Advances in Computer Science and Information Technology (ACSIT)
Print ISSN : 2393-9907; Online ISSN : 2393-9915; Volume 2, Number 7; April – June, 2015 pp 27 – 32
Krishi Sanskriti Publications
http://www.krishisanskriti.org/ACSIT.html

A Review of Clone Detection in UML Models

Balwinder Kaur[1], Er. Harpreet Kaur[2]

[1]M.Tech Scholar, University College of Engineering, Punjabi University, Patiala, Punjab, India.
[1]balwinder0909@gmail.com
[2]Assistant Professor, University College of Engineering, Punjabi University, Patiala, Punjab. India.
[2]khasria.harpreet@gmail.com

Abstract: *Model Driven Engineering has become standard and important framework in software research field. Unified Modeling Language (UML) domain models are conceptual models which are used to design and develop software in software development life cycle. Models contain design level similarities, these are called model clones. Model clones are harmful for software maintenance as code clones and also lead to bad design. So number of clones need to be detected from UML domain models. Awareness of clones helps in reusable mechanism. Many techniques have been proposed for code clone detection but a few work has been done on model clone detection. In this paper review has been provided related to various techniques for detection of clones in UML models. Tree comparison technique is used to find similarity in two fragments of a model. Tree is less false positive because of minimum non-relevant matches. Suffix array technique is used to detect clones in class diagrams. Suffix array consumes minimum memory. NiCad Clone detector tool is a scalable and flexible tool to detect type-3 near-miss clones in behavioural models. This paper provides comparative features of these above different techniques.*

Keywords: *Model clones, UML models, Code clones, Model clone detection.*

1. INTRODUCTION

Copying existing code and pasting it with or without any change into other sections of programme is a popular process in software development. The copied code is called a software clone and process is called software cloning [14]. Model clones are the clones which are detected at the design phase. Code clones are the clones which are detected at the implementation phase. Clones increases redundancy, probability of bugs and maintenance cost.

Model Driven Development defines domain models also called conceptual models which mainly focuses on modeling rather than computer programming. There are various UML models such as class diagram, use case diagram, activity diagram, state chart diagram, sequence diagram etc [2]. The UML i.e. Unified Modeling Language gives us a standard way to define system's view including many conceptual details such as business processes. Because of large adaptability of it by software developer, it is necessary to understand the importance of modeling. Use of UML makes modeling more efficient and effective. It is very important to understand the definition of model clones and to derive a formal framework for model clones. Many techniques have been applied to detect code clones but a few techniques have been proposed to detect model clones[3], [7]. In this paper techniques for clone detection have been discussed.

1.1. Motivation

1.1.1 Higher level of abstractions and a number of modeling languages has made modeling a key industry practice in different domains and different phases of software development life cycle.
1.1.2 Models attain substantial size [10] thereby increasing complexity and higher rate of duplications.
1.1.3 There is a dearth of studies suggesting techniques for detection of clones in UML models.

2. Model Cloning

2.1. Model Clone

In code based development source code clone detection is a big problem, the same problem also occur in model based development for duplicated parts of models. There is a difference between programming language code and models so algorithms and notations used for code clone detection are difficult to implement to model clone detection [2], [12].
A model fragment is a set of model elements that is closed under some closure property of similarity. Model Clone is a pair of model fragments that contains high degree of similarity.

2.2. Types of model clones

There are four types of model clones.
2.2.1. Type 1- Exact Model Clone: In exact model clone, model fragments that are identical except from internal identifiers and layouts.

2.2.2. Type 2- Modified Model Clone: In Modified model clone, model fragments with changes to the attribute names and element names.

2.2.3 Type 3- Rename Model Clone: In rename model clone, model fragments with changes such as addition or removal of parts.

2.2.4. Type 4- Semantic Model Clone: In semantic model clone, model fragments that are due to model part copying or language constraints etc [3], [7].

