IJITKM 0973-4414 Volume-11, Issue-1 June-Dec 2017 pp. 93-99 available online at www.csjournals.com

Clone Detection Techniques in Software Code and Models

Neha Saini[1], Sukhdip Singh[2]
[1]Research Scholar, [2]Associate Professor
Computer Science & Engineering Department, Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Sonipat Haryana.

Abstract: Two fragments of source code which are identicalto each other either syntactically or semantically are called code clones. Code clones make software maintenance very difficult and cause bug propagation from original code to all copied fragments. There are so many reasons of occurrence of code clones like code reusability by copyinglibrary and functions which are already in existence, various coding practices like inheritance and repeated computation using pre-existing functions with minor changes in variable names and types or data structures used. If a code fragment is changed or modified by deleting or adding code statements, we need to check all the related code clones to see if they need to be updated as well.Nowadays Model based methodology is used for development. Models are designed before coding. As clones exist in the source code, similarly clones exist in the models. Itaffects the quality of software and increase maintenance cost. In this study, code clones, model clones, various types of clones, different phases in clone detection, techniques and tools for clone detection and shortcomings of clone detection techniques have been discussed.

1. INTRODUCTION

Copy and Paste of fragments has been used in software development. This strategy is known as cloning. Cloning can be either at designing level or at implementation level. At implementation level clones exist in the source code and during designing clones exist in the models. Clones increase redundancy in the software which cause problem in software maintenance. And cloning also increase probability of bugs and maintenance cost. So clones need to get removed. Many Solutions have been proposed to remove clones.

2. CODE CLONE:

During the implementation phase of software development, fragments of code identical to each other are known as code clonesotherwise identical fragments exist in the form of models and are known as model clones. Code clones can be broadly categorized in to four on the basis of similarity.

2.1 Types of Code Clones:-

Type 1 (Exact clones): Program sections which are identical to each other except for differences in white space and comments etc.

Type 2 (Renamed/parameterized clones): Program fragments which are structurally or semantically alike except for differences in name of identifiers, literals, layout, types, and comments.

Type 3 (Near miss clones): Programfragments that have been copied with some additional changes like statement inclusions or removals in addition to modifications in identifiers, types, literalsand layouts.

Type 4 (Semantic clones): Program coding parts which are functionally alike without being textually related.

Code clones minimizes the variations and risk, however it identifies them at advanced phase.Though code clones must be considered when the software to be formed is compatible with reference to its time and cost limits.When the system needs accuracy with high recall then code clones must be chosen.

3. MODEL CLONE:

A model fragment is a set of model elements which are identical to each other on the basis of some similarity criteria. A model clone is a set of various model fragments such that there is a great degree of similarity among those fragments. The two similar model fragments may be result of direct copy and paste operation or accidental copy and paste operation.

3.1 Types of Model Clones:-

Type I (Exact clone): A model fragment that is identical except for some minor differences like comments and whitespaces.

Type II (Renamed clone): A duplicated part of model with consistent changes to identifiers, variables, types, or functions.

Type III (Parameterized clone): A duplicate allowing arbitrary changes, additions or removals of parts. This type of clone is also called "near-miss clone".

Type IV (Semantic clone): A duplicate in content only that may be owed to code copying, convergent growth, or other methods.

4. Benefits and Need of Clone Detection

Clone detection not only helps in improving the code quality through refactoring of duplicated code, there are various other advantages of clone detection:

i. Detection of Library Candidates

If a code segment is duplicated and reused many times in a software system, then it demonstrates its usability. Therefore, this segment can be used as library candidate.

ii. Helps in Reducing Code Size

If the detected code clones are replaced by function calls to a generic code segment performing the same functionality as that of the code clone, then it results in reducing the complexity and size of software system. It also improves maintainability and readability of code.

iii. Helps in Finding Usage Patterns

When all cloned segments of a piece of code are detected, then the usage patterns of that piece of code can also be found out.

iv. Better Understanding of Problem

If working of a cloned segment is apprehended, one is able to understand the working of all duplicate code segments of the cloned segment.

v. Detects Malicious Software

Clone detection can be helpful in detecting malicious code. This can be done by matching one malicious piece of code with other and finding which piece of code matches with the malicious code.

