Model Clone Detection based on Tree Comparison

Dhavleesh Rattan
Department of Computer Science and Engineering and Information Technology
Baba Banda Singh Bahadur Engineering College,
Fatehgarh Sahib – 140407, Punjab, India
dhavleesh.rattan@bbsbec.ac.in

Rajesh Bhatia
Department of Computer Science and Engineering
Deenbandhu Chhotu Ram University of Science and Technology, Murthal (Sonepat) -131039, Haryana, India
rbhatiapatiala@gmail.com

Maninder Singh
Computer Science and Engineering Department
Thapar University, Patiala – 147004, Punjab, India
msingh@thapar.edu

Abstract—**Model driven development has become a key industry practice. With higher levels of abstraction and advent of domain specific languages, models find their presence in every field. Latest software engineering practices lead to large models which are really hard to design and manage. Significant overlaps in large models are really a matter of concern. Anecdotal evidences suggest that clones in models poses similar threats as in code. The paper introduces an approach to detect clones in UML models. The technique is based on finding similarities between two object oriented diagrams. Firstly, UML models are encoded as XMI files. Subtree comparison is applied after the XMI file is filtered and represented as a tree. Similarity between two diagram elements in a model is determined and reported as a clone.**

Keyword-model clones; tree comparison; XMI file;

I. INTRODUCTION

Copying the existing code and pasting with or without modifications is a continuous process in software development. As a result, similar sections of code appear in software systems, known as software clones [16], [9]. Clones in source code have been extensively studied. There are number of studies showing effects of code clones. Generally they are considered harmful and increases maintenance cost. Recent studies [3], [7], [17] showed the presence of clones in other software artifacts.

The graphical *Unified Modeling Language* (*UML*) is increasingly replacing conventional programming languages for developing software systems [20]. UML models have become an integral part of large software development and maintenance process. In large and complex systems, models attain substantial size. Unexpected overlaps in large complex systems are present. It is important to detect those duplications in model so that models can be refactored [2], thus stepping towards a better design. Though there is lack of empirical studies and model clone detection tools showing consequences of cloning in UML models, yet large number of studies carried out in source code confirms the consequences and effects of cloning in general [1],[15]. The presence of clones in models has similar side effects as in source code.

This paper presents an approach to detect clones in UML models. In this paper we propose a technique that finds the similarity between two UML class diagrams. Firstly, UML class diagram is encoded as an XMI file. The class diagrams are stored in the form of XML tree similar to DOM tree using XML parsing. Detection is done using subtree comparison in XML tree. Similarity between two fragments is reported as a clone. Similar techniques have been used in the past to detect differences between versions of UML diagrams [8], [13]. The applicability of our approach is demonstrated by taking UML class diagram as example.

A. Motivation

- 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 [20].
- Models attain substantial size [17], thereby increasing complexity and higher rate of duplications.
- There is a dearth of studies suggesting techniques for detection of clones in UML models.

The remainder of this article is structured as follows. Section 2 presents the background, definitions and general theories on model clones and model clone detection. Section 3 describes our approach of clone detection. Section 4 presents the related work. Section 5 concludes and provides recommendations for further research in the area of model clone detection.

II. BACKGROUND

A. Cloning

The Merriam-Webster dictionary defines a clone as one that appears to be a copy of an original form, thus being synonymous to a duplicate. In source code and other software artifacts, the original (code) fragment is copied and pasted with or without modifications. The pasted (code) fragment is said to be a clone and this activity is known as (code) cloning [1], [9], [15], [16]. However in software engineering field, the term

code clones is still searching for a suitable definition. Its vagueness was properly reflected in Ira Baxter's words:

"Clones are segments of code that are similar according to some definition of similarity."

A bug or correction detected in one section of code multiplies across all the replicated fragments of code. Thus, it is important to find those places throughout the source code. Considering high maintenance cost, software clone detection has emerged as an active research area. Different programming paradigms and languages have led to number of clone variants and detection techniques. Just as in code based development, presence of model clones increase maintenance cost.

B. Model Clones

The previous research has proved the presence of clones in models. Nowadays graphical languages are replacing the code as core artifacts for system development. Unexpected overlaps and duplications in models [3], [20] are termed as model based clones. Since models give a visual representation of the concept, generally models are defined with the help of graphs. Desissenboeck et al. [3] defined model graph as:

The clone in a model can be defined as a weakly connected sub graph $G_1 = (V_1; E_1)$ of G that is isomorphic to at least one other sub graph $G_2 = (V_2; E_2)$ of G w.r.t. the labeling function L. Fig 1 and Fig 2 show two packages. They have two classes namely, *student* and *course* similar, thus clones.

