International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3 Issue No.6, pp : 765-766 1 June 2014

Clone Removal V/S Clone Avoidance

Ritu Garg, Rajesh Bhatia

Deenbandhu Chhotu Ram University of Science and Technology, Murthal
Punjab Engineering College University of Technology,Chandigarh
Ritugarg.engg@gmail.com, rbhatiapatiala@gmail.com

Abstract -- Cloning occurs in software when there is redundancy in it in form of any similarity. So, the clones need to be removed or it can be avoided from the software in order to mitigate the negative impacts of clones. In this paper we have studied the various factors that affect the decision of avoidance or removal of clones for handling them.

Keywords -- Software clones, Code clones, Model clones, Software system

I. Introduction

The clones [1] can occur as the redundant component in form of fragments in the software.During the software development process there is a high probability of cloning due to time, budget constraints or programmer limitations [2, 11]. Mainly two types of clones exist during software development life cycle [14]:-

1. Code clones [3] 2. Model clones [4]

The code clones are the redundant component that exists in the implementation phase during coding the software. This stage occurs after designing phase. There are four types of code clones – exact code clone, renamed or modified code clone, near miss code clones and semanticcode clones [1].

Similarly the model clones are the redundant component that exists in the design phase during the software designing after feasibility study and before implementation phase. Similar to code clones these can also be divided in four categories- exact model clone, renamed or modified model clone, near miss model clones and semanticmodel clones [5].

These clones if present in the software can lead to update anomalies [6, 7] and change at one place needs to be changed at all the other duplicate components. Due to this, cloning can increase the cost and time [7, 12] associated with the software under construction as the risk involved with the softwareincreases. In order to decrease the impacts of cloning, the clones need to be identified and removed [13] but it is not always possible that we can remove these clones. So, they can be avoided to mitigate its effects. The detection process must be followed by either avoidance or removal of clones.

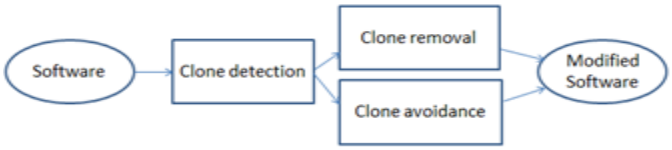

Figure 1: Different phases in clone handling

2. Selection Technique

The selection of clone avoidance or removal depends on the following factors:-

Nature of software :- A software with many reusable components allow more reusing and less cloning so the clones can be avoided easily in this case while the software with less reusable components allows less reusing and more cloning so, the clones should be removed. Thus, the amount of reusing in the software determines the clone avoidance or clone removal process.

A. Independent component: - Between two dependent systems during the clone detection phase the clones are represented in form of clone pairs. These clone pairs are analyzed if they can exist as an independent component that is any type of internal dependence to and fro from the component is prohibited. If the fragment represented in form of clone pairs has high cohesion and is in less coupling with other components of the software then only it can exist as an independent component. If clone pair can be represented in form of independent component then clones must be removed otherwise the clone should be avoided.

B. Cost/Profit Metric: - This metric is based on the assumptions where cost represents the money which will be involved in removing the clones from the software while the profit represents the benefits that will be provided on removing of these clones. The value of profit parameter can be decided on basis of current software between which the clones have been detected while the value of cost parameter can be provided by the previous similar software where such similar type of clone pairs have been removed.

If the value of cost/profit metric is positive so the cost for removals of these clone pairs would be more than the profit that can be achieved by removing them. Therefore the clones must be avoided to develop or maintain the software within strict cost and time schedule. If the value of cost/profit metric is negative so the cost for removals of these clone pairs would be less than the profit that can be achieved by removing them. Therefore the clones must be removed to develop or maintain the software within strict cost and time schedule. Here removal would be beneficial for the software development life cycle.

Using all these three metrics, we can differentiate between the clones that should be removed from the software.

3. Results (How clones can be removedbetween dependent systems using independent component approach)

The clone pairs in form of independent component can be removed from the software's and put into the library as a library candidate so that it can be reused by current software

