International Journal of Engineering Research and General Science Volume 3, Issue 3, May-June, 2015
ISSN 2091-2730

Detecting model Clones using C K Metrics Suite

Aastha Saini[1], Vivek Sharma[2]
M.Tech Student[1], HOD[2]
Computer Science & Engineering Department
JMIT, Radaur / Kurukshetra University, India
[1]aasthasaini99@gmail.com
[2] vivek@jmit.ac.in
9416814057

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. 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. This paper is about detecting Clones utilizing the Model architectures using C K Metric suite.

Keywords — UML Domain Models, Model Clones, Clone Detection, Object oriented metrics, CK metrics, Metric based clone detection, Artificial Neural Network.

1.INTRODUCTION

The exercise of reusing of program by programmers employing easy duplicating and gluing is common in the software. Such exercise leads to program clones. Cloning in the software introduces countless problems. If the duplicated program encompasses bug the alike bug propagates in the supplementary duplicated servings of the software. Program cloning aftermath in increased program lengths and additionally makes the maintenance a challenging task.
In the progress of the software, models frolic a momentous role. The use of conceptual models is recurrent in the software progress phases. In fact the past experiences alongside colossal scale models counsel that the occurrence of clones arises in models in a quite comparable manner as in basis code [10]. Therefore recognizing ideal clones plays a vital role.

A. Code clones

Software clones are spans of basis program that are exceedingly similar [1], [2]. If two basis program fragments are comparable to every single supplementary next they are denoted to as 'clone pair' in that one fragment is oftentimes a duplicate of supplementary alongside or lacking tiny adjustments completed alongside the code. Modification to one program fragment could depart the supplementary cloned fragment unchanged therefore depreciating the quality of program and making the maintenance of program a tough task.
The reasons behind code cloning involve:
1) To reuse existing code by copy and paste.
2) The risk involved in the development of new code.
3) Writing reusable code is error prone.

```
If (c >= b)                    If (c>= b)
{                              {
 a = d + b; // Comment1          //comment1
  d= d + 1;                       a= d + b;
                                   d = d + 1;
}                              } Else
Else                           {
{                                //comment2
 a = d -c; //Comment2             a=d –c;
}                               }
```

Figure 1.1: A simple code clone

B. Model Clones

Model clone can be described as a set of comparable or identical fragments in a model. Clones in an ideal emerge in a comparable manner as program clones and so are disapproving to the quality of software.

As an Example, we can illustrate a model clone in Figure 1.2. We can clearly see the similarity between the reference model and its possible model clone. The classes and attributes of the reference model are identical to the classes and the attributes of the model clone candidate. For example the address class attributes in the reference and cloned candidate respectively have identical 'name', 'street', 'city', 'country' attributes.

