Using Squarified Treemaps for Structural Clones' Visualization

Umber Nisar1, Hamid Abdul Basit2

[1]Department of Computer Science Forman Christian College University, Lahore, Pakistan

[2]Department of Computer Science Lahore University of Management Sciences, Lahore, Pakistan

Abstract

Novel programming languages propose numerous abstraction methods to promote reuse of code fragments but this often leads to several duplicated code fragments so called clones. Clones recur in a particular fashion in software to constitute structural clones. This paper presents a squarified treemaps visual technique to effectively represent higher level structural clones to the programmers to facilitate further maintenance or reengineering activities based on these clones.

Keywords— Code Clones, Structural Clones, Squarified Treemap, Higher Level Clone.

I. Introduction

Software clones, replicated code blocks, duplicated code fragments___ all such terms are used to refer code clones. According to Ira Baxter "Clones are the segments of code that are similar according to some definition of similarity" [1]. The emergence of clones is the result of copying and pasting the code once written to reuse for some other task within a larger software systems [2]. The dark side of reusing the same code fragment is that it adds a lot to the maintenance cost of the system, it is one of the reasons of introducing bugs by reusing a buggy code and its presence also leads to poorly designed system.

When the code that is being copied forms some recurring pattern within a system then such clones become part of advanced clones' type called structural clones [3]. Structural clones give bigger picture of similarity situation than simple clones alone [3]. Figure 2 shows an example of structural clones referenced from the paper on Detecting Higher-Level Clones in Software [3].

Consider a1, a2 and a3 are code blocks and are copy of each other and together they form a group of simple clones. Similarly b1, b2, b3 form another group and this continues till the group of g's comprising of g1, g2 and g3. When the configurations of simple clones recur in files X1, X2 and X3, then we call the group of such configurations a file level structural clone set. Similarly, the file level structural collaboratively may further form still higher level structural clones.

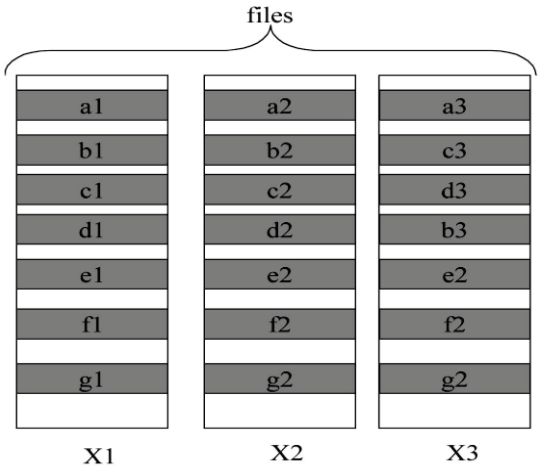

Figure 1 Structural clone example

In order to decrease software maintenance cost, complexity and bug propagation the software should be made free of the clones [4]. Clone detection has several advantages e.g. it improve software quality, detects library candidates, increases program understanding, finds patterns, detects plagiarism and helps in making the code compact. The papers [6], [3] explain well the advantages of detecting simple and structural clone respectively.

To achieve the advantages, a proper clone detection mechanism is required. Several software clone detection techniques [5] have been developed which help developers in code refactoring. The best way however is to make them visualize at one glance. A paper on survey of clone visualizations [7] explains well the clone visualization techniques. All the techniques represent the detected clones in their own ways but there are also associated limitations of each one. Some are not suitable for larger data, some cannot well represent the structural clones, some are not appropriate for hierarchical data and some cannot simply well represent all the data nodes simultaneously for larger data.

According to the paper [8], the rooted tree is one of the most well-known tree representations for hierarchical information like directory structures. It says that the tree is based on node-link visualization in which the relationship between parent and child nodes is depicted with line connections.

The paper [9] gives sound reasoning of the requirement to move from simple tree to treemaps. It says that the tree views are very effective for small data but they fall short

in case of larger data which is to be viewed simultaneously. And the main reason of this limitation is the inefficient use of display space as the background covers most of the pixels. To cope with this problem treemaps [10] were introduced by Ben Shneiderman. The full display space is used efficiently. According to the research in this paper, the simple treemap is the result of recursive subdivision of original rectangle. The resulting sub-rectangles' sizes depict the corresponding node sizes. At each level, the direction of subdivision varies, may be first there is a horizontal subdivision then vertical subdivision etc. The figure 2 below is an example of simple treemap.

