# Chapter 44
# Code Clone Detection and Analysis in Open Source Applications

**Al-Fahim Mubarak-Ali**
*Universiti Teknologi Malaysia, Malaysia*

**Shahida Sulaiman**
*Universiti Teknologi Malaysia, Malaysia*

**Sharifah Mashita Syed-Mohamad**
*Universiti Sains Malaysia, Malaysia*

**Zhenchang Xing**
*Nanyang Technological University, Singapore*

## ABSTRACT

*Code clone is a portion of codes that contains some similarities in the same software regardless of changes made to the specific code such as removal of white spaces and comments, changes in code syntactic, and addition or removal of code. Over the years, many approaches and tools for code clone detection have been proposed. Most of these approaches and tools have managed to detect and analyze code clones that occur in large software. In this chapter, the authors aim to provide a comparative study on current state-of-the-art in code clone detection approaches and models together with their corresponding tools. They then perform an empirical evaluation on the selected code clone detection tool and organize the large amount of information in a more systematic way. The authors begin with explaining background concepts of code clone terminology. A comparison is done to find out strengths and weaknesses of existing approaches, models, and tools. Based on the comparison done, they then select a tool to be evaluated in two dimensions, which are the amount of detected clones and run time performance of the tool. The result of the study shows that there are various terminologies used for code clone. In addition, the empirical evaluation implies that the selected tool (enhanced generic pipeline model) gives a better code clone output and runtime performance as compared to its generic counterpart.*

DOI: 10.4018/978-1-5225-3923-0.ch044

## INTRODUCTION

Software maintenance is an important phase in preserving quality and relevancy of software due to advances in technology. Maintenance of a software system is defined as a modification of software product after the implementation of the software to improve performance or to adapt the product to a modified environment (Ueda, Kamiya, Kusumoto, & Inoue, 2006). Software maintenance consumes a substantial amount of the software development life cycle costs. Maintainability is one of the issues in software maintenance. One of the factors that affects maintainability of software is code clone (Roy & Cordy, 2007). Code clone refers to similar copies of the same instances or fragments of source codes in software. Code clone also causes an increase in software maintenance cost. This happens due to frequent changes carried out on clone instances (Deissenboeck, Hummel, Juergens, Pfaehler, & Schaetz, 2010). If a source code in a program contains bugs, there is a possibility that other code clone contains the same bug that requires a fix. Hence, this increases maintenance work not only due to the increase of the number of code clone but also the number of bugs that exist in the code clone itself (Roy & Cordy, 2007).

Although code clone increases software maintenance tasks, software community also acknowledges it as a practice in software development. Software developers tend to clone the codes for various reasons. One of the reasons is to speed up the development process (Hou, Jacob, & Jablonski, 2009). This occurs especially when a new requirement is not fully understood and a similar piece of code is present in the software that is not designed for reuse. Programmers usually clone the code instead of adopting the costly redesigning approach. Other reasons of cloning a code during development includes the application of design pattern or implementation of the same requirement of a software (Gang, Xin, Zhenchang, & Wenyun, 2012).

Current code clone research focuses on the detection and analysis of code clones in order to help software developers in identifying code clones in source codes and reuse the source code in order to decrease the maintenance cost. Many approaches such as textual based comparison, token based comparison, and tree based comparison approaches are available to detect code clone. As software grows and becomes legacy, the complexity of these approaches to detect code clone increases, thus makes it more cumbersome to detect code clones.

The issues that occur in current code clone detection research include conflicting, less distinguished terminology and definition on types of code clone. Furthermore, the evaluation differs as most of the code clone detection tools have their own set of code clone definition that is used for evaluation purposes. Therefore, this chapter aims is to provide a comparative study on current state-of-the-art in clone detection approaches and tools, and also to perform an empirical evaluation on selected clone detection tools. In order to achieve this aim, this chapter focus three main aspects that are:

1. **Code Clone Terminology:** There are various terminologies and definitions regarding the type of code clone. This chapter attempts to unify existing terminologies and definitions. This chapter also looks into scenarios that contribute to code clone.
2. **Code Clone Detection Approaches and Models:** Various approaches and models have been proposed and implemented as code clone detection tools in order to detect code clone. This chapter aims to study the best approach or model that can be used for a comparative study. These approaches are compared and evaluated based on their strengths and weaknesses. Only tools that have a complete set of code clone detection process will be used for the evaluation process.