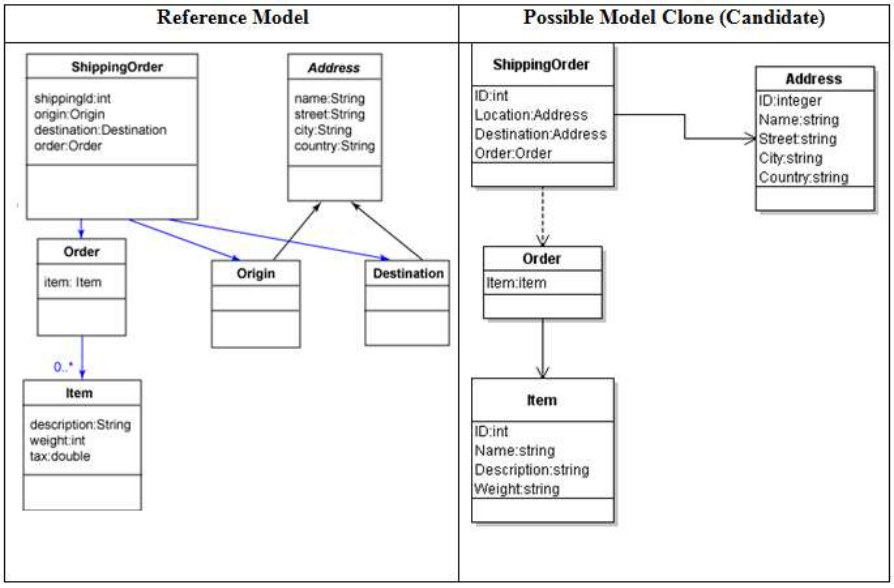


Figure 1.2: Depiction of Model Clones in UML Diagrams

Four major challenges regarding model clones are:

1. Understand the structure of real clones and derive a practical definition of model clones.
2. Quantitatively analyze the structure of medium to large scale models and develop method to detect clones.
3. Derive a formal framework for model clones and develop an algorithm to detect clones in models of realistic size and structure.
4. Implement the algorithm and method, balancing precision and recall against acceptable run time.

C. Object oriented metrics

To accomplish quality in the software procedures and products the object oriented metrics are utilized that are measured employing the basis code. The OO arranging consists of objects that are the generalization for the real objects and interact alongside every single supplementary to process the data [6], [7], [8].

The software metrics for OO paradigm compute disparate software characteristics like cohesion, quality of software, intricacy etc. Intricacy and cohesion are believed the most vital characteristics of all.

Cohesion can be described as the degree to that the methods and qualities of class are associated together. A class with an elevated worth of cohesion way that its methods and qualities are exceedingly connected alongside every single supplementary and is tough to divide. Cohesion discovers inadequate sketches of classes. The number of methods in a class provides alongside the compute of intricacy of a class.

D.CK Metrics Suite

Today, the works provides a collection of metrics to compute the intricacy of software. Amongst them, we can remark one of the early suites of OO design compute was the Chidamber & Kemerer (CK) metrics suite counseled by Chidamber & Kemerer in 1994 [5], [9]. This suit of metrics can be functional to recognize arranging errors, the disparate software quality parameters such as maintenance price, reusability etc and additionally in discovering the intricacy of the system.

Chidamber and Kemerer have counseled a suite of six object oriented design metrics:

WMC - Weighted Methods per Class – indicates the sum of complexity of all the classes. It determines the complexity of a class. A class with higher WMC is believed extra complicated than a class alongside a lower WMC value. Consider a class C, having methods M1,..., Mn described in the class and c1,...,cn to be the complexity of methods M1….Mn. Next:

$$\sum_{i=1}^{n} c_i$$

DIT - Depth of Inheritance Tree -The DIT is the maximum length from the node to the basis of the tree. A class that is deeper in the pecking order inherits extra methods becoming extra convoluted for predicting its behavior. The design intricacy increases alongside the length of the tree.

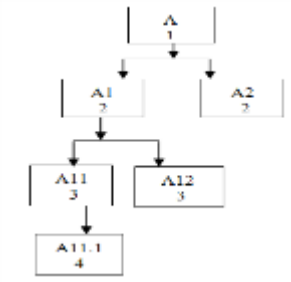


Figure 1.3: Inheritance tree with DIT=4

NOC - Number of Children-The NOC metric is the number of instant subclasses subordinated to a class in the class hierarchy. The larger the number of children, larger is the level of reuse and more testing effort is required.

CBO - Coupling Between Object Classes -The CBO metric for a given class is a count of the number of supplementary classes to that it is coupled. Interpretation: two classes are coupled afterward methods uttered in one class use methods or instance variables delineated by supplementary class. Higher coupling amid the objects of the classes prevents reuse and additionally needs an precise assessing effort. In order to advance encapsulation and enhance modularity, class coupling have to retain to the minimum.

RFC - Response for a Class -The reply for a class is given by the number of methods that can be potentially gave in reply of a memo consented by an object of that class.

The larger the number of methods that can be implored by a class, the larger is the intricacy of the class. RFC gives a compute of the power needed for uphold a class in words of assessing time.

