Mining Software Repositories for Defect Categorization

Sakthi Kumaresh, and Ramachandran Baskaran

Original scientific paper

***Abstract*— Early detection of software defects is very important to decrease the software cost and subsequently increase the software quality. Success of software industries not only depends on gaining knowledge about software defects, but largely reflects from the manner in which information about defect is collected and used. In software industries, individuals at different levels from customers to engineers apply diverse mechanisms to detect the allocation of defects to a particular class. Categorizing bugs based on their characteristics helps the Software Development team take appropriate actions to reduce similar defects that might get reported in future releases. Classification, if performed manually, will consume more time and effort. Human resource having expert testing skills & domain knowledge will be required for labeling the data. Therefore, the need of automatic classification of software defect is high.**

This work attempts to categorize defects by proposing an algorithm called Software Defect CLustering (SDCL). It aims at mining the existing online bug repositories like Eclipse, Bugzilla and JIRA for analyzing the defect description and its categorization. The proposed algorithm is designed by using text clustering and works with three major modules to find out the class to which the defect should be assigned. Software bug repositories hold software defect data with attributes like defect description, status, defect open and close date. Defect extraction module extracts the defect description from various bug repositories and converts it into unified format for further processing. Unnecessary and irrelevant texts are removed from defect data using data preprocessing module. Finally grouping of defect data into clusters of similar defect is done using clustering technique. The algorithm provides classification accuracy more than 80% in all of the three above mentioned repositories.

***Index Terms*— Software Defect, Defect Classification, Bug Repository, Clustering, Bug Categorization.**

I. INTRODUCTION

Software defect can be defined as imperfections in software development process which disenables the software to fail to meet the desired expectations [14]. Software defects are inherent in the software process development, and it is also a significant factor contributing to software quality.

Developing defect free software product is impossible, but the organization can aim at minimizing defects by investing majority of the effort in detecting and preventing defects through effective defect management process. Software defect detection is done at various phases of software development through activities like Requirement's Review, Code Inspection, walkthrough and testing. Detected defects are then stored in software repositories for further analysis.

Software bug repositories holds large amount of useful information about software defects [5]. It contains attributes of defects like defect id, description, status, open date, close date etc. Software description holds text data which tells us details about the defect. Bugs are reported to these repositories by non-technical people who cannot assign correct class to these bugs. Developers may be proficient in one particular domain, assigning a particular bug to relevant developer is important as it could save time and would help to maintain the interest level of developers. However, assigning the right bug to the right developer is quite difficult for tri-ager without knowing the actual class; the bug belongs to. [6]

Data mining techniques can be applied to handle large amount of data and text mining in particular to extract the knowledge from bug repositories. Categorizing defects into types and performing analysis may be beneficial to software organizations, but defects are not grouped into categories as it involves huge effort and time [10]. Thus there is a need for an automated approach that could help developers assign category labels to defects during defect analysis [10].

This work attempts to categorize bugs into classes by proposing an algorithm called SDCL. The algorithm proposed, works in three modules namely Defect extraction module, Data pre-processing and clustering module. Defect extraction module extracts the defect data from various software repositories like Bugzilla, JIRA and Eclipse and converts it into a unified format. Defect details collected from various bug repositories are then pre-processed for efficient defect categorization. Parsing the defect data, Stop word removal and Stemming is performed through data pre-processing module.

A. Defect Classification

Classification is one of the popular data mining techniques to categorize the defects into classes. Software bug classification is the process of classifying the software bugs into different categories. Classification of the software defect data is done in order to get deeper insight into defect details. Software defect classification is an essential part of improving software quality. Manual classification consumes lot of time and effort and people with domain knowledge and testing skills are required. Numerous techniques and algorithms are available

Manuscript received February 27, 2015; revised April 12, 2015.