C. Why Clones in UML Models

UML modeling has become a key industry practice. Though modeling is a well established field but research in detecting clones in those models is still in infancy. We highlight some of the reasons for the presence of model clones:

- **Model clones through copy/paste**: Ad hoc reuse through copy/paste creates model clones. Developers adopted this as the fast and immediate way of meeting the change [18].

- **Model clones through language limitations:** Sometimes the modeling language fail to factor out the common parts. Thus they essentially appear in the model [16], [18].

- **Programmers Limitation and Time Constraints:** The modeling is sometimes done under severe time constraints. Limitations of programmer's skills and hard time constraints inhibit proper modeling [9].

- **Complexity of the System:** The difficulty in understanding large systems only promotes copying the existing functionality and logic [20].

- **Lack of Restructuring:** Developers delay restructuring (refactoring, abstraction, etc.) of model due to time limits. Often, restructuring gets delayed indefinitely.

D. Challenges in identifying clones in UML models

- **Notion of model clone**
  There is no standard and agreed upon definition of model clone.
- **No standard classification of model clones**
  There is no general classification of types of model clones
- **Large sizes of the model**
  Large size models need highly scalable techniques of model clone detection.

III. OUR APPROACH IN IDENTIFYING CLONES IN UML MODELS

The technique is based on exporting the model to an XMI file. XMI files contain large number of tool specific and auxiliary data. It stores a large amount of layout information. So textual clone detection based on XMI file contains a number of irrelevant repetitions [8]. Converting the XMI file to a tree representation filters irrelevant data and prunes the search space. In this way most important elements are identified as nodes of the tree. Comparing the subtrees helps in recognizing the similar diagram elements of a UML model, thus helping in clone detection. Importantly, in a UML class diagram, we have made one subtree for one class. Every path of the tree contains a sequence from arbitrary root down to class name, attributes and operations.

Then model is stored in the form of an XML tree. Since the approach uses XMI standard, it applies to all models generated using any modeling tool. We used tree comparison to find similarity in two fragments of a model. The basic steps followed are:

- Firstly, the model is created using any modeling tool.

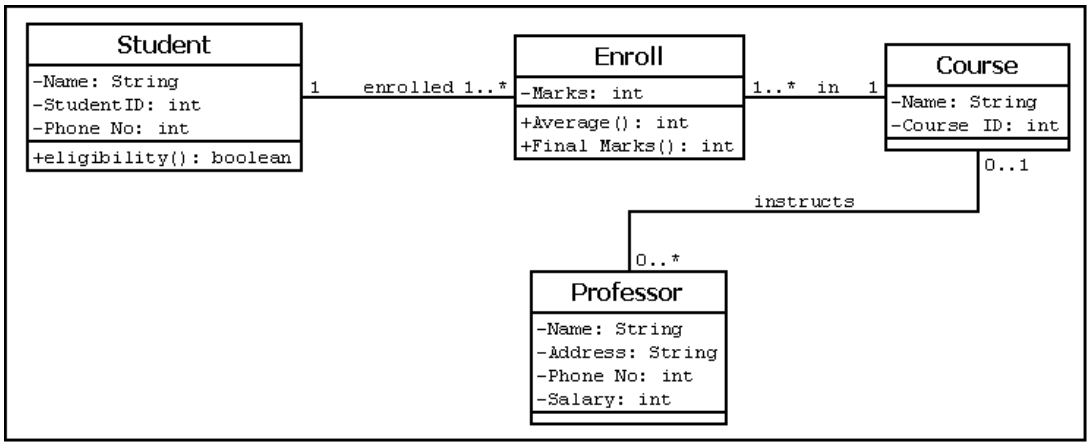

Fig 1. Class Diagram I (Package: Enrollment)