LCOM - Lack of Cohesion in Methods Definition-The LCOM metric provides alongside the degree of similarity in methods. The larger the number of comparable methods, supplementary cohesive is the class.

High LOCM way extra methods are coupled alongside every single supplementary managing to convoluted classes. LOCM ought to have a low value.

2.RELATED WORK

Florian Deissenboeck [3] proposed techniques improving scalability and relevance of results. The detection results are evaluated using tools. The work concentrates on challenges which occur while dealing with model based clone detection in industrial perspective and their solutions. The existing detection algorithms are compared and it has been shown that scalability is most important factor.

Harald Störrle [10] work provides the definition of model clones through a systematic study. A clone detection algorithm is proposed for UML domain models. The different heuristic used in the algorithm is investigated and the performance is calculated. The work was restricted to UML models.

Yoshiki Higo [5] in this scrutiny work a mechanism is provided to calculate different source code metrics. With this work the need to use different tools for measuring different metrics is overcome using a software tool MASU. Using MASU makes it easier to develop plug-in for CK metrics suite.

Unlike the previous related work, this paper formulate an algorithm that can identify similarity in aggregated CK metric using the similarity structure and Predict and visualize similar fragments of CK metric's of aggregated components.

3. PROPOSED WORK

A. Artificial Neural Network

It is a computational system stimulated by the composition, processing method and learning skill of biological brain.

It is composed of a colossal number of exceedingly interconnected processing agents (neurons) working in together to resolve specific problems.ANN learns by example.

- Basic neural network architecture consists of a huge number of processing neuron like processing elements. Similar to the human brain neurons, the neurons in the neural network transports the incoming information on their outgoing connections to the other neurons.
- The constituents are related by unidirectional contact channels 'connections'.

- The network gains knowledge from a learning process.

B. Learning

In ANN, to present a specific task in a proficient manner it is needed to notify the web design and the associated weights across a discovering process. The web normally learns the related weights from obtainable training data. The striking feature of ANN is the skill of ANN to discover automatically from the obtainable data and examples. Instead of pursuing a law set endowed by human expert, ANNs discover the laws from the obtainable set of examples.

Different web design need disparate discovering algorithm.

Supervised Learning: In Supervised learning the arrangement is given an output for every single input sample. The arrangement next predicts the outputs to the recognized examples as close as probable to the recognized outcomes and learns from its mistakes.

Unsupervised Learning: In unsupervised learning no initial output result is provided with the input samples. The system itself explores the pattern in which the data is organized or the correlations existing between the data patterns and also organizes the patterns into the categories.

C. Software Design Pattern Analysis

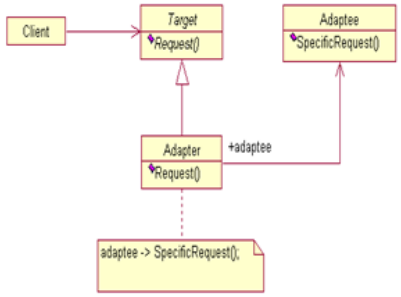
1) Calculating assorted matrices like WMC (Weighted Methods each Class), DIT (Depth of Inheritance Tree), NOC (No. of Children), RFC (Response For Class), CBO (Coupling Amid Objects) are computed for every single class of design outlines.

2) Summation of matrices for every single class is completed for every single design pattern. Additionally one more matrix NC (No. of classes utilized in every single pattern) is computed at run period.

Table 3.1: Design Pattern with Metric Values.

