Automatic Clone Recommendation for Refactoring Using History and Code Metrics

By
Md. Jubair Ibna Mostafa
MSSE0602

Supervised By
Dr. Kazi Muheymin-Us-Sakib
Professor
Institute of Information Technology
University of Dhaka

A Proposal for Research for the

Master of Science in Software Engineering Degree

Institute of Information Technology
University of Dhaka
Dhaka Bangladesh

5th March, 2019

Research Proposal

Automatic Clone Recommendation for Refactoring Using History and Code Metrics

Author
Md. Jubair Ibna Mostafa
MSSE 0602
bsse0614@iit.du.ac.bd
Institute of Information Technology
University of Dhaka

Supervisor
Dr. Kazi Muheymin-Us-Sakib
Professor
sakib@iit.du.ac.bd
Institute of Information Technology
University of Dhaka

Contents

List of Tables

List of Figures

1 Introduction

Duplicate source codes are called code clones. Code clones are created due to copying and pasting source code with or without modification by a developer. It is common in software development because of its usefulness to implement functionality faster and fulfill initial design requirement of similar functionality [1]. However, in the maintenance phase, code clone is responsible for late propagation [2], hidden bug propagation [3] and unintentional inconsistencies [4] problems. So, it is a controversial practice due to its applicability in development and negative impacts on software evolution and maintenance.

To support maintenance of code clone, researchers have conducted many studies towards effective management of cloned code. These management activities can be broadly categorized into three types namely - code clone detection [5, 6, 7], tracking [8] and refactoring [9]. Detection of code clone is required to know the presence of duplication in the source code. Tracking is useful while refactoring is not possible due to association with other non-cloned codes [10]. And, refactoring is needed to remove those clones [11]. However, in practice a small portion of clones are refactored. and refactoring of all clones are not desirable in terms of cost and benefit [12].

In repositories, almost 7% to 23% codes are cloned codes [13]. From this large amount of clones, manually deciding which clones should be refactored and which clones are not, is not desirable. Moreover, excessive refactoring of clone without considering relation with other codes can break the application. In this situation, analyzing the previous history of refactored and non-refactored clones may suggest the current need. Since refactoring preserves external behavior of source code while improving internal structure [14], so that, certain code metrics of cloned refactored and non-refactored codes can aid in taking refactoring decision. Building a classifier through these features (i. e., metrics) and recommending clones for refactoring may assist in this task. However, the research has several challenges to identify desirable clones for refactoring. Automatically detecting refactored and non-refactored clones from version history, covering all types of refactoring strategy that a clone can be refactored, identifying appropriate features for characterizing refactored and non-refactored clones, analyzing the impact of particular character means feature and building an appropriate classifier to perform well in both within and cross-project evaluation etc. are worth mentioning. To do these tasks, processing commit histories also incurs challenges.

Researchers have conducted clone management related research from different perspectives such as visual representation of clones, differences among those, suggesting important clones for refactoring, automatically refactor of clones etc [15]. Lin et al. have conducted a research to find out differences between multiple clone instances (MCIDiff) [16] to support refactoring decision of muli-clone instances. To automatically refactor clones, authors built different plug-in for Integrated Development Environment [17, 18, 19]. Tsantalis et al. used the lambda expression to merge behavioral change clones of type-2 and type-3 [20]. However, in these research, the authors focused on clone refactorability, not desirability. Extracting important clones for refactoring is necessary rather than removing all clones. Eunjong Choi et al., used clone metrics for example length, group size to extract clones for refactoring [21]. However, authors did not use any version history. Mondal et al. proposed an approach to automatically identify important clones for refactoring through mining association rules [10]. Authors used historical change coupling to identify important clones. They extend their work by suggesting clones for refactoring and tracking [11] with similar fashion. To identify tracking candidates, they used cross-boundary relation of changing clones. However, their approach is limited in within project analysis and they did not use any machine learning based approach in their study. Wang et al conducted research to recommend clones for refactoring [12]. Authors used Design, Context, and History to capture many aspects of refactoring decision. However, they didn't use evolving co-changing relation of cloned codes. Moreover, their approach is limited with fewer features and one classifier. Performance in the cross-project is also pure. The state of the art [22] used a more comprehensive study on clone refactoring recommendation with 34 features and 5 classifiers. Authors also compare their result with Wang et al.,[12] and showed that their approach performs better than the previous one. However, their approach is limited to heuristic-based and threshold system. Moreover, the approach can not capture all types of clone refactoring such as form template method refactoring. So, covering all types of clone refactoring and characterizing clone refactoring desirability by history and code metrics help to build a clone recommendation system for refactoring.

