Code Cloning: The Analysis, Detection and Removal

Mohammed Abdul Bari
Senior Lecturer
Department of Computer Science
College of Science & Arts
University of Al-Kharj
Wadi Al-Dawasir-11991
Kingdom of Saudi Arabia

Dr. Shahanawaj Ahamad
Assistant Professor
Department of Computer Science
College of Science & Arts
University of Al-Kharj
Wadi Al-Dawasir-11991
Kingdom of Saudi Arabia

ABSTRACT

The coping, modifying a block of code is identified as cloning and is the most basic means of software reuse. It has been extensively used within the software development community. An official survey which is carried out within large, long term software development project suggested that 25-30% of modules in system may be cloned. This paper begins with background concept of code cloning, presents overcall taxonomy of current techniques and tools, and classify evolution tools in two different format as static code clone and dynamic code cloning, this together presented with program analysis, secondly as a solution the static code is divided into four parts as T1, T2, T3, T4, to finally develop a process to detect and remove code cloning.

Keywords

Code Clone, Static Code Clone, Dynamic Code Clone, Legacy Program, Program Analysis.

1. INTRODUCTION

A code fragment CF1, which is a sequence of code line is clone to another code fragment CF2, if they have similar properties i.e. F (CF1) = F(CF2), where "F" is a similar function .Two fragments that have similar properties are referred as clone pair (CF1, CF2) and when many fragment are similar then they form clone class or clone group. [2] Although it has got some short term advantages (e.g. time) [2], but it has a crucial drawback in long run. Several studies show that software with code cloning is more difficult to maintain, then the software without code [3, 4, 5], because the code clone increases maintenance costs [2]. Code cloning is found to be more serious problem in industrial software [2, 6]. It is observed to have negative impact on software evolution. [5] It may adversely affect the software system quality, maintainability and comprehensibility. [6] This paper provides an improved analysis, identification and removal technique for these code clones.

2. CODE ANALYSIS

Program analysis is carried out to extract the embedded knowledge in program code and is divided into two parts. Static code analysis and Dynamic code analysis, which is used to improve software quality and productivity.

2.1 Static Code Analysis (SCA)

This code analysis is performed without execution of program; normally analysis is performed with a formal method with human analysis being called *code review*. [7] SCA is divided into different parts which are shown below in a table 1:

Table 1: Parts of SCA

T1	Identical code fragments
T2	Syntactically identical fragment
T3	Copied modules which is modified in identifier ,types ,layout
T4	Two or more module that are performed same function but are written in different places

2.1.1 Analysis tools

There are no specific tools which can be used for all languages to detect code cloning [8]. The tools are completely language independent. Some of the static code analysis tools are given below:

- RATS: Rough Auditing Tools for security for C, C++ .[8]
- FXCOP: Free Static Analysis for Microsoft from Microsoft.[8]
- CAST: Application intelligent platform which used for Oracle, People soft, .NET,[8]
- Model Checking: It is considered for the systems which have finite state or may reduce to finite state. It checks whether the model meets a given specification based on it input. [8]

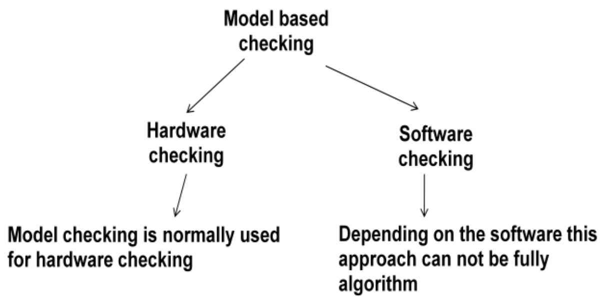
Model based checking
Hardware checking
Software checking
Model checking is normally used for hardware checking
Depending on the software this approach can not be fully algorithm

Fig 1: Model based checking

2.1.2 Data flow analysis

This is the technique for gathering information about the possible set of value that can be calculated at various points in the computer program, since it is easy to compute the information at this point. Source units are to be represented as sub graph in program dependent graph [2]. The techniques that look in to this graph to find clone [10, 11], some matrix based approach is presented to calculate data and control flow metric [12, 13].