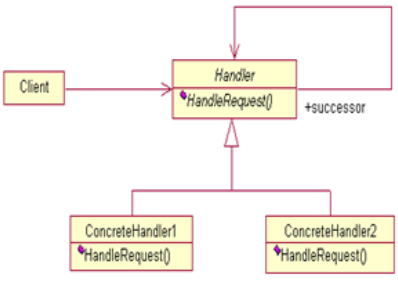
Design Pattern | **Metric Values**

Adapter



Class/Matrix	NOM	DIT	NOC	CBO	RFC
Target	1	0	1	0	1
Adapter	1	1	0	0	2
Adaptee	1	0	1	0	1

Chain of Responsibility



Class/ Matrix	NOM	DIT	NOC	CBO	RFC
Handler	1	0	2	0	2
Concrete Handler 1	1	1	0	0	1
Concrete Handler 2	1	1	0	0	1

D. Self-Organizing Feature Maps

The Self-Organizing Feature Chart (SOFM) is competent way for the visualization of high-dimensional data. In its open form it produces similarity graph of input. It converts the nonlinear statistical connections among high-dimensional data into simplistic geometric connections of their picture points on a low- dimensional display, normally a usual two-dimensional grid of nodes. The data is compressed by SOFM and also maintains the most vital topological and metric connections of the main data agent on the display. It may also generate some abstractions. Visualization and abstraction can be utilized in a number of methods in convoluted tasks such as domination, link, speech trust, vector quantization, adaptive equalization and combinational optimization.

E. Clone Detection Work Flow

Complexity of the inner composition of the component is helpful in giving an approximation of effort related to progress of component. In our ongoing scrutiny, our main focus is on achieving the quality of internal design of component and its connection to the external quality characteristics of the component. The complete work is classified in following phases.

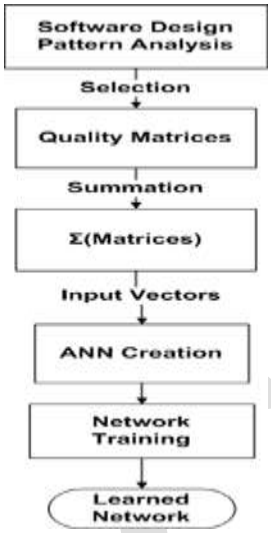


Figure 3.1: System Flow chart for SOM training for Detecting Clones

- **Analysis Phase**: Calculating various matrices like WMC (Weighted Methods per Class), DIT (Depth of Inheritance Tree), NOC (No. of Children), RFC (Response for Class), CBO (Coupling Between Objects) are computed for every class of design patterns.
- **Summation** of matrices for each class is done for every design pattern. Also another matrix NC (No. of classes used in each pattern) is calculated at run time.
- **Network Creation** An unsupervised neural Network is created for this Classification Problem.

4.ALGORITHM FOR DETECTING CLONES USING SOFMS

The training of SOFM C. K matrices as feature maps is as follows

1. SOM Initialization: Choose random values for the initial weights Wj (0).
2. Set size of neurons in to SOM ***MxN***
3. input matrices selected from the input space is denoted by

$\mathbf{X} = [x_1, x_2\ldots, x_m]$

Here m is the dimension of the input data space. Weight vector of any neuron j can be denoted as

$\mathbf{W_j} = [w_{j1}, w_{j2},\ldots, w_{jm}]$, j = 1, 2 …… N

Where N is the total neurons in SOM layers

4. $\boldsymbol{I_{mxn}}$ is a matrix of Patterns of Code Matrices, derived from Software Components
   where M is the number of rows and n is no of columns in ***I.***

The main task of SOM based clone detector is to find out the cluster centers for each input x from input space X,

5. Do, for each C.K matrices from software component with n no. of features add row ***r*** in ***I***

Find the neuron for each ***i*** such that the C.K matrices weight vector is closest to input vector, i.e
then i(x) may be determined as follows:

$\boldsymbol{i}(\mathbf{x}) = \arg\min \|X - W_j\|$, j = 1,2,…,L

6. Update the weights w_i , i = 1…m
   $w_i(t+1)= w_i(t)* n(t)*X(I,i*(t))$, *n* is the SOM Learning Rate
7. until, Change Sum of weights is greater than threshold $\sqrt{\sum_{i=1}^{m}|\Delta w_{ij}|^2} > \in$

