can be behavioral, interactional, or structural models. During the