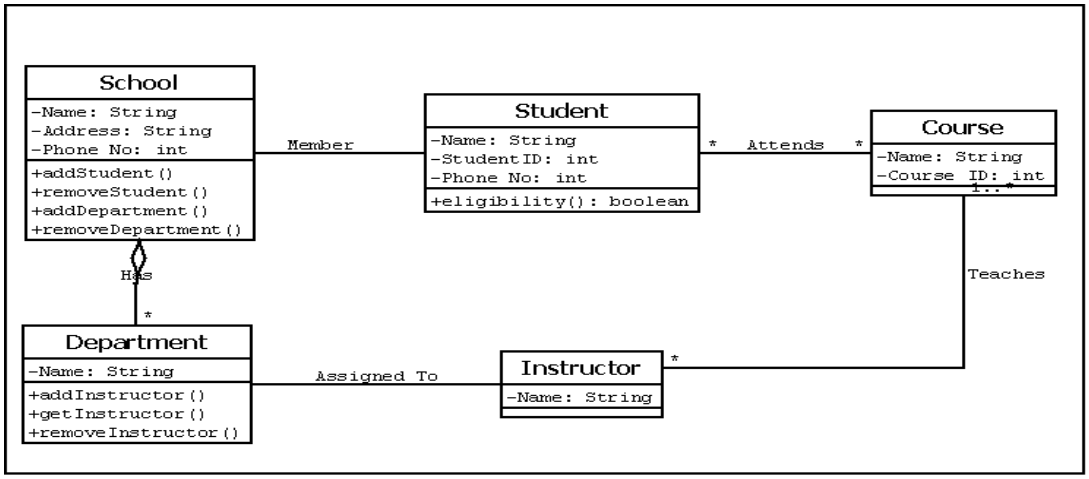

Fig 2. Class Diagram II (Package: Teaching)

- Secondly, the model is exported to XMI (XML Metadata Interchange) file format. Since XMI is a standard given by OMG, it is built in most of the modeling tools.

- Thirdly, XMI file is preprocessed and is stored in the form of tree using DOM API's and XML parsing.

- Fourthly, subtrees are compared and similarity is reported in the form of model clones.

The block diagram showing various steps of clone detection is shown in figure 3. UML models encoded as XMI files have been used in the past to detect and visualize differences between pairs of UML diagrams [8], [12], [13], [19]. We have explored those techniques to detect clones in UML models. The elements of diagram element are compared and similarity is reported in the form of clone. Comparison starts from the root of the tree and works level by level. Whereas the differencing algorithms [8], [19] work on the basis of unique ID's, our algorithm compared every subtree with other subtree. Our technique works within the diagram elements of a package and between two packages as shown in figure I and figure II.

A. Salient Features of our Approach

- The technique is scalable.
- We capture meaningful attributes of XMI file and store them in the form of tree. Though UML model can be represented as a graph where nodes represent diagram elements and edges represent connections between them. For instance, in a class diagram, nodes represent a class and relationships between two classes are represented as edges. Storrle [18] concludes that UML models contain heavy and dense nodes as large information is stored in these nodes. So any model based on exploring these nodes for clone detection will give better results. Model clone detection techniques applied on Matlab/Simulink models [3], [14] considered model as a graph. In

