Reducing Cloning in Software Models Using UML Parsing

Anisha Rani
M.Tech (IT)
Guru Nanak Dev Engineering College, Ludhiana (Punjab)

Raninder Kaur Dhillon
Asst. Prof. (IT)
Guru Nanak Dev Engineering College, Ludhiana (Punjab)

Pankaj Goel
Asst. Prof. (CSE)
RIMT-IET, Mandi Gobindgarh (Punjab)

ABSTRACT

This paper presents a way for reducing cloning in software models using a high level language which parse XML of models. Model cloning is usually a strategic means by which evolution of the software product is measured. Model clone detection deals with the identification of duplicated parts in models [4]. Cloned code is considered harmful for various reasons. A clone contains multiple and unnecessary duplicate fragments of code. Due to cloning, maintenance costs are increased. They make software products inconsistent as making changes to a cloned code, can create faults and lead to unexpected behaviour. Similarly, duplicated fragments and parts of models are also harmful in model-based development such as unified modeling language (UML). In order to remove duplicity in models, model clone detection technique uses.

Keywords: *Software Engineering, UML Domain Models, Model Clones, UML Parsing, Clone Detection.*

1. INTRODUCTION

1.1 Code Clones

The copying of code has been studied within software engineering mostly in the area of clone analysis. Software clones are regions of source code which are highly similar; these regions of similarity are called clones, clone classes, or clone pairs [17]. Clone identification has great potential in the maintenance and re-engineering of legacy systems [7].

While there are several reasons why two regions of code may be similar, the majority of the clone analysis literature attributes cloning activity to the intentional copying and duplication of code by programmers; clones may also be attributable to automatically generated code, or the constraints imposed by the use of a particular framework or library [9].

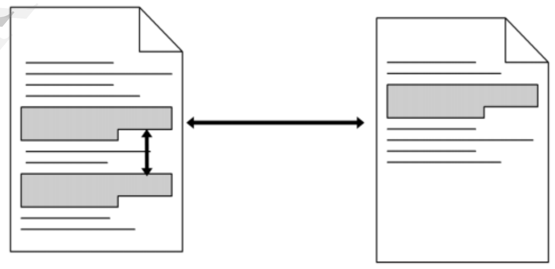
Figure 1: Example of Cloning

Cloning code [4] fragments that are similar known to hamper productivity of software maintenance in code-based development environments. Obviously, the same problems also occur for duplicated parts of models in model-based development. Moreover, the identification of duplicated model elements can help to reduce program size which is beneficial in a domain where limiting the required hardware resources is still a central objective.

2. RELATED WORK

Baker S. et al. [1] in their paper entitled "A Program for Identifying Duplicated Code" described a program called dup that found occurrences of duplicated or related code in large software systems. The motivation is that duplication may be introduced into a large system as modifications are made to add new features.

Boehm B. W. et al. [2] in their paper entitled "A Spiral Model of Software Development and Enhancement" presented one candidate for improving the software process model situation. The major distinguished feature of the spiral model is that it creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process.

D'souza, Desmond F., and Alan Cameron Wills et al. [3] in their paper entitled "Objects, components, and frameworks with UML: the catalysis approach" described that software systems must met those business needs, work properly, be effectively developed by teams, and be flexible to change.

Deissenboeck, Florian, Benjamin Hummel, Elmar Juergens, Michael Pfaehler, and Bernhard Schaetz, et al.[4] in their paper entitled "Model clone detection in practice" detailed on the challenges and solutions to the most pressing ones, namely scalability and relevance of the results. Moreover, the tool supported that eases the evaluation of detection results and thereby helps to make clone detection a standard technique in model based quality assurance.To automatically identify duplicates in graphical models.

Harold, Elliote Rusty, et al. [5] in their paper entitled"Processing XML with Java" XML was enthusiastically adopted by programmers who needed a robust, extensible, standard format for data. For the most part, this was not narrative data like stories and articles, but record oriented data such as that found in databases.

J. Johnson, et al. [6] in their paper entitled" Identifying redundancy in source code using fingerprints" metric fingerprints was built on the idea that you can characterise a code fragment using a set of numbers. These numbers were measured by identify the functional structure of fragment and sometimes the layout.

J. Mayrand, C. Leblanc and E. Merlo, et al. [7] in their paper entitled "Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics," presented a technique to automatically identify duplicate and near duplicate functions in a large software system.

3. PRESENT WORK

3.1 Clone Detection Procedure

In order to achieve task, select Reference Model and Candidate model from UML 2.0 class models which extract internal structure, in the form of XML. After that, parse that file to store the internal structure of XML and form internal XML data structure. A parser is a software library that knows how to read XML documents and handle the entire mark up it finds [5]. Then, compare the parse and stored value with Heuristics which means selection criteria matching of clones. Finally, the result is obtained which means the clone is detected.

