Determining the Best Weightage Feature in Parameterization Process of GCCD Model for Clone Detection In C-based Applications

Nurul Syafiqah Zaidi
Faculty of Computing,
Universiti Malaysia Pahang Al-Sultan Abdullah
26600 Pekan, Pahang
nsyafiqahzaidi11@gmail.com

Al Fahim Mubarak-Ali
Faculty of Computing,
Universiti Malaysia Pahang Al-Sultan Abdullah
26600 Pekan, Pahang
fahim@ump.edu.my

Abdul Sahli Fakhrudin
Faculty of Computing,
Universiti Malaysia Pahang Al-Sultan Abdullah
26600 Pekan, Pahang
sahli@ump.edu.my

Rahiwan Nazar Romli
Faculty of Computing,
Universiti Malaysia Pahang Al-Sultan Abdullah
26600 Pekan, Pahang
rahiwan@ump.edu.my

2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS) | 979-8-3503-1093-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSECS58457.2023.10256395

***Abstract*— The term "code clone" relates to code that has been replicated many times in a program. Primarily, Type-1, Type-2, Type-3, and Type-4 serve as the four distinct categories for the classification of code clones. Distinct code clone approaches and tools have been implemented for identifying code clones over the years. To overcome the limitation of generalization in recognizing all types of clones, Generic Code Clone Detection (GCCD) model is developed. The five procedures that make up the GCCD model's foundational structure are pre-processing, transformation, parameterization, categorization, and match detection. However, the preceding GCCD model can only detect all types of code clone in Java applications. In light of this limitation, the study proposes a code clone detection model based on the GCCD model, which has the capability to support other programming languages in various applications. The primary objective of this proposed research is to enhance the process in Generic Code Clone Detection (GCCD) model that can improve the code clone detection result, specifically in C-based applications. To achieve the desired objective, some enhancements in the GCCD model have been recommended which are to propose a constant and weightage for Pre-processing and Parameterization process in GCCD model. The proposed work will be tested in a case study involving four C applications. As determined by the code clone detection results from the proposed enhancement, void with its weightage is the preeminent constant and weightage for the Generic Code Clone Detection Model in C-based applications.**

Keywords—Code clone, GCCD, C-based applications, Computational Intelligence

I. INTRODUCTION

The entanglement of software evolution differs with each other as the technology and software environment expands exponentially [1]. In software development, code cloning is a concept where it is commonly observed to replicate source code through direct duplication, either by duplicating and transferring it or by utilising copy-paste operations, regardless of whether any modifications are made [2]. Within the discipline, experts have identified four terminologies for code clones. Type-1 code clone represents a pair of closely resembling clones, whereas Type-2 involves a pair of clones that are similarly structured but exhibit vaguely modified elements. Type-3 is a code clone with extra modifications from Type-2, including sentence updates and insertions. Type-4 clones exhibit functional equivalence despite semantic changes and syntactic variations. Consequently, upon identifying a defect in a specific portion of the source code, it becomes necessary to test all associated segments in order to pinpoint the identical bug [3][4].

This cloning practice may consequently contribute to the spread of bugs, which greatly influences the operation cost and the safety of the software [5][6]. Various research reviews proposed that very nearly 20-50 percent of huge programming frameworks comprise of cloned code [7]. Subsequently, code clone detection tends to become a popular field of studies in addressing this issue [8]. Code clone can be detected in various programming languages. C is one example of a programming language. C is a extensively employed versatile programming language that finds application in the development of software such as operating systems, databases, and compilers [9][10]. An experimental effort to determine code clone occurrences in C-based applications, clones were found in 5-10% of 400 000 lines of code from evaluated C applications using the prevailing clone detection technique. The lack of clones' removal detection technique in C is one of the causes of this issue [11].

Distinct code clone approaches have been presented primitively. The text-based approach [12], token-based approach [13], metric-based approach [14], tree-based approach [15][16], graph-based approach [17][18], and hybrid approach [19][20] are the six major code clone approaches. The majority of approaches, nonetheless, are unable to identify all varieties of code clones [21]. As a corollary, the code clone detection model was proposed as a mechanism for efficiently recognizing code clones. A code clone detection framework is a systematic technique comprising various code clone approaches or tools designed to aid in the identification of clones [22]. Generic Clone Model [23], Generic Pipeline Model [24] and its enhancement [30], Unified Clone Model [25], and Generic Code Clone Detection (GCCD) Model [26]