ISSN NO. 2320-5407

Journal Homepage: - www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI: 10.21474/IJAR01/1331
DOI URL: http://dx.doi.org/10.21474/IJAR01/1331

RESEARCH ARTICLE

Predicting Bug severity using Classification on Clustered Bugs Data.

***Rajalakshmi R [1] and Dhanya P M[2].**
Department of Computer Science and Engineering, Rajagiri School of Engineering and Technology Kochi, India.

Manuscript Info

Manuscript History

Received: 18 June 2016
Final Accepted: 15 July 2016
Published: August 2016

Key words:-
Bug Severity Prediction,
Clustering,
Classification.

Abstract

Bug Triaging is an inevitable step in software testing process. It assigns a potential developer to resolve the bug and if it is done manually, it takes lot of time and resources. Hence a automatic bug triage system is developed to address the issue. In addition to predicting the developer, it is important to understand how quickly the bugs are addressed based on severity factor. Bug severity prediction based on classification algorithms in data mining helps to predict the severity of the bug. The system also studies the effect of clustering before classification. Here we group similar bugs to a cluster based on the problem title attribute. Thus classification was then applied to the clusters obtained, to assign severity labels to bugs based on priority, product and component attributes. A comparative study of different combinations of classification and clustering algorithm on the performance of prediction is also undertaken in this work.

Introduction:-

Data mining for software engineering consists of collecting software engineering data, extracting some knowledge from it and, if possible, uses this knowledge to improve the software engineering process. Software repositories are large scale databases for storing the output of software development like source code, bugs, emails and other specifications. Bug repository is a kind of repository which handles software bugs.

Bug triage, an important step for bug fixing, is to assign a new bug to a relevant developer for further handling. Assigning correct priority to the bugs plays a significant role in fixing the critical bugs on time. When clustering is performed before classification step, it can significantly improve the performance. Thus prediction of developer with correct severity helps in bug management process.

Text mining, a kind of data mining is the process of deriving high-quality information from text. Text mining or knowledge discovery from text (KDT) deals with the machine supported analysis of text. It uses techniques from information retrieval, information extraction as well as natural language processing (NLP) and connects them with the algorithms and methods of Knowledge Discovery in Data (KDD), data mining, machine learning and statistics. Cluster analysis or clustering technique used in the paper is the task of assigning a set of objects into groups(called clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters. The preprocessed data sample is initially used for further clustering task. Either feature selection or extraction can be used to obtain an appropriate set of features to use in clustering. Pattern proximity is usually measured by a distance

Corresponding Author:- Rajalakshmi R.
Address:- Department of Computer Science and Engineering, Rajagiri School of Engineering and Technology Kochi, India.

function defined on pairs of patterns. Euclidean distance, Minkowski distance, Manhattan distance are some distance measures used. Partitional Clustering (dividing data objects into non overlapping subsets or clusters) and Hierarchical clustering (set of nested clusters organized as a hierarchical tree which maintains class-subclass relationship) are broadly two types of clustering approaches. Kmeans and Expectation Maximisation clustering studied here fall under partitional clustering in which each object belongs to a unique cluster. The area of research undertaken here is that whether clustering algorithms can help supervised learners to specialize their treatments according to different specific areas in the input space. Here experiments focus on general framework of classifier combination when one learner is unsupervised.

This paper focus on clustering the bug reports which has similar severity attributes, preceding classification. The objective of this paper is to study which combination of clustering and classification algorithms are suitable to produce best results for bug severity prediction.

The content of the paper is split into various sections each describing the topic in detail. The division of the contents is made as follows:
Section 1 gives an Introduction and Highlights on the area of research.
Section 2 gives an overview on Literature Survey.
Section 3 deals with proposed method and design.
Section 4 gives the implementation details of the work.
Section 5 provides the results obtained.
Section 6 provides the concluding remarks.