S. Kumaresh is with the MOP Vaishnav College for Women, Chennai, India (e-mail: sakthi.kma@ gmail.com).

R. Baskaran is with the Department of Computer Science and Engineering, Anna University, Chennai, India. (e-mail: dr.baskaran10@gmail.com).

to automate the defect classification process. SDCL algorithm proposed in this paper aims to categorize defects into classes using clustering technique. Once categorization of software defects are done, defect pattern can be analyzed which in turn helps to find out the root causes of the defects. The scope of this study is limited to categorizing defect into various classes using text clustering technique.

B. Clustering Technique

A cluster is a collection of similar objects which shows some similar characteristics between them and shows some dissimilar characteristics between the objects in other clusters. There are number of clustering algorithms available and various techniques exist for measuring the distances for the clusters data points [7].

Clustering technique is very useful in the text domain, where the objects to be clustered can be of different type such as documents, paragraphs, sentences or terms. Clustering can be used for number of tasks like information retrieval, web analysis, marketing and medical diagnostic, corpus Summarization and Document Classification [12]. In this work, clustering is done using vector space model, in which the closeness between the defect data is identified using cosine similarity measure. The efficiency of our algorithm is tested on data collected from three software repositories Bugzilla, JIRA and Eclipse. The results show that our defect categorization algorithm could achieve an accuracy of more than 80%.

The rest of this paper is structured as follows. Section 2 gives details about bug repository. Section 3, reviews the related work on software defect classification using clustering technique. Section 4 details about the proposed Software Defect CLustering (SDCL) through three modules namely, Defect Extraction module, Data pre-processing and clustering module. Section 5 briefs about implementation details and section 6 presents the performance measure of the proposed algorithm. Finally section 7 concludes the paper with future research.

II. BUG REPOSITORY

Organizations aim to develop defect free software, but defects are inevitable during software development. So, all software projects need defect tracking system to track the defects related to software. Software bugs are managed using bug tracking tools like Bugzilla, Eclipse, JIRA etc. These are online repositories which contain useful information about software defect in HTML or XML format. Knowledge discovery from such software bug repositories is essential to know about the details of the defects. These repositories posses useful information about defects like defect summary, description, open date, closed date, severity, priority, user comments etc. Many of the defect related attributes like defect description are in textual form. To extract knowledge from bug repositories data mining techniques can be used. In this work, an attempt has been made to categorize the defect in different labels on the basis of defect description using clustering technique. In order to take corrective actions for the defects, the defects have to be assigned to the right person. This can be accomplished by categorizing the bugs into various categories.

III. LITERATURE SURVEY

Mining software repositories is essential as it contains information about defects. Software defect prediction, classification is done using software repositories. An approach to automatically identify the duplicate bug reports in the software bug repositories is proposed by Jalbert and Weimer [1]. Naresh kumar et al [5] presented a software bug classification algorithm, CLUBAS (Classification of Software Bugs Using Bug Attribute Similarity). CLUBAS is a hybrid algorithm, and is designed by using text clustering, frequent term calculations and taxonomic terms mapping techniques thereby categorizing bug data. Neelofar et al [6] realizes that assigning a particular bug to relevant developer could save time and would help to maintain the interest level of developers by assigning bugs according to their interest. However, assigning right bug to right developer is quite difficult for tri-ager without knowing the actual class, the bug belongs to. Hence in this paper, they have classified the bugs in different labels on the basis of summary of the bug. Lian Yu et al [9] discusses that it is hard to understand the crux of the problem and the debuggers must be well equipped with domain knowledge. In [7] Naresh Kumar Nagwani et al used Suffix Tree Clustering (STC) algorithm for software bug classification. STC can be applied to create the clusters of software bug record. Surendra Naidu [8] proposed the system for classifying various defects using decision tree based defect classification technique, which is used to group the defects after identification. P. V. Ingle et al [13] have presented different types of clustering methodology and categorize cluster as simple, medium and complex cluster. They does defect clustering using K means clustering algorithm. NLP based Information Retrieval system for mining defect data repository based on the quality of text description was proposed by Schugertl P et al [4]. Ferdian Thung [10] has worked on defect categorization technique by analyzing both texts from bug reports and code features from bug fixes. Analyzed data from three software system and classified them according to three categories of Orthogonal Defect Classification (ODC). The categories include control and data flow, structural and non-functional. Thung was able to achieve 77.8% accuracy by using SVM multi class classification algorithm.