6. CODE CLONE DETECTION TECHNIQUES:

6.1 Text Based Techniques

The code sections are considered as sequences of text and are compared line by line after applying various transformations like elimination of whitespaces and comments etc. Various researchers have developed tools and techniques to find clones on the basis of text. **Baker [1]**has line-based string matching algorithm to find clones. In this method tokens are generated out of each sequence/line of the text with the help of a tool namedDup. This method makes use of all the basic properties of text-based approach. Additionally, it replaced variables, identifiersand types with a separate parameter. Therefore it is possible to detect the clones even if variable name differs. But one of the limitation of this tool is that it cannot support investigation and navigation among the copied fragments of codes. Also, it cannot identify the clones if code is printed in different styles. This shortcoming was addressed by **Koshke et al.by [3]** by finding the clones based on the tokens rather than lines.

6.2 Lexical or Token Based Techniques:

This technique converts the entire source code in to tokens to overcome the limitations faced in text-based method like variation in white space, identifier name etc. **Kamiya et al.[2]**developed a tool called as CCFinder in which every line is converted in to tokens and all the tokens are combined into single token. Even though **Baker [1,5]**also made use of tokenization but his technique for clone detection contained so many false positive. To add more flexibility in tokenization RTF made use of suffix array instead of suffix tree to remove unnecessary tokens and decrease the rate of false positives but implementation of this method is very difficult.

6.3 Tree based Clone Detection Technique:

This method focuses on generation of subtrees of a particular section of code and matching those subtrees for clone detection. After that abstract syntax tree is generated out of source code. CloneDRis a widely used tool which makes use of this technique for clone detection. First parse tree is produced and after that subtrees are matched using hash function. It was unable to detect similar clones.CCdim [3] tool given by Bauhausovercomed this limitation of CloneDR tool, but it could not detect renamed identifiers. Yang also used tree based method to detect the syntactic differences among various versions of the system by generating their parse tree.

6.4 Graph based Clone Detection Method:

Program Dependency Graph (PDG) [4] is able to detect both syntactic and semantic clones, thus overcoming the shortcomings of tree based method.**Komondoor and Horwitz [4]** developed a method calledPDG-DUP which made use of program slicingto detect the clones without making any change to its semantics. Gallagher and Lucas modified the work of Komondoor et. al by relating program slices on the basis of all the variables of a code but it did notresult in any improvement in results. PDG technique was considered as an recursive approach by Krinkeordetecting the maximal analogous subgraph but it was unable to give a formula to be applied for any kind of system to detect the clone. All the researchers who have used PDG based

concluded that PDG based method cannot be applied to large systems although they can detect noncontiguous clones.

6.5 Metric based Clone Detection Technique:

This method does not compare the code source code directly. Instead, metrics are generated out of source code and those metrics are used for comparison. **Mayrand et. al[5]** used metric based approach to compute the metrics from names, expression,layoutsand control flow butsegment based copy-paste action could not be detected using this approach. **Kontogiannis et. al made use of** markovapproach but it could not detect exact clones. After that they made some changes to the approach by calculating metrics on the basis of begin-end structure of the code.The code is called cloned code if there is similarity between metrics on the basis of some predefined criteria.

6.6 Hybrid Clone Detection Technique:

These techniques are the combination of the techniques discussed above. **Koschke et. al**[6] hybridized tree and token based techniques for clone detection. First a suffix tree is constructed using tree based method and after that comparisons are done using token based method. Only exact and type-II clones can be detected using this method. Microsoft's new phoenix framework adopted similar kind of approach. It can only detect clones with change inidentifier name but unable to detect clones with change in type of variables. **Greenan**used same technique using sequence matching algorithm. Jiang et al. used the Abstract Syntax Tree in Euclidean space and then Locating SensitiveHashing is used for clustering the vectors based on some similarity criteria.

7. MODEL CLONE DETECTION TECHNIQUES:

Dhavleesh Rattan et al.[8]developed a new technique for clone detection in UML class diagrams. These days Unified Modeling Language (UML) is widely used in software development. In the graph of UML models, nodes are classes and edges are relationship amongst those classes. Due to wide variety of information stored in the nodes of the graph corresponding to UML model, they are very dense. Due to this, clone detection in UML models give better results. Weight of nodes of Matlab/Simulink models are comparatively lighter. Therefore, isomorphicgraph comparison is applied in Simulink models but can't be applied in UML models. Basic steps for clone detection in UML class diagram are:

1) First of all create the UML model using any tool.
2) After that export the created UML model to XMI (XML Metadata Interchange)format of file. Since XMI is a standard given by OMG, it is available in most of the tools used for model creation.
3) Preprocessing of XMI file is done and it is stored in the form of tree using DOM API's and XML parsing.
4) Atlast, sub-trees created in the previous step are compared and similarity is calculatedin the form of model clones.

UML diagrams are encoded in XMI files to find out differences between these diagrams. This technique has been explored to detect clones. All the elements of diagramare compared and similarities are measured between those elements. Based on the values of similarities these elements are reported as clones.