2.1.3 Normalization

This is method is optional to remove space, comments.

- **Space:** almost all approach takes less attention to white space, although line base approaches retain break. Some metrics-based approaches however use formatting and layout as part of their comparison [14, 15].
- **Comments:** Most of the approaches remove or ignore comments but Marcus and Maletic [15] used to comment as part of their concept similarity method.

2.1.4 Manual analysis

After selecting the original source code, the clones are subject to manual analyses [16] which are filtered by human experts. Visualization of cloned source code [17] can help in speed up manual filtering step.

2.2 Dynamic Code Analysis

This analysis of code is performed by executing program which is build for particular software. The target program is executed with sufficient test i.e. given input given to produce interesting behavior. The main objective of dynamic code cloning is to reduce debugging time by automatically time by automatically pinpointing the cloning. The cloning program is executed with the tools in order to find the different code clone which is shown in the figure 2. The used tool depends on the type of cloning is wanted. Different types of tools are used for different cloning procedure.

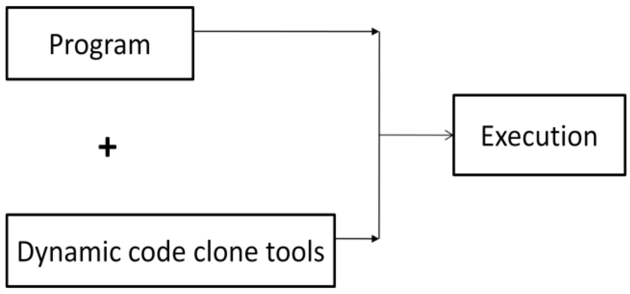

Fig 2: Analysis in executing program

- **Tokenization Tools:** Each line of source code is divided into token based on the lexical rules of the programming language. They are many different rules in acquiring the token for the particular lexical. ex: java strip package name and read class names when the function is only implicit. All white space and comment between the token are remove CCfinder [18] and dup [19]. These are leading rules which are used in tokenizing the source code.
- **Parsing Tools:** The entire source code is parsed to build a parse tree or ATS (abstract syntax tree). The source unit is compare with the parse tree in order to find clone [20, 21]. Metric base approach also be used in order to find cone based on metrics. [13, 15]
- **Dataflow Analysis and Control:** The semantic approaches programs the dependence graph [PDG] [22] from the source code. The nodes are statements and edges represent the control and data dependency, and then look for isomorphic subgroup to find clone [10, 11]. Backward and forward pieces are added to increase the size of isomorphic sub graph until it is not possible to add more pieces. This algorithm is very slow it takes 13 minutes to run 3419 loc. It is more accurate but takes time to process.
- **Match Detection Tools:** These algorithms all strikes between providing more accurate results and running in a useable amount of time. The transformed code is fed into comparison algorithm where comparison is done to find matches, often small comparison units are joining together to find large unit. The output is match detection, which is the list of matches represent to form a set of clone's pairs. Each clones pairs are mark and represented separately [19].
- **Automated Heuristic**: The heuristic rules are defined based on length, diversity, frequency and others characters of clones in order to rank or filter the clone automatically.

