International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-5, Issue-9)

Research Article | **September 2016**

Using Fuzzy C Means Clustering to Support Program Comprehension

Vishakha Yadav
Department of Computer Science & Engineering, Jaypee Institute of Information Technology,
Noida, Uttar Pradesh, India

Abstract—

This paper presents a methodology that focuses on software understanding using the program files. C code contains .c and .h files. To find the relation among them, there is a need to investigate the semantics of source code. Clustering technique can be used to support program comprehension and to find the software architecture. Fuzzy C means clustering is used here for program comprehension. It extracts useful attributes (Function names included header files) from the C source code files and store into a temporary file with relevant names. Match these temporary files and find the dissimilarity among them. The proposed approach makes use of these dissimilarities as distance to fuzzy clusters. Cluster heads are selected from the files using cluster head threshold value. Membership of the source code file to cluster head shows the degree to which source code file is relevant to cluster head and included in that cluster. List of related files of particular file represent with the increasing order of dissimilarity.

Keywords—semantic dependency; structural dependency; attribute mining; threshold; cluster head; header files; membership function.

I. INTRODUCTION

Process of software development involves maintenance phase which play vital role because it takes approximately more than half time of the software development cycle. Maintenance occurs after the delivery of software system to the end user. In maintenance phase, there is need to do modification in the software system either to add new functionality in the system or fixing the defects of the system. Software development contains different activities where many programmers work on these activities. The task of maintenance becomes more challenging as code is written by many programmers over a period of time. The novice programmers have no idea about the architecture of the software system and how the program entities are related to each other.

As the proper documentation of the software systems are not available the difficulties will arise to understand the whole code. Program Comprehension is the area that concern about extracting the knowledge from the source code and help the novice developer at the time of maintenance. Clustering is the process that gathers the similar objects in one place and dissimilar objects in distinguish place. Clustering process can be used in the area of program comprehension. Software clustering is the process that helps novice programmer by clustering the similar entities of the source code. After clustering the source code novice, developer get the information about software system architecture and find how one file of source code is related to other files.

Many researcher have contributed in the area of program comprehension. Programs have the structural dependency (flow of the source code entities) and semantic dependency (comments, identifiers, function name and variables etc.).These dependencies help to find the similarity among the source code entities for the program comprehension. Major research in the area of program omprehension is done for the software systems written in object oriented programming because objected oriented programs can easily be comprehended. Object oriented software system are enriched with program entities(class, package,object) and there relationship(inheritence, sub class of class and implements imported function etc.) Procedural software system have less program entities then object oriented language. Even today, we are dependent on many programs that have been written 10 to 15 years back and most of them are written in procedural language like C.

This work contributes:

i. Data mining on the C source code that extract only useful attributes that help to find similarities among source code files.
ii. Finding the dissimilarities among files that use as distance among the files.
iii. Give more priority to the files by making them cluster head which have less distance to other files.
iv. Applied fuzzy c means clustering so, that one source file can reside more than one cluster if it show strong relationship to more than one cluster.

The rest of paper is organized as follows: related work is described in section 2, Section 3 illustrates the proposed method, experimental setup is shown in section 4, Section 5 describes the results, conclusion is provided in section 6n and finally future work is presented in section 7.

II. RELATED WORK

In [1], Dimitris Rousidis provide technique which applies on the java source code. Java code contains classes, packages, function and parameters. Extract the essential property of these entities. These properties consider as attribute. Using these attributes data mining engine create table for each entity. Attribute value can be nominal or numerical. All the nominal values converted into numerical value. Distance between two records calculates using these attributes values. Apply the hierarchical agglomerative clustering algorithm to cluster similar elements using distance function. One case study also provided on fragment of the application Keep in touch (KIT) and result show the 86.67% precession of the given software clustering technique.