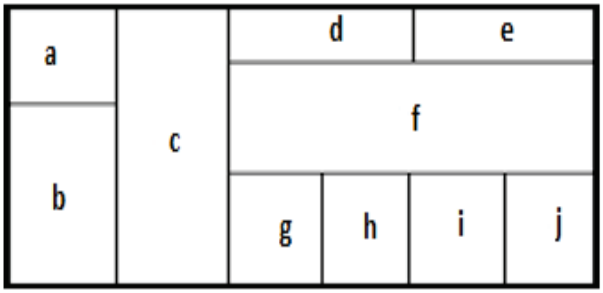

Figure 2 Treemap

The paper on Squarified treemaps [9] explains that the subdivision of original rectangle in case of simple treemaps results in a problem of producing thin and lengthened rectangles. The nodes at one level are treated similar so the appearance of small file is degraded relative to its bigger sibling nodes. To solve the above problem advanced form of treemaps emerged called “Squarified treemaps. The squares in Squarified treemaps have low aspect ratio, help in using the display space still more efficiently, improves the accuracy of representation and no doubt the square nodes are easier to detect and point out. An example of squarified treemap with steps involved in squarification can be seen in a well known squarified treemaps paper [9]. The aim of this work is to effectively visualize the structural clones. The methodology used to attain the aim is by using an advanced form of treemaps called Squarified Treemaps.

II. Structural Clone Visualization Results

Keeping in view the vast advantages of representing hundreds of hierarchical nodes simultaneously, we chose Squarified treemaps to represent the structural clones effectively. The input data to our software is the output produced by a clone detector called Clone Miner. The paper [11] describes well the clone miner. It says that it does not only finds simple clones but also their repeating configurations in files or even in directories using data mining techniques. The paper describes that first it detects Simple Clones Classes (SCC) from a tokenized representation of the source code. And then by using frequent item set mining technique finds repeated configurations of simple clones across method and files to form first level of cloning abstractions. At the end the Clone Miner uses clustering techniques to recognize next level of cloning abstractions. The paper further describes the types of structural clones that can be found using clone miner, which are:

- Repeat configurations of simple clones
  - o in the same method (SCS_In_Method)
  - o across different methods (SCS_Acoss_Method)
- Repeated configurations of simple clones
  - o in the same file (SCS_In_File)
  - o across different files (SCS_Across_File)
- Method clone classes (MCC)
- File clone classes (FCC)
- Repeated configurations of method clones
  - o in the same file (MCS_In_File)
  - o across different files (MCS_Across_File)
- Repeated configurations of file clones
  - o in the same directory (FCS_in-Dir)
  - o across different directories (FCS_Across_Dir)

The explanations for simple, method and file clone classes and their higher level configurations are well explained in paper on structural clones.

Our structural clone visualization software is implemented in php (Hypertext Preprocessor), server scripting language, to make it dynamic and interactive. The architecture used is MVC (Model View Controller) and framework is Codeigniter.

The software takes the output table produced by clone miner as input data to work on. After that it converts the data into a json (JavaScript Object Notation) array. Then it creates treemap using JavaScript library d3. It also uses Html (HyperText Markup Language), SVG (Scalable Vector Graphics) and CSS (Cascading Style Sheets) to bring the data to life.

Each structural clone is represented with different levels which are System View With Directory, System View with Directories and Files, Directory View with Files, File view with Methods. The user may filter results by selecting the structural clones of his choice and also can define the threshold values for Average Percentage Coverage (APC) and Average Token Coverage (ATC) and the software outputs the structural clones fulfilling the selected criteria.

