Software Visualization of Text Content in Ecosystem

Karthiga Mohanmani[1], Chamundeswari Arumugam [2],
P.G. Student, Professor (Computer Science and Engineering),
Department of Computer Science and Engineering,
SSN College of Engineering, Rajiv Gandhi Salai, Kalavakkam-603110,
Chennai, Tamilnadu, India.

Abstract
Software ecosystem analysis is on focus due to open source communities. It need to be analyzed at the high level through visualization techniques for the better understanding of the interactive relationship. Analysis of ecosystem has been usually represented using developer's data, project's data, based on the dependencies that exist. Dependencies at high level are explored, while the low level fine grained information is unexplored. Thus an attempt has been done in this paper, to visualize the low level information related to open source ecosystem GRAPPA. This ecosystem's source files are analyzed with the low level vocabularies such as non-keywords and represented as tag cloud. Tag cloud visualizes all the non-keywords such as class name, methods, and data in GRAPPA. This analysis exposes the project vocabularies with different weight based on the frequency count.

Keywords— software ecosystem, text visualization, tag cloud, tag operation.

I. INTRODUCTION

Software Visualization is the static or animated 2D or 3D visual representation of information about software systems based on their structure, history or behavior. It is defined as a discipline that makes use of various forms of imagery to provide insight and understanding and to reduce complexity of the existing software system under consideration [1]. Tools for the software visualization can be used to visualize source code and quality defects during software development and maintenance activities. It has been applied in various areas like algorithm animation, software engineering, concurrent program execution, static and dynamic visualizations of object-oriented code, fault diagnostics, debugging, and requirements analysis. A software ecosystem is a collection of software projects which are developed and co-evolved together in the same environment [2]. The need for tools and techniques to recover the source code traceability in an ecosystem is particularly important for a variety of software engineering tasks. Some examples of ecosystem are CinCom, SCG, REVEAL, SOOPS has 288, 210, 55, 249 projects respectively. The Small Project Observatory (SPO) is a tool

that is used to analyze the software ecosystem. The SPO [2] [3] tool visualizes the different type of views such as project timelines, project dependency map, project vocabulary map, and developer collaboration map. The first visualization tools SeeSoft [4] summarizes changes in the software at the level of lines of code. MOOSE is a tool that functions as a repository for software models, and it provides numerous services for importing, viewing, querying and manipulating these models [5].

The visual perspective of project vocabulary map [2] represented in SPO, show information only about the elements that are relevant at the high level abstraction of an ecosystem. However, there are cases in which visualization should support navigating at a lower level abstraction that enables to view the vocabulary perspective inside an ecosystem. The objective of the paper is to analyze the vocabularies in an ecosystem and visually represent it as a tag cloud. Non-keywords in the ecosystem's project are extracted and based on the frequency count, weight are assigned. Based on weight, the vocabularies are represented in vocabulary tag cloud. Thus the vocabulary tag cloud summaries all vocabularies or of an ecosystem.

The organization of the paper is as follows. Section 2 discuss with the related work. Section 3 discuss analyzes of GRAPPA software ecosystem. Sections 4 discuss tag cloud. Section 5 describes the tag operation. Section 6 describes conclusion and future work.

II.RELATED WORK

A software ecosystem can be defined as a collection of software projects that belong to an organization and are developed in parallel by the organization [6]. The organizations can be a company, a research group or an open source community. When analyzing software ecosystems, exploration and visualization are important because of the large amounts of information that can be made available about the ecosystem. The various goals[8] to analysis the ecosystem are to understand the past, control the present, shape the future, etc. Kawaguchi et al. [13] used Latent Semantic Index (LSI) to categorize software systems in open-source software repositories.

Maletic et al. [9] worked on software reverse engineering and applied LSI to analyze the semantic clusters in Mosaic documents files. M Lungu [6] differentiated between that the two levels of abstraction such as intra-system and inter-system in

reverse engineering software ecosystems. To visualize the source code the author developed a static hierarchical decomposition model, at intra level. Explored three view points at inter system level such as overview viewpoints, developer viewpoints, project viewpoints using the various concepts like projects structure, relationship and evolution. Softwarenaut [7] is a tool that provides a high-level view on the system that supports various gardening operations, generating new views such as expanding nodes, collapsing nodes, filtering out nodes and edges. This approach visualizes the high level view of the reverse engineering software ecosystem at two levels and it not applicable in forward engineering ecosystem.