In [2], Yiannis Kanellopoulos proposed framework applicable on C++ source code. Data mining apply on the source code and the class information and method information extracted. Attribute can be binary and qualitative. Four main entities selected that are - classes, member function, parameters and member data. All the entities information considered on the basis of level. Levels from high to low are- Preprocess class information, Preprocess member Data information, Preprocess member functions, and Preprocess functions parameters information. Information refines from one level to another and uses to cluster the entities. Three case study provided to result evaluation that on- CAccess Report, CompDB and FlightGear Flight Simulator.

In [3], Vassilios Tzerpos design an algorithm to support clustering large software system. Source files are organized in the form of graph. Sub graphs contain the denominator nodes according to the pattern driven approach. First step of ACDC algorithm is Skeleton construction. Skeleton of the source files created that contain sub system. These sub system represent as a cluster itself. Some sub system with very small cardinality discarded. Next step is Orphan adoption; Orphans are the resources that are not clustered in previous step. Placed Orphans in most appropriate subsystems, so that whole system files clustered. Algorithm validation given on two large systems namely: TOBEY and Linux. Algorithm takes 54 seconds to cluster 939 source files of the TOBEY and 84 seconds to cluster 955 source files of Linux system. TOBEY's 604 files out of 939 files (64.3%) and Linux's 488 files out of 955 (55.7%) are considered in skeleton construction rest are added into orphan adoption step.

In [4], Yu Liu proposed an approach for program summarization. Organization of the packages evaluated using latent semantic indexing (LSI) and Hierarchical clustering. Approach contains three steps: preprocessing information in the program, clustering the retrieving topics and recovering semantic information to generate summaries. In preprocessing inputs are class names, identifier names, method names and comments. Splitting all inputs into small terms. Matrix between document and terms prepare. Weighing each term on basis of occurrence of term into the document. Clustering technique use the cosine similarity between the documents. Apply hierarchical clustering based on similarity to find out the structure of the software system. Now, summary of the package using previous two steps. For each package some cluster documents are present. Each document focuses on some topic and term. Summaries of package are given by the topics and terms.

In [5], Ural Erdemir,Umut Tekin an object oriented software clustering provided. Author use community network phenomena to cluster object oriented software system. Design graph model from source code according to the program entities and their relationship. If the graph contains many sub graphs or disconnected components. Select largest sub graph as input to the clustering algorithm. If the clustering algorithm works for undirected graph. Then convert the directed graph into undirected graph using the symmetry. Here is five set of algorithm apply on the graph. Set of clustering algorithm work on graph and used in this approach are - Fast Community, Bunch, ACDC, LIMBO and K-Means. Result of clustering given to the clustering analysis. Clustering result is given on the basis of clustering result from the previous version and expert decomposition. Cluster analysis provides three types of results: authoritativeness result, stability result and NED result. This approach applies on the four medium sized java software systems that are: Weka, JUnit, E-Quality and JFreeChart. Authoritativeness of Fast community algorithm is best from all other four clustering algorithms. Stability also checks between the versions of the software system. Stability of fast community algorithm is much better than others. Execution time of the Fast community algorithm is 418 ms which is far better than other algorithm execution times.

III. PROPOSED METHOD

The method proposed here use the C source code as input. C source code contains both header and C files. This method has following steps:

i. Attribute Mining from the C code
ii. Design the dissimilarity matrix among the source code files.
iii. Cluster head selection.
iv. Design membership matrix between cluster heads and source code files.
v. Assigning of the source code file to the selected cluster head according to membership matrix.
vi. Reassignment of source code files to the clusters.

Attribute Mining

Similarity among the C source code files is mainly measured using the user defined function names and the included header files. C file includes both user defined header files and the standard header files. To measure the similarity among C file and user defined header files, we have extracted the names of included user defined file from the C file. Function name in the c code provide the flow of the source code. So, we have also extracted user defined function name from the C files and declared function name from the header files. This extracted data is stored in the temporary text files with relative names.