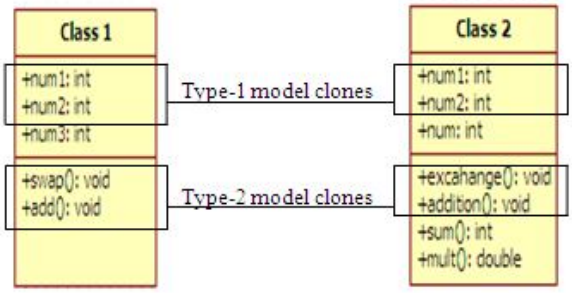

Figure 1. Clones in class diagrams

2.3. Reasons of model clones

2.3.1. Model clones through copy/paste: Editing commands copy/paste is used for copying a diagram element so that create a new copy of model element. Model clones are created by copy/paste to reuse the model elements. This is the fast and immediate way of adapting the change by software designers [7].

2.3.2. Model clones through language loopholes: Due to some language limitations, parts of models are repeated by mistake [14].

2.3.3. Complexity of the system: Copying the existing model elements is used, because there is difficulty in understanding the large systems [7].

***2.3.4.* Time limits assigned to software designers:** One of the major causes of cloning in the system is the time frame allowed to its software designers. In many cases, the software designers are assigned a specific time limit to finish a certain project or part of it. Due to this time limit, software designers look for an easy way of solving the problems at hand and consequently look for existing design. They just copy and paste the exiting one and adapt to their current needs [14].

2.3.5. Wrong method of measuring software designer's productivity: Sometimes the productivity of a software designer is measured by number of model elements produces per hour. In such circumstances, the software designer focus is to increase the no. of model elements of the system and hence tries to reuse the same model element again and again by copying and pasting instead of following a design strategy.

2.3.6. Software designer's lack of knowledge in a problem domain: Sometimes the designer is not familiar to the problem domain at hand and hence looks for existing solutions of similar problems. Once such a solution is found, the designer just adapts the existing solution to his/her needs. Because of the lack of knowledge, it is also difficult for designer to make a new solution even after finding a similar existing solution and thus reusing the existing one gets higher priority than making a new one [14].

2.3.7 Model clones by intention of programmer: Some clones are created with purpose by the programmers to create the parts of models [7].

2.4. Code clones versus Model clones

2.4.1. Language/Tool Integration: Program is a text file. Programming languages are independent of a development environment, a Java program created with one IDE can easily be transferred to another. Models, on the other hand, are tightly integrated with some tool which is used to create the models.

2.4.2. Structure: The code of a system can be represented as a directory tree of text files, each of which is considered as a long string of characters or tokens. Models, on the other hand, have a graph structure stored in a repository.

2.4.3 Identification: In source code, elements like types, procedures and so on are identified by their names. Textual representation is used to identify code fragments. Code clones are identical because copying a text fragments retains their identity. In many modeling environments, on the other hand, model elements have internal identifiers. Model clones are equal but not identical because these identifiers are understood as globally unique, a copy of a model element will consistently change the identifiers in the duplicate.

2.4.4. Syntactic representation: Source code is represented as a string of characters. There are white spaces and indentations but source code remains sequential text. Models, on the other hand, have a dual structure. Internally, they are a

set of linked meta model class instances. Externally, they are a set of diagrams where secondary notation and layout play a important role in understanding and using diagrams as used in visual modeling languages [7].

3. LITERATURE REVIEW

Various techniques has been provided till now but each with its pros and cons. The following table 1 gives overview of available techniques.