Marcus et al. [10] used LSI to detect high-level conceptual clones, using the LSI, to spot the similar terms. LSI was also used to recover links between external documentation and source code [11]. Marcus et al. [12] employed LSI as a search engine and formulated the queries to detect concept location in the code. M Lungu et al. [2] proposed SPO tool to analyze ecosystem engineering. This tool will visualize the different type of views such as project timelines, project dependency map, project vocabulary map, developer collaboration map, and set of timeline views [2] [3]. The vocabulary map [2] presents the summary of the terms used in the Moose software. The code is analyzed, the identifiers split in component words, these words are stemmed and then the final statistics on the frequency of occurrence of the words is presented as a tag cloud. Tag cloud is a text-based visual representations depicting tag importance by font size. In this approach, the vocabularies in a tag cloud represent only the high level abstraction words.

M Lungu et al. [8] worked on the software ecosystem in the context of the developer contribution. SPO tool automatically recover the domain of expertise of the individual developers in the ecosystem. Domain of expertise is the vocabulary to the tag cloud representation. Recent trends in social and collaborative software have greatly increased the popularity of tag cloud visualization [14].

P Alberto et al. [15] proposed a method to identify interrelated tags based on the textual content of tagged web documents by means of Self-Organizing Maps (SOM). It extracts the most relevant terms for each group of tags by means of language modeling techniques. The resulting SOM turns into a richer tag cloud that provides an alternative way to visualize and navigate the tags and terms. This methodology, organize and visualize the tag cloud, making it easier to analyze relations between web documents.

Thus through the literature it is clear that many researchers, worked on tag cloud in different context. Project vocabulary map presented in SPO show information only about the elements that are relevant to the high level abstraction of an ecosystem. There are cases in which the application should support navigating at a lower level abstraction that enables to view the vocabulary perspective inside a project's in ecosystem. Here in this paper, an attempt has been done to analyze the vocabularies for GRAPPA ecosystem and visually represent it as a tag cloud. The major advantage in visualizing the vocabulary dependencies in ecosystem is to understand the way how the code fits into the ecosystem.

III. ANALYSING SOFTWARE ECOSYSTEM

The GRAPPA open source code ecosystem is visually represented as a tag cloud.

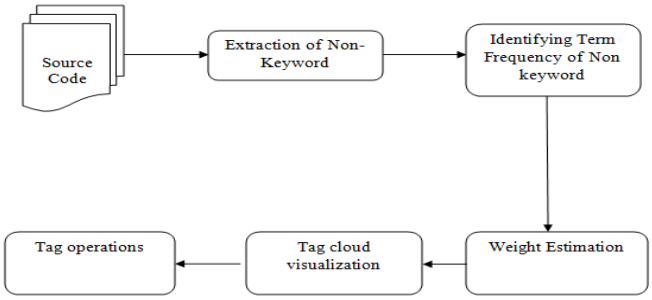

Figure 1: visualization concepts of an ecosystem.

Figure 1 represents the visualization concepts of an ecosystem. The various steps are discussed in detail in this section.

The following steps are used to visually represent the various vocabularies in this ecosystem at low level.

1. Low level information such as non-keywords are extracted from the ecosystem.
2. The frequency count for each non-keyword are obtained and stored in the database.
3. The weights for each frequency count are estimated.
4. Project vocabulary tag cloud is represented.
5. Perform the tag operation on the tag cloud.

A. Extraction of Non-Keyword

GRAPPA [16] is an open source code graph package written in java**.** The package comprises of classes that implements graph representation, presentation and layout services. It provides an application programming interface on top of which web-based application that need to visualize information in terms of graphs such as process flows, business workflows or program dependencies.

The first hierarchy, whose root is the class DotGraph, contains the classes for graph definition (i.e., for defining graphs, subgraphs, nodes, edges, and associating attributes with nodes and edges and associating attributes with nodes and edges). DotGraph defines a graph as a set of DotElements, each of which is a node, an edge, or a subgraph. Each instance of DotElement has set of attributes associated with it, such as shape, style, color, and so on. DotGraph also refers to the class DrawPane, whose instances contain information about displayed of DotGraph.

The second hierarchy, whose root is the abstract class DrawObject, includes classes for graph drawing. Class defines for drawing 36 node shapes (Box, Circle, Ellipse,etc.) and several edge types are included in this hierarchy. Each instance of DrawObject refers to a specific Graphic Context, which provides information for drawing an object on specific canvas.

The third hierarchy, rooted at AppObject is a application-specific classes, and it constitutes the application program interface for GRAPPA. An instance of AppObject refer to an instance of DotElement and has a reference to an instance of DrawObject, depending on the value of the type attribute of the object (e.g. ,the node shape), the methods of the appropriate subclass of DrawObject are called to draw the object on the client canvas.

