Code Clone v/s Model Clones: Pros and Cons

Ritu Garg
Computer Science Department
Deenbandhu Chhotu Ram University of Science and Technology, Murthal
Haryana, India

Rajesh Bhatia
Punjab Engineering College University of Technology
Chandigarh

ABSTRACT
Every software has time and budget constraints associated with it.The time and budget of the software also depends on the risk and inconsistencies during the software life cycle phases.These risks and inconsistencies can be reduced by detecting clones in form of redundancy between the software systems.This paper provides a brief overview to the detection of these risk and inconsistencies in either of the two phases of software development system i.e.design phase or the implementation phase along with their pros and cons.

Keywords
Software System, Clone detection, Model based Clone detection, Code based Clone detection

1. INTRODUCTION
Whenever two similar type ofsoftware's are developed, they have much common functionality so, the new software is developed with the existing one.Sometimes the new version is released from the previous version then also they have common functionality between them.Such a reuse which results in copy and paste activitieswith minor modifications without changing the functionalities is known as cloning. Cloning can create problems associated with update anomalies exist which requires change at one place in one original copy to be duplicated in all other duplicate copies that are similar to the original one.It can hamper the maintainability and Comprehensibility of the software systems.In addition to time and effort, the cost of developing and testing the software systems also increases. Thus there is a great need to detect the clones and remove them.It can be done at two levels in software development process- code clones and model clones.When clones are detected at the implementation phase of two software's then such a clone is called code clones while when clones are detected at the design phase of two software's then such a clone is called model clones.

2. CODE CLONE AND MODEL CLONES
If the fragment is in form of code during the implementation phase then it is termed as code clone [1] otherwise the fragments will be in form of models which is termed as model clones.There exist four types of code clones on basis of similarity [2].The two similar code fragments may be based on copy and reuse approach or the accidental cloning which is not the result of direct copy or paste activity.One such example of code based cloning is shown in figure 1 on basis of type-2 clones as discussed below.

Types of Code Clones [see 1]:-

Type-1:- It represents the redundancy in the code fragments except for the differences in the whitespace, layouts and comments.

Type-2:- It includes type-1 clones within it.It also represents the redundancy in the code fragments except for the differences in the naming of variables, constants, keywords, literals, types.

Type-3:- It includes type-2 clones within it.It also represents the redundancy in the code fragments except for the differences in the modification of statements.It reflects change in the form of addition, deletion or modification of the statements within a block.

Type-4:- It includes type-3 clones within it.It also represents the redundancy in the code fragments in form of semantic relation.The redundant code performs same computation but different implementations using different syntaxes.

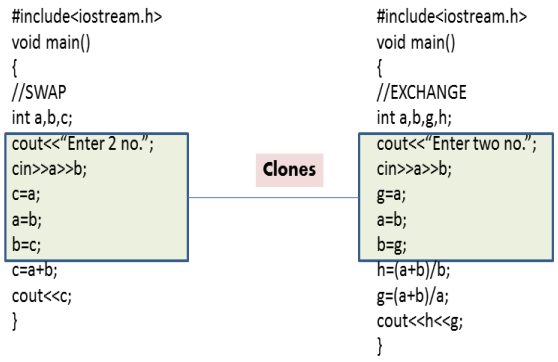

Figure1: Code based clone detection

A model fragment is a set of model elements that is closed under containment relationship.A model clone is a pair of model fragments such that there is a high degree of similarity between the fragments.The two similar model fragments may be based on copy and reuse approach or the accidental cloning which is not the result of direct copy or paste activity.One such example of model based cloning is shown in figure 2 on basis of type-2 clones as discussed below.

Types of Model Clones (See [2]):-

Type 1:- It represents a model that is identical except for layout, secondary notation, internal identifiers and notes.

Type 2:- It represents a model that is identical except for changes such as changes to element names, attribute names and parts.

Type 3:- It represents a model that is identical with changes such as addition or removal of parts (sub model inside the model elements).

Type 4:- It represents a model that is identical in content only that may be due to model fragment copying, methodology or language constraints, convergent development or other processes.