An algorithm, Lingo is proposed by Osinski et al. [2] for clustering search results of text documents, Lingo uses algebraic transformations of the term-document matrix, and frequent phrase extraction using suffix arrays. Qinbao Song et al [3] have proposed a FAST algorithm for selecting the Feature subset for clustering and evaluated empirically. In this they have adopted the efficient Minimum-Spanning Tree (MST) method.

IV. SOFTWARE DEFECT CLUSTERING (SDCL) ALGORITHM

Software defect reports that are extracted from bug repositories contain lot of defect related attributes. Attributes like Defect description and summary are the important one in

the bug repository which holds text data. For Effective categorization of the software defect, a text classification scheme is required. This work, attempts to categorize software defects by proposing an algorithm called Software Defect CLustering (SDCL) algorithm. This algorithm contains three modules to perform defect categorization using bug data from software repositories. The various modules used in this algorithm are Defect Extraction, pre-processing and clustering which is depicted in fig 1. Defect Extraction module extracts defect data from various bug repositories and stores in primary database. Stemming the defect data and stop word removal is done during pre-processing. Finally defect clusters are created based on the cosine similarity measure.

There is no universal correct way to categorize bugs, it is recommended to adapt some taxonomy to categorize bugs. By categorizing bugs, we can pick bug category that has more number of bugs, focus on it, and take preventive actions in future projects in order to avoid such defects from recurring [15]. While adopting taxonomy to categorize bugs, this work attempts to use Logical and syntax defects in addition to assignment and interface bug taxonomy given by Orthogonal Defect Classification (ODC). The following algorithm presents the various steps involved in SDCL.

Algorithm: SDCL (**S**oftware **D**efect **CL**ustering)
Input: Defect data from software Bug Repositories
Output: Software Defect Clusters
START
def_Extract(bug_Db)
u_def_Rep[ ] ←Conv_DefRep(defect_Rep[ ]) // Convert the data sources from various software repositories in unified (Common) format and store it in database for further analysis.

pre-Process(u_def_Rep) // Pre process the defect description from various defect repositories
pre_proc_Def[ ] ← stm_ BagOfWords[ ]

def_Cluster (pre_proc_Def[ ]) // Finds defect clusters and assigns class labels.
END

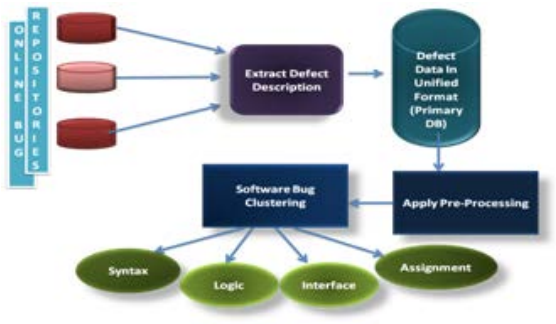

Fig. 1 **S**oftware **D**efect **Cl**ustering (SDCL)

A. Module 1: Defect Extraction

Fig 2 shows the interface created for defect extraction. In this module, the steps involved in extracting defect reports from various sources are taken one by one. Initially bug reports from various bug repositories are taken as input. Later these reports are converted into unified format to enable them to store in database for further analysis.

Fig. 2 Defect Extractions

Steps involved in defect extraction are shown below:

Module 1: (Defect Extraction)
def_Extract(defect_Data)
START
defect_Rep[ ] ← Get bugz(dr), Eclipse(dr),Jira(dr) // Get the defect reports(dr) from Bugzilla, Eclipse and JIRA.
Convert the data from software repositories into unified format
u_def_Rep[ ] ←Conv_DefRep(defect_Rep[ ]) // Convert the data sources from various software repositories in unified (common) and store it in primary database for further analysis.
For each record retrieved from the primary database, do the following:
Def_des[ ] ← extract_desp (u_def_Rep[ ])
End.
STOP

B. Module 2: Data Pre-processing

Defect description is one of the important attribute of the defect reports. It gives details about defect. As these data are available in the form of text, we need to do some processing in order to make it available for defect analysis. Here, every defect description is treated as Bag-of-words. Data pre-processing involves three steps namely parsing the data, stop word removal and applying stemming to the text. Parser does the process of removing unnecessary text from the defect description. The unnecessary text may include text found between hyphens, after bullets and between parentheses.[19] Def_Parser function outputs each defect description as a set of words by removing the unnecessary semicolons, colons, exclamation marks etc. Interface created for doing the pre-processing task is shown in figure 3.