Literature Survey:-

First of all, an automatic bug triage system is implemented based on the work done by (Cubranic and Murphy,2006) which uses classification techniques. To predict the severity of bugs, previous works usually adopted machine learning algorithms. (Menzies and Marcus, 2008)designed and built a tool named SEVERIS which is based on a rule-learning technique to effectively predict the severity of the bug. (Lamkanfi et al,2010) utilized a Naïve Bayes classifier to predict whether a new bug report belongs to severe or non severe category. In a later work, (Lamkanfi et al,2011)compared different text mining algorithms for predicting the severity of a reported bug and concluded that Naive Bayes Multinomial is best suited for the purpose of classifying bug reports. In Information Retrieval paper by (Tian et al, 2012) predicted bug severity first based on BM25 similarity measure and then using KNN. (Pamela Bhattacharya et al, 2012) proposed that graph-based approaches can help to better understand software evolution, and to construct predictors that facilitate development and maintenance. Graph metrics can detect significant structural changes, and can help estimate bug severity, prioritize debugging efforts, and predict defect-prone releases.

Researches had explored that when clustering is used as preprocessing step before classification step is applied on the data, it improves the performance of machine learning algorithms. (Ricardo Vilalta, Murali-Krishna Achari, and Christoph F. Eick,2003) has shown that experimental results on real-world domains show an advantage in predictive accuracy when clustering is used as a preprocessing step to classification.

Proposed Method and Design:-

Bug triage is an important step in the process of bug fixing. The existing bug triage approaches are based on machine learning algorithms, which build classifiers from the training sets of bug reports. Although using these techniques have provided good results for bug triage, there is still scope for improvement. Assigning correct priority to the bugs play a significant role in fixing the critical bugs on time. For this , a new method is proposed here. When clustering is used as a preprocessing step in certain areas before other machine learning algorithms are applied on data, such as classification, it can increase the performance of the machine learning algorithms. This is used for automated classification of severity of software bugs. It is being proposed to cluster the the bugs based on the similarity of the problem title attribute instead of directly applying classification algorithm on the data. Clustering can be considered the most important unsupervised learning problem; it deals with finding a structure in a collection of unlabelled data. A cluster is therefore a collection of reports which are coherent internally, but clearly dissimilar to the reports belonging to other clusters.

The approach used for automated severity classification of software bugs as part of software bug triaging is discussed below, it addresses following two areas related to bug triaging
1. It clusters similar bugs based on their problem title using clustering algorithm.
2. In each cluster, severity label is assigned to each bug using classification technique.
The system architecture is shown below in the figure 1

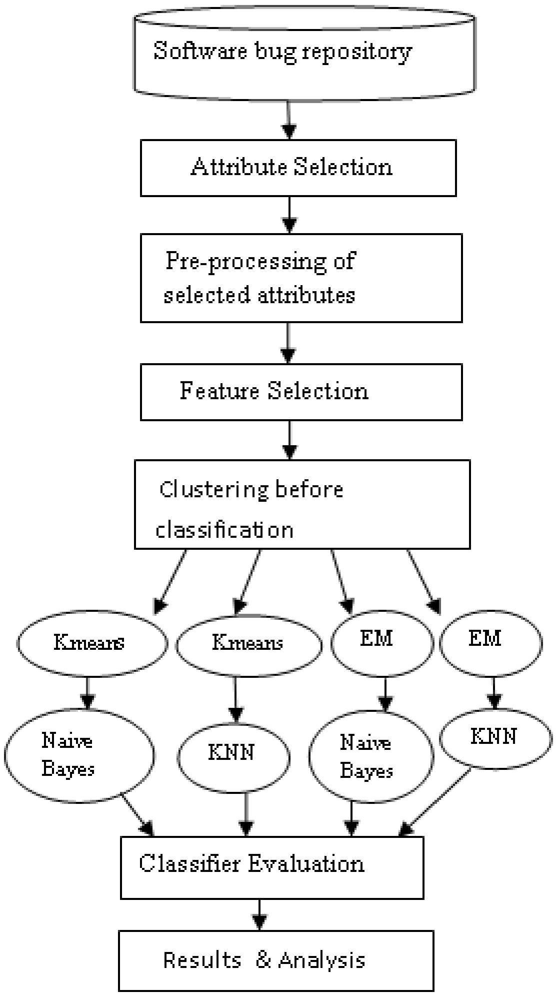

Fig 1:-System Architecture

Implementation:-
The system is implemented in Java language over MYSQL database. The different modules of the system are
Data collection, Attribute Selection, Pre processing of selected attributes, Clustering and Classification.