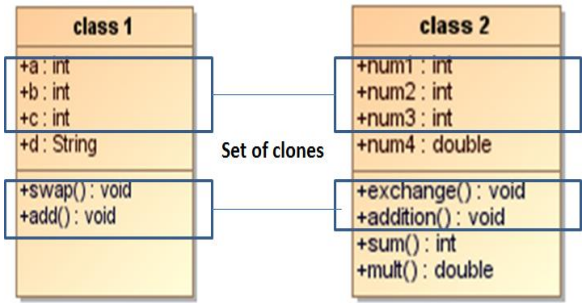

Figure 2: Model based clone detection

3. RELATED WORK RELATED TO CLONES

There are many tools related to code clone detection.Ducasse et al.[3] developed a language independent clone detection tool duploc which does line by line comparison.Simian [4] takes every file as plain text file in order to find clones.Johnson [5] applied fingerprinting technique for source code comparison.Nicad [6] is a text based hybrid clone detection tool that can detect type-3 clones (Near-Miss Clones) effectively via identification and normalization phase.CCFinder [7] uses tree matching technique in order to find similarity to find clones of large size.CloneDR [8] is capable of detecting the type-2 and type-3 clones by using hashing and dynamic programming technique.Deckard [9] converted AST (abstract syntax trees) to characteristic vectors using locality sensitive hashing.Komodoor and Horowitz's PDG-DUP [10]use program slicing to find isomorphic PDG subgraphs.Hummel et al.[11] implemented a tool in ConQAT which uses a hybrid incremental index-based clone detection technique.It is highly scalable, incremental and takes less time for execution.

There are only some tools related to model cloning as not much work has been done in its context.Liu et al.[12] developed a tool DuplicationDetector in order to detect duplications in sequence diagram.Deissenboeck et al.[13] developed a tool Clone Detective to detect clones in Simulink/Matlab Models.Pham et al.[14] presented a tool ModelCD for Matlab/Simulink Models that is able to efficiently and accurately detect both exactly matched and approximate model clones via two algorithms escan and ascan.Storrle et al.[see 2] developed a tool MQlone which produces XMI files from UML domain models and these files are transformed into prolog files.Hummel et al.[15] pioneered a tool that is based on incremental instead of batch mode clone detection.There is vagueness in model clone detection on notions of similarity which hinders the understanding of clone detection [16].

4. RESULTS IN SUPPORT OF MODEL CLONE DETECTION

In this paper the differences between the code based clone detection and the model based clone detection has been observed for the various parameters such as versioning systems which represents the same system as another version of the previous with some additional functionality.The identification, impact and refinement of risk are taken into the account.So as the cost, effort and time associated with the software.The cost for RMMM

(Risk Mitigation, Monitoring and Management) Plan is determined for them as shown in table 1.Accordingly the cost/benefit analysis is also good for the model clones because they detect the clones earlier.

Table 1:- Result in support of Code Clone Detection

Parameters	**Model Clone**	**Code Clone**
Versioning	More efficient	Less efficient
Risk identification	Earlier	Later
Cost for handling risk	Less	More
Effort in handling risk	Less	More
Time to overcome risk	Less	More
Cost associated with amplified errors	Less	More
Elimination or control of potential hazards	Earlier stage	Later stage
Risk impact	Less	More
Risk Refinement	Earlier analysis and response	Later analysis and response
Cost for RMMM	Less	More
Cost/benefit	Less	More

5. RESULTS IN SUPPORT OF CODE CLONE DETECTION

Clones are detected to reduce the inconsistency within the software but this should be done with accuracy which occurs in code clone.The errors/inconsistency detected by code clone is more for code clone as compared to models.There are more chances to detect clones in systems based on polymorphism and overriding in code clones.Further accounting for completeness, recall (true positives within the software) and precision (false positives within the software) is higher for code clones than to models as shown in table 2.