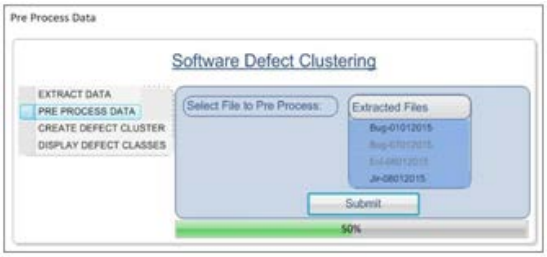

Fig. 3 Data Pre-processing

Another pre-processing task involves the elimination of “stop” words. Stop words are very common words like the articles “the” and “a” that do not add to the information content of a text string. These words are irrelevant to the defect analysis and are therefore eliminated. A further task that is performed in text mining is “stemming.” Stemming algorithm attempts to replace a word, to the “stem” or main root of the word. It reduces the frequency of unique words. Commonly used

algorithm for stemming is the "Porter's Algorithm" [18]. Module 2 gives the steps involved in pre-processing.

Module 2 (Pre-processing)
pre-Process(m_def_Rep)

START

Repeat the following until EOF
Treat every defect description in the bug database (Def_des[ ]) as a "bag-of-words"
BagOfWords[ ] ← DefParser(*Def_des[ ]*) // DefParser function outputs the each defect description as a set of words by removing unnecessary semicolons, colons, exclamation marks etc.
stp_ BagOfWords[ ] ← BagOfWords[ ] // Eliminate stop words from the BagOfWords[ ] ; Words that are not relevant to Software Defect
stm_BagOfWords[ ] ← stp_BagOfWords[ ] // Apply stemming over the BagOfWords [ ]; Stemming words are reduced to their root.
pre_proc_Def[ ] ← stm_ BagOfWords[ ] // outputs pre-processed defects
term_Data[ ] ← tokenizer(pre_proc_Def) // tokenizer divides the data from the pre-processed defects into components and stores in term Data[ ]

STOP

C. Clustering

Clustering is the process of grouping a set of data object into classes of similar objects [11]. It is the most common form of unsupervised learning, meaning, finding natural grouping of instances given un-labeled data. Clustering help users understand the natural grouping or structure in data set. The quality of a clustering result depends on the similarity measure used by the clustering method and its implementation [16].

As software defect description holds text data, Clustering of defects based on the similarity of terms in defect description is done in this study. For this purpose, distance based clustering algorithm is used to find the similarity measure between the defect description. One of the popular similarity function called cosine similarity is employed to find the closeness between two defect descriptions.

Cosine similarity comes under vector space model. The set of defect description in a collection then is viewed as a set of vectors in a vector space. Vectors deals only with numbers. In this work, we are dealing with defect description which holds text data. Hence TF and IDF are used to convert text into numbers so that it can be represented by a vector. In reality each defect description will be of different size. On a large defect description, the frequency of the terms will be much higher than the smaller ones. Hence normalization of text based on its size is necessary. A simple way to achieve this is to divide the term frequency by the total number of terms. Using (1), we can find out the similarity between two defect descriptions. Various steps involved in clustering are shown in module 3.