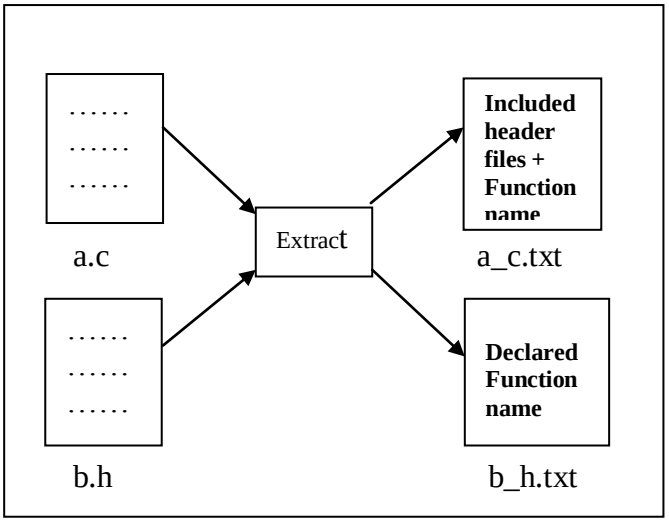

Fig 1: Attribute mining from .c and .h files

Design dissimilarity matrix

Create the N*N matrix where N is the no. of extracted files (both c and h files). Match the data of one file to the other file line by line. Dissimilarity between the two files suppose i and j is calculated as-

$$\boldsymbol{Dismatrix(i,j) = \frac{Loc(i,j) - matched(i,j)}{Loc(i,j)}}$$

$$\boldsymbol{0 \leq Dismatrix(i,j) \geq 1}$$

0 means fully matched.

1 means fully distinct.

Where,

Loc(i,j)= total no. of lines in i and j files matched(i,j)=no. of lines matched in files i and j.

Put this value into two place of dissimilarity matrix dismatrix(i,j) and dismatrix(j,i) because of symmetry. All the diagonal elements are 0 because file is fully matched to itself.

Cluster head selection

In the cluster head selection, the files having less dissimilarity with other files are selected to be the cluster heads. The threshold of dissimilarity is used for further selecting the cluster heads among them. As finding appropriate value for threshold is difficult, we have experimented with different threshold values. For each source code file take sum of dissimilarity to all other files. We have replicated the experimented with 10 different threshold values given by user side ranging between 0 to 1.If the dissimilarity sum of the file less than the average threshold value then it consider as cluster head. From the N source code files get the set cluster head files.

$$\boldsymbol{avg(i) = \frac{\sum_{j=1}^{N} dismatrix(i,j)}{N}}$$

if, avg(i)< Threshold(CH) then i th file is the cluster head

otherwise, i th file is not cluster head.

Design membership matrix

To decide membership of the element in different clusters. There is need to find the degree of the file with respect to different cluster heads. The membership function is described as-

$$\boldsymbol{mf(i,j) = \frac{1}{\sum_{k=1}^{M} \frac{dismatrix(i,j)}{dismatrix(i,k)}^{2/M-1}} \times 100}$$

Where, M= Number of cluster heads

N= Total number of files

A matrix of dimension M*N is created which shows the membership of the source file with the cluster head. Each cell indicates value of membership function which varies from 0 to 100%. Membership of the file which is also a cluster head to itself is 100%. Membership function decides that source code file should go to which set of clusters according to the membership threshold.

Assign source code files to clusters

Assigning of the source code to the cluster head based on the membership function value. If the value of the membership function of element to cluster head greater than the threshold then that element reside in that cluster. To decide the threshold value takes the median of all the membership function values and use median as threshold for deciding membership. According to the membership value and threshold value source code reside in one or more cluster.

$$\boldsymbol{Threshold(membership)} = \frac{\boldsymbol{M \times N}}{\boldsymbol{2}} + \frac{\boldsymbol{(M \times N) - 1}}{\boldsymbol{2}}$$

If $M \times N$ *is even*

Otherwise,

$$\boldsymbol{Threshold(membership)} = \frac{\boldsymbol{M \times N}}{\boldsymbol{2}}$$

If, $\boldsymbol{mf(i,j)} > Threshold(membership)$ then i th cluster head file contains j th file

Otherwise, i th cluster head file does not contain j th file.

