Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 6 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

A Review of Clone Detection Techniques using Model Semantics

By Yachna Arora & Er. Sarita Choudhary

Abstract - A model clone is a set of similar or identical fragments in a model of the system. Understanding and Identifying model clones are important aspects in software evolution. During the Evolution of the Software product Cloning is often a strategic means for the same. 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 for model clones detection we will have to study all the models defined in UML including internal and External Structure of UML This paper reviews some of the techniques available for the Model Clone Prevention and Detection.

Keywords : *UML, semantic clones, model clone, clone detection, parsing, XML.*

GJCST-C Classification *: D.2.9*

Strictly as per the compliance and regulations of:

A Review of Clone Detection Techniques using Model Semantics

Yachna Arora [α] & Er. Sarita Choudhary [σ]

Abstract - A model clone is a set of similar or identical fragments in a model of the system. Understanding and Identifying model clones are important aspects in software evolution. During the Evolution of the Software product Cloning is often a strategic means for the same.

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 for model clones detection we will have to study all the models defined in UML including internal and External Structure of UML This paper reviews some of the techniques available for the Model Clone Prevention and Detection.

Keywords : *UML, semantic clones, model clone, clone detection, parsing, XML.*

I. Introduction

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering. The Unified Modeling Language includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems.

The Unified Modeling Language was developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in the 1990s.[1] Unified Modeling Language is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system underdevelopment.[2]

UML combines techniques from data modeling, business modeling, object modeling and component modeling. It can be used with all processes, throughout the software development life cycle and across different implementation technologies.[3]

a) Definition of a Clone

Software clones are regions of source code which are highly similar; these regions of similarity are called clones, clone classes, or clone pairs. . Cloning is the unnecessary duplication of data whether it is at design level or at coding level.

b) Clone Types

i. *Code Clones*

(i.e., duplicate fragments of source code) have been identified as a major source of software quality issues. As a consequence, a large body of research has been developed on how to prevent, or spot and eliminate code clones. The problem with code clones is of course that they are linked only by their similarity, i.e., implicitly rather than explicitly which makes it difficult to detect them.

ii. *Model Clones*

Model Clones are duplicate fragments of Architecture of Model of the project. It is difficult to formulate the actual definition of a model clone because of the abstract nature of a model. We can however define a model clone as a set of similar or identical fragments in a model.

c) Clone Detection

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.

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.

Cloning is the unnecessary duplication of data whether it is at design level or at coding level.

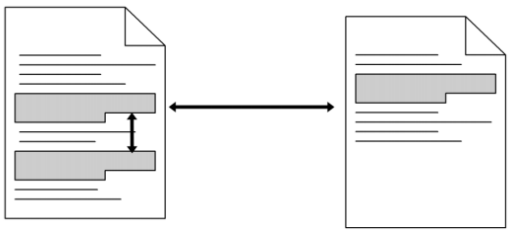

Figure 1 : An Example of Code Clones

It results to excessive maintenance costs as well. So cut paste programming form of software reuse deceivingly raise the number of lines of code without expected reduction in maintenance costs associated with other forms of reuse.

The reasons why programmers duplicate codes include the following reasons:

1. Making a copy of a code fragment is simpler and faster than writing the code from scratch. In addition, the fragment may already be tested so the introduction of a bug seems less likely.

Author α σ : Dept. CSE, DIET.

2. Evaluating the performance of a programmer by the amount of code he or she produces gives a natural incentive for copying code.

Efficiency considerations may make the cost of a procedure call or method invocation seems too high a price. In industrial software development contexts, time pressure together with first and second points lead to plenty of opportunities for code duplication

II. Related Work

Störrle Harald[4], describes that, Code clones, have been identified as major source of software quality issues. Evidence suggests that 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. In this article, we develop and propose a definition of the notion of "model clone" based on the thorough analysis of practical scenarios.

The Author proposes a formal definition of model clones, specify a clone detection algorithm for UML domain models, and implement it prototypically. The problem with code clones is of course that they are linked only by their similarity, i.e., implicitly rather than explicitly which makes it difficult to detect them.

The paper also discusses that the clones are a substantial problem for code based development, and model clones are increasingly becoming a problem for model-based development. However, currently, there is not much published work on model clones, and next to no work on UML model clones. Therefore, this article started out analyzing actual model clones in UML domain models, and proposed a terminological framework, a pragmatic definition, and a clone classification schema adapted from work on source code clones.

Liliane Jeanne Barbour[5],describe that, Two identical or similar code fragments form a clone pair. Previous studies have identified cloning as a risky practice. A clone pair experiences many changes during the creation and maintenance of software systems. A change can either maintain or remove the similarity between clones in a clone pair.

If a change maintains the similarity between clones, the clone pair is left in a consistent state. However, if a change makes the clones no longer similar, the clone pair is left in an inconsistent state. The set of states and changes experienced by clone pairs over time form an evolution history known as a clone genealogy.