2 Objectives

Since code clone has controversial benefits from the maintenance perspective, it is important to know which clone should be refactored and which clones are not. Analyzing version history of refactored and non-refactored clones with their code metrics may help to recommend appropriate clones for refactoring in terms of cost and benefit. This work aids in clone management by answering the following research questions.

- **RQ1:** How do automatically recommend clones for refactoring using clone refactoring history and different code metrics by covering all types of clone refactoring?
  To recommend clones for refactoring, refactored and non-refactored clones need to be identified from version history. By answering the following sub-questions this question can be answered.
  - **SQ1:** How do automatically detect clone refactoring instances from version history?
    By answering this question, we will get an approach to automatically detect refactored and non-refactored clone instances from version history. This aids to further investigate the reasons behind refactoring of clones.
  - **SQ2:** How many clones are refactored in a repository throughout the version history?
    By answering this question, we will get insight about which clones are refactored in development history and which are not. This way labeling clone, helps to identify important characteristics of clones from refactoring perspective.
- **RQ2:** How can the performance of clone refactoring recommendation be improved by characterizing clones?
  Characterizing clone refactoring and building a recommendation system may help in clone management by suggesting clone for refactoring. This question can be answered by answering the following sub-questions.
  - **SQ1:** Which features or metrics can characterize clone refactoring desirability to build a recommendation system?
    Identification of important metrics or reasons of clone refactoring can help to characterize clone desirability. This helps to build a recommendation system, which help to recommend clone for refactoring.

- **SQ2:** How does the recommendation system perform in terms of accuracy in within and cross-project evaluation?
  This can be answered by analyzing the performance of recommendation system in within project and cross-project that how accurately the system recommend desirable clone. The effect of any particular feature also have to be measured to know the impact in recommendation system.

3 Research Methodology

The purpose of this research is to recommend important clones for refactoring by analyzing the evolution of refactored and non-refactored cloned codes. In order to attain this goal, Table 1 represents a list of research activities, a brief description and relation of each activity to the research questions. Below activities are described shortly.

The first activity is performing the literature study. It helps to acquire knowledge about the research domain. Without knowing the existing knowledge it is difficult to conduct research on that domain. In this state, supporting research areas are also investigated. It helps to think broadly and differently about a problem. It is an ongoing process because it supports every step of the research activities. During the whole research period, it will be continued.

The second activity is to dig down into the research domain and related works. It helps to attain knowledge from existing research works, techniques and tools. Moreover, one can know the state-of-art techniques and tools with existing problems and future research direction in that particular domain. It also helps to identify the limitation of existing works and opens the scope of improvement.

The third activity is developing an approach to recommend clones for refactoring. In this step, challenges and issues are listed and ways of solution to those problems will be identified. A methodology to achieve the goal will be discussed.

To evaluate the proposed approach an experimentation field called testbed will be created. This helps to identify applicability and correctness of the proposed approach. Based on the feedback from experimentation on the test bed, approach, considered metrics will be modified and corrected.

After evaluating the proposed approach in terms of accuracy and effectiveness of code metrics, interpretation will be provided. Our approach will be compared with the state of the art techniques like CREC [22].

In the end, a technical report will be written that contains details of every phase of the research work. Significant result or finding will be published in renowned conferences such as ICSE, APSEC, ICSME. A thesis will be compiled at the end of the research, as a requirement for the Master of Science in Software Engineering degree provided by the Institute of Information Technology (IIT), University of Dhaka.

Step No.	Activity Name	Activity Description
1	Literature Survey	• Reviewing the existing literature and building the background of the research • Find out limitation and scope of improvement in existing research • Writing research proposal • Presenting initial ideas and receiving feedback
2	Understanding code clone's Refactoring, code metrics and recommendation system (i.e, classifier)	• More specific knowledge gain by exhaustively studying the existing works in the research area. • Reporting on the survey
3	Developing an approach to automatically recommend clone for refactoring using history and code metrics	• Assessing issues in clone refactoring recommendation
4	Test bed creation and evaluation of proposed approach	• Standard experimental data selection for experimentation • Evaluate the accuracy of the recommendation system
5	Developing a recommendation model	The model will be evaluated by different projects
6	Technical Report Writing	A technical report will be written and delivered for each of the major activity
7	Publications and Presentations	Review reports and research papers will be prepared for renown international conferences like ICSE, ASE, ICMS, SCAM, APSEC etc
8	Thesis Compilation	At the end, thesis will be compiled in order to fulfillment of Master of Science in Software Engineering

Table 1: Research Methodology

4 Research Timeline

To conduct the research, research timeline starts from June 2018 to mid of 2019. During this whole period, literature study will be continued as an evolving process. Firstly, literature will be studied from the broad domain to gain broader knowledge in the domain. Then a specific domain will be focused and related studies will be conducted. After getting the overall knowledge, an approach will be developed within 2018. At the beginning of 2019, a testbed will be created for evaluating and testing the proposed approach. After that, empirically evaluate the impact of different code metrics on clone refactoring recommendation. A technical report will be prepared based on each activity. Finally, the research paper will be presented to the evaluation committee in mid of 2019. The above description of the timeline is presented in the Figure 1.