Data Set:-
The dataset for the work is the bug data of open source bug repositories such as eclipse/Mozilla. The bug data is a collection of bug reports entered in the bug repository entered by users when they encounter bugs. These bug reports contain different fields or attributes providing the details of the bug. The details of bug report essential to carry out the work are extracted for each bug report. Bug reports that have status FIXED OR DUPLICATE are chosen for training. The training set used here includes 2393 bug reports which are of status FIXED.

Attribute Selection
The features/attributes required for the work are extracted in this stage. These data includes various fields (or attributes) of a software bug like bug-id, summary of the bug, description of the bug, assigned- To field (developer

to whom the bug is assigned), product, priority, severity and component. The extracted attributes are stored in the database.

Pre processing of selected attributes:-
After the software bug records are extracted and made available at local system, pre processing of problem title attribute or summary field of the bug is performed. The pre-processing takes places in two stages: elimination of stop words and tokenisation. In stopping the first step is to identify the suitable stop list, which consists of the terms (words) not relevant for the classification of the software bugs. Terms with numeral and special characters are also eliminated in the process of stopping. However, the terms like "not" etc. are not removed during stopping, since such terms are relevant for software bug classification. Stop word removal helps to remove unnecessary and uninformative words which do not help text categorisation purpose. This process reduces the size of document by 30 percent.

Tokenisation: Tokenization is a process in which it simply segregates all the words, numbers, and their characters from a given document. These identified words, numbers, and other characters are termed as tokens. Along with token generation this process also computes the frequency of occurrence of all these tokens present in the input documents.

Feature Selection using term frequency concept:-
Obtain bag of words (top list of words) using term frequency concept. Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization using equation (1)

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}} \qquad (1)$$

Clustering:-
The bug reports were clustered on the basis of similarity of their problem title attributes. To compute this, bug reports needs to be represented as feature vectors. Preprocessing step of problem title attribute performs tokenisation and removes unnecessary words. Then a representative set of words is determined which acts as feature vector. Different clustering algorithms such as Simple KMeans and Expectation Maximisation are inputted with pre-processed bug attributes from the previous step. The idea of clustering as shown in fig 2

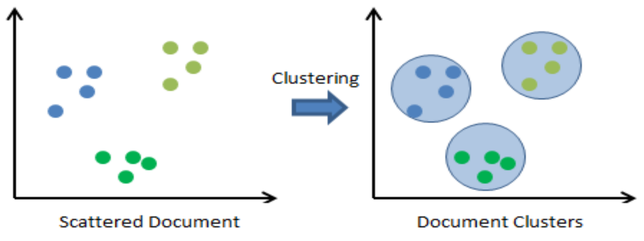

Fig 2:- Document Clustering

K-Means Clustering Algorithm:-
The k-means clustering algorithm is known to be efficient in clustering large data sets. The K-Means algorithm aims to partition a set of objects, based on their attributes/features, into k clusters, where k is a predefined or user-defined constant. The main idea is to define k centroids, one for each cluster. The centroid of a cluster is formed in such a way that it is closely related (in terms of similarity function; similarity can be measured by using different methods such as cosine similarity, Euclidean distance, Extended Jaccard) to all objects in that cluster.

Algorithm Steps:-

```
1. Specify the no: of clusters, let it be $k$ and $d$ be the distance measure between bug reports.
2. Randomly select $k$ reports as seeds $\{s_1, s_2 \dots s_k\}$ and form initial clusters based on the seeds.
3. Until clustering converges or other stopping criterion:
     For each bug report $x_i$:
        Assign $x_i$ to the cluster $c_j$ such that $d(x_i, s_j)$ is minimal
        (Update the seeds to the centroid of each cluster)
     For each cluster $c_j$
          $s_j = \mu(c_j)$
```

Centroid or Mean point of cluster $\mu(c)$ using formula $\vec{\mu}(c) = \frac{1}{|c|}\sum_{\vec{x} \in c} \vec{x}$ (2)

Reassignment of bug instances to cluster based on Euclidean distance formula $L_2(x,y)$

$$L_2(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \quad (3)$$

Expectation Maximisation Clustering Algorithm:-

Iterative method for learning probabilistic categorisation model from unsupervised data. EM assigns probability distribution to each instance which indicates the probability of it belonging to each of the clusters.