$$\text{Cosine Similarity}(Dd_x, Dd_y) = \text{Dot product}(Dd_x, Dd_y) / \| Dd_x \| * \| Dd_y \| \text{ [17]} \quad (1)$$

$$\text{Dot product}(Dd_x, Dd_y) = Dd_x[0] * Dd_y[0] + Dd_x[1] * Dd_y[1] * \ldots * Dd_x[n] * Dd_y[n]$$

$$\| Dd_x \| = \text{square root}(Dd_x[0]^2 + Dd_x[1]^2 + \ldots + Dd_x[n]^2)$$

$$\| Dd_y \| = \text{square root}(Dd_y[0]^2 + Dd_y[1]^2 + \ldots + Dd_y[n]^2)$$

Module 3 (Clustering)

def_Cluster (pre_proc_Def[ ])

START
For each pair of bug description Dd_x and Dd_y,
Compute normalized_frequency of term_data obtained from module 2
text_Sim(Dd_x, Dd_y) // calculate the textual similarity between the bug description using cosine similarity measure
If sim(Dd_x, Dd_y) > t (threshold), then assign Dd_x, Dd_y to same cluster else create a new cluster and assign Dd_y to the new cluster.
End
For each cluster Ci, do the following:
cl_Data ← collect[def_desc] //collect the defect description under each cluster
FC[ ] ← FCount[cl_Data]) // Apply frequency count to the cluster data
Assign the term with highest frequency term as class labels for the cluster.
End
STOP

V. IMPLEMENTATION DETAILS

The entire algorithm is implemented in java eclipse environment with a maximum of 2500 defect reports. Defect reports with missing data are eliminated. Defect reports are taken from three different repositories with 495, 1083, 2095 records for experimentation of the algorithm. Java supports traversing into the xml using parsers. The bug reports that are retrieved from bug repositories are in the xml format which are later stored in primary database using java program. The Document Object model and Simple Application for XML (SAX) are the popular parsing tools in java. DOM is comparatively slow and supports adding new elements to the xml file. But here the requirement is to store the xml content in a database. Using SAX the data from the xml file are transferred to the local variables and then using JDBC, the data are stored in the database.

2. PERFORMANCE MEASURE

Table 1 illustrates a confusion matrix to evaluate the performance of classification problem. Accuracy and F-measure are the two performance measure that is used in this study.

TABLE I.
CONFUSION MATRIX

	Same Cluster	Different Clusters
Same Class	True Positive (TP)	False Negative (FN)
Different Classes	False Positive (FP)	True Negative (TN)

TABLE II.
PERFORMANCE MEASURE

Measure	Formula	Meaning
Precision	TP / (TP + FP)	Precision is the ratio of the number of correctly classified defects and the actual number of defects which was assigned to the type.

Recall	TP / (TP + FN)	Recall is the ratio of the number of correctly classified software defects and the number of software defects which belongs to the type.
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Accuracy is defined as the number of defects that are correctly classified to the total number of defects.

A. *Accuracy and Rand index*

The Rand index measures the percentage of decisions that are correct. It is simply the measure of accuracy [20]. The accuracy measure for the JIRA, Eclipse and Bugzilla are shown in fig 4. It is observed from the experimental results that accuracy wise SDCL performs well and maintains more than 80% accuracy for different samples in all of the above mentioned repositories.

TABLE III:
RESULTS

Data Sources	Instances	Precision	Recall	F-Measure	Accuracy
JIRA	495	0.279	0.354	0.312	84.84%
Eclipse	1083	0.419	0.372	0.394	89.38%
Bugzilla	2095	0.253	0.306	0.277	82.24%

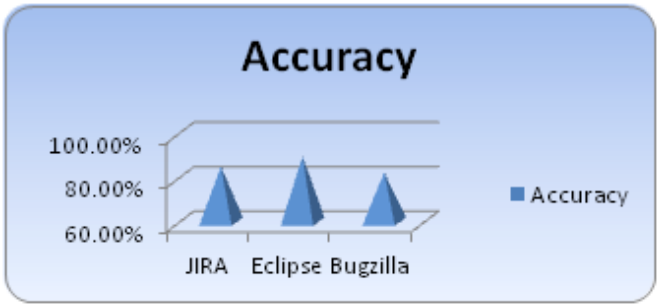

Fig. 4 Accuracy Measure

B. *Precision and Recall*

The following chart shows the performance of SDCL algorithm in terms of precision and Recall. It gives good precision for Eclipse data records than other two repositories.