Table 1. Literature of different available techniques

S. No.	Year	Author	Title	Work done	Results	Future scope
1.	2014	S. mythili, Dr. S. Sarala	Efficient weight assignment method for clone detection in sate flow diagrams [1]	Clone detection in state flow diagrams takes query model as input. The weight identification of the query model is done. The weight is compared 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 when weight of the query model matches with one of the weight in the database.	This process will be implemented to check for clones in various industrial projects for evaluation. It can also be extended to detect clones in process oriented models.
2.	2014	Harjot Kaur, Manpre et Kaur	Detecting clones in class diagrams using suffix array [2]	Class diagrams are encoded as XML files. Tokens are extracted and matched using suffix array technique. This approach is based on finding similarities in tokens known as clones.	Class diagrams contain redundant elements. Similar attributes or operations present in two different classes are known as clones.	This technique will be extended for clone detection in state chart and activity diagrams.
3.	2013	Antony. E.P., Alafi.M. H., Cordy.J. R.	An approach to clone detection in behavioural models [4]	An approach for reverse engineered UML behavioural models to detect near-miss interaction clones consists of four steps: 1) Behavioural models are represented as XMI file. 2) TXL is used to do transformation/ contextualization. 3)Normalization is performed. 4) NiCad is used to do clone detection analysis.	This approach has detected type 3-1(exact near-miss) conversation clones. The result, obtained from clone detector are presented in NiCad's default XML and HTML text formats.	Work will be planned to trace the clones back to the original diagrams and visualize them in the model.
4.	2012	Rattan. D, Bhatia. R, Singh.M	Model clone detection based on tree comparison [6]	To find similarity in two fragments of a model, tree comparison is used. The basic steps are: 1) Any modeling tool is used to create the model. 2) The model of class diagram is exported to XMI file format. 3) XMI file is stored in the tree form using DOM API's and XML parsing. 4) Comparison of subtrees are done and duplicity is reported in the form of model clone.	Anecdotal evidence suggests that clones in models hamper maintenance. A technique is presented to detect clones in class diagrams encoded as XMI file. The similarity is reported as output.	Work will be extended to the prototypical implementation to display the clone results with percentage of similarity. The present technique can be extended to Activity diagrams.

S. No.	Year	Author	Title	Work done	Results	Future scope
5.	2011	H. Storrle	Towards clone detection in UML domain models [7]	Clone detection algorithm is implemented as MQlone tool, a plugin in Magic Draw UML CASE tool which reports the clones to the user. Steps of clone detection approach: 1) A model is created manually or by transformation. 2) Built in facilities of the Magic Draw UML CASE tool are used to export the model into XMI file format. 3) Prolog code is generated after transformation of XMI file. 4) Next the clone detection is based on similarity and model matching.	Formal definition of models, model clones and implemented approach in the MQlone tool have been provided. The clone detection quality and runtime of algorithm were validated experimentally.	Future work will focus on areas such as: first, MQlone can be tuned by large number of parameter settings that have not been fully explored. Second, more algorithms will be explored.
6.	2011	J.R. Cordy, C.K. Roy	The NiCad Clone Detector [8]	NiCad Clone detector involves three stages: 1) Parsing: To extract all fragments of given granularity input sources are parsed. 2) Normalization: Extracted fragments can be normalized, filtered before comparison. 3) Comparison: To detect similar fragments, LCS (Longest Common Subsequence) is used to compare linewise extracted and normalized fragments.	It provides output results in both XML and HTML forms. NiCad is new clone detection method that has been used in detecting near-miss clones with high precision and high recall.	To perform comparison of NiCad clone detector with other clone detection tools.
7.	2010	Deissen boeck.F, Hummel.B, Pfaehler.M, Schaetz. B.	Model clone detection in practice [9]	An industrial case study for BMW group has been presented that provides an efficient clone detection technique in model based quality assurance. Model clone detection is done in three steps: 1) Preprocessing and Normalization 2) Detection 3) Post processing. Model quality Assessor tool allows to execute the model clone detection within an integrated environment that is also used for visualizing detection results.	Model Quality Assessor tool is provided that eases the evaluation of detection results and thereby helps to make clone detection a standard technique in model based quality assurance.	The detection approach will be extended to other data flow oriented forms of models, example: State oriented models like state charts or state flow.
8.	2009	Nam H. Pham, Hoan Anh Nguyen, TungThang Nguyen	Complete clone detection in Graph-based Models [10]	ModelCD, a novel clone detection tool has been proposed for graph based models. The core of ModelCD is two clone detection algorithms i.e. eScan and aScan. Two algorithms detect clones through three steps: generating, grouping and filtering.	Empirical evaluation on large scale Simulink systems is shown that it is able to handle both exact and approximate clones.	In future, to compare usefulness of this approach with other tools.

