SOURCE CODE OPTIMIZER USING LOCAL OPTIMIZATION APPROACH FOR JAVA BASED GENERIC CODE CLONE DETECTION

IKHWAN MUHAMMAD BIN HILLMEE

BACHELOR OF COMPUTER SCIENCE
(SOFTWARE ENGINEERING)
WITH HONORS

UNIVERSITI MALAYSIA PAHANG

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis/project* and in my opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science in Software Engineering with Honors.

_____

(Supervisor's Signature)

Full Name :

Position :

Date :

_____

(Co-supervisor's Signature)

Full Name :

Position :

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : IKHWAN MUHAMMAD BIN HILLMEE

ID Number : CB15054

Date :

SOURCE CODE OPTIMIZER USING LOCAL OPTIMIZATION APPROACH FOR JAVA BASED GENERIC CODE CLONE DETECTION

IKHWAN MUHAMMAD BIN HILLMEE

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science (Software Engineering) with Honors

Faculty of Computer Systems & Software Engineering

UNIVERSITI MALAYSIA PAHANG

JUNE 2019

ACKNOWLEDGEMENTS

In the name of Allah SWT, the Most Gracious and Most Merciful

First and foremost, praise be to Allah SWT and Alhamdulillah for He has provide me with perseverance, sufficient knowledge, and good health conditions in completing this research.

I would like to say a million thanks to both of my parents; Hillmee Lockman and Elainee Hassan for having the patience, giving endless motivation and resources as well as many other things in the process of completing my bachelor degree in Universiti Malaysia Pahang. Not to forget my siblings; Nimi Hillmee, Diyana Hillmee, and Nadiah Hillmee for helping me in giving thoughtful advices along the way.

To my supervisor; Dr. Al-Fahim Mubarak Ali, I would also like to say a million thanks for all the guidance, advices, ideas and utmost support in completing my research.

Thank you to all the lecturers from FSKKP and fellow colleagues who have directly and indirectly involved in the process of completing this research as well. May Allah bless you of everything good in this world and akhirah.

ABSTRAK

Teknik pengoptimuman kod adalah teknik yang digunakan secara meluas oleh banyak penyelidik dan pengaturcara untuk menambahbaik sistem perisian. Ia digunakan untuk penambahbaikan terutamanya dari segi prestasi, pelaksanaan masa, mengoptimumkan saiz kod dan sebagainya. Terdapat tiga pendekatan yang tersedia untuk pengoptimuman kod, iaitu pengoptimuman tempatan, pengoptimuman global, dan pengoptimuman antara Prosedur. Di samping itu, kajian ini bertujuan untuk meningkatkan model Pengesanan Klon Kod Generik (GCCD). GCCD adalah prototaip model yang digunakan untuk mengesan klon kod. Sebelum meningkatkan model GCCD dengan pengoptimuman kod, kajian ini akan mengkaji mengenai pelbagai teknik pengoptimuman kod, pendekatannya dan juga keseluruhan struktur dan proses model GCCD untuk menggunakan pengoptimuman kod sebagai sebahagian daripada proses di dalam model GCCD. Selain itu, kajian ini hanya memberi tumpuan kepada pendekatan pengoptimalan tempatan. Untuk mendapatkan hasil, dataset dari penanda aras Bellon untuk Java akan digunakan untuk tujuan penilaian. Terdapat empat (4) langkah dalam kaedah yang digunakan dalam penyelidikan ini, langkah pertama ialah mengkaji model GCCD dan fungsinya untuk memahami prosesnya dan bagaimana model berfungsi. Langkah kedua ialah mengkaji teknik pengoptimuman kod. Langkah ketiga ialah menerapkan teknik pengoptimuman kod sebagai sebahagian daripada proses dalam model GCCD. Langkah keempat adalah membandingkan versi prototaip GCCD yang diperbaharui dengan melaksanakan dengan mengintegrasikan proses pengoptimuman kod ke dalam model GCCD yang akan membantu alat prototaip itu untuk mengesan klon kod yang lebih berkualiti. Sebagai kesimpulan, alat prototaip GCCD yang telah diperbaharui dan dipertingkatkan akan berjalan mengikut fungsi asalnya tanpa perlu mengubah hasil keluarannya tetapi boleh dilaksanakan dengan prestasi yang lebih baik.

ABSTRACT

Source code optimization techniques is a technique that is widely used by many researchers and programmers to enhance a software system. It is applied for enhancements especially in terms of performance, time execution, optimizing size of code etc. There are three main available approaches for source code optimization, that is Local Optimization, Global Optimization, and Inter-Procedural Optimization. In addition, this research aims to improve the Generic Code Clone Detection (GCCD) model. GCCD tool is a prototype which is used for detecting code clones. Before enhancing the GCCD model with source code optimization, this research will study about multiple code optimization techniques, its approaches and also the overall structure and processes of GCCD model in order to apply code optimization as part of the process in the GCCD model. Besides that, this research focuses only on the local optimization approach. To obtain the results, datasets from the Bellon's benchmark for Java will be used for evaluation purposes. There are four (4) steps in the method used in this research, first step is to review the GCCD model and its functions in order to understand its processes and how the model works. Second step is to review code optimization techniques. Third step is to apply code optimization technique as part of the process in GCCD model. The fourth step is to compare the enhanced and original version of the GCCD prototype tool as an evaluation. As a result of this research, an enhanced version of the GCCD prototype is created by implementing and integrating code optimization processes into the GCCD model which will aid the prototype tool to detect better quality code clones. As a conclusion, the improved and enhanced GCCD prototype tool will run in accordance with its original functions without having to change its expected output but can be executed with a better performance.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GCCD	Generic Code Clone Detection
UVE-1	Unused Variable Elimination-1
UVE-2	Unused Variable Elimination-2
UVE-3	Unused Variable Elimination-3
UVE-4	Unused Variable Elimination-4
UVE-5	Unused Variable Elimination-5
UVE-6	Unused Variable Elimination-6
UVE-7	Unused Variable Elimination-7
UVE-8	Unused Variable Elimination-8