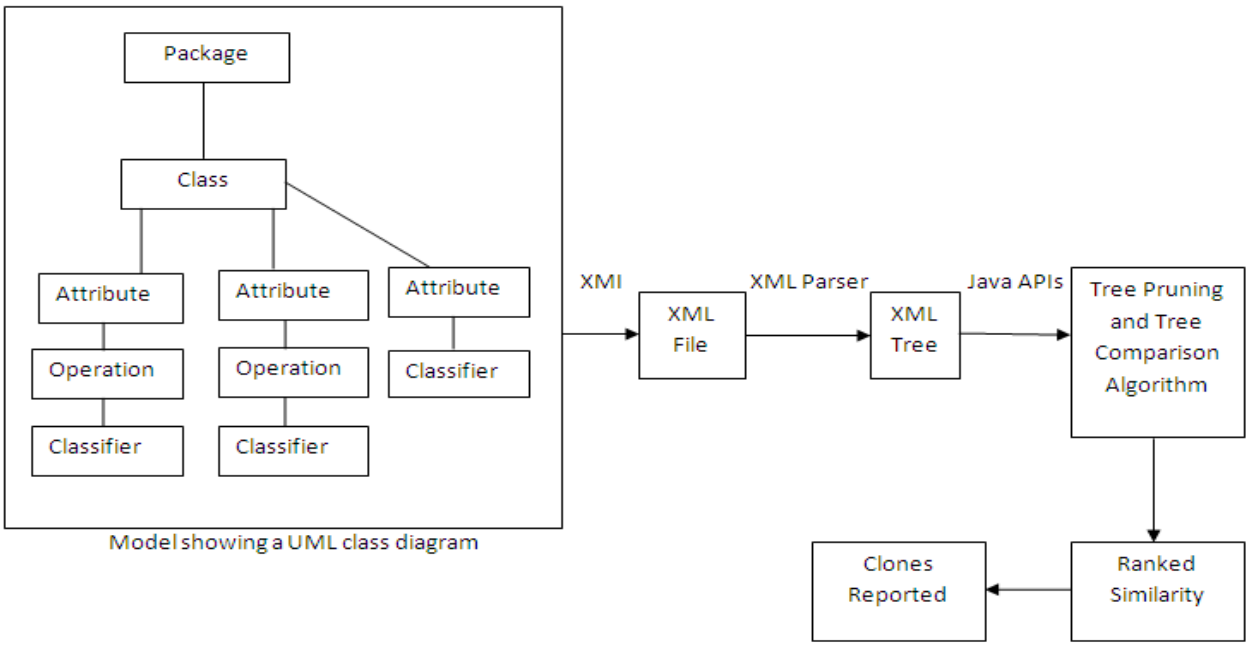

Fig 3. Block Diagram of Clone Detection

those models, nodes are comparatively lightweight. So computationally demanding algorithm of graph isomorphism will not be useful in case of UML models as UML models are loosely connected fat nodes rather than densely connected graphs of lightweight nodes [18].

- Existing model clone detection tools for Matlab/Simulink models report many clones as irrelevant. Since we are storing key elements of UML diagram in the form of a tree, irrelevant clones are not reported.
- The technique is implemented in Java. Figure 3 shows the block diagram of our methodology. Fig 1and Fig 2 reports two clones: Class *Student* and Class *Course* in both packages Teaching and Enrollment are reported as clones with high similarity.

IV. RELATED WORK

In this section we present an overview of existing algorithms for clone detection in models. We also highlight the advantages, shortcomings and applications of every study. Upon assessing the literature, we came across clone detection techniques for Matlab/ Simulink and only one study to detect UML domain models.

Liu et al. [10] detected duplications in sequence diagrams by converting the 2-dimensional sequence diagram to 1- dimensional array. Then, 1-dimensional array is used to build suffix tree. Common prefixes are found out from the suffix tree in the form of reusable sequence diagram as refactoring candidates. The study confirmed the presence of 14% duplication in sequence diagrams of sample industrial projects.

The automatic detection of clones in models leads to identification of potential domain specific library elements [3]. Deissenboeck et al. [3] used ConQAT as an integrated framework to detect clones in Matlab/Simulink models especially in automotive domain. The tool CloneDetective works by representing the model as labeled multi-graph as labels are assigned to relevant blocks. Similarity between blocks is checked by maximum weighted bipartite matching technique using heuristics. After detection, clones are clustered based on set of nodes using union function. It leads to large number of false positives in the result.

Pham et al. [14] gave two algorithms namely ‘escan’ and ‘ascan’ to detect clones in models. Firstly, the model was preprocessed to be represented as sparse, labeled directed graph. ‘escan’ was used to detect exact matching using an advanced graph matching technique called canonical labeling and ‘ascan’ was used to detect approximate matching by counting vector of sequence of nodes and edges’ labels. The technique is incremental in the way it generates candidate cloned subgraphs. Their tool ModelCD was compared with other state of the art model clone detection tool CloneDetective [3]. For same clone granularity and subject systems, both the tools are compared based on 4 parameters: Precision, completeness, scalability, incrementality. ModelCD performs better. In nutshell, most of the suggested improvements by Deissenboeck et al. [3] are carried out by Pham et al. [14].

H. Storrle [17], [18] pioneered the detection of clones in all types of UML domain models. The technique was based

on model querying. Using any of the UML case tools, XMI files are generated from UML domain models. These files are transformed into Prolog files, syntactically. A small model is input in the query and using model matching, the output is generated. The tool is capable of comparing models originating from diverse sources including a variety of UML versions. Hummel et al. [5] introduced an incremental algorithm for model clone detection. A Simulink/ Matlab model is preprocessed by flattening the model into a directed multigraph. Then, relevant edges and blocks are labeled. A clone index is created for all subgraphs of same size. Canonical label of all subgraphs in the clone index is calculated and similar labels are hashed. Clone retrieval and index update are integrated for fast retrieval.

Deissenboeck et al. [4] mentioned the presence of large number of false positives in Simulink/ Matlab models, thereby extracting relevant clones is important. Model clones suffer from the problem of scalability, clone inspection and relevance. The study provided useful insights in addressing these problems in real time industrial context. The authors proposed reducing the size of models to make clone detection speedier. Firstly, removal of obvious cloned sub-systems is carried out. Similar to the approach proposed by Pham et al. [14], all nodes with high degree are removed. After detection is over, the algorithm tries to connect smaller nodes that are connected to each other over high degree nodes

The aim of clone detection in models is to identify duplicate elements in models. Maintenance of models gets improved if the maintainer is aware about of presence of clones in models. If functionality of one clone is comprehend, then it become easy to get overall idea of other files containing the instances of the clone. Irrelevant clones can be removed. The libraries can be constructed for blocks which are used at multiple places so that these can be used by proper reusable mechanism. Detection of model clones can also help to decrease resource requirements of a model. Figure 1 and figure 2 shows an example of clones in two different UML class diagrams.