4. RESULTS

On the basis of literature, strengths and weakness of different techniques has been gathered in table 2.

Table 2. Review

S. No.	Approach for model Clone detection	Advantages	Disadvantages
1.	Weight assignment method	This method is used to identify the clones with accuracy and rapidity.	This clone detection technique does not execute a graph based approach.
2.	Clone detection using suffix array	Suffix array is considered to be better than suffix tree in terms of memory space and access speed.	There is no functionality added to the algorithm that can rate the clones relevant or non-relevant for better maintenance.
3.	Clone detection in behavioural models using NiCad tool	Clones in reverse engineered behavioural models from web applications can be used to find worrisome patterns such as security violations.	This approach is not applied to other kinds of UML behavioural model representations.
4.	Model clone detection using tree comparison	Key elements of UML diagrams are stored in the form of a tree, therefore irrelevant clones are not reported.	This algorithm is not implemented on large number of class diagrams as well as other object oriented diagrams.
5.	Clone detection algorithm implemented as MQlone tool, a plugin in Magic Draw UML CASE tool	This approach works well for clone detection in small and medium sized models.	This approach cannot be applied on large and very large models. This approach has inaccuracy and low degree of completeness in clone detection.
6.	NiCad Clone detector	NiCad clone detector is a code clone detection tool i.e. scalable and flexible. It is used to detect Type-3 (Near-Miss) model clones. NiCad is efficient in its resource usage and on a standard single processor can handle the largest systems in 2GB of memory.	NiCad clone detection stages such as parsing and extraction are most expensive stages.
7.	Model clone detection in practice using Model Quality Assessor tool	This technique mainly focuses on improvement of scalability and relevancy.	This clone detection is not applicable to process oriented models for clone detection.
8.	Clone detection in graph-based models using ModelCD tool.	This tool is used to detect clones with high degree of accuracy, scalability and completeness.	This model clone detection for Matlab/Simulink models report many clones as irrelevant.

This crux will be helpful to users/ readers to choose and apply clone detection technique according to particular need.

5. CONCLUSION AND FUTURE SCOPE

Clone detection is an active research area. Large adaptability of model based development in software field is promoting model based clone detection. In this paper, a survey on the area of clone detection research is made, putting emphasis on the types of clones used, their detection mechanism and evaluation of the techniques. Different techniques have been presented for model clone detection in class diagrams and other models. Each methodology has its own advantages and limitations. From the survey it has been observed that tree comparison technique reports minimum irrelevant clones.

Clone detection using suffix array is considered to be better than suffix tree in terms of memory space and access speed. NiCad is a scalable and flexible clone detection tool to detect type-3 clones. So we can conclude that finding clones from the models will help the developer for better maintenance and understandability of model because most of the developer interacts with the systems through diagrams only.

In future, clone detection techniques will apply for detection of clones in other UML diagrams such as sequence diagrams and activity diagrams etc. Further, a lot of work need to be done to detect type-3 and semantic clones.

6. REFERENCES

[1] S. Sarala, S. Mythili, "Efficient weight assignment method for detection of clones in state flow diagrams," International Journal of Software Engineering Research & Practices, vol. 4, Issue 2, October 2014.

[2] Harjot Kaur and Manpreet Kaur, "Detecting clones in class diagrams using suffix array," International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249-8958, Vol. 3, Issue 4, April 2014.

[3] D. Rattan, R. Bhatia, M. Singh, "Software clone detection: A systematic review," Information and Software Technology, pp. 1165-1199, 2013.