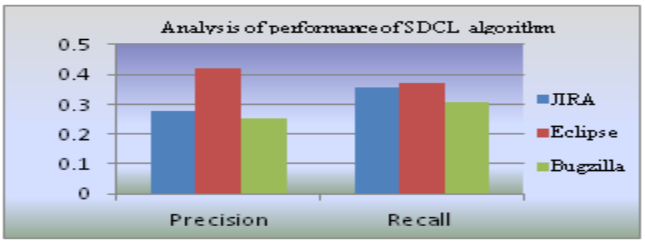

Fig 6: Performance of SDCL algorithm

C. *F-Measure*

F-Measure is a combined measure of Precision and Recall parameters. F-Measure is calculated using (2). Fig 5 shows the F-Measure for the three repositories taken for study. From the F-Measure view point, the values are consistent for JIRA and Eclipse, and it is slightly lesser for Bugzilla. The higher value of F-measure indicates higher quality of the classifiers. SDCL performed well for JIRA and Eclipse when compared to Bugzilla.

$$F = \frac{(2*precision*recall)}{precision+recall} \quad (2)$$

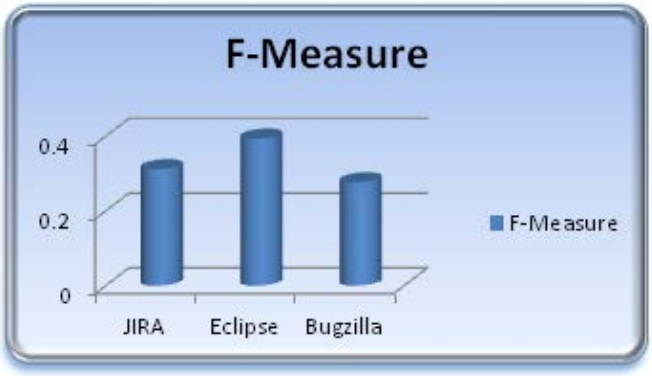

Fig. 5 F-Measure

VII. CONCLUSION

Defect classification is a difficult task to perform unless and otherwise the person involved in classifying defect has extensive knowledge about software testing. Moreover classifying defect involves lot of time, cost and effort. Therefore, this work attempts to do automatic defect classification through clustering techniques. The clustering technique helps the project managers to know the category of defect and helps them to assign right defect to the right personnel.

Realizing the need for automatic defect categorization, in this work, SDCL algorithm is proposed for categorization of software defects using clustering technique. Here, clustering is done based on vector space model using cosine similarity to judge the closeness between text data. Once the similar defects are grouped according to the above technique, from each clustered defect description, frequent terms are generated, and the term with highest frequency is assigned as a class label to the cluster. Defect clusters are created under the category, syntax and logical in addition to assignment and interface defect taxonomy given by ODC.

In order to improve the efficiency of the algorithm, the future scope of the work is related to extracting key phrases from the defect description and then applying clustering technique to group defect data into various categories. By doing so, the time involved in similarity calculation and the comparison of terms in the text can be reduced thereby reducing the response time of the automatic defect classification system.

REFERENCES

[1] N. Jalbert and W. Weimer, “Automated Duplicate Detection for Bug Tracking Systems,” IEEE International Conference on Dependable Systems & Networks, Anchorage, 24-27 June 2008, pp. 52-61.

[2] S. Osinski, J. Stefanowski and D. Weiss, “Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition”, Proceedings of the Springer International Intelligent Information

Processing and Web Mining Conference, Zakopane, 17-20 May 2004, pp. 359-368.

[3] Qinbao Song, "A FAST Clustering-Based Feature subset selection algorithm for High Dimensional Data", IEEE transactions on knowledge and Data Engineering, Volume 25, Issue 1, pp 1-14, August 2011,

[4] Schugerl P, Riling J, Charland P, "Mining Bug Repositories – A Quality Assessment", Proc of the International Conference on Computational Intelligence for Modelling control & Automation, Vienna, 10-12 December 2008, pp 1105-1110.