Reassignment of source code files to the clusters

Calculate centroid of each cluster and make the centroid as new cluster head of the cluster. Repeat step iv to vi until the difference between old and new cluster head to particular termination threshold. Centroid of the cluster can be calculate as-

$$\boldsymbol{Cj} = \frac{\sum_{i=1}^{N} \boldsymbol{mf(i,j)^M \times dismatrix(i,j)}}{\sum_{i=1}^{N} \boldsymbol{mf(i,j)^M}}$$

Where M= No. of clusters

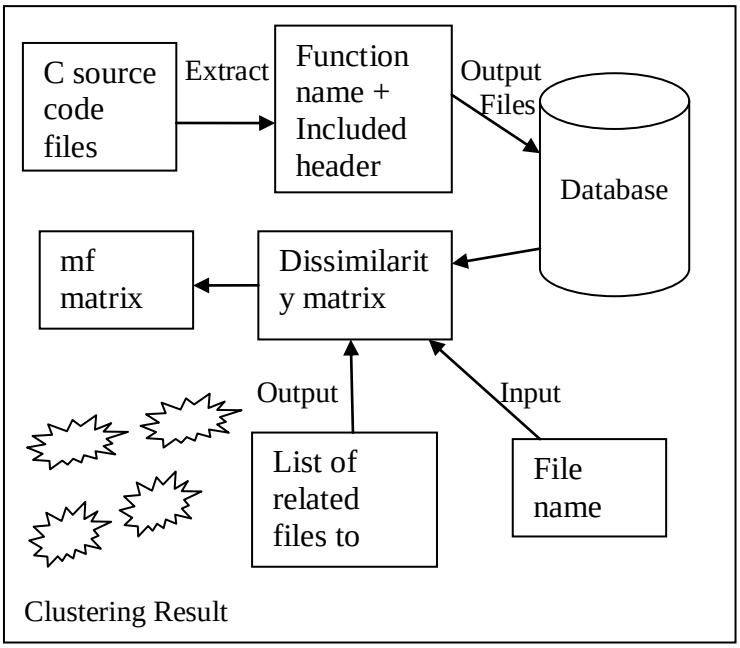

Figure 2: Overall architecture of the framework

Figure 2 shows the overall framework of our methodology with C source code files as input. C source code files are of two types: .c and .h files. Name of the functions and names of the included header file are extracted from the .c and .h files. The data is saved in the temporary text files. These text files have been given the same name as was present in the .c and .h extension files. Dissimilarity matrix is created using the matching of these temporary text files. Cluster heads are selected on the basis of dissimilarity and membership function is calculated between set of cluster heads and all the files. Assign the file to cluster heads according to the membership function value. To find the related file for a particular file, we can input the name of the file and it will provide the top ten related files with their dissimilarity value.

IV. EXPERIMENTAL SETUP

Our methodology is designed for the C source code. The experiment is conducted on the open source software system: Mosaic web browser source code. The mosaic dataset written in C language and it is big in size. This dataset is selected because it is difficult to understand the structure of this software system as no proper external documentation is present for this dataset. Coding standard used in this dataset is poor because it is made and modified by multiple programmers. Different programmers have their own style of programming it is difficult to understand the overall architecture of the software system.

Table 1: Description of Mosaic data set

File Type	No. of file
.c file	135
.h file	143
Total	278

Our approach use java apache io utils to fast extraction of useful features from the .c and .h files. Extracted information of the files saves in the output files using java output file writer. Then the java io reader read the output file line by line and matched to another file using the dismatrix formula mentioned above dissimilarity between the files calculated and save into the respective place in the dismatrix.

Cluster head selected using the average threshold of dissimilarity matrix. In the last, calculating the median of membership function matrix check the membership of all files with selected cluster heads if it greater than median then that file can reside in that cluster.

V. RESULT

Our methodology use the fuzzy c means clustering. If the file related to more than one cluster then it can reside in all those clusters. Our findings are as follows - Finds no. of cluster head selected on different input threshold value on the mosaic dataset. A detailed description of the clusters and selection of the suitable value of the cluster head selection threshold. We have manually looked into the code To validate the overlapping between two clusters. In the end validation of the top 3 related files of particular input file is shown.