[4] Antony.E.P., Alafi.M.H., Cordy.J.R., "An approach to clone detection in behavioural models," Queen's University, Kingston, Canada, AAC-WCRE, 2013.

[5] M.H. Alafi, J.R. Cordy, M. Stephan, T.R. Dean and A. Stevenson, "Models are code too: Near –miss clone detection for Simulink models," IEEE, in ICSM, pp. 295-304, 2012.

[6] Rattan.D, Bhatia.R, and Singh.M, "Model clone detection based on tree comparison," India Conference (INDICON), IEEE, ISBN: 978-1-4673-2270-6, pp. 1041-1046, Dec. 2012.

[7] H. Storrle, "Towards clone detection in UML domain models," Software and Systems Modeling, doi 10.1007/s10270-011-0217-9, pp. 39, 2011.

[8] R. Cordy and C.K. Roy, "The NiCad clone detector," in Proceedings of the Tool Demo Track of the 19th International Conference on Program Comprehension (ICPC 2011), IEEE, Kingston, Canada, pp. 219-220, June 2011.

[9] Deissenboeck.F, Hummel.B, Pfaehler.M and Schaetz.B., "Model clone detection in practice," IWSC'10, Cape Town, South Africa, pp. 37-44, 2010.

[10] H. Storrle, "Towards clone detection in UML domain models," Proceedings of European Conference on Software Architecture (ECSA'10), Copenhagen, Denmark, pp. 285-293, 2010.

[11] Nam H. Pham, Hoan Anh Nguyen, Tung Thang Nguyen, Tien N. Nguyen, Jafar M. Al-Kofahi, "Complete and accurate clone detection in Grap-based Models," Proceedings of the IEEE 31st International Conference on Software Engineering (ICSE 2009), pp. 276-286, 2009.

[12] C.K. Roy, R. Koschke and J. R. Cordy, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," Science of Computer Programming, vol. 74, no. 7, pp. 470-495, 2009.

[13] Lin.H.J. and Peng.L.F., "Quick similarity measurement of source code based on suffix array", IEEE, International Conference on Computatinal Intelligence and Security", DOI 10.1109/CIS.2009.175, 2009.

[14] C.K. Roy, R. Koschke and J.R. Cordy, "A survey on software clone detection research," Technical Report 2007-541, Queen's University at Kingston Ontario, Canada, pp. 115, 2007.

[15] S. Bellon, E. Merlo, J. Krinke, G. Antoniol and R. Koschke, "Comparison and evaluation of clone detection tools," IEEE, Transactions on Software Engineering, vol. 33, no. 9, pp. 577-591, 2007.

[16] F. Deissenboeck, B. Hummel, E. Juergens, B. Schatz, S. Wagner, S. Teuchert and J. F. Girard, "Clone detection in automotive model based development," Proceedings of 30th International conference on Software Engineering, Leipzig, Germany, pp. 603-612, 2008.

[17] Abdul.H.B., Puglisi.S.J., Smyth.W.F., Turpin.A. and Jarjabek.S., "Efficient token based clone detection with flexible tokenization," ESEC/FSE'07, ACM, Cavtat Croatia, 2007.

[18] H. Liu, W. Shao, L. Zhang and Z. Ma, "Detecting duplications in sequence diagrams based on suffix trees", IEEE CS, Proceedings of 13th Asia- Pacific Software Engineering Conference (APSEC'06), Bangalore, India, pp. 269-276, 2006.

[19] Abdul.H.B. and Jarzabek.S., " Detecting higher-level similarity patterns in programs," ESEC-FSE'05, ACM, Lisbon, Portugal, 2005.

[20] I.D. Baxter, L. Bier, M. Sant'Anna, L. Moura and A. Yahin, "Clone detection using abstract syntax trees," in ICSM'98, IEEE Computer Society, pp. 368-377, 1998.