Specifically, two cases are most risky:

1. When a clone experiences inconsistent changes and then a re-synchronizing change without any modification to the other clone in a clone pair; and
2. When two clones undergo an inconsistent modification followed by a re-synchronizing change that modifies both the clones in a clone pair.

Cloning has been identified by previous researchers as a risky practice in software development and maintenance.

However, software projects have limited resources for reviewing and testing code. Identifying the clones most at risk of faults can help allocate the limited Resources.

Florian Deissenboeck, Benjamin Hummel Elmar Juergens, Michael Pfaehler, Bernhard Schaetz[6], describe that, Cloned code is considered harmful for two reasons:

1. Multiple, possibly unnecessary, duplicates of code increase maintenance costs.
2. Inconsistent changes to cloned code can create faults and, hence, lead to incorrect program behavior.

Likewise, duplicated parts of models are problematic in model-based development. Recently, we and other authors proposed multiple approaches to automatically identify duplicates in graphical models.

While it has been demonstrated that these approaches work in principal, a number of challenges remain for application in industrial practice. Moreover, we present tool support that eases the evaluation of detection results and thereby helps to make clone detection a standard technique in model based quality assurance.

In many application domains for embedded software systems, model-based development-the specification of the functionality of the software using (graphical) models and the automatic generation of production code from these models-is a state-of-the-art technique.

In this paper, techniques have been presented to improve scalability by an adapted subsystem detection, to improve relevance of detected by clones by providing use case specific rankings, and finally tool-support to ease inspection of the instances of the detected clones.

Finally, the detection approach can also be extended to other, less data flow oriented forms of models, e.g., state-oriented models like State Charts or State Flow, or process-oriented models like BPEL or ARIS, however, requiring adapted definitions of similarity as well as means of normalizing models and ranking clones.

Arun Lakhotia, Junwei Li, Andrew Walenstein and Yun Yang[7],describe that, Source code clones are copies or near-copies of other portions of code, often created by copying and pasting portion of source code. This working session is concerned with building a communal research infrastructure for clone detection. The intention of this working session is to try to build a

consensus on how to continue to build a benchmark suite and results archive for clone- and source comparison related research and development. Several automated and semi-automated clone detection techniques have been devised. Clone detection is done as an information retrieval (IR) task. The standard measure for IR techniques can be applied in the form of the detector's precision and recall. In the case of clone detection, "recall" refers the percentage of the clones that are found, and "precision" refers to the percentage of correct results as compared to "false positives" (code falsely reported to be clones). Different clone detectors report clones in different formats.

III. Conclusion

Cloning works at the cost of increasing lines of code without adding to overall productivity. Same software bugs and defects are replicated that reoccurs throughout the software at its evolving as well its maintenance phase.

Software clones are important aspects in software evolution. If a system is to be evolved, its clones should be known in order to make consistent changes. Cloning is often a strategic means for evolution.

Clone detection techniques play an important role in software evolution research where attributes of the same code entity are observed over multiple versions.

IV. Future Scope

In UML Literature There exists a plenty information regarding Model clones. How Code clones can be identified and removed. However, a development of a system capable of detecting a complete Model is still in research domain.

As we know UML Models have two parallel structures; an external, visual representation as diagrams; and an internal, tree-like structure. The Tree like structure is usually presented in XML like modeling language, because of the free nature of XML format.

In future I would like to achieve a basis for comparisons of UML Domain Models. The process will be done using XML parsing of UML domain Models, with Referenced and Candidate Models.

References Références Referencias

1. Marc Hamilton, Software Development: A Guide to Building Reliable Systems, p.48, 1999.
2. FOLDOC (2001).Unified Modeling Language last updated 2002 01-03. Accessed 6 February, 2009.
3. Satish Mishra (1997). "Visual Modeling & Unified Modeling Language (UML): Introduction to UML". Rational Software Corporation. Accessed 9 November 2008.
4. Störrle Harald, "Towards clone detection in UML domain models", Springer-verlag, Softw Syst Model. 18 September 2011.
5. Liliane Jeanne Barbour, "EMPIRICAL STUDIES OF CODE CLONE GENEALOGIES", Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada. January, 2012.
6. Florian Deissenboeck, Benjamin Hummel Elmar Juergens, Michael Pfaehler, Bernhard Schaetz, "Model Clone Detection in Practice", Technische Universität München Garching b. München, Germany, fortiss GmbH München, Germany.
7. Lakhotia Arun, Li Junwei, Walenstein Andrew and Yang Yun, "Towards a Clone Detection Benchmark Suite and Results Archive", Software Research Laboratory Center for Advanced Computer Science University of Louisiana at Lafayette, IEEE, 2003.

Global Journals Inc. (US) Guidelines Handbook 2013

www.GlobalJournals.org