UNIT TEST CODE GENERATION TOOL FOR LOWER LEVEL PROGRAMMING LANGUAGES

Kangana Mudiyanselage Parakrama Danesh Rasika Bandara

(179307J)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2020

UNIT TEST CODE GENERATION TOOL FOR LOWER LEVEL PROGRAMMING LANGUAGES

Kangana Mudiyanselage Parakrama Danesh Rasika Bandara

(179307J)

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Computer Science specializing in Software Architecture

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2020

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

....

Kangana Mudiyanselage Parakrama Danesh Rasika Bandara Date

The above candidate has carried out research for the Masters thesis under my supervision.

....

Dr. Indika Perera Date

ABSTRACT

In the software development life cycle, there are a few, well-known, major phases and their sub-phases. Those are, namely, requirement engineering, feasibility study, design, development, testing, deployment and maintenance. Within those, the most likely sub-phase to be overlooked is the unit testing, in the testing phase. One of the main reasons for such negligence is that the cost it takes for unit testing. This cost is in the aspect of the effort, the amount of human resources to be put for unit testing. Most of the time, the project managers and the other responsible personnel, trade-off between carrying out unit testing and the cost it would take, and, either, neglect unit testing or carry out unit testing lightly.

In the case where carrying out unit testing lightly, it could be either writing trivial unit test code, or carrying out unit testing by debugging the code for various cases for functional units. In these cases, the code could not be tested enough at all. Or, there would not be any valid evidence to prove that a comprehensive unit testing has been carried out.

It is important to have a very good balance between carrying out comprehensive unit testing, keeping solid evidence of unit testing and saving the cost it would incur for unit testing. Considering all the aspects mentioned above, this research suggests a spreadsheet format as a unit test specification and offers a unit test code generator tool to generate unit test code based on the said unit testing specification. This research has considered using Microsoft Excel as the spreadsheet software to create the unit test specification, as it is widely used and popular. The unit test code is generated for the C++ programming language, which does not have that much of good unit testing frameworks. And, the unit test code is for the emerging unit test framework, Google Test.

The outcome of this research has been applied to five different industrial software system projects and six target functions of each of those projects, which ultimately sums up to 36 target functions. The results have been presented to an expert software architect and his judgement of the generated unit test code has been obtained.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Dr. Indika Perera, who guided me from the initial stages from this work to complete it successfully, providing the much-needed advice, material, and the knowledge.

Then, my sincere thanks go to my mother, brother, and my wife, not only being patient with my academic work committing my family time but by supporting me and encouraging me to continue this work till it succeeded.

At last but not the least, I would like to thank my colleagues, both in my MSc batch and in my office, Metatechno Lanka Company (Pvt.) Ltd., who helped me in various ways for making this work a success. Thank you all.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
CPU	Central Processing Unit
DB	Database
GUI	Graphical User Interface
ID	Identification
IDE	Integrated Development Environment
JML	Java Modelling Language
JPF	Java PathFinder
MPOS	Mobile Point of Sales
NASA	National Aeronautics and Space Administration
NFC	Near Field Communication
OOP	Object Oriented Programming
POC	Proof of Concept
OS	Operating System
QA	Quality Assurance
RFID	Radio Frequency Identification
SAM	Secure Access Module
SQL	Structured Query Language
UI	User Interface
VM	Virtual Machine
XML	Extensible Markup Language