Algorithm Steps:-

1) Initially assume random assignment of bug instances to different bug severity categories.
2) Learn an initial probabilistic model by estimating model parameters from this randomly labelled data.
3) Iterate below steps until convergence
 E-step: Compute posterior probability for each instance using the current model and probabilistically relabel the instances based on the estimates
 M-step: Re estimate the model parameters from the probabilistically relabeled data.

Classification algorithms:-

In text classification, we are given a description d ∈ X of a document, where X is the document space; and a fixed set of classes C = $\{c_1, c_2, .. , c_j\}$.Classes are also called categories or labels. Typically, the document space X is some type of high-dimensional space, and the classes are human defined for the needs of an application. We are given a training set D of labelled documents <d,c>, where <d, c> ∈ X * C. Using a learning method or learning algorithm, we then wish to learn a Classifier or classification function g that maps documents to classes:

g : X -> C. This type of learning is called supervised learning because a supervisor (the human who defines the classes and labels training documents) serves as a teacher directing the learning process.

Naive Bayes text classification algorithm:-

The first supervised learning method introduced is multinomial Naïve Bayes or multinomial NB model, a probabilistic learning method.

Here Bug reports are instances, words are features and severity label indicates class ,c. For a bug report d and a class c, $P(c|d)=\frac{P(d|c)P(c)}{P(d)}$ or P(d|c).P(c) since P(d) is negligible, it can be neglected.

Furthermore, the bug reports are represented as "bags of words": each bug report consists of words drawn from vocabulary $V_n = \{w_1, \dots, w_n\}$ The Naive Bayes assumption is that the words are independently and identically distributed: the probability of each word is independent of its context and position in the document. To find most likely class i.e. severity label for prioritizing the bug, use the equation (4),

$$C_{MAP} = argmax_{c \in C} P(d|c)P(c) \quad (4)$$

Assuming the conditional independence between the term t in bug report and class c, the probability of terms with classes is computed as (5)

$$P(t_1, t_2 \dots t_n|c) = P(t_1|c).P(t_2|c).P(t_3|c)*..P(t_n|c) \quad (5)$$

To determine the class/ the severity label of bug report j using Naive Bayes, denoted by C_{NB} , find the maximum prior probability for each of the posterior probability of terms in bug report using the equation (6),

$$C_{NB} = \text{argmax}_{c \in C} P(c_j) \Pi_{t \in V} P(t_j|c) \qquad (6)$$

Learning the Multinomial Naive Bayes Model:-
Here, it is a multi-class, single label classification problem, hence
Prior Probability can be estimated using equation (7) $P(c) = \frac{N_c}{N}$ (7)

Nc = Number of documents labeled as c , N=total no. of docs.
Conditional probability, using equation (8) defined as fraction of times word appears among all words in document of class c indicated by w and t_k denotes term in vocabulary V,

$$P(t_k|c) = \frac{Count\ (t_k,c)}{\sum_{t \in V} count\ (w,c)} \qquad (8)$$

After Laplace smoothing, $P(t_k|c) = \frac{Count(t_k,c) + 1}{\sum_{t \in V} count(w,c) + |V|}$

Best class found by equation (9), $C_{NB} = \text{argmax}_{c \in C} P(t_1, t_2, \ldots)|c) * P(c)$ (9)

Algorithm for Classifying a New Bug:-

```
Input:  A new vocabulary v(words in description of new bug)
        A fixed set of classes C = { $c_1$, $c_2$, .. , $c_j$ }
        A training set of labeled set ($v_1$,$c_1$),....,($v_n$,cj)
Output: a learned classifier µ:v → c
Method:
Extract vocabulary from training bug dataset
For each bug report in the cluster set
   i. Find prior probability (p) for each term in the vocabulary
   ii. Find conditional probability cp = p + cp
Return minimum value of cp and assign class label to new vocabulary set.
```