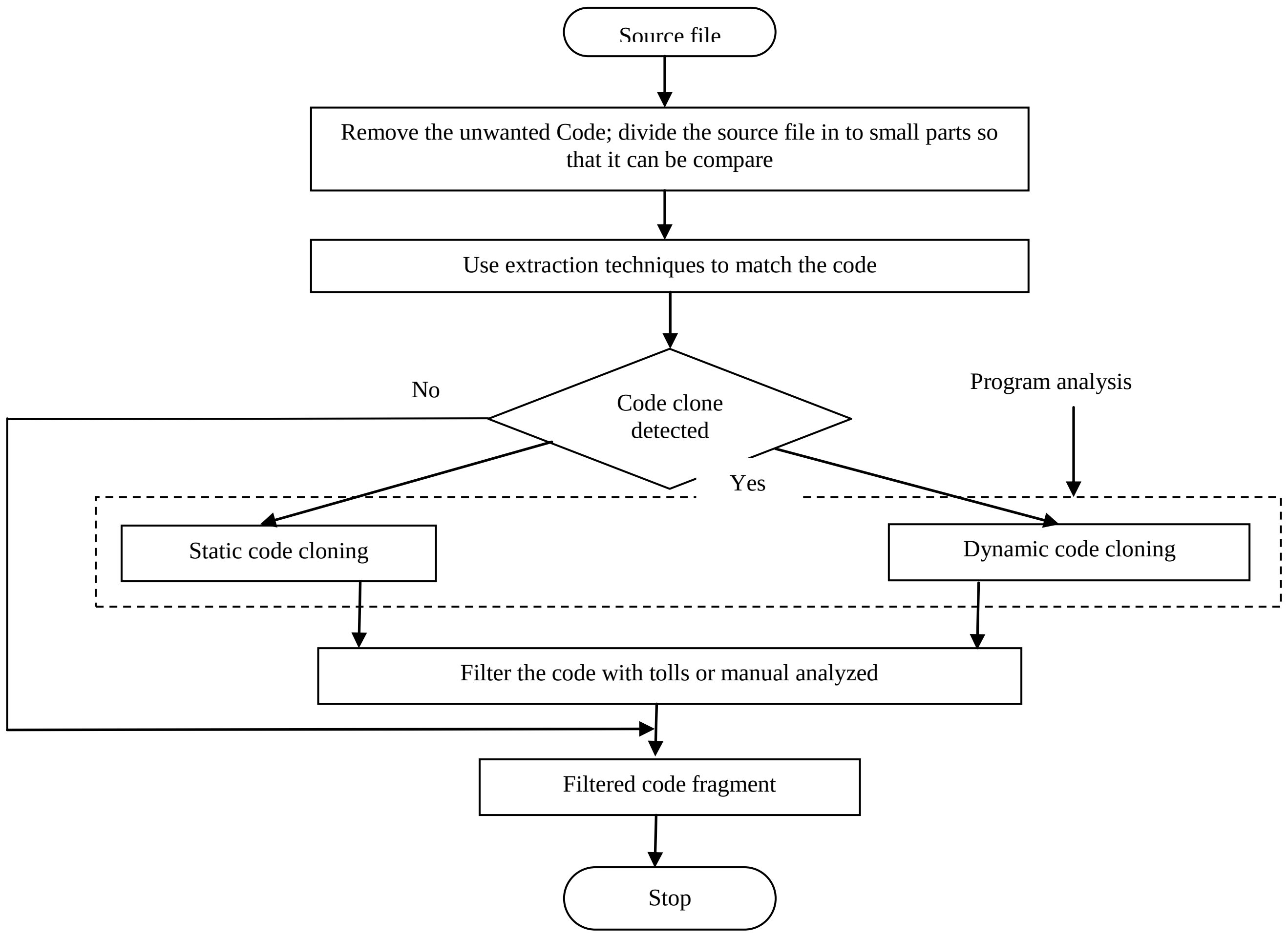
Source file
Remove the unwanted Code; divide the source file in to small parts so that it can be compare
Use extraction techniques to match the code
Code clone detected
No
Yes
Program analysis
Static code cloning
Dynamic code cloning
Filter the code with tolls or manual analyzed
Filtered code fragment
Stop

Fig 3: Clone detection process

3. CLONE DETECTION

A code detector tries to find pieces of code which have high similarities in the system source text. The detector compares every possible clone and tools supports may be required to identify the actual clone .This section provides the basic steps in clone detection process. The process is presented in fig 3.

4. CLONE REMOVAL

The fig 4 depicts the process of code clone removal. When the clone is detected, only the conceptual view of the clone is seen, if it is similar to the clone which was detected before, then the similar old method is applied.