Result for different cluster head threshold

Table 2: No. of cluster head form on Threshold value in between 0 to 1

Input Threshold for cluster head selection	Selected Cluster Head
0	0
0.5	0
0.9	0
0.95	4
0.96	22
0.97	62
0.98	94
0.99	116
1	273

As shown in table 2, with user defined threshold value from 0 to 0.94, 0 or no cluster heads are selected that means the entire source code file reside in single cluster. At the threshold value 0.95, four source code files are selected as cluster head. If user selects threshold value 1 then all files becomes cluster head that means each source code file makes its own cluster. The most suitable value of cluster head threshold is 0.96 that make 22 cluster heads. Result shows that as soon as increment in the threshold value no. of cluster heads also increase.

Table 3: Description of the cluster created and there elements for mosaic dataset

No. of files in cluster	No. of cluster
0-10	0
11-20	0
21-30	15
31-40	1
41-50	1
51-60	0
61-70	0
71-80	1
81-90	4
91-100	0
Total	22

Table 3 shows the structure of the clustering result on the mosaic dataset. First column contains number of element in the cluster and second column show number of cluster that contain element in the given range. As shown in the table 3 there is no cluster with 0 elements. Most of the clusters have the files in range of 21 to 30.There is no cluster with files more than the range of 51-60, 61-70 and 91-80. Largest cluster created with 98 files using our approach.