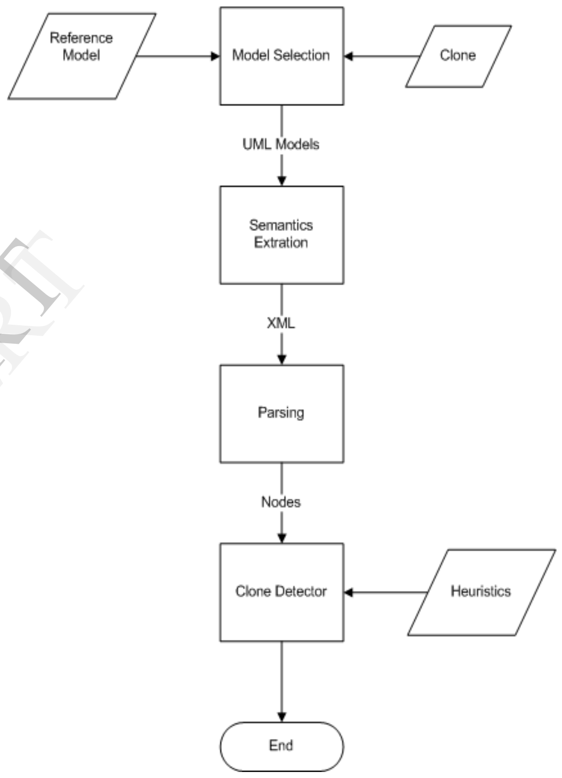

Figure 3.1: Flow Chart of the Clone Detection Process

3.2 Algorithm

Model clone detector algorithm reports its detection results as a list of qualified names of the affected model elements. To inspect a clone group and assess its relevance and opportunities for consolidation, a developer needs to inspect all involved clone instances and understand their commonalities and differences [4].

```
Rank: Initialize ranking Data Structure for Clone Detection.
Ref = Import Reference Model as XML
Candidate = Import Candidate Model as XML
If (Ref = NULL)
    Error = XML Parsing Failed for Reference Model
    Exit
If (Candidate = NULL)
    Error = XML Parsing Failed for Candidate Model
    Exit
For each node in Reference does
Reference_node = parse_node()
If (node matches selection criteria)
    Update ranking data structure
Else
  Continue parsing
End
If (Rank! = NULL)
Prob: Candidate Clone Probabilities
If (Prob>Expected)
Report: Clone Detected
Report: Probability for each Clone Node in Reference Model
Else
Report: No Clone Detected.
```

Figure 3.2: Model Clone Detector Algorithm

3.3 Example of Reference and Candidate Models

To create example for the clone detector using a Visual paradigm UML modeling tool and parse the resultant XML file with the help of Microsoft Visual C# on Visual Studio using MSXML interface.

3.3.1 Detecting Clones in Models (Modeled Hospitals)

According to the definition of cloning [17], there can be different notions of similarity. They can be based on text, lexical or syntactic structure as shown in Figure1.1 or can be semantics, model based Figure 3.3 and 3.4, or functionally. They can even be similar if they follow the same pattern, that is, the same building plan.

This phenomenon occurs similarly in models, suggesting that model clones are as detrimental to model quality as they are to code quality. However, programming language code and visual models have significant differences that make it difficult to directly transfer notions and algorithms developed in the code clone arena to model clones [16]. The structural clone analysis extends the benefits of analysis based on simple clones in the areas of program understanding, maintenance, reuse, and refactoring [4].

Taking Reference Model i.e. Civil Hospital that is a simple abstraction of the real word hospital. In Reference Model, there is one Package Diagram i.e. Civil Hospital which includes five classes. These classes have attributes, operations and show the relationships with each other.

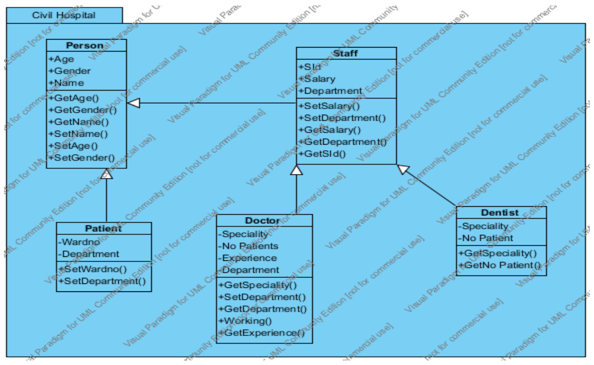

Figure 3.3: Reference Model (Civil Hospital)

Taking Candidate Model i.e. Hospital. This model includes same classes as reference model. Mostly both models have same attributes and operations of all the classes but the difference is that this candidate model does not include the class Dentist. As the class dentist is not include in the Candidate Model so there is no cloning between Person with Dentist, Staff with Dentist, Doctor with Dentist and so on. The comparing Reference Model with Candidate Model means that the Classes of both model is to compare and also the attributes and operations of each and every class is to be compare. For more clones detected, more attributes and operations are required.

