Chapter 6

Conclusions and Future Work

In this thesis, we have carried out a thorough systematic literature review of software clones in general and software clone detection in particular. Systematic literature reviews are carried out in accordance with predefined search strategy and provides a thorough and complete review of existing literature. The thesis has presented a technique for clone detection in UML models. We have also devised a tool to detect high level similarities in source code and between UML class models. Section 6.1 concludes the thesis and section 6.2 discusses future scope of work.

6.1 Conclusions

We believe that the results of our systematic literature review will be useful for any researcher who wants to carry forward the research in any domain pertaining to software clones such as clone management, clone detection, clone analysis, impact of software clones on software quality, etc. Further model and semantic clone detection techniques have been investigated. We noticed that the series of the International Workshops on Detection of Software Clones have made a significant contribution towards promotion of research in the field of software clones.

Software developers in industry deal with large amounts of process and project data. So clone management tools should be scalable and integrated into development environments, to help programmers understand the behavior of cloning patterns. A clone management tool having integrated detection and developer friendly visualization of clones would help the developers observe clones as and when they proliferate during development. The tool should be able to detect real clones i.e. clones which are really interesting and useful to the developer and should not report non-useful and uninteresting clones.

There are many language specific issues which hinder code clone classification. Subjective studies carried out by several human experts vary a lot on creating reference data for creating different benchmark suits. There is disagreement between human experts as to whether the candidate code is or is not a clone. It is a definite challenge as it is a difficult and time-consuming task to manually classify the candidates as clones or not. Thus, we believe that experts from industry and academia related to diverse domains of
Clone detection should come together to create a verified reference corpus of clone candidates in standard subject systems. The study should be carried out differently for each type of clone and depending upon use case. Such benchmark suites would make the results of empirical comparison consistent and reliable for use in research and industry.

There is a lack of research in cloning beyond source code. There are different software artifacts where cloning may occur. Clones do occur in requirement specifications, models and test cases too. There is an urgent need to explore the reasons for clones and efficient clone detection techniques in these software artifacts. Different artifacts have inherent characteristics which have to be exploited to apply the clone detection algorithm for that artifact. Empirical studies need to be carried out to understand the patterns in clone evolution for various artifacts. We realised the importance by proposing the techniques for clone detection for UML models. Clones detection and removal in earlier phases of software development life cycle will reduce the maintenance costs.

Software developers tend to copy the existing code into other sections of code. They may intentionally modify it to bring the desired behavior. But, if there is any bug detected in one section of code that has been copied at other places, the same bug will propagate to other sections. Software clones affect system quality. Model driven development is a standard practice in industry. Since, models are usually created before the source code, so these are closer to real world concepts. Hence, the presence of clones in models like UML models will affect the quality of the system, further.

The objective of our work is to explore UML models for the presence of clones. In our thesis, we have devised a strategy in which UML models are created using any standard modeling tool. These models are then exported to XMI representation. The XMI file is parsed to extract different attributes of the UML class model like name of the class, name of the field with data type, name of the method and return type, etc. The core of our technique is the construction of labeled, ranked tree such that its subtrees represent field and its data type and method signatures. An existing algorithm identifies various repeating subtrees of different sizes from the constructed tree. By grouping and clustering of these repeating subtrees, the proposed technique is able to detect model clones of different granularity. The novel classification of model clones provide useful insights into the software modeling practices. Another highlight of the proposed technique is its ability
to detect clones at three different levels of granularity in a UML class model i.e. complete class, attributes with their data types and methods with their signatures and cluster of such repeating attributes and methods.

It has been observed that there is clear need to address the lack of empirical studies to examine the effects of cloning in real world models. There is no open source repository of UML models of real world industrial applications due to different proprietary reasons. No UML based model clone detection tool is available for open source subject systems. So, we carried out the empirical evaluation of the proposed technique using open source and forward designed models. We firmly believe that our results will help the researchers and practitioners to compare and evaluate the model clone detection techniques in choosing the right technique based on application.

A large number of clone detection tools in the literature focus on fragments of duplicated code with smaller granularity (Chapter 2). To comprehend and understand large systems, one needs to see the bigger picture. Thus the second technique of our thesis is focused on identifying the similarity at higher levels in source code and UML models. We termed these clones of high granularity as concept clones. Since, users may choose different implementations in the form of class models for the same real world concept. Thus, we extended our technique to find similarity between two different UML class models. Detection of concept clones finds its applications in different domains like software product lines, clone evolution, software plagiarism detection, etc.

The proposed technique is able to identify concept clones by applying principal component analysis and latent semantic indexing. The distance between two fragments of source code or UML models is calculated by applying cosine similarity. It is more important to identify concept clones at abstract model level than at lower levels. So we evaluated the tool by applying it to sample UML class models down to method level within same source code file. It is important to understand code clone evolution to know whether clones are harmful or not and to know the impact of code clones on maintenance. Thus our tool is also able to estimate the extent of similarity between class files across versions of an open source software system. The results will assist the software engineer to understand as to why two different domain concepts are so close.
The major contributions of our work are:

- A systematic literature review was carried out to show the current status of clone detection, various techniques of clone detection and all allied areas.

- Proposed the classification of model clones as type-1 model clones due to standard modeling practice, type-2 model clones due to purpose and type-3 model clones due to design practices.

- Developed two model based clone detection techniques and their validation on open source subject systems.

- Both the proposed techniques have successfully detected model clones of different granularity level.

6.2 Future Scope of Work

- More studies need to be undertaken to know how harmful actually clones are. Automatic tools for investigation and visualization of clone genealogies across different versions of the software will be helpful for developers.

- The behavior and impact due to code clones is still not known precisely. The study of changeability of cloned vs non-cloned system depends on the choice of application as some applications are continuously restructured. Such studies should be empirically carried out using different types of clone detectors on large subject system base to conclude general remarks. Code fragments do not appear consistently in all revisions of the software. The code fragment skips one version and again reappears in the next revision. This non-continuous code appearing across revisions of the software need to be validated with developers’ intent as to why code has been copied from the old revisions.

- The analytical results of any clone detection tool need to be complemented with empirical behavioral studies of the practitioners. The use of self learning technique and history based log techniques can be explore in the field of software clone detection.

- The reliable and scalable detection of behaviorally similar code is an open research area. In software reuse we need to identify the most relevant component for given
context. From existing software repository, developer may find many candidate components for given context, the components that are semantically same but may differ in one or other criteria such as cyclomatic complexity, algorithmic complexity, time/ space trade-off, known bugs and many more. With the help of semantic clone detection, developer will be able to find out most suitable and efficient component out of available candidate components.

- Different forms of models have individual features which need to be exploited for clone detection. We noticed vagueness in the definition of model clones. There is apparent need to classify the comparative and empirical studies differently for Matlab/Simulink models, UML models and data flow models as the application area is different. We foresee the use of multiple threads or parallel machines to speed up and distribute the task of retrieval and detection in future.

- Currently, the tool detects clones in UML class models only. In future, work can be carried out to know how type-3 clones can be used to improve the design? UML model refactoring has emerged as an allied area similar to code refactoring. Detection of patterns for model refactoring with the help of our model clone detection approach together with semantic preservation may help in improving the design and structure of the UML class model. Further, the proposed technique can be extended for other UML models.

- A mapping of UML constructs from syntactical to semantic domain may help in detecting clones having same behavior but different syntactical structure. Our work can be extended further to detect semantic similarities by using formal methods such as object constraint language, Object–Z, etc.