K-nearest neighbor algorithm (KNN):-
K-nearest neighbour algorithm (KNN) is part of supervised learning that has been used in many applications in the field of data mining, statistical pattern recognition and many others.
KNN is a method for classifying objects based on closest training examples in the feature space.
K is always a positive integer. The neighbours are taken from a set of objects for which the correct classification is known. The textual similarity between new bug report and clustered bug reports using Euclidian distance measures.
The algorithm on how to compute the K-nearest neighbours is as follows:
1) Determine the parameter K, number of nearest neighbours beforehand.
2) Calculate the distance between the bug instance and all the training samples.
3) Sort the distances for all the training samples and determine the nearest neighbour based on the K-th minimum distance.
4) Since this is supervised learning, get all the Categories of your training data for the sorted value which fall under K.
To classify a class-unknown document X, the k-Nearest Neighbour classifier algorithm ranks the document's neighbours among the training document vectors, and uses the class labels of the k most similar neighbours to predict the class of the new document. The classes of these neighbors are weighted using the similarity of each neighbour to X, where similarity is measured by Euclidean distance.

Result and Discussion:--
The results of classification were obtained both with and without performing clustering before applying classification on the data. In case of classification of previously clustered data, the bug data was first clustered on the basis of problem title. Each cluster thus obtained was then classified separately based on the severity or component attribute of the bugs. The result comprised of details such as 'total number of instances' in each cluster,' number of instances took as test data' out of total number of instances, percentage of correctly classified instances' etc. The precision and recall of each cluster was computed.

The performance measures viz accuracy, precision and recall for severity based classification for all the combinations of clustering and classification algorithms used are shown in fig 3, 4 ,5 and 6.