$\in$ is maximum threshold

8. wt = calculate the weighted sum the Neurons in SOM layers
9. Sort weights wt in descending order

10. Plot sorted weighted sums and exited neuron structure.

5.RESULTS AND ANALYSIS

The steps to enhance the performance are followed as stated earlier. Training data in experiment is 21 x 6 i.e. a total of 126 elements. The experiment results for training of neural web in the Matlab are shown in table 5.1. In the table 5.1 Training method that is used, No. of training data, No. of epoch seized to encounter, No. of output data, time period taken for execution are shown.

Table 5.1: The experimental results using neural network analysis

Experiment	**Experiment**
Training method used	trainbuwb
No. of training data	6 x 21 = 126
No. of epoch taken to converge	2000
Time taken to execute	7.14773 seconds
No. of outputs	6

For training aim all 21 design outlines and 6 metrics are taken.2000 epochs are taken for attaining accuracy. Quantity of epochs grabbed is 2000 to finish elevated accuracy.

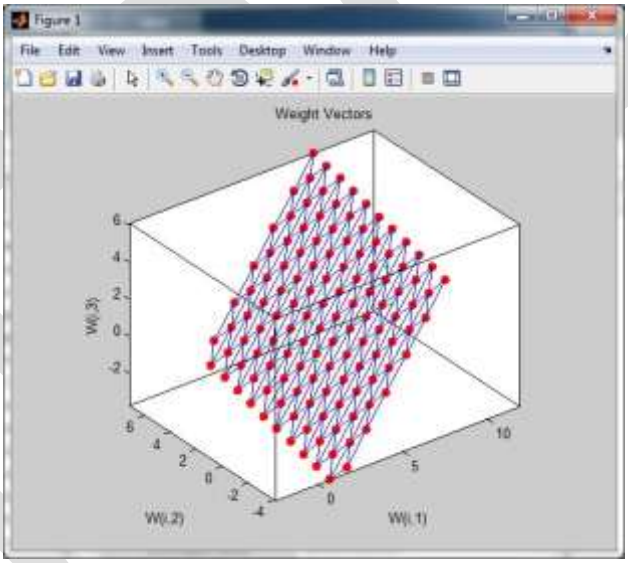


Figure 5.1: SOM weight positions before training

According to figure 5.1 it is aligned diagonally. As every single the aftermath shown above, the model proposes for the design chart presentation can be enhanced by owning weighted method every single class as 6, depth of inheritance tree as 6, answer for class as 9, number of children as 6, number of classes as 6 and coupling amid objects as 4. Total time elapsed in entire execution: 7.14773 seconds.

Adding up the metrics values for every single constituent, number of classes is additionally computed at run era and concatenated alongside the final matrices value. This in finished seized as input for the neural network. Subsequently neural web is utilized to train the self coordinating map (SOM) neural web and highest performance can be attained. By retaining the example of design outlines and unsupervised neural web, a model giving an enhanced performance for the component quality is proposed.

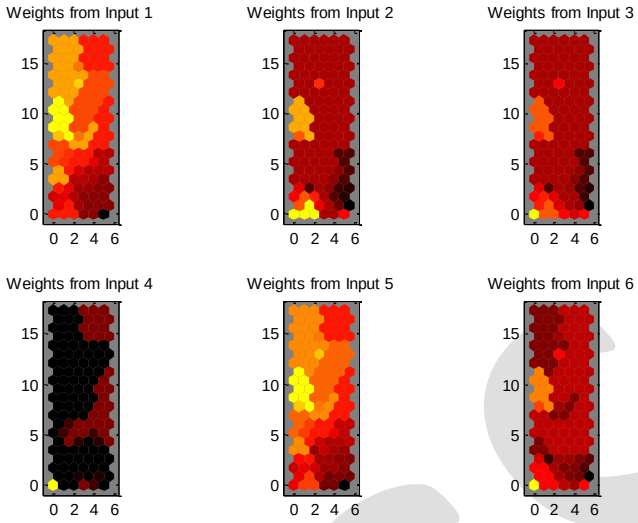


Figure: 5.2 SOM weights for all inputs after Training phase.