Manar H. Alalfi et al.[7]has introduced SIMONE. SIMONE uses text based clone detector NICAD to detect near miss clones in the Matlab/Simulink model. Simulink stores textual

representation of models on disk. This representation is given as input to SIMONE. Following steps are used in this technique:

1) Simulink TXL grammar
2) Extractor Plug in
3) Filtering
4) Sorting
5) Renaming

In the first step Simulinks are converted into TXL grammar. NICAD is language sensitive clone detectors. It uses TXL parser. So it needs to convert Simulink in TXL grammar. Grammar inference techniques are used for this. This grammar identifies all simulink constructs, including models, systems, block, lines, ports etc. In the second step Extractor Plug in are used to extract potential clones. In the third step filtering is done. When simulinks are converted into textual form, meaning of models is changed.In the fourth step Sorting is done. Even after filtering some clones are not detected, because when simulinks are converted into textual form then order of block, lines, branches and ports will be changed in textual form. So canonical sorting is implemented on models.In the fifth step sophisticated blind renaming plug-ins are used for Simulink. Problems of linear representation are resolved by sorting. SIMONE can find out exact and nearmiss exact subsystem clones.

Florian Deissenboeck et al. [9]has used graph based theory to detect clones. This technique works on Matlab/Simulink models. This approach consists of three steps: In the first preprocessing is done then at second level Simulinks are normalized and clone pairs are extracted. And in the last step pairs are clustered to find substructure used more than twice in the model.

Liu et al.[10]proposed a tool DuplicationDetector. It detects duplications in sequence diagram. Sequence diagrams are used as interaction diagrams to describe behaviors of use cases, operations and collaborations. It describes how the processes operate and in what order. The duplications occur because of system's complexity , poor design and reluctance to restructure the design and due to various existing scenarios with a main execution flow and several alternate flows. These duplications hamper maintainability and reusability.

Hummel et al.[11]developed a toolbased on incrementalclone detection. It takes input Simulink/matlab model. In preprocess phase, it converts the model into a directed multi graph and assigns labels to relevant blocks. In detection phase, graph isomorphism is determined which is based on canonical labeling (unique code invariant to ordering of vertices and edges). A small change need not entire detection using index- based algorithm that is incremental anddistributable. Hash code is used as a heuristic.To reduceruntime hash code is generated using md5 hashing. In post-processing phase, cloning information is filtered, preventedor used by the clone management tools. It is reused by clone detector ConQat [9].

Clone index is calculated for all the sub-graphs which are of equal size. Clone index is calculated on the basis of canonical labeling and similar labels are passed through hash function for hashing. Due to updation of index and clone retrieval takes less time. It has only been verified for small models.

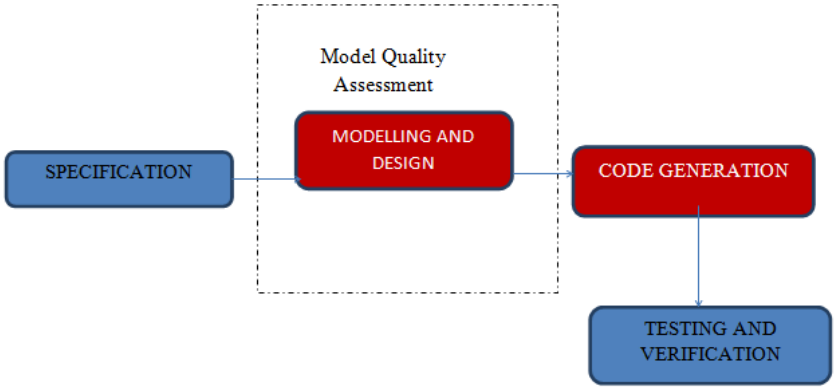

Figure 1: Model based development process

As the quality of code generated from a model is directly depending on the quality of that model, using a model-based approach enables developers to assess the software's quality already in the design phase.