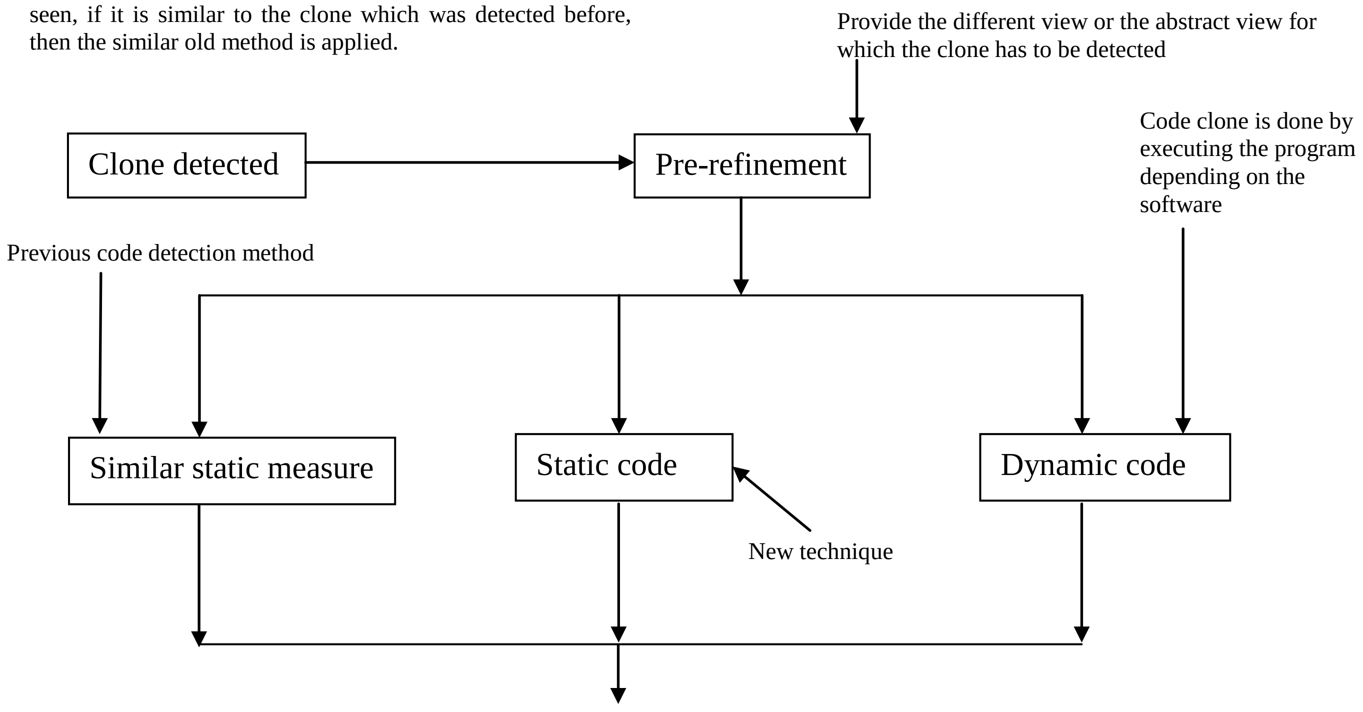
Provide the different view or the abstract view for which the clone has to be detected
Clone detected
Pre-refinement
Code clone is done by executing the program depending on the software
Previous code detection method
Similar static measure
Static code
Dynamic code
New technique

Fig 4: Clone removal process

If it is static code, as it is language independent a new method is applied. If it is a dynamic code then the clone program is executed with the tools, different type is code clone tools are there for different software.

5. CONCLUSION

Clone detection is live problem in industry and an active research area with plenty of work on detecting and removing clones from software. Code clone detection and removal is still not settled well. Code clone tools should be incorporated into standard IDE to achieve widespread adoption; CCFinder is one of the few tools that do a good job and providing only interesting clone while still running in a reasonable time. The result of this paper may serve as a roadmap to a potential user of clone detection techniques, to help them in selecting the right tools or techniques for their interests.

6. REFERENCE

[1] N. Davey, S.D.H Fied, R.J.Frank and D.S.W.Tansley " The Development of a Software Clone Detector", University of Hertfordshire , UK, 1995.

[2] Chanchal K.Roy,James R. Cordy, Rainer Koschke ," Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach ", School of Computing ,Queen's University , Canada ,2009.