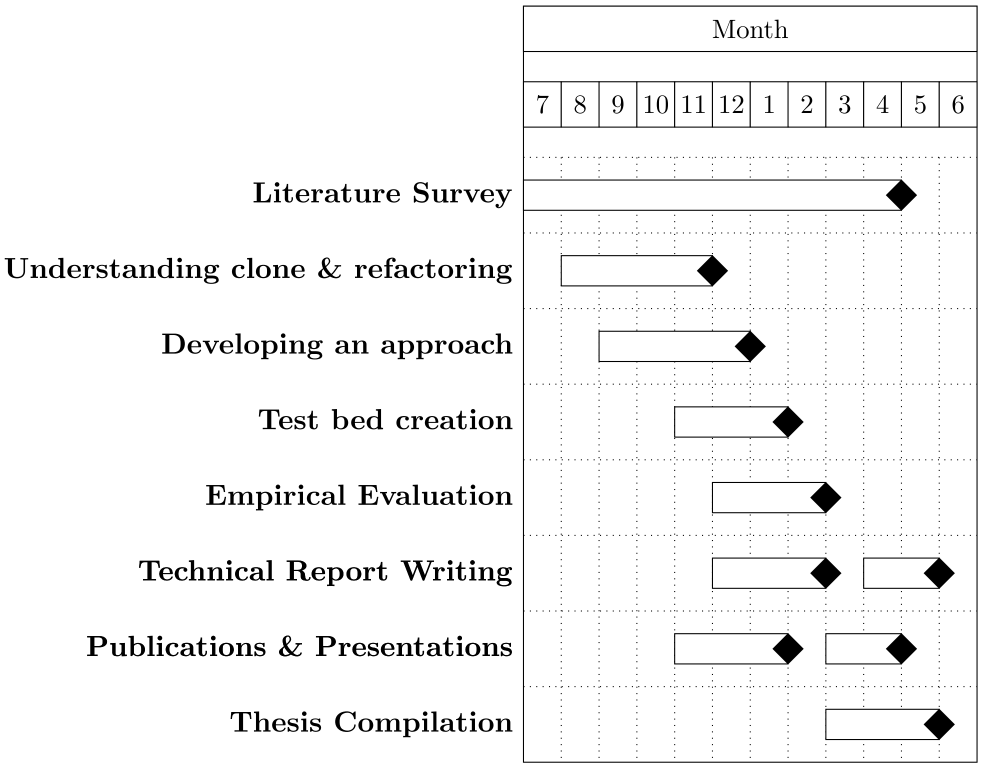

Figure 1: Research Timeline

5 Rationale of the Research

In software development, code cloning is a common technique to implement similar functionality. However, clone existence indicates the low quality of the source code and has a potential risk to create new bugs. So that clone code needs to be managed. Clone refactoring is one of the ways of clone management. Analyzing historical data of refactored and non-refactored clone codes may provide insight of clone management. Moreover, it may suggest which clones should be refactored next in terms of risks and benefits by analyzing different metrics of those data.

Recently Bangladesh has fulfilled all requirements to be a developing country. According to the IMF, Bangladesh's economy is the second fastest growing major economy of 2016, with a rate of 7.1% [23]. Among different contributing sectors like agriculture, manufacturing industry, pharmaceutical industry etc. information technology plays an important role in such economic growth. Bangladesh has earned $800 million by exporting locally made software and providing ICT-related services like outsourced and freelance work in 2017 [24].

The government aims to get the earnings up to the $1 billion this year and $5 billion within 2021 [24]. To achieve this goal, the software industry already focusing on different aspects like code quality, reusability, and maintainability.

To provide quality software, source code should contain less smell and defects. The duplicated source code is one of the types of code smell which is responsible for increasing hidden bug propagation or introduce new bugs on cloned fragments. This ultimately increases maintenance cost and effort. However, all clones in a repository do not have a negative impact on software. So, from a huge list of clones, it is necessary to identify which clones should be removed to get better maintainable software. Thus, this research work can be helpful to accomplish the 20-21 vision of the government.

References

[1] C. J. Kapser and M. W. Godfrey, “”cloning considered harmful” considered harmful: Patterns of cloning in software,” *Empirical Softw. Engg.*, vol. 13, pp. 645–692, Dec. 2008.