S.mythili .et.al in her study "Efficient weight assignment method for clone detection in state flow diagrams", has taken query model as input parameter. After that weights are identified for query model. The weight is collated with the weights of all the models available in the database. If both the weight matches then the whole model is said to be cloned. Clone is displayed even if weight of the query model matches with one of the weight available in the database

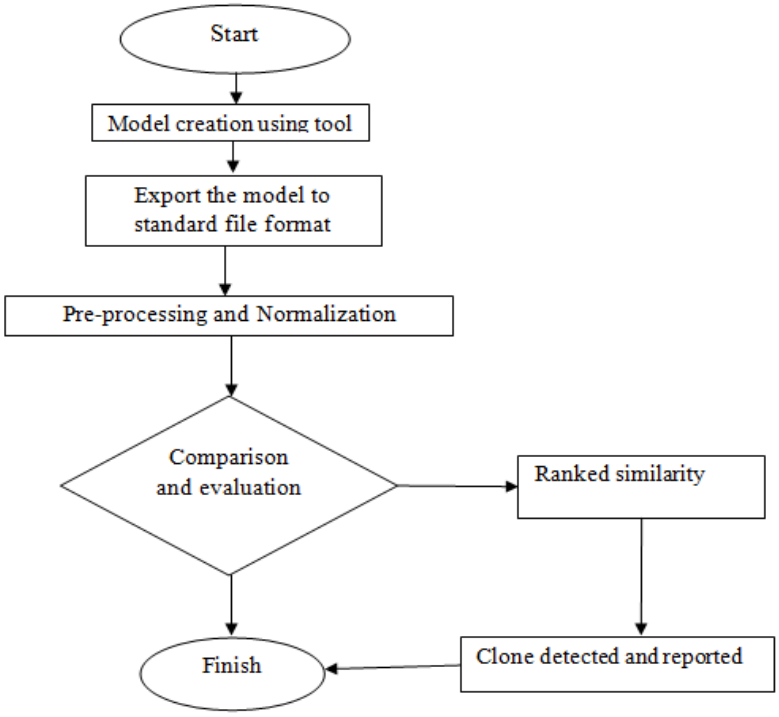

Figure 2: Flowchart for the model clone detection method

8. CONCLUSION

This study includesdiscussion in terms of various categories of clones, classification of various clone detection approaches such as text based, token based, tree based, PDG based, metric based and hybrid technique on the basis of their properties. However, so many algorithms have been developed based on these approaches but they lack in terms of accuracy.

Model clones are as harmful as code clones. Model clones also increase maintenance cost and possibility of errors. So these need to be removed. Different tools have been proposed for model clone detection. Each tool is designed for particular model. Different methodologies are used by tools. Each methodology has its own advantage and limitations. Still very few solutions have been offered for model basedclone detection.

References:

1. Baker, B.S.: A program for identifying duplicated code.de. In: Proceedings of Computing Science and Statistics, 24th Symposium on the Interface, pp. 49–57, March 1993
2. Kamiya, T., Kusumoto, S., Inoue, K.: CCFinder: a multilinguistic token-based code clonedetection system for large scale source code. IEEE Trans. Softw. Eng. 28(7), 54–67 (2002).
3. Baxter, I.D., Yahin, A., Moura, L, Anna, M.S.: Clone detection using abstract syntax trees.In: Proceedings of the 14th IEEE International Conference on Software Maintenance (ICSM1998), Maryland, pp. 368–377, November 1998
4. Ferrante, J., Ottenstein, K.J., Warren, J.D.: The program dependence graph and its use inoptimization. ACM Trans. Program. Lang. Syst. 9(3), 319–349 (1987).
5. Mayrand, J., Leblanc, C., Merlo, E.: Experiment on the automatic detection of functionclones in a software system using metrics. In: Proceedings of 1st IEEE InternationalConference on Software Maintenance, Monterey, CA, pp. 244–254 (1996)
6. Koschke, R., Falke, R., Frenzel, P.: Clone detection using abstract syntax suffix trees. In:Proceedings of the 13th IEEE Working Conference on Reverse Engineering, Italy, pp. 253–262, October 2006.
7. M. H. Alalfi, J. R. Cordy, T. R. Dean, M. Stephan, A. Stevenson, SIMONE: Models are Code too, Near-miss Clone Detection for Simulink Models, IEEE, 2012.
8. D. Rattan, R. Bhatia, M. Singh, Model Clone Detection based on Tree Comparison, IEEE, 2012.
9. F. Deissenboeck, B. Hummel, E. Juergens, B. Schätz, S. Wagner, J.-F. Girard and S. Teuchert, Clone detection in automotive model-based development, Proceedings of 30th InternationalConference on Software Engineering, Leipzig, Germany, 2008.
10. H. Liu, Z. Ma, L. Zhang, W. Shao, Detecting duplications in sequence diagrams based on suffix trees, in: Proceedings 13th Asia-Pacific Software Engineering Conference (APSEC‟06), Bangalore, India, 2006.
11. B. Hummel, E. Juergens, D. Steidl, Index-based model clone detection, in:Proceedings of 5th International Workshop on Software Clones, Honolulu,USA, 2011.