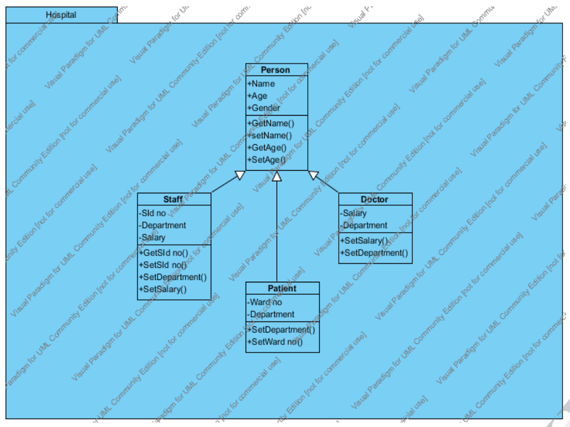

Figure 3.4: Candidate Model (Hospital)

3.3.2 Final Results and Foreseeable Cloning between Hospital Models

The final result is that cloning occurs between the Reference and Candidate Model. Model clone detection deals with the identification of duplicated parts in models [4]. This process is done through XML parsing. The parser takes the responsibility for checking documents, well-formedness and validity. The code reads XML document only through the parser's API. Taking information in the form that is convenient to use without worrying excessively about low level serialization details [5].As the results shown in table, Cloning occurs in Person with Person, Staff with Staff, Patient with Patient and Patient with Doctor but there is no Cloning between Doctor with Doctor and so on. This Cloning is obtained with the help of score value. The present technique for detecting clones work well for model clones.

Table 3.1: Final Results and Foreseeable Cloning Between Hospital Models

Reference Model	Candidate Model	Cloning (%)
Person	Person	95%
Staff	Staff	75%
Patient	Patient	70%
Patient	Doctor	65%

3.3.2.1 Percentage Cloning between Hospital Models

This graph shows between both the Hospital Models. In this graph, X-axis shows the relation between the reference and candidate model and Y-axis shows the percentage for cloning. The graph shows only the cloning value that is obtained by the score value. The result of cloning obtained only when the score value is greater than 60%. Some score values are less than 60%, those values in graph not shown.

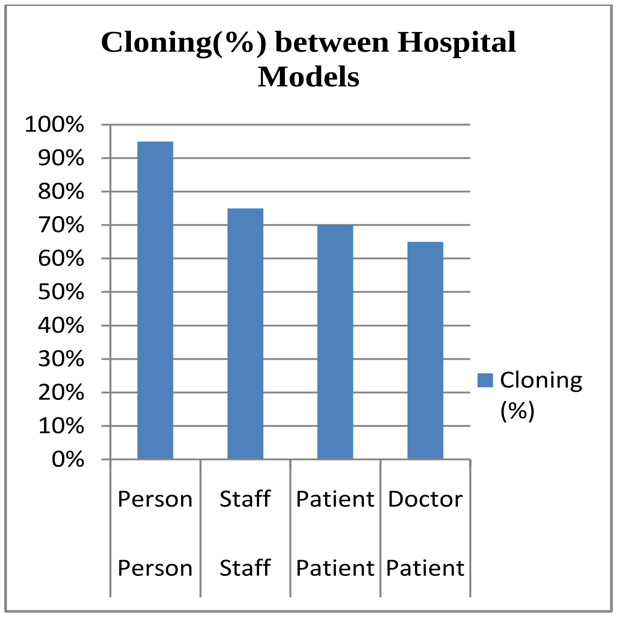

Figure 3.5: Percentage Cloning between Hospital Models

As shown in graph, cloning between reference and candidate model means that the Classes of both model is to compare and also the attributes and operations of each and every class is to be compare. Hence formed a cloning between Person with Person is 95%, Staff with Staff is 75%, Patient with Patient is 70%, Patient with Doctor is 65% and so on. For cloning, more attributes and operations are required.

4. CONCLUSION AND FUTURE WORK

Clone detection techniques play an important role in software evolution research where attributes of the same code entity are observed over multiple versions. To successfully create any method or technique [3] for model clones detection, study all the models defined in UML including internal and external structure of UML. This paper presents the technique available for the model clone prevention and detection. While this approach has demonstrates the practical relevance of clone-detection in model-based development [4].The semantic modelling elements are used for code generation, validity checking, and complexity metrics and so on [8].

As a result, present technique for detecting clones work very well for model clones and able to detect software clones for the same. Some Heuristics, such as 60% constraints are still required to begin with clone detection but the system produces adequate results for any number of models. The future work is to automated generation of models from code to detect model clones, and also work on interaction and activity clone detection in UML Domain Models.