[2] L. Barbour, F. Khomh, and Y. Zou, “Late propagation in software clones,” in *IEEE 27th International Conference on Software Maintenance, ICSM 2011, Williamsburg, VA, USA, September 25-30, 2011*, pp. 273–282, 2011.

[3] M. Mondal, C. K. Roy, and K. A. Schneider, “Bug propagation through code cloning: An empirical study,” in *2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, Shanghai, China, September 17-22, 2017*, pp. 227–237, 2017.

[4] E. Jürgens, F. Deissenboeck, B. Hummel, and S. Wagner, “Do code clones matter?,” in *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*, pp. 485–495, 2009.

[5] C. K. Roy and J. R. Cordy, “NICAD: accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization,” in *The 16th IEEE International Conference on Program Comprehension, ICPC 2008, Amsterdam, The Netherlands, June 10-13, 2008*, pp. 172–181, 2008.

[6] N. Göde and R. Koschke, “Incremental clone detection,” in *2009 13th European Conference on Software Maintenance and Reengineering*, pp. 219–228, March 2009.

[7] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes, “Sourcerercc: scaling code clone detection to big-code,” in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pp. 1157–1168, 2016.

[8] E. Duala-Ekoko and M. P. Robillard, “Tracking code clones in evolving software,” in *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*, pp. 158–167, 2007.

[9] D. Mazinanian, N. Tsantalis, R. Stein, and Z. Valenta, “Jdeodorant: clone refactoring,” in *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume*, pp. 613–616, 2016.

[10] M. Mondal, C. K. Roy, and K. A. Schneider, “Automatic identification of important clones for refactoring and tracking,” in *14th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2014, Victoria, BC, Canada, September 28-29, 2014*, pp. 11–20, 2014.

[11] M. Mondal, C. K. Roy, and K. A. Schneider, “Automatic ranking of clones for refactoring through mining association rules,” in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering, CSMR-WCRE 2014, Antwerp, Belgium, February 3-6, 2014*, pp. 114–123, 2014.

[12] W. Wang and M. W. Godfrey, "Recommending clones for refactoring using design, context, and history," in *30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014*, pp. 331–340, 2014.

[13] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Sci. Comput. Program.*, vol. 74, no. 7, pp. 470–495, 2009.

[14] M. Fowler, *Refactoring - Improving the Design of Existing Code.* Addison Wesley object technology series, Addison-Wesley, 1999.

[15] C. K. Roy, M. F. Zibran, and R. Koschke, "The vision of software clone management: Past, present, and future (keynote paper)," in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering, CSMR-WCRE 2014, Antwerp, Belgium, February 3-6, 2014*, pp. 18–33, 2014.

[16] Y. Lin, Z. Xing, Y. Xue, Y. Liu, X. Peng, J. Sun, and W. Zhao, "Detecting differences across multiple instances of code clones," in *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, (New York, NY, USA), pp. 164–174, ACM, 2014.

[17] R. Tairas and J. Gray, "Increasing clone maintenance support by unifying clone detection and refactoring activities," *Information & Software Technology*, vol. 54, no. 12, pp. 1297–1307, 2012.

[18] K. Hotta, Y. Higo, and S. Kusumoto, "Identifying, tailoring, and suggesting form template method refactoring opportunities with program dependence graph," in *16th European Conference on Software Maintenance and Reengineering, CSMR 2012, Szeged, Hungary, March 27-30, 2012*, pp. 53–62, 2012.

[19] N. Tsantalis, D. Mazinanian, and G. P. Krishnan, "Assessing the refactorability of software clones," *IEEE Trans. Software Eng.*, vol. 41, no. 11, pp. 1055–1090, 2015.

[20] N. Tsantalis, D. Mazinanian, and S. Rostami, "Clone refactoring with lambda expressions," in *Proceedings of the 39th International Conference on Software Engineering*, ICSE '17, (Piscataway, NJ, USA), pp. 60–70, IEEE Press, 2017.

[21] E. Choi, N. Yoshida, T. Ishio, K. Inoue, and T. Sano, "Extracting code clones for refactoring using combinations of clone metrics," in *Proceedings of the 5th International Workshop on Software Clones*, IWSC '11, (New York, NY, USA), pp. 7–13, ACM, 2011.

[22] R. Yue, Z. Gao, N. Meng, Y. Xiong, X. Wang, and J. D. Morgenthaler, "Automatic clone recommendation for refactoring based on the present and the past," *CoRR*, vol. abs/1807.11184, 2018.

[23] G. tenders, "Economy of bangladesh." `https://www.globaltenders.com/economy-of-bangladesh.php/`. Online; accessed 14 April 2018.

[24] T. D. Star, "Ict export." `https://www.thedailystar.net/frontpage/ict-export-fetches-800m-2017-1492510`. Online; accessed 14 April 2018.