The abstraction of low level fine grained information of non-keywords such as class name, function name, data are identified from the GRAPPA input source code. The open source code of GRAPPA contains 65 java files and 479 methods. AppObject is one of the java file of GRAPPA. The non-keywords like class name, function name, data all are extracted from that file. The various low level non-keywords that are extracted from AppObject includes *AppObject, classesByTag, tagsByClass, dotElement, getsetDrawDefaults, getReDrawFlag, getDrawObject, draw, getElement, getDrawObject.* The extracted vocabularies are stored in the database.

B. Weight Estimation of Grappa

In section 3.1 the low level abstraction of GRAPPA non-keywords that frequently occurred in the java file AppObject.java has been extracted. Check the reputation of the extracted non-keywords in other java files in GRAPPA . When the same non-keyword is found then the count will be incremented automatically in the data base. Table 1 shows the *frequency count* of the non-keyword in the file AppObject.java in GRAPPA.

For example, '*tagByClass'* non-keyword exists only one time in AppObject.java but it is repeated 3 times in other files of GRAPPA ecosystem. In GRAPPA, '*copy'* is a non-keyword that occurs in many files which has the maximum *frequency count* of 130. This gives a clear picture regarding the usage of non-keyword in a particular project.

Table1: Weight estimation of the file AppObject.java in GRAPPA.

Ecosystem : GRAPPA		
File name : AppObject.java		
non-keywords	**frequency count**	**weight**
AppObject	3	2.9059
classesByTag	3	2.9059
tagsByClass	4	3.5762
dotElement	17	16.7589
getsetDrawDefaults	14	13.7629
getReDrawFlag	6	5.4038
getDrawObject	30	28.5762
Draw	15	13.9234
getElement	4	3.5762
getDrawObject	5	4.4038

To optimize the *frequency count* for various non-keywords, weight need to be estimated. The weight of the non-keyword can be estimated using the formula [17] given below.

Weight=maxPercent + ((max-(max-(count-min)))(maxPercent-minPercent)/(max-min))*

maxPercent = max/total number of files. minPercent = min/total number of files.

In the formula, where *max* denotes the *maximum frequency count* in GRAPPA , m*in* denotes the *minimum frequency count* in GRAPPA, *count* denotes the value of the *frequency count* of this non-keyword. In table 1 third column shows the weight estimation of the file AppObject. Java in GRAPPA. For example , *'tagbyclass'* non-keyword has the value of *maxpercent= 0.4615*; *minpercent=0.0153; max=30; min=1; count=3* from this the weight is calculatedas 4.5762.

IV.TAG CLOUD

The GRAPPA open source code is given as the input to the *tag cloud tool*. This tool will analyze each and every word of the input code and extract the non-keywords such as class name's, methods and data's. The *frequency count* is calculated for each and every non-keyword using the tool. Based on the *frequency count* weight is calculated. According to the weight, vocabularies or non-keywords are visualized as a *Tag cloud*. Figure 2 shows the number of classes, number of methods, number of data and number of lines in a GRAPPA ecosystem. The non-keywords like class name, function name and data are extracted as shown in figure 2. Whenever the same non-keyword is identified, *frequency count* is automatically incremented in the database. Then the weight is estimated according to the *frequency count*.

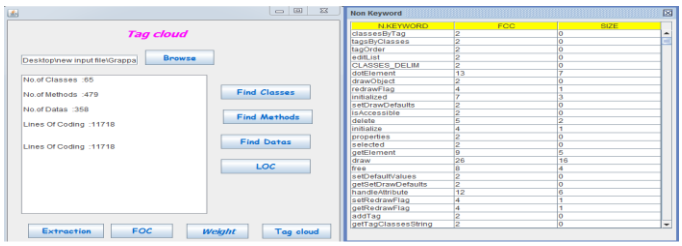

Figure 2: Non-keyword extraction

The *Tag cloud tool* gives the general overview of the vocabularies used in the source code of the project. Figure 3 shows the Tag cloud of GRAPPA ecosystem. This tool gives the information to the developer who wants to obtain a general overview of the domain language of a project. The tag cloud tool analyses and visualizes some of the most important vocabularies used in the GRAPPA ecosystem project are *'copy'*, 'graph', '*subgraph*'.

Figure 3: Tag Cloud of GRAPPA ecosystem.

V. TAG OPERATIONS

A tag operation will displays the list of java files of the particular non-keyword. The tag operation will display in which file it is used as a class name, function name and data. By this the user can differentiate the each occurrence of the non-keywords that is present in a particular java file. The tag operation aids the simplicity for analyzing a large software ecosystem. In Figure 3 the non-keywords are extracted and the file location of every non-keyword is visualized and the source code is displayed. For example '*handleAttribute' h*as occurred in the list of java files AppObject.java, DrawEdge.java, DrawNode.java, DrawObject.java, DrawSubgraph.java and Table.java. Then the source code of the AppObject.java is displayed.