An empirical study [7] showed significant cloning in Matlab/Simulink models. This study had used ConQAT [6] to find some useful patterns of model clones in thirteen Matlab/Simulink models ranging from 218 blocks to 73888 blocks.

V. CONCLUSION AND FUTURE WORK

Anecdotal evidence suggests clones in models hamper maintenance as in code based development. In this paper we present a technique to detect clones in UML class diagram encoded as XMI file. Currently the similarity is reported as output. In future we will extend the prototypical implementation to display the clone detection results in more user friendly manner with percentage of similarity.

Our algorithm will work for all object oriented diagrams as modeling tools provide the facility to export the model to XMI format. The present technique can be extended to state chart diagrams, activity diagrams, etc.

We will try to classify clones for UML models so that efficient model clone detection techniques can be developed. This will help in reporting the percentage of similarity for different clones with proper classification.

Martin and Cordy [11] proposed representing models as XML file using XMI format. The paper proposed using text based clone detection tool to detect clones in XML file.

We are currently trying to implement our algorithm on large number of class diagrams as well as other object oriented diagrams. To overcome the scalability issues in large size models, we are also working on a technique of XML clustering using principal component analysis and latent semantic analysis to report the distance between two principal components made of XML subtrees.

REFERENCES

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

[2] A. Correa and C. Werner, "Applying refactoring techniques to UML/ OCL models" *The Unified Modeling Language*, LNCS Vol. 3273, pp. 173-187, 2004.

[3] 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 International Conference on Software Engineering*, Leipzig, Germany, pp. 603-612, 2008.

[4] F. Deissenboeck, B. Hummel, E. Juergens, M. Pfaehler and B. Schaetz, "Model clone detection in practice", *Proceedings of 4th International Workshop on Software Clones*, New York, U.S.A., pp. 37-44, 2010.

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

[6] E. Juergens, F. Deissenboeck, and B. Hummel, "CloneDetective–A workbench for clone detection research", *Proceedings of 31st International Conference on Software Engineering*, Vancouver, Canada, pp. 603-606, 2009.

[7] M. Kaur, D. Rattan, R. Bhatia, and M. Singh, "Clone detection in models: An empirical study", *3rd IBM Collaborative Academia Research Exchange (I-CARE) 2011*, New Delhi, India, 2011.

[8] U. Kelte, J. Wehren and J. Niere, "A generic difference algorithm for UML models" *Proceedings of SE 2005*, Essen, Germany, pp. 105-116, 2005.

[9] R. Koschke, "Survey of research on software clones", *Duplication, Redundancy, and Similarity in Software*, *Dagstuhl Seminar Proceedings,* pp. 24, 2006.

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

[11] D. Martin and J. R. Cordy, "Analyzing web service similarity using contextual clones", *Proc. of 5th Int'l workshop on Software Clones*, pp. 41-46, 2011.

[12] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook and P. Zave, "Matching and merging of statecharts specifications" *Proceedings of 29th International Conference on Software Engineering*, pp. 54- 64, 2007.

[13] D. Ohst, M. Welle and U. Kelter, "Differences between versions of UML diagrams", *Proceedings of 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of Software Engineering*, Helsinki, Finland, pp. 227 – 236, 2003.

[14] N. H. Pham, H. A. Nguyen, T.T. Nguyen, J.M. Al-Kofahi and T. N. Nguyen, "Complete and accurate clone detection in graph-based models". *Proceedings of 31st International Conference on Software Engineering,* Vancouver, Canada*,* pp. 276- 286, 2009.

[15] C. K. Roy, J.R. Cordy and R. Koschke, "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.

[16] C. K. Roy 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.
[17] H. Storrle, “Towards clone detection in UML domain models”, *Proceedings of European Conference on Software Architecture (ECSA’10),* Copenhagen, Denmark, pp. 285-293, 2010.
[18] H. Storrle, “Towards clone detection in UML domain models”, *Software and Systems Modeling*, doi 10.1007/s10270-011-0217-9, pp. 39, 2011.
[19] Y. Wang, D. J. DeWitt and J. Cai, “X-Diff: An effective change detection algorithm for XML documents”, *Proceedings of 19th International conference on Data Engineering*, pp. 519-530, 2003.
[20] T. Weilkiens, Systems Engineering with SysML/ UML, Morgan Kaufmann, 2007.