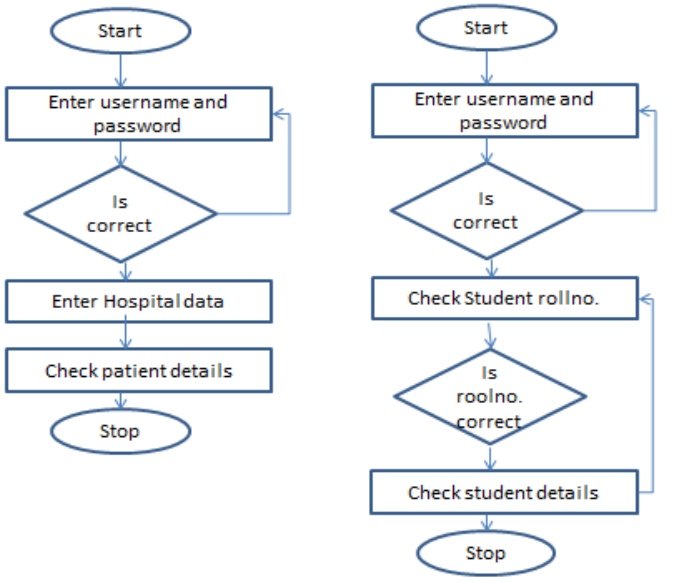

Figure 2: Software's design before clone removal process

and for the future use. So, in current software from which clone pairs were detected reuses that library candidate in all occurrences. It includes both original occurrences and all duplicate occurrences. It must be carefully determined that the size of independent component should be large enough otherwise making every very small component and then reusing could increase the effort, time and cost of the software which is not desirable.

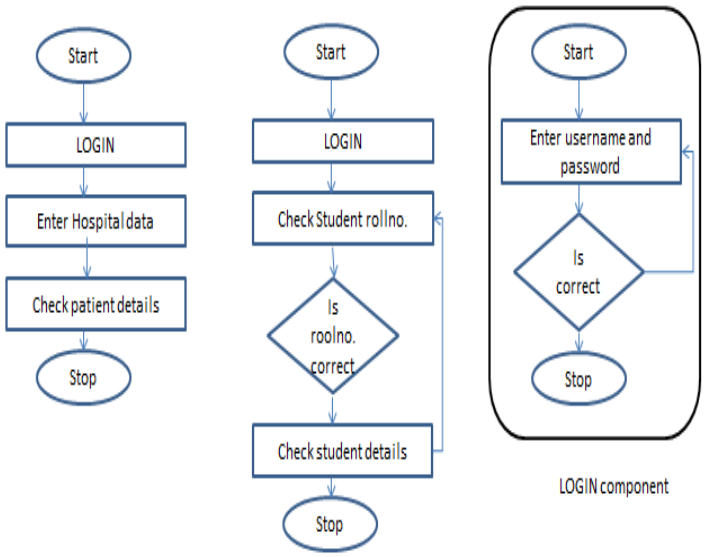

Figure 3: Software's design after clone removal process

4. Conclusions

In this paper we have identified the clones that should be kept in the software by avoiding them. Also it has identified the clones that should be deleted from the software by removing them. It is also studied how these clones can be removed and why there is a need for avoidance or removal arises.

References

i. *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.*

ii. *M. Kim, L. Bergman, T. Lau, D. Notkin, An Ethnographic study of copy and paste programming practices in OOPL, in: Proceedings of 3rd International ACM-IEEE Symposium on Empirical Software Engineering (ISESE'04), Redondo Beach, CA, USA, 2004, pp. 83–92.*

iii. *S. Bellon, R. Koschke, G. Antoniol, J. Krinke, E. Merlo, Comparison and evaluation of clone detection tools, IEEE Transactions on Software Engineering 33 (9) (2007) 577–591.*

iv. *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.*

v. *H. Storrle, Towards clone detection in UML domain models, in: Proceedings of European Conference on Software Architecture (ECSA'10), Copenhagen, Denmark, 2010, pp. 285–293.*

vi. *R. Koschke, Frontiers of software clone management, in: Proceedings of Frontiers of Software Maintenance (FoSM'08), Beijing, China, 2008, pp. 119– 128.*

vii. *J. Mayrand, C. Leblanc, E.M. Merlo, Experiment on the automatic detection of function clones in a software system using metrics, in: Proceedings of the 12th International Conference on Software Maintenance (ICSM'96), Monterey, CA, USA, 1996, pp. 244–253.*

viii. *C.J. Kapser, M.W. Godfrey, Supporting the analysis of clones in software systems: a case study, Journal of Software Maintenance and Evolution: Research and Practice 18 (2) (2006) 61–82.*

ix. *B. Baker, On finding duplication and near-duplication in large software systems, in: Proceedings of the 2nd Working Conference on Reverse Engineering (WCRE'95), Toronto, Ontario, Canada, 1995, pp. 86–95.*

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

xi. *D Rattan, R Bhatia, M Singh, Model clone detection based on tree comparison, in: India Conference (INDICON), 2012 Annual IEEE, pp.1041-1046.*

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

xiii. *Y. Sharma,R. Bhatia,R . K .Tekchandani ,Thesis on Hybrid technique for object oriented software Clone detection ,in Electronic Theses & Dissertations @ TU, 2011.*

xiv. *R.Garg,R. Bhatia,Code Clone v/s Model Clones: Pros and Cons, in:International Journal of Computer Applications (IJCA),Volume 89 – No 15, 2014, pp. 20-22*