IJCSC 0973-7391 Vol- 6 • Issue-1 Sep - Mar 2015 pp.155-158 available at www.csjournalss.com

A Review on Code Clones Detection Techniques for Software Systems

Jai Bhagwan[1]
[1]Assistant Professor, Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, India

Abstract: An extensive demand for software professional is increasing because industries are demanding a lot of software systems in order to carry out their work. The professionals are not trained as per need which causes software crisis. This force to reuse the source code for development and the reuse process gives birth to the code clones. Code clones are responsible for making complex software systems and it is difficult to manage these complex systems. So, documentation of clones is required by using detection algorithms or tools. Various tools and algorithms have been invented using many techniques like the textual comparison, metrics analysis, token-based process, data-mining techniques etc. In this paper, various code clones techniques or tools have been analysed in order to choose better technique for future use and research.
Keywords: Code Clones, Textual Analysis, Metrics, Tokens, Clones Types.

1. Introduction

The demand for software applications are increasing rapidly because of the maximum pieces of works in all kind of organizations is going to be computerized. The programming professionals required for this task are not increasing proportionately. In order to overcome this crisis, software professionals choose reusability concept for software development [8]. This creates a lot of clones in a software system, due to which software maintenance cost might be increased as code clones make complex to a system. It requires detecting clones among various software systems and refactoring those clones in order to reduce the software complexity. So code clone detection is one of the most demanded topics of research nowadays. A code segment in a source file that is similar or identical to other segment is called a Code Clone. Due to reusability or copy and paste of code, software code clones come into existence. Source files become hard to modify due to these clones. The modification can be easily done with the help of proper documentation of the clones [1]. In order to detect and organize the clones, various tools have been invented in literature. One of them is the text-based technique which is lightweight and is capable to detect the clones accurately. But it is not as good to detect syntactic based clones or units of code. Another token-based technique outperforms with a higher recall rate but it is lower in case of precision. Parser-based methods of clone detection are good enough to detect semantic and syntactic clones, but with a lower rate of recall values. Metrics-based techniques perform well for the detection of semantic as well as syntactic code clones with a good rate of precision but some of the actual clones cannot be detected using these approaches [3].

1.1 Process of Clones Detection

A clone detection tool or approach can follow the steps shown in figure 1 in order to find out the clones in software source codes. These steps are described below [12]:

- Pre-processing – This is the first step of detection in which the code is partitioned on the basis of its development languages. Any uninterested part is also removed in this step.
- Transformation – In this step, the source code is transformed into an intermediate language. Extraction and normalization are other tasks involved in this step.
- Match Detection – Here, the transformed code is provided to an algorithm in order to find out the matches.
- Formatting –In this step, the clones' pair list obtained by comparison algorithm is converted into a corresponding pair list on the basis of original code.
- Post-processing – In this phase, the clones are filtered by manual or heuristic analysis.
- Aggregation – Clones can be combined in clone classes here in order to reduce the data amount.
-

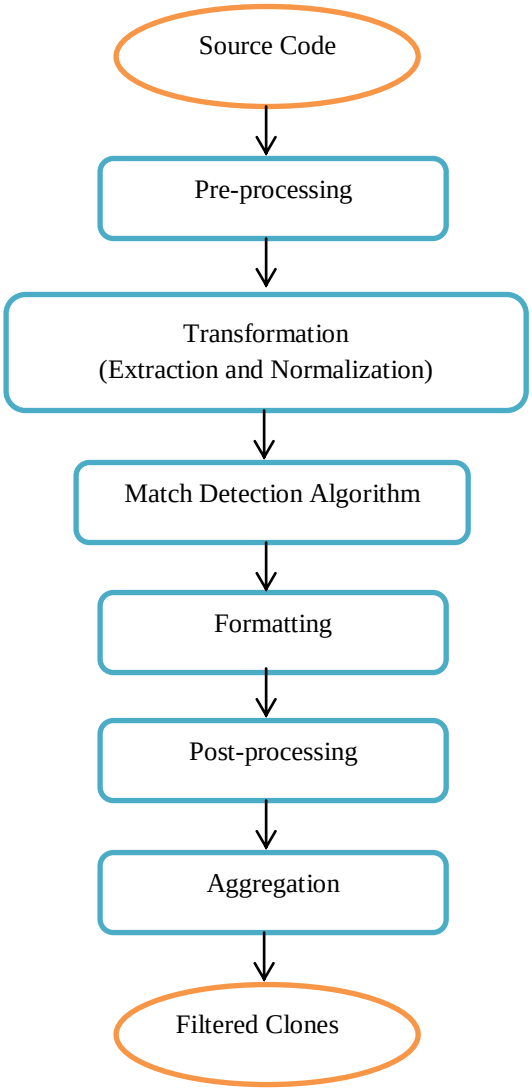

Figure 1. Clone Detection Process

Rest of this research paper coversan overview of existing techniques, evaluation of algorithms, conclusion, and references.