CHAPTER 1

INTRODUCTION

1.1 Background of Study

In software development, it is normally introduced with two related but distinct notion; which is the functional and non-functional requirements. Functional requirement is defined as function of a system or component, which mainly may involve inputs, calculations, processing, and outputs. Non-functional requirement refers as to how it acts as a supporting role to the delivery of functional requirements, for example, robustness, efficiency, performance and optimization status of a software.

Source code optimization can be categorized under non-functional requirements. The main purpose and goal of code optimization is to utilize source codes by making some changes in the code for a better software quality and efficiency. Code length reduction and the removal of repeatable and unnecessary lines of code are one of the few approaches to source code optimization (Goss, 2013). After a program has been optimized, it will become smaller in size, uses less and minimal resources and has better execution for all input and output operations (Goss, 2013).

Optimization techniques can be categorized into two levels; high level and low level optimization techniques or process. High level optimization is performed in order to optimize the design structure as its goal. This process is usually performed by professional programmers who are able to handle changes in classes, methods, functions, loops in source code and etc (Goss, 2013). Actions such as source code which have been compiled into the machine code by the presence of a tool compiler is called low level optimization (Goss, 2013).

There are numerous existing code optimization techniques which are readily available to be applied for any un-optimized software that needs attention. Constant folding, Constant

propagation, Useless Expression Elimination, Dead Code Elimination are existing optimization techniques which have been used by researchers and programmers to name a few. Each and every technique have different functions and produces different results. Therefore, it is crucial to know what suitable technique to apply to an un-optimized software.

1.2 Problem Statement

Code optimization is a technicality which have always been left out by developers during a software development. Some or few developers tend to use and reuse lengthy and repeatable codes without thinking the possible problematic outcome which will affect the behaviour, quality and efficiency of a software. There are also possibilities where programmers are not capable of writing high quality codes which will result in low quality, performance and bad software executions. Truly understanding a software design structure and writing up the right source codes is key in achieving an optimized and well running software.

For the purpose of this research, code optimization techniques will be applied as part of the the generic code clone detection model. The generic code clone detection model was developed in 2015 (Al-Fahim, 2015). It was specifically developed to detect code clones in java programming language only. The main and primary goal of code optimization is to improve the performance and efficiency of a software as well as maximizing the optimum size of code needed. Hence, the generic code clone detection model will be tested and compared if codes which have been applied with code optimization will bring any difference as to the original version.

1.3 Research Objective

The purpose of this research is to apply and to prove the workability of source code optimization on the generic code clone detection model. Below are the sub-objectives that must be achieved in order for this research to produce satisfactory results:

i. To study the abilities and effects of source code optimization on developed softwares.

ii. To develop Java source code optimizer for GCCD model.

iii. To evaluate the prototype of enhanced Generic Code Clone Detection Model (GCCD).

1.4 Research Scope

In accordance with the research, the scope is as follows:

i. Focuses on Java based programming language.

ii. Focuses on understanding the structure and processes of generic code clone detection prototype.

iii. Focuses on local optimization approach for GCCD Model.

1.5 Thesis Organization

This thesis consists of five (5) chapters. Chapter 1 covers the introduction of Source Code Optimizer Using Local Optimization Approach for Java Based Generic Code Clone Detection. This chapter mainly explains about the background of study for this research, problem statement, objective and scope of research as well as thesis organization.

REFERENCES

Al-Fahim. (2015). Generic Code Clone Detection Model for Java Applications, *16*(December).

Goss, C. F. (2013). Machine Code Optimization-Improving Executable Object Code. *ArXiv Preprint ArXiv:1308.4815*, *8*(June 1986), 2015–2017.

Gotarane, P., & Pundkar, S. (2015). Smart Coding using New Code Optimization Techniques in Java to Reduce Runtime Overhead of Java Compiler, *125*(15), 11–16.

Sarma, A. K. (2015). New trends and Challenges in Source Code Optimization. *International Journal of Computer Applications*, *131*(16), 27–32. Retrieved from http://www.ijcaonline.org/archives/volume131/number16/23535-2015907609

Cordy, J. R., & Roy, C. K. (2011). The NiCad clone detector. In *IEEE International Conference on Program Comprehension* (pp. 219–220). https://doi.org/10.1109/ICPC.2011.26

Bilhaqi, A. Z. (2018). *Enhancing Generic Code Clone Detection Model for C#.NET Platform Applications*.

Enyindah, P., Uko, O. E., Harcourt, P., & Harcourt, P. (2017). The New Trends in Compiler Analysis and Optimizations, (May).

Bellon, S., Koschke, R., Antoniol, G., Krinke, J., & Merlo, E. (2007). Comparison and Evaluation of Clone Detection Tools, *33*(9), 577–591.