[5] Naresh Kumar, CLUBAS: An Algorithm and Java Based Tool for Software Bug Classification Using Bug Attributes Similarities. Journal of Software Engineering and Applications, 5, 436-447,2012.

[6] Neelofar, Muhammad Younus Javed, Hufsa Mohsin, "An Automated Approach for Software Bug Classification" in proceedings of the Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS),2012

[7] Naresh Kumar Nagwani , "Software Bug Classification using Suffix Tree Clustering (STC) Algorithm IJCST Vol. 2, Issue 1, March 2011

[8] M. Surendra Naidu et al. , " Classification of Defects in software using Decission Tree Based Algorithm',International Journal of Engineering Science and Technology (IJEST), ISSN : ISSN : 0975-5462 Vol. 5 No.06 June 2013

[9] L. Yu, C. Kong, L. Xu, J. Zhao and H. Zhang, "Mining Bug Classifier and Debug Strategy Association Rules for Web-Based Applications," in *08* Proceedings of the 4th international conference on Advanced Data Mining and Applications , 2008.

[10] Ferdian Thung, David Lo, and Lingxiao Jiang, "Automatic Defect Categorization" (2012). Research Collection School of Information Systems. Available at: http://ink.library.smu.edu.sg/sis_research/1681

[11] Ashish Moon and T Raju " A Survey on Document Clustering with Similarity Measures", published in International Journal of Advanced Research in Computer Science and Software Engineering" Volume 3, Issue 11, November 2013.

[12] Charu C Aggarwal, 'A Survey of Text Clustering Algorithms"

[13] P. V. Ingle, M.M Deshpande, "Software Quality Analysis with Clustering Method" International Journal of Applied Information Systems (IJAIS), Foundation of Computer Science FCS, New York, USA. International Conference & workshop on Advanced Computing 2013 (ICWAC 2013) – www.ijais.org 8

[14] Sakthi Kumaresh and Baskaran R, "Defect Analysis and Prevention for Software Process Quality Improvement", published in International Journal of Computer Applications vol 8 – No 7, October 2010.

[15] http://www.khannur.com/stb3.3.htm

[16] http://www.cs.put.poznan.pl/jstefanowski/sed/DM-7clusteringnew.pdf

[17] https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

[18] Atika Mustafa, Ali Akbar and Ahmed Sultan, "Knowledge Discovery using Text Mining: A Programmable Implementation on Information Extraction and Categorization". International Journal of Multimedia and Ubiquitous Engineering, Vol4 No 2, April 2009.

[19] Ayaz Ahmed Shariff K, MohammednAli Hussain, Sambath Kumar, "Leveraging Unstructured Data into Intelligent Information Analysis. Proceedings of the International Conference on Information and Network Technology", IPCSIT volume 4, 2011.

[20] http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

Sakthi Kumaresh is a Research Scholar at Bharathiar University, Coimbatore, India; she obtained her Master's Degree from Madurai Kamaraj University, Madurai, TN, India in 1996 and M Phil in Computer Science from Periyar University, Salem, TN, India in 2006.She is currently working as Associate professor at Department of Computer Science in MOP Vaishnav College at Chennai. She has more than a decade of teaching experience. Her areas of specialization include Software Engineering, Software Project Management, Software Testing, Software Quality Management, Unified Modeling Language, Data Mining and knowledge Engineering. She is a researcher in the area of Software Quality Engineering. She has publications in National and International conferences and in several International journals.

Ramachandran Baskaran is working as the Associate professor in Department of computer science, Anna University, Chennai. He has obtained his M.E. and Ph.D. in the field of Computer Science and Engineering in Anna University at Chennai, India. He is having more than a decade of experience as an academician and his research areas include Multimedia and principles, Software quality engineering, Software Agents and Distributed networking. He has published around 75 research papers in National and International Journals and Conferences. He is a member of various forums. He is the editor and a reviewer of various journals. He is guiding research scholars working in area of software standards for Attributes Specific SDLC Models & Evaluation and Metric Based Efficient Traffic Management.