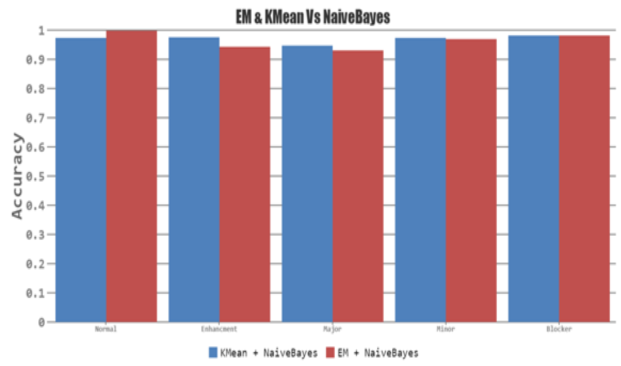

Fig 3:- Accuracy of Clustering algorithms with NB

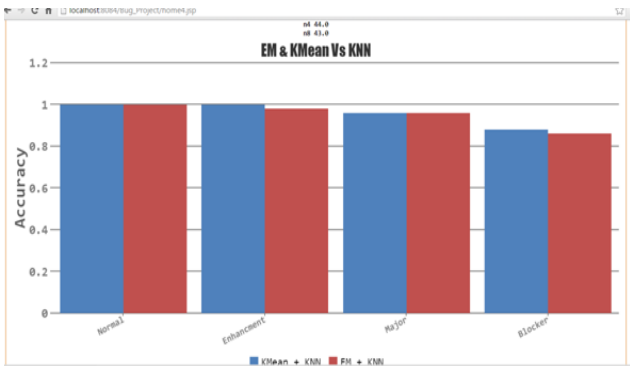

Fig 4:- Accuracy of Clustering algorithms with KNN

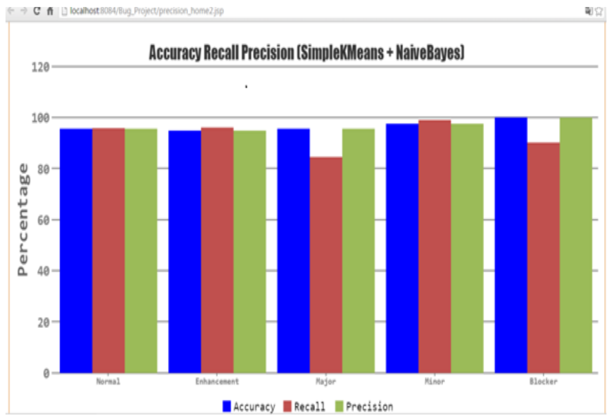

Fig 5:- Performance measures of Kmeans with Naïve Bayes

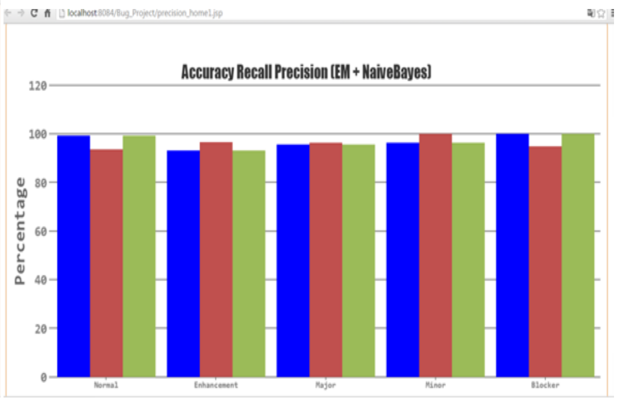

Fig 6:- Performance measures of EM with Naïve Bayes

The graph showing comparison of classification algorithms on accuracy measure with clustering and without clustering is shown below in fig 7.

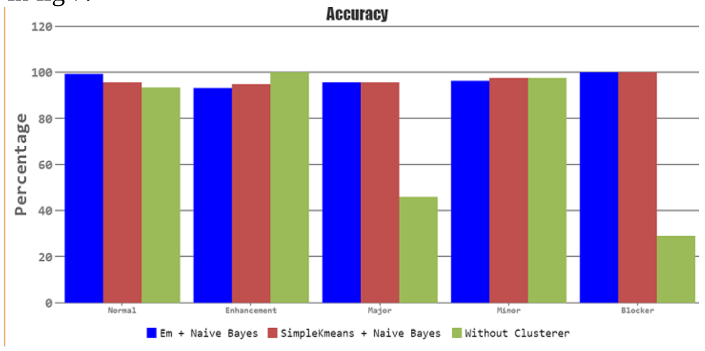

Fig 7:- Comparison of classification with and without clustering before classification

Conclusion:-

Bug Triaging is an indispensable part of software testing. It helps to identify the potential developer to resolve the bug. In addition to prediction of developer, it is necessary to predict the severity as it specifies how soon the bug needs to be resolved. In this study, bug triage system is improved with bug severity prediction. Data mining techniques such as Information retrieval, classification and clustering are used to implement the system. Researchers have found that when clustering technique is used as pre-processing step before any other machine learning techniques are applied, it can increase the performance of the algorithm. Results obtained here shows that Kmeans clustering performs better than Expectation Maximization when used with Naive Bayes classifier. The classification without clustering experiment result showed poor performance when compared to clustering prior classification.

The approach used here can be extended to other classification and clustering algorithms with other types of data. Software bug data from open source software repository eclipse is undertaken for this work.

References:-

1. **Geunseok Yang,Tao Zhang and Byungjeong Lee(2014),** “Towards Semi-automatic Bug Triage and Severity Prediction Based on Topic Model and Multi-Feature of Bug Reports,” Software and Applications Conference, IEEE.
2. **W. Zou, Y. Hu, J. Xuan, and H. Jiang(2012),** “Towards training set reduction for bug triage, ” IEEE Int. Computer Soft.Appl.
3. **D.Cubranic, G.C.Murphy,(2004),** “Automatic bug triage using text categorization ” , Intl. Conf. Software Engineering and Knowledge Engineering.
4. **A. Lamkan, S. Demeyer, E. Giger and B. Goethals(2010)**, “Predicting the Severity of a Reported Bug”, Proc. of IEEE Working Conference on Mining Software Repositories.
5. **A. Lamkan, S. Demeyer, E. Giger and B. Goethals(2011)**, “Comparing Mining Algorithms for Predicting the Severity of a Reported Bug”, Proc. of European Conference on Software Maintenance and Reengineering.
6. **Y. Tian, D. Lo and C. Sun(2012),** “Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction”, Proc. of Working Conference on Reverse Engineering.
7. **T. Menzies and A. Marcus(2008),** “Automated Severity Assessment of Software Defect Reports”, Proc. of of IEEE International Conference on Software Maintenance.
8. **P. Bhattacharya, M. Iliofotou, I. Neamtiu and M. Faloutsos(2012),** “Graph-Based Analysis and Prediction for Software Evolution”, Proc. of International Conference on Software Engineering.
9. **Ricardo Vilalta, Murali-Krishna Achari, and Christoph F. Eick(2003),** “Class Decomposition Via Clustering: A New Framework For Low-Variance Classifiers”, Proc. of International Conference on data mining.
10. **Bugzilla**, http://www.bugzilla.org/