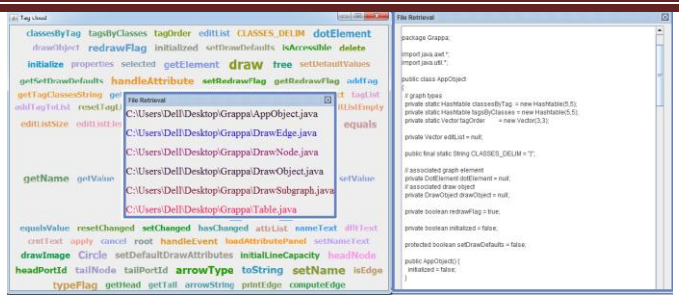

Figure 3: Tag operation – 'File Retrieval'

VI. CONCLUSION AND FUTURE WORK

In this paper, visualization of GRAPPA ecosystem has been done for low level fine grained non-keyword. This visualization has been performed with extraction of non-keywords, frequency count, and weight calculation. To achieve this GRAPPA open source code is used and the number of class files,methods, data and number of lines are extracted. The low level non-keywords are extracted and stored in data base. The *frequency count* has been calculated for all the non-keywords. Weight is estimated according to the *frequency count*. The new developer can easily find about the project based on low level non-keywords which has been obtained in the tag cloud. This will help to get the non-keyword terms present in the large complex software. Here, in this work, the non-keywords is restricted to class name, methods, and preliminary data but it can further expanded. Apart from this dependency graph can be plotted based on the dependencies between classes, methods and data.

REFERENCE

[1] D. Gracanin, K. Matkovic, M. Eltoweissy, (2005),"Software visualization", Springer-Verlag, pp. 221-230.
[2] M. Lungu, M. Lanza , T. Girba , R. Robbes, (2010), " The Small Project Observatory: Visualizing software ecosystems", pp. 264-275.
[3] M.Lungu, T. Girba,(2007), " A small observatory for super-repositories ", in : IWPSE'07: Proceedings of the 9th International Workshop on the Principles of Software Evolution, pp.106-109.
[4] S. Eick, J. Steffen, (1992), " Seesoft a tool for visualizing line oriented software statistics", IEEE Transactions on Software Engineering, pp.957-968.
[5] O. Nierstrasz, S. Ducasse, T. P. Girba. (2005), "The story of moose: An agile reengineering environment", SIGSOFT Software Engineering, pp. 1-10.
[6] M.Lungu, (2008), "Towards Reverse Engineering Software Ecosystem", ICSM , pp.428-430.
[7] M. Lungu and M. Lanza, (2006), "Softwarenaut: cutting edge visualization", Proceedings of . ACM symposium on Software visualization,.pp.179-180.
[8] M. Lungu, M. Lanza, (2010), " The Small Project Observatory - A tool for Reverse Engineering Software Ecosystems".pp.289-292.
[9] J. I. Maletic and A. Marcus, (2000), "Using latent semantic analysis to identify similarities in source code to support program understanding", Proceedings of the 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00), pp.1-46.
[10] A. Marcus and J. Maletic, (2001), "Identification of high-level concept clones in source code", Proceedings of the 25th International Conference on Software Engineering (ICSE'03),pp.1-30.
[11] A. Marcus and J. I. Maletic,(2003), "Recovering documentation-to-source-code traceability links using latent semantic indexing", Proceedings of the 25th International *Conference* on Software Engineering, pp.125-135.
[12] V. R. Andrian Marcus1, A. Sergeyevl and J. I. Maletic, (2004), "An information retrieval approach to concept location in source code", Proceedings of the 11th Working Conference on Reverse Engineering, pp.1-10.
[13] M. M. S. Kawaguchi, P. K. Garg and K. I. Mudablue, (2004), "An automatic categorization system for open source repositories", Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC.04), pp.1-10.
[14] C. Seifert, B. Kump, M.l Granitzer, (2008), "On the beauty and usability of tag clouds", 12th International Conference Information Visualization, pp. 17-25.
[15] A. Zubiaga, A. P. Garcla-Plaza, V. Fresno, R. Martinez, (2009), "Content-based clustering for Tag Clould Visualization", Advances in social Network Analysis and Mining, pp. 316-319.
[16] W. Lee, N. Barghouti, and J. Mocenigo, (1997), "GRAPPA: A graph package in Java", Proceeding on Symposium Graph Drawing, pp.1-8.
[17] http://blog.jeremymartin.name/2008/03/efficient-tag-cloud-algorithm.html