ACKNOWLEDGEMENTS

The author thanks to Asst. Prof. Raninder Kaur who is currently working in Guru Nanak Dev Engineering College, Ludhiana (Punjab) and also Asst. Prof. Pankaj Goel who is currently working in RIMT-IET, Mandi Gobindgarh (Punjab) for their fruitful suggestions on this work. Thanks also to friends for their valuable suggestions to improve this paper.

REFERENCES

[1] Baker S., “A Program for Identifying Duplicated Code”, Computing Science and Statistics, vol. 24, pp. 49-57, 1992.

[2] Boehm B. W., “A Spiral Model of Software Development and Enhancement”, Software Engineering Notes, vol. 11, no. 4, 1994.

[3] D'souza, Desmond F., and Alan Cameron Wills. Objects, “Components, and frameworks with UML: the catalysis approach” Vol. 1 Reading: Addison-Wesley, 1998.

[4]Deissenboeck, Florian, Benjamin Hummel, Elmar Juergens, Michael Pfaehler, and Bernhard Schaetz, "Model clone detection in practice "In Proceedings of the 4th International Workshop on Software Clones, pp. 57-64. ACM, 2010.

[5]Harold, Elliote Rusty, “Processing XML with Java” Addison-Wesley Longman Publishing Co., Inc., 2002.

[6]J.Johnson,”Identifying redundancy in source code using fingerprints”In Cascon,1993.

[7] J. Mayrand, C. Leblanc and E. Merlo, “Experiment on the Automatic Detection of Function Clones in a Software System Using Metrics,” in Proceedings of the 12th 45 International Conference on Software Maintenance (ICSM’96), pp. 244–253, November 1996.

[8]James Rumbaugh, Ivar Jacobson, Grady Booch.The Unified Modeling Language Reference Manual. Addison-Wesley, 1998.

[9] Lakhotia A., Li J., Walenstein A. and Yang Y, “Towards a Clone Detection Benchmark Suite and Results Archive”, in Proceedings of the 11th IEEE International Workshop on Program Comprehension.

[10]OMG Unified Modeling Language (UML) Superstructure, V2.1.2, p.154.

[11]Object Management Group UML specification 1.4, September 2001

[12]P. Jablonski, D.Hou, “Aiding Software Maintenance with Copy-and-Paste Clone-Awareness” in the Proceedings of IEEE 18th International Conference on Program Comprehension, 2010.

[13]Paradigm, Visual, "UML CASE tool for software development - Visual Paradigm" Hong Kong: Visual Paradigm International. Available at: http://www.visual-paradigm.com/product/vpuml/. Accessed April (2013): 2013.

[14]Rieger,Matthias,"Effective clone detection without language barriers" Inaugural dissertation der Philosophisch-naturwissenschaftlichen Fakultat der Universitat Bern 10 (2005).

[15]Roy C.K., Cordy, J.R. Scenario-Based Comparison of Clone Detection Techniques, In:,2008. ICPC 2008. In: The 16th IEEE International Conference on Program Comprehension,June 2008

[16] Roy, Chanchal K., James R. Cordy, and Rainer Koschke,"Comparison and evaluation of code clone detection techniques and tools: A qualitative approach." Science of Computer Programming 74, no. 7 (2009): 470-495.

[17] Storrle, Harald, "Towards clone detection in UML domain models" In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, pp. 285-293. ACM, 2010"

[18]Sparks, Geoffrey. "Database Modeling in UML" Retrieved 8 September 2011.

[19]Scott W. Ambler UML 2 Class Diagrams. Webdoc 2003-2009. Accessed Dec2,2009.

[20]Satish Mishra (1997) "Visual Modeling & Unified Modeling Language (UML): Introduction to UML" Rational Software Corporation. Accessed 9November 2008.

[21] Tiarks R., Koschke R. and Falke R, "An Assessment of Type-3 Clones as Detected by State-of-the-Art Tools", in Proceedings of the 2009 Ninth IEEE International Working Conference on Source Code Analysis and Manipulation, pp. 67-76, 2009.

[22]Tairas R.,"Representation, Analysis and Refactoring Techniques to Support Code Clone Maintenance", PhD Thesis, University of Alabama, 2010.

[23] UML Superstructure Specification Version 2.2. OMG, February 2009.

[24]W3C Extensible Markup Language (XML) 1.0 (2nd edition) http://www.w3.org/xml, 2000.

[25]William Harrison, Charles Barton, Mukund Raghavachari."Mapping UML Designs to Java" The 15th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications-OOPSLA'2000, October 15-19 2000, Minneapolis, Minnesota, United States. ACM SIGPLAN Notices, 35(10): 178-187. ACM Press, New York,NY, USA.