The results shown using MatLab through execution of Neural Network are:
Weighted Method per Class (WMC): 6
Depth of Inheritance Tree (DIT): 6
Response For Class (RFC): 9
Number Of Children (NOC): 6
Number of Classes (NC): 6
Coupling Between Objects (CBO): 4
Total time elapsed in entire execution: 7.14773 seconds.

6.CONCLUSION AND FUTURE SCOPE

In the scrutiny work, we counseled a quantitative Clone Detection model alongside respect to the Constituent Instituted Progress (CBD) methodology retaining SOFMs. We used C. K. metrics to find clones varied kinds of design outlines (components). While adding the value of metrics for every component, number of class is additionally computed at run time and concatenated alongside the final matrices value. This becomes input for the neural network. Subsequent neural web is utilized to train the self coordinating chart (SOM) neural web and in that case, maximum presentation can be achieved. By retaining the example of design outlines and unsupervised neural web, we have proposed a model that provides improved performance of software model. The results can be more enhanced if realistic data is obtainable to as training set and also if neural network techniques with fuzzy techniques are considered. Also, if output might be understood from the past benefits, a supervised neural web might give larger result. Full advantage of component-based clone clustering approach will be achieved when not only the C. K matrices, but also hybrid approaches with SOMs and iterative methods. This approach may lead to easier and more accurate predictability of the Clone Detection and visualization.

REFRENCES:

[1]. Miryung Kim, Vibha Sazawal, David Notkin, and Gail Murphy. "An empirical study of code clone genealogies." In ACM SIGSOFT Software Engineering Notes, vol. 30, no. 5, pp. 187-196. ACM, 2005.
[2]. Zhen Ming Jiang and Ahmed E. Hassan. "A framework for studying clones in large software systems." In Source Code Analysis and Manipulation, 2007. SCAM 2007. Seventh IEEE International Working Conference on, pp. 203-212. IEEE, 2007.
[3]. Florian Deissenboeck, 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.
[4]. Vinaya Sawant and Ketan Shah. "Automatic Generation of Test Cases from UML Models." In International Conference on Technology Systems and Management (ICTSM), pp. 7-10. 2011.
[5]. Yoshiki Higo, Akira Saitoh, Goro Yamada, Tatsuya Miyake, Shinji Kusumoto, and Katsuro Inoue. "A pluggable tool for measuring software metrics from source code." In Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA), pp. 3-12. IEEE, 2011.
[6]. Sanjay Misra, Ibrahim Akman, and Murat Koyuncu. "An inheritance complexity metric for object-oriented code: A cognitive approach." Sadhana 36, no. 3 (2011): 317-337.

[7]. Sanjay Misra, "Evaluation criteria for object-oriented metrics." Acta Polytechnica Hungarica 8, no. 5 (2011): 110-136.
[8]. Deepak Arora, Pooja Khanna, Alpika Tripathi, Shipra Sharma, and Sanchika Shukla. "Software quality estimation through object oriented design metrics." Int. J. Computer Science and Network Security 11, no. 4 (2011): 100-104.
[9]. Brij Mohan Goel, and Pradeep Kumar Bhatia. "Analysis of Reusability of Object-Oriented System using CK Metrics." Analysis 60, no. 10 (2012).
[10]. Harald Störrle, "Towards clone detection in UML domain models." Software & Systems Modeling 12, no. 2 (2013): 307-329.
[11]. Roy, Chanchal K., Minhaz F. Zibran, and Rainer Koschke. "The vision of software clone management: Past, present, and future (keynote paper)." In Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014, Software Evolution Week-IEEE Conference on, pp. 18-33.IEEE, 2014.
[12]. Chanchal K. Roy, 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