Validation for Overlapping between clusters

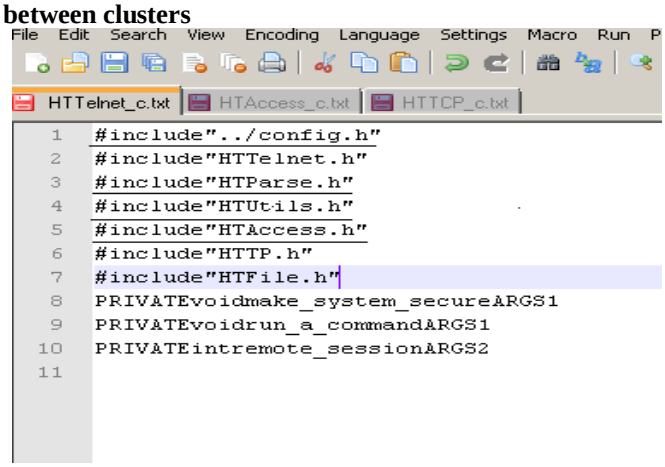

Figure 3: HTTelnet_c.txt common lines with respect to HTAccess_c.txt

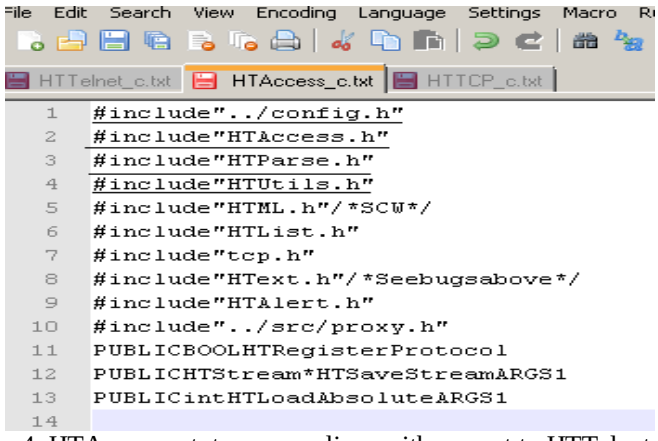

Figure 4: HTAccess_c.txt common lines with respect to HTTelnet_c.txt

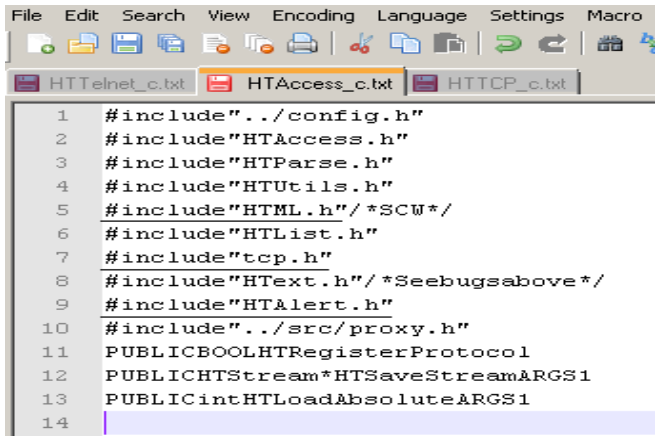

Figure 5: HTAccess_c.txt common lines with respect to HTTCP_c.txt

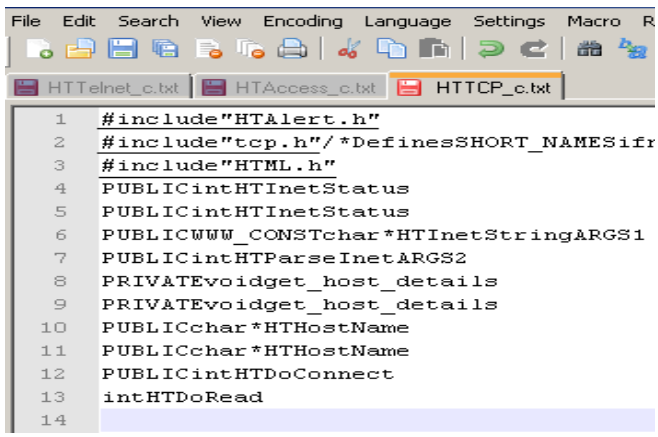

Figure 6: HTTCP_c.txt common lines with respect to HTAccess_c.txt

In above figure 3 and figure 4 shows the relationship between the files HTTCP_c.txt and HTTelnet_c.txt underline line represents matched lines between them. 4 lines are matched. In above figure 5 and figure 6 shows the relationship between the files HTAccess_c.txt and HTTelnet_c.txt underline line represents matched lines between them. In those files 3 lines are also matched. HTTCP_c.txt represents the cluster 17 and HTAccess_c.txt represents the cluster 7 HTTelnet_c.txt is common between both clusters.

Validation for top 3 related files of input file

Table 4: List of file related to the gui-dialogs_c.txt with their dissimilarities

File Name	Dissimilarity
gui-dialogs_c.txt	0.0
gui_c.txt	0.01
history_c.txt	0.08
gui-news_c.txt	0.09
gui-ftp_c.txt	0.16
mailto_c.txt	0.30
gui-extras_c.txt	0.49
whine_c.txt	0.51
audan_c.txt	0.66
gui-menubar_c.txt	0.67

Table 5: Common included header files and function name between the gui-dialogs and their top three related files

Name of file	Matched included header files	Matched function name
gui_c.txt	6	3
history_c.txt	5	2
gui-news_c.txt	5	1

As shown in the table 4 top related files of the gui-dialogs_c.txt. To validate the result of our technique manual validation has been done. We checked the source code files and as shown in table 5 find that 6 header file names and 3 function names are common between the gui-dialogs_c.txt and gui_c.txt. Like this 5 header file names and 2 function names common between history_c.txt and input files. 5 included header files and 1 function name common in gui-news_c.txt and gui-dialogs.txt.

VI. CONCLUSION

This work explore new dimension of the research where fuzzy clustering incorporated in the area of program comprehension. In this project, semantic information extracted from the C code. Relevance among the C code files calculated with the help of dissimilarity among them. Experimental result show that technique used here helps programmer to find the relationship among source code files of the software system. As fuzzy clustering applied so, overlapping of cluster elements (files) is also possible among the clusters. If the file related to more than one files that it can reside more than one cluster.