Table 2:- Result in support of Code Clone Detection

Parameters	**Model Clone**	**Code Clone**
Errors detected	Less	More
Accuracy	Less	More
Systems based on Polymorphism	Less chances to detect clones	More chances to detect clones
Systems based on Overriding	Less chances to detect clones	More chances to detect clones
Completeness	Low	High
Recall	Low	High
Precision	Low	High

6. CONCLUSIONS AND FUTURE WORK

Cloning is widely used in order to reduce risk and inconsistencies within the software.The model clone detection is done to reduce redundancy and inconsistency in the software systems at an earlier phase which doesn't hamper the cost, effort and time associated with the software as the code clone does.Code clones also reduces inconsistencies and risk but it detects them at later stage.Model clones should be used when the software to be build is strict with respect to its time and budget constraints.Also, when the software suffers a higher risk then it should be mitigated at the earlier stage with model clones.However code clones should be used when the software to be build is soft/loose with respect to its time and budget constraints.When the system needs accuracy with high recall then code clones should be preferred.This analysis has been done on the detection phase only but it can be extended to the removal of clones after detection by keeping only one original copy and automatically removing the duplicate copies.

7. REFERENCES

[1] C.K.Roy, J.R.Cordy, A Survey on Software Clone Detection Research, Technical Report 2007-541, Queen's University at Kingston Ontario, Canada, 2007, p.115.

[2] H.Storrle,Towards clone detection in UML domain models, in: Proceedings Software & Systems Modeling, Volume 12, Issue 2, 2013,pp.307-329.

[3] S.Ducasse, M.Rieger, S.Demeyer, A language independent approach for detecting duplicated code, in: Proceedings of the 15th International Conference on Software Maintenance (ICSM'99), Oxford, England, UK, 1999,pp.109–119.

[4] Tool Simian <http://www.harukizaemon.com/ simian/index.html> (accessed April 2012).

[5] J.H.Johnson, Substring matching for clone detection and change tracking, in:Proceedings of the 10th International Conference on Software Maintenance,Victoria, British Columbia, Canada, 1994, pp.120–126.

[6] C.K.Roy, 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'08), Amsterdam, The Netherlands, 2008, pp.172–181.

[7] T.Kamiya, S.Kusumoto, K.Inoue, CCFinder: a multi-linguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering 28 (7) (2002) 654–670.

[8] I.D.Baxter, A.Yahin, L.Moura, M.Sant'Anna, L.Bier, Clone detection usingabstract syntax trees, in: Proceedings of the 14th International Conference on Software Maintenance (ICSM '98), Bethesda, Maryland, USA, 1998, pp.368– 378.

[9] L.Jiang, G.Misherghi, Z.Su, S.Glondu, DECKARD: Scalable and accurate treebased detection of code clones, in: Proceedings of 29th International Conference on Software Engineering (ICSE'07), Minneapolis, MN, USA, 2007, pp.96–105.

[10] RaghavanKomondoor.Automated Duplicated-Code Detection and Procedure Extraction.Ph.D.Thesis, 2003.

[11] B.Hummel, E.Juergens, L.Heinemann, M.Conradt, Index-based code clonedetection: Incremental, distributed, scalable, in: Proceedings of the 26th IEEE International Conference on Software Maintenance (ICSM'10), Timisoara,Romania, 2010, pp.1–9.

[12] 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, pp.269–276.

[13] F.Deissenboeck, B.Hummel, E.Juergens, B.Schätz, S.Wagner, J.Girard, S.Teuchert, Clone detection in automotive model-based development, in: Proceedings of 30th International Conference on Software Engineering (ICSE'08), Leipzig, Germany, 2008, pp.603–612.

[14] N.H.Pham, H.A.Nguyen, T.T.Nguyen, J.M.Al-Kofahi, T.N.Nguyen, Complete and accurate clone detection in graph based models, in: Proceedings of 31st International Conference on Software Engineering (ICSE'09), Vancouver, Canada, 2009, pp.276–286.

[15] B.Hummel, E.Juergens, D.Steidl, Index-based model clone detection, in: Proceedings of 5th International Workshop on Software Clones, Honolulu, USA, 2011, pp.21–27.

[16] D.Rattan, R.K.Bhatia, M.Singh: Software clone detection: A systematic review, in: Information & Software Technology, Volume- 55, 2013, pp.1165-1199.