2. Overview of theExisting Techniques

A lot of techniques have been designed and developed for software systems code clone detection. The researchers in [1] proposed a novel method for the detection of software code clones using source text transformation and token comparisons. By these techniques as well as a few optimization methods, a tool was developed which was named CCFinder to detect the software clones from C, C++, Java, and COBOL based software projects. The authors in [2] proposed a method for few higher-level clones in source code based on data-mining method. The tool which was developed using the data-mining technique was given a name as Clone Miner which was experimented with various case studies. In [3], the authors have designed a tool named CloneManager for code clone detection that works based on a LWH (Light Weight Hybrid) technique which is a combination of textual and metrics based approaches. After experiments with C and Java Projects, the tool detected method-level clones accurately. The scientists in [4], proposed a new approach which is based on token generation. This new approach was implemented by developing a tool named Deckard found accurate and scalable while experimenting with Linux and JDK 7 source codes. The scientists in [5] suggested a practical solution for higher-level clones' detection among various files and classes. The authors found clones effectively and accurately using a Frequent Itemset data-mining based approach. The scientists in [6] introduced a technique which is a hybrid method of textual and metrics based approaches. The authors

found that the designed technique is accurate by extracting and comparing various metrics. In [7], the authors described that a new method automatic categorization is effective for software archive. It was found that the function-oriented approach outperformed than object-oriented for software modules classification. Naïve Bayes technique was found effective in term of Precision and Recall over SVD based retrieval method. The scientists [9] introduced a hybrid technique for code clone detection that works on the principal of template conversion and metrics identification. It found by the experiments that the designed technique performs well than existing techniques and it is less complex too. The researcher [10] described three algorithms for software clones detection. The author worked using sub-trees and sequence transformed similarity formula. The authors in [11] designed a technique which a combination of textual and metrics based methods. The newly designed method worked better and was less complex. The authors of [12] gave a deep comparison of various effective tools and techniques in their research paper.

3. Evaluation of Various Approaches

After a literature survey of various research papers, the comparison of existing techniques is shown in table 1.

Table 1. Evaluation of Existing Approaches for Code Clones Detection

Algorithms/ Methods	Techniques Used	Parameters	Findings
Multilinguistic Token Based Approach [1]	Text Transformation Rules, Tokens, Metrics Analysis	CPU Time, Memory, Clone Pairs	CCFinder tool worked effectively
Data Mining Based Clone Detection [2]	Data Mining Techniques	Structural Clones	Scalable Technique
Light Weight Hybrid Technique [3]	Textual and Metrics Analysis	Actual Clones, Detected Clones, Correctly Detected Clones, Precision, Recall	The proposed method found functional clones efficiently
Token Based Approach [4]	Tokens	Execution Time, Clones Quality	The proposed method is faster than previous one
Frequent Itemset Mining Based Technique [5]	Tokens Analysis, Frequent Itemset Mining, Clusters	Similarity Patterns, Structural Clones	Found Similarities at methods, classes and Files level
Clone Detection using Textual and Metrics Analysis [6]	Textual Comparison, Metrics Analysis	Recall, Precision	Average Recall Value
Automatic Categorization [7]	Function Oriented, Object Oriented, Machine Learning	Precision, Recall, F-Measure	Function Oriented is better than Object Oriented Technique, Naïve Bayes is better than SVD based retrieval method
Hybrid Metrics and Template Conversion Technique [9]	Metrics and Template Conversion	Precision, Recall	The proposed technique is accurate in terms of Precision and Recall

4. Conclusions and Future Scope

Many scientists have designed a lot of techniques for code clone detection at different levels like methods level, classes' level, files level etc. In this paper, various clone detection methods or techniques have been compared and found that clones can be detected using textual analysis, metrics analysis, tokens comparisons, data-mining and with various other techniques too. A few authors developed tools like Clone Miner, Clone Manager, CCFinderetc. and found type-1, type-2, type-3 and type-4 clones with accurate rate and complexity. In the future, a method can be designed either by extending an existing technique or combination of more than one methods.

References

[1] T. Kamiya, S. Kusumoto and K. Inoue, "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code," IEEE Transactions on Software Engineering, Vol. 28, No. 7, pp. 654-670, 2002.

[2] H. A. Basit and S. Jarzabek, "A Data Mining Approach for Detecting Higher-level Clones in Software," IEEE Transactions on Software Engineering, pp. 1-18, 2007.

[3] E. Kodhai and S. Kanmani, "Method-level Code Clone Detection through LWH (Light Weight Hybrid) Approach," Journal of Software Engineering Research and Development, Vol. 2, pp. 1-29, 2014.

[4] Y. Yuan and Y. Guo, "Boreas: An Accurate and Scalable Token-Based Approach to Code Clone Detection," ASE 12, Essen, Germany, pp. 286-289, 2012.

[5] H. A. Basit and S. Jarzabek, "Detecting Higher-level Similarity Patterns in Programs," European Software Engineering Conference, ACM SIGSOFT, 2005.

[6] E. Kodhai, S. Kanmani and A. Kamatchi, "Detection of Type-1 and Type-2 Code Clones Using Textual Analysis and Metrics," Internation Conference on Recent Trends in Information, Telecommunication and Computing, IEEE, pp. 241-243, 2010.

[7] P. S. Sandhu, M. Bala and H. Singh, "Automatic Categorization of Software Modules," International Journal of Computer Science and Network Security, Vol. 7, No. 8, pp. 114-119, 2007.

[8] P. S. Sandhu, J. Singh, H. Singh, "Approaches for Categorization of Reusable Software Components," Journal of Computer Science, Vol. 3, No. 5, pp. 266-273, 2007.

[9] G. R. Goda and A. Damodaram, "An Efficient Software Clone Detection System based on the Textual Comparison of Dynamic Methods and Metrics Computation," International Journal of Computer Applications, Vol. 86, No. 6, pp. 41-45, 2014.

[10] K. Greenan, "Method-level Code Clone Detection on Transformed Abstract Syntax Trees Using Sequence Matching Algorithms," Department of Computer Science, University of California, 2005.

[11] E. Kodhai, A. Perumal and S. Kanmani, "Clone Detection using Textual and Metric Analysis to figure out all Types of Clones," International Journal of Computer Communication and Information System, Vol. 2, No. 1, pp. 99-103, 2010.

[12] K. R. Chanchal, J. R. Cordy and R. Koschke, "Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach," Science of Computer Programming, Elsevier, Vol. 74, pp. 470-495, 2009.