VII. FUTURE WORK

In this paper fuzzy clustering applied on the C source code it can also be applied on the source code written in any other language. More attribute can extract to refine the result of clustering. Textual data (source code) need to be normalized before extraction. Source code can be written with bad smells. Structural dependencies can also be included to find the relationship among the source code files.

REFERENCES

[1] Rousidis, D., & Tjortjis, C. (2005, March). Clustering data retrieved from Java source code to support software maintenance: A case study. In Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on (pp. 276-279). IEEE.

[2] Kanellopoulos, Y., & Tjortjis, C. (2004, June). Data mining source code to facilitate program comprehension: experiments on clustering data retrieved from C++ programs. In Program Comprehension, 2004. Proceedings. 12th IEEE International Workshop on (pp. 214-223). IEEE.

[3] Tzerpos, V., & Holt, R. C. (2000, November). ACDC: An algorithm for comprehension-driven clustering. In 2013 20th Working Conference on Reverse Engineering (WCRE) (pp. 258-258). IEEE Computer Society.

[4] Liu, Y., Sun, X., Liu, X., & Li, Y. (2014, June). Supporting program comprehension with program summarization. In Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on (pp. 363-368). IEEE..

[5] Erdemir, U., Tekin, U., & Buzluca, F. (2011, December). Object oriented software clustering based on community structure. In Software Engineering Conference (APSEC), 2011 18th Asia Pacific (pp. 315-321). IEEE.

[6] Misra, J., Annervaz, K. M., Kaulgud, V., Sengupta, S., & Titus, G. (2012, October). Software clustering: Unifying syntactic and semantic features. InReverse Engineering (WCRE), 2012 19th Working Conference on (pp.113-122).IEEE.

[7] Liu, X., Sun, X., Li, B., & Zhu, J. (2014, June). Pfn: A novel program feature network for program comprehension. In Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on (pp. 349-354). IEEE.

[8] Guerrouj, L. (2013, May). Normalizing source code vocabulary to support program comprehension and software quality. In Proceedings of the 2013 International Conference on Software Engineering (pp. 1385-1388).IEEEPress.

[9] Nguyen, H. A., Nguyen, T. T., Nguyen, H. V., & Nguyen, T. N. (2011, November). iDiff: Interaction based program differencing tool In Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on (pp.572-575).IEEE.

[10] O'brien, M. P. (2003). Software comprehension–a review & research direction.Department of Computer Science & Information Systems University of Limerick, Ireland, Technical Report. Sridhara, G., Hill, E., Pollock.

[11] L., & Vijay-Shanker, K. (2008, June). Identifying word relations in software: A comparative study of semantic similarity tools. InProgram Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on (pp.123-132).IEEE.

[12] Maletic, J. I., & Marcus, A. (2001, July). Supporting program comprehension using semantic and structural information. In Proceedings of the 23rd International Conference on Software Engineering (pp. 103-112). IEEE Computer Society.

[13] anne Storey, M. (2014). Theories, Methods and Tools in Program Comprehension: Past, Present and Future.

[14] M.H. Dunham, Data Mining, Introductory and advanced topics, Prentice Hall, 2002.

[15] U. Fayyad, G. Piatetsky-Shapiro, and R. Uthurusamy, " From Data Mining to Knowledge Discovery: An Overview" , in Advances In Knowledge Discovery and Data Mining, AAAI Press/The MIT Press, 1996.

[16] A. Takang, P. A. Grubb, and R. D. Macredie, "The effects of comments and identifier names on program comprehensibility: an experiential study," Journal of Program Languages, vol. 4, no. 3, pp. 143–167, 1996.

[17] C. Tjortjis, N. Gold, P.J. Layzell and K. Bennett, "From System Comprehension to Program Comprehensions" Proc. IEEE 26th Int'l Computer Software Applications Conf. (COMPSAC 02), IEEE Comp. Soc. Press, 2002, pp. 427-432.

[18] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo, "Recovering traceability links between code and documentation," IEEE Trans. on Software Engineering, vol. 28, pp. 970–983, Oct 2002.

