THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Is Model Cloning a Replacement for Code Cloning?

Ritu Garg
Deenbandhu Chhotu Ram University of Science and Technology, Murthal, India
Dr. Rajesh Bhatia
Punjab Engineering College University of Technology, Chandigarh, India

Abstract:
Model cloning refers to the process of finding duplication in the models whereas in the code cloning, the duplication is detected in the code of the system. Model cloning works at design while the code cloning works at the implementation phase of the software development life cycle. In this paper we will discuss the impacts of both and analyze if they are a substitute or a support for one another. It also describes the nature of software where which type of cloning (Code or Models) must be applied in order to achieve accuracy in less time as the software development is restricted with tight time and budget constraints.

1. Introduction

Clone is termed as a duplicate copy of the original one. Clone detection is a process of detecting clones in the phases of software development process [17].

Types of clones in software [18] are:-

- Code clones
- Model clones

Code clone is the duplication that occurs in the software during the implementation phase in the code. Various types of code clones are [1]:-

- Type 1: Exact clones- These represents the code clones that are identical in all ways excluding white spaces and comments.
- Type 2: Renamed clones- These represents the code clones that are identical in all ways excluding change in identifiers, types, literals, comments and layouts.
- Type 3: Near miss clones- These represents the code clones that are identical in all ways excluding modifications in terms of addition, deletion or modification of one or more statements in the fragments.
- Type 4: Semantic clones- These represent the code clones which are not textually but semantically similar.
- Model clone is the duplication that occurs in the software during the design phase. Various types of model clones are [2]:-
- Type A: Exact model clones- These represents the model clones that are identical in all ways excluding layout, secondary representations, internal identifiers and notes.
- Type B: Modified model clones- These represents the clones that are identical in all ways excluding changes to element names, attributes and parts.
- Type C: Renamed model clones- These represents the clones that are identical in all ways excluding modifications in terms of addition, deletion or modification of parts.
- Type D: Semantic model clones- These represent the clones which are identical in all contexts excluding methodology or language constraints on copying the fragment.

These clones can create update anomalies where change in one copy must be applied in all the duplicate copies which increase risk, time and effort so these clones must be detected and removed to decrease risk, time and effort for the development of the software [16]. It also increases the quality of the software.

2. Literature Review

There are various techniques for clone detection. Some of them are provided here. The code clone detection can be achieved by using various methods such as textual comparison, token based comparison, abstract syntax tree [15], program dependency graphs, metric based and hybrid approach. The works that have been proposed are as under. Cordy et. al. [3] detected near-miss clones through textual comparison. Ducasse et. al. [4] presented a tool duploc on basis of line based comparison. Kamiya et. al. [5] developed CCFinder which uses suffix tree matching to match the longest common prefix. CP-Miner [6] is a tool that uses itemset mining in software to detect bugs using cloning. A tool named CloneDR [7] can detect type 1, 2 and 3 clones using hashing. Wahler et. al. [8] developed a Clone Detection tool that has the capability to detect clones using Abstract Syntax Tree.

Komondoor et. al. [9] presented PDG-DUP that finds clones through isomorphic subgraphs using program slicing. Hummel et. al. [10] uses ConQAT to detect clones using hybrid incremental index based approach.
A little amount of work has been done as compared to code clones. For model clone detection the methods have been proposed are as under. Mainly the work on model clones has been accomplished on Matlab/Simulink models. Pham et. al. [11] developed a tool ModelCD that uses canonical matching in graphs to detect clones in models. Deissenboeck et. al. [12] presented CloneDetective tool to detect clones using bipartite matching in Matlab/Simulink models. Hummel et. al. [13] transformed models to graph and uses canonical matching with index based hashing to detect clones in models. Some work on model clones has been performed on UML diagrams. Liu et. al. [14] uses suffix tree approach to detect clones in UML models. It can work only with sequence diagram using tool DuplicationDetector. Störrle [2] developed a tool MQlone using model matching in UML models.