A. System View with Directories

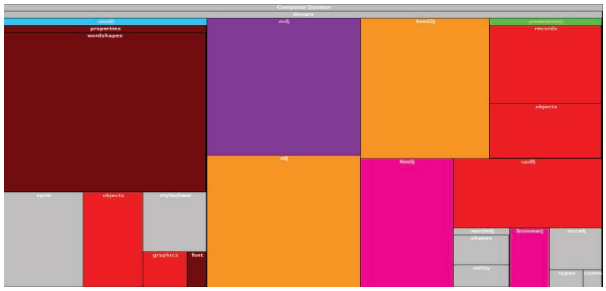

Figure 3 System View with Directories

The figure 3 shows the squarified treemap having the whole system as its base. The base rectangle is then subdivided to show directories within the system. Each rectangle within the system depicts a directory. One can also see the hierarchical view of different directories. The size of parent directory is the sum of the sizes of its child directories. Each rectangle is labeled with the name of the directory. The colors are used to show the instances of the structural clones. In this figure three instances of the structural clone are depicted with different colors. Also if one rectangle involves the intersection of more than one instance then a separate color is used to help the viewer understand the intersection at one glance. The instance of first selected scs instances are shown with maroon, second with pink, third with purple, the intersection of first and second with red, the intersection of second and third with orange, the intersection of third and first with green and the intersection of all three with blue color.

B. System View with Directories and files

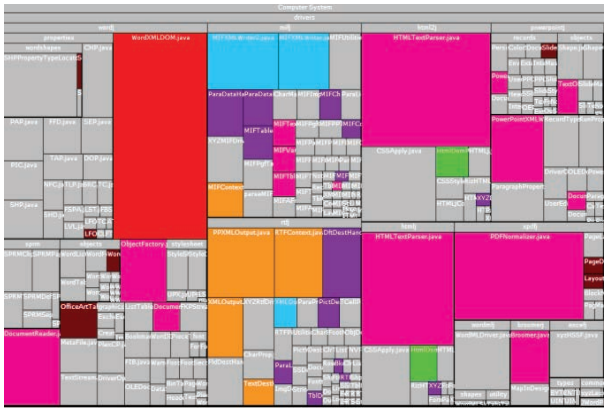

Figure 4 System View with Directories and Files

The figure 4 is view involving computer system at its base but includes directories and files also within the directories. So this figure comprises three hierarchical levels, the system having the directories and directories having files inside. Now one can clearly see that the same space used in figure (1a) is now being used to represent a lot more information. This is because the squarified treemaps are formed as a result of recursive subdivision of the rectangles. Now the rectangles here in this figure, represent the system, the directories and files. Each directory size represents the space it consumes in that system. One can easily grab the idea about the size of the files compared to others by the amount of space it is consuming of the whole view. All the files are shown within the directory rectangle of which they are part of. Here again the three instances are shown with the instance of first selected scs instances shown with maroon, second with pink, third with purple, the intersection of first and second with red, the intersection of second and third with orange, the intersection of third and first with green and the intersection of all three with blue color.

C. Directory View with files

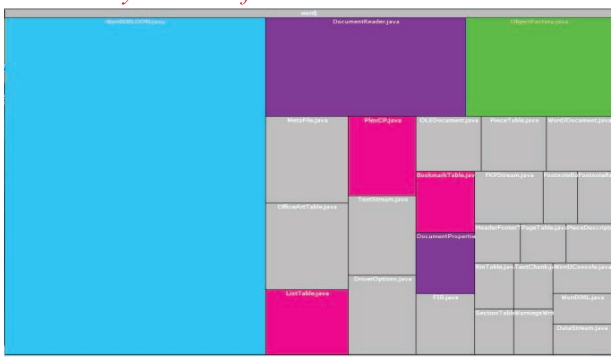

Figure 5 Directory View with Files

It is not necessary to have the whole system as the base of this squarified treemap. One can have a clearer view of any directory by having it at the base of the treemap as shown in figure 5. Now that the directory is the base one can view the files inside it more evidently. The size of each file is the inkling of the space it utilizes within a directory. Again at this level we have shown the first selected scs with maroon color, second with pink, third with purple, and the intersection of first and second with red, the intersection of second and third with orange, the intersection of third and first with green and the intersection of all three with blue color.