[3] Chanchal Kumar Roy and James R. Cordy ," A Survey on Software Clone Detection Research ", School of Computing ,Queen's University ,Canada, 2007.

[4] J Howard Johnson," Identifying Redundancy in Source Code Using Fingerprints", In Proceeding of the Conference of the Centre for Advanced Studies Conference (CASCON'93), Toronto, Canada, October 1993.

[5] Brenda Baker, "On Finding Duplication and Near-Duplication in Large Software Systems", In Proceedings of the Second Working Conference on Reverse Engineering (WCRE'95), Toronto, Ontario, Canada, July 1995.

[6] B. Baker, "On Finding Duplication and Near-Duplication in Large Software Systems", in Proceedings of the 2nd Working Conference on Reverse Engineering, WCRE 1995, (1995)

[7] Wikipedia, "Static Code Analysis ", 2010.

[8] Thomas LaToza, "A Literature Review of Clone Detection Analysis ", 2005.

[9] Martin Johns, "A Practical Guide to Vulnerability Checkers ", University of Hamburg, 2006.

[10] R. Komondoor and S. Horwitz, "Using Slicing to Identify Duplication in Source Code", in: Proceedings of the 8th Int. Symposium on Static Analysis, (2001).

[11] J. Krinke, "Identifying Similar Code with Program Dependence Graphs", in Proceedings of the 8th Working Conference on Reverse Engineering, (2001).

[12] K. Kontogiannis, R. DeMori, E. Merlo, M. Galler, and M. Bernstein, "Pattern Matching for Clone and Concept Detection, Journal of Automated Soft. Engg., 1996.

[13] J. Mayrand, C. Leblanc and E. Merlo," Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics, in proceedings of the 12th International Conference on Software Maintenance, 1996.

[14] N. Davey, P. Barson, S. Field and R. Frank," The Development of a Software Clone Detector", International Journal of Applied Software Technology, 1996.

[15] J. Mayrand, C. Leblanc and E. Merlo," Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics", in Proceedings of the 12th Int. Conference on Software Maintenance, ICSM 1996.

[16] R. Koschke, R. Falke and P. Frenzel," Clone Detection Using Abstract Syntax Suffix Trees" in Proceedings of the 13th Working Conference on Reverse Engg. WCRE 2006.

[17] C.K. Roy and J.R. Cordy, NICAD, "Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization" in Proceedings of the 16th IEEE International Conference on Program Comprehension, ICPC 2008.

[18] 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,2008.

[19] B. Baker," A Program for Identifying Duplicated Code", in: Proceedings of Computing Science and Statistics 24th Symposium on the Interface, 2004.

[20] I. Baxter, A. Yahin, L. Moura and M. Anna," Clone Detection Using Abstract Syntax Trees", in Proceedings of the 14th International Conference on Software Maintenance, 2008.

[21] V. Wahler, D. Seipel, J. Gudenberg and G. Fischer, "Clone Detection in Source Code by Frequent Itemset Techniques", in Proceedings of the 4th IEEE International Workshop Source Code Analysis and Manipulation, 2004.

AUTHORS PROFILE

Mr. Mohammed Abdul Bari is an Information System Architect and expert in handling software process improvement. His research area includes Business Process Reengineering, Process Modeling, Information System Redesign and Reengineering. He did B.E. in Computer Science & Engineering from Bangalore University, INDIA and M.S. in Information Systems from London South Bank University, United Kingdom, currently pursuing Ph.D. in Computer Science from University of Newcastle, District Columbia, U.S.A.

Dr. Shahanawaj Ahamad is an active academician and researcher in the field of Software Reverse Engineering with experience of ten years, working with Al-Kharj University's College of Science & Arts in Wadi Al-Dawasir, K.S.A. He is the member of various national and international academic and research groups, member of journal editorial board and reviewer. He is currently working on Legacy Systems Migration, Evolution and Reverse Engineering, published more than twenty papers in his credit in national and international journals and conference proceedings. He holds M. Tech. followed by Ph.D. in Computer Science major Software Engineering, supervised many bachelor projects and master thesis, currently supervisor of Ph.D. theses.