3. Selection Technique
This paper assumes three natures of the software between which the clones are to be detected:-

- Independent system
- Dependent system
- Version based system

3.1. Independent System
In Independent system, it's very difficult to detect clones earlier because the system as a whole is different and also its components but it may be the case that a part of the system or its components may be the clone so it can be detected with accuracy only when code cloning is used. In case of model cloning the false positives will be more.
Here two code clones are represented between two different system student and teacher. The similarities have been detected in later stage i.e. implementation phase where only four attributes have resulted in clones while there is hardly any operation that is similar in any context.

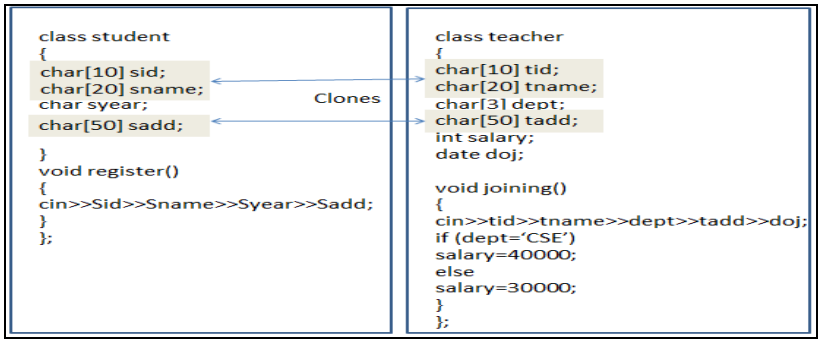

Figure 1: Code cloning in independent systems

3.2. Dependent System
If case of dependent system is considered then the system as a whole is not similar but it's components or subcomponents of the components may be similar so, the components can be detected easily through the model clones while the accuracy for the detection of similarity in subcomponents can be achieved via the code clones so to attain accuracy in the less time, model is assisted with code clones. First of all the model cloning is detected in the design phase which reduces lesser space for the code cloning to be detected thus making it faster and accurate.

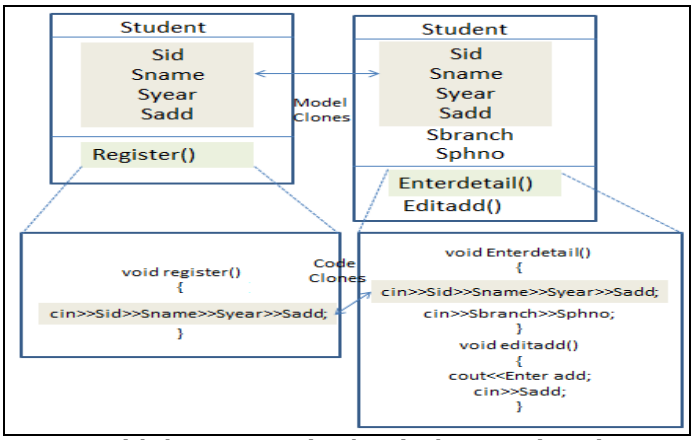

Figure 2: Model cloning assisted with code cloning in dependent systems

Here two model clones are represented where both exact and renamed model clones are identified and then for near miss and semantic clones, they are detected in the implementation part.

3.3. Version Based System

In terms of version control system the paper has detected clones between various versions of a single system. So, the functionality is almost similar with some extra functionality due to which clones can be detected in the design phase only as there will be large number of components that are clones. Then there is no need to detect the subcomponent cloning. Thus only model cloning is preferred to detect clones between various versions of a system without hampering accuracy.

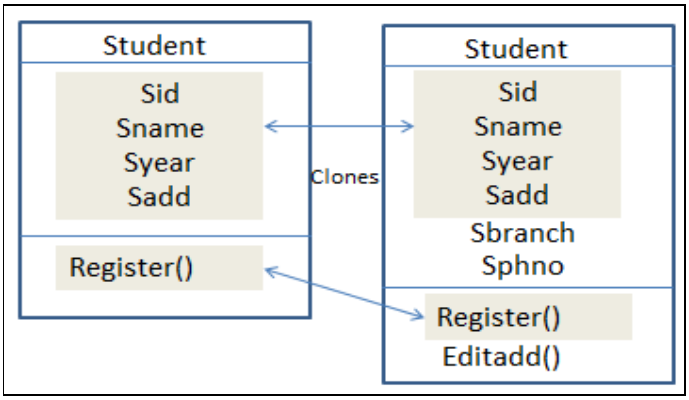

Figure 3: Model cloning in version control system

In model cloning with version control, two versions of same system are represented with student class. It identifies the clones in form of attributes and operation. There are four attributes and one operation that are identical and are thus results in class pairs.