D. File View with methods

Figure 6 File View with Methods

Now the figure 6 is file view with methods. If one wants to view all the methods within a file then it can also be seen with the file, one is interested to view as the base and all the methods inside the file. It is amazing that one can get the idea of size of method and structural clones inside within the squarified treemap. The rectangles sizes within the file view are the clue of the space each individual method is consuming within the file. The structural clones' instances and their intersection can clearly be seen with different color shades. The ones used in the figure are like that of first selected scs shown in maroon color, second with pink, third with purple, and the intersection of first and second with red, the intersection of second and third with orange, the intersection of third and first with green and the intersection of all three with blue color.

III. Discussion

Below is the performance analysis of software. The response time of the software depends on the number of nodes in the hierarchy. The software provides well organized representation of structural clones for hundreds of nodes. In case the number of nodes is very big and the area of very small file is degraded as compared to larger nodes in hierarchy, the hierarchical zooming feature helps in getting an improved version of that view. The user can make the parent of the very small file as the root node and then can easily get approximate view of the space consumed by its child node. Similarly he can unset that root node by just simple clicks. The visualization of structural clones with squarified treemaps utilizes the display space very efficiently. It provides a compact view of all hierarchy nodes. The area taken by node is directly proportional to its size. It no doubt provides good approximation of space consumption by a particular directory/file with all its file/method nodes. At one glance, the user gets an idea that which of my directory/file contains structural clone and then can perform maintenance tasks etc. Treemaps as compared to the different visualization techniques like rooted tree view, wheel view, dot plot etc. does not waste the display space in the background and the view also does not go beyond a page. All the view is confined within certain area with each and every pixel representing useful information. In comparison with other treemapping algorithms, the squarified treemaps also provide better view.

IV. Conclusion

The visualization software is no doubt a compact view for hierarchical information like directory, file structures. It provides an efficient and well-organized view for hundreds of nodes. The different types of structural clones are well represented at multiple levels of the hierarchy. To provide better depiction of structural clones various color schemes have been incorporated. The different types of filtering mechanisms also assist user in getting the useful information. Multiple enhancements like zooming, textual tooltips, highlighting etc. have been added to give improved display of the structural clones. It is also made interactive using different animation phenomenon. It helps user to detect structural clones with less time and effort. The user can thus improve software quality, increase software understanding, detect plagiarism, find patterns, detect library candidates, analyze changes impacts and perform refactoring. The structural clone visualization software provides simple, interactive and efficient display of structural clones for hierarchical data.

References

[1] Tom Mens, Serge Demeyer, Software Evolution, 2008.

[2] Ira D.Baxter, Andrew Yahin, Leonardo Moura, Marcelo Sant' Anna, Lorraine Bier, "Clone Detection Using Abstract Syntax Trees", 1998.

[3] Hamid Abdul Basit, Stan Jarzabek, "A Data Mining approach for Detecting Higher-Level Clones in Software", 2009.

[4] Cory J. Kasper, "Toward an Understanding of Software Code Cloning as a Development Practice", 2009.

[5] Rainer Koschke, "Survey of Research on Software Clones", 2007.

[6] Chanchal Kumar Roy, James R. Cordy, "A Survey on Software Clone Detection Research", 2007.

[7] Muhammad Hammad, Hamid Abdul Basit, Stan Jarzabek, "Evaluating Clone Visualization Techniques from User Goals' Perspective", 2012.

[8] Danny Holten, "Hierarchical Edge Bundles:Visualization of Adjacency Relations in Hierarchical Data", 2006.

[9] Mark Bruls, Kees Huizing, Jarke J. vanWijk, "Squarified Treemaps".

[10] B. Johnson and B. Shneiderman. "Treemaps: a space-filling approach to the visualization of hierarchical information structures".

[11] Yali Zhang, Hamid Abdul Basit, Stan Jarzabek, Dang Anh, and Melvin Low, "Query-based Filtering and Graphical View Generation for Clone Analysis", 2008.

[12] Hamid Abdul Basit, Usman Ali, Sidra Haque, 2012, "Things Structural Clones Tell that Simple Clones Don't", 2012.