4. Conclusion

The selection of the model cloning or code cloning depends on the nature of the software. If clones are detected between the two versions of a system then model cloning can provide good results. In two independent systems the code cloning should be preferred while the dependent system i.e. the system developed by two different teams but on same functionality, the model cloning assisted with code cloning provides the better results. This selection is based on the proper balancing of accuracy and time constraints.

5. Future Scope

This technique for selection of type of cloning preferred for the system doesn't produce accurate results always, so there must be some other parameters for this selection in order to produce accurate results. Moreover it can be further extended to clone detection and management for its proper working.

6. References

1. Roy, C. K. , Cordy, J. R. (2007). A Survey on Software Clone Detection Research, Technical Report 2007-541, 115.
2. Störrle, H. (2010). Towards clone detection in UML domain models, European Conference on Software Architecture (ECSA'10), 285–293.
3. Cordy, J.R., Dean, T.R., Synytskyy, N. (2004). Practical language-independent detection of near-miss clones, 14th IBM Centre for Advanced Studies Conference (CASCON'04), 1–12.
4. Ducasse, S., Rieger, M., Demeyer, S. (1999). A language independent approach for detecting duplicated code, 15th International Conference on Software Maintenance (ICSM'99), 109–119.
5. Kamiya, T., Kusumoto, S., Inoue, K. (2002) .CCFinder: a multi-linguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering ,28 (7), 654–670.
6. Li, Z., Lu, S., Myagmar, S., Zhou, Y. (2006). CP-Miner: finding copy–paste and related bugs in large-scale software code, IEEE Transactions on Software Engineering, 32 (3), 176–192.
7. Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M., Bier, L. (1998). Clone detection using abstract syntax trees,14th International Conference on Software Maintenance (ICSM '98), 368–378.
8. Wahler, V., Seipel, D., Gudenberg, J.W., Fischer, G. (2004). Clone detection in source code by frequent itemset techniques, 4th IEEE International Workshop Source Code Analysis and Manipulation (SCAM'04), 128–135.
9. Komondoor, R., Horwitz, S. (2001). Using slicing to identify duplication in source code, 8th International Symposium on Static Analysis (SAS'01), LNCS 2126, 40–56.
10. Hummel, B., Juergens, E., Heinemann, L., Conradt, M. (2010). Index-based code clone detection: Incremental, distributed, scalable, 26th IEEE International Conference on Software Maintenance (ICSM'10), 1–9.
11. Pham, N.H., Nguyen, H.A., Nguyen, T.T., Al-Kofahi, J.M., Nguyen, T.N. (2009). Complete and accurate clone detection in graph based models, 31st International Conference on Software Engineering (ICSE'09), 276–286.

12. Deissenboeck, F., Hummel, B., Juergens, E., Schätz, B., Wagner, S., Girard, J., Teuchert, S. (2008). Clone detection in automotive model-based development, 30th International Conference on Software Engineering (ICSE'08), 603–612.
13. Hummel, B., Juergens, E., Steidl, D. (2011). Index-based model clone detection, 5th International Workshop on Software Clones, 21–27.
14. Liu, H., Ma, Z., Zhang, L., Shao, W. (2006). Detecting duplications in sequence diagrams based on suffix trees, 13th Asia-Pacific Software Engineering Conference (APSEC'06), 269–276.
15. Rattan, D., Bhatia, R., Singh, M. (2012). Model clone detection based on tree comparison, India Conference (INDICON), 2012 Annual IEEE, 1041-1046.
16. Rattan, D., Bhatia, R., Singh, M. (2013). Software clone detection: A systematic review, Information & Software Technology, 55, 1165-1199
17. Sharma, Y., Bhatia R., Tekchandani , R . K. (2011). Thesis on Hybrid technique for object oriented software Clone detection , Electronic Theses & Dissertations @ TU
18. Garg, R., Bhatia, R. (2014). Code Clone v/s Model Clones: Pros and Cons, International Journal of Computer Applications (IJCA), 89, No 15, 20-22