International Journal of Soft Computing 5 (3): 109-115, 2010
ISSN: 1816-9503

Automation of Software Artifacts Classification

Yuhanis Yusof and Qusai Hussein Ramadan
College of Arts and Sciences Information Technology Building,
University of Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia

Abstract: With the huge increase of software functionalities, sizes and application domain, the difficulty of categorizing and classifying software packages for reuse and maintenance purposes is on demand. This research includes the use of structure information contained in source code programs to automate program classification. Three software metrics namely; Line Of Codes (LOC), McCabe's Cyclomatic Complexity (MVG) and Weighted Methods per Class (WMC1) are used to automatically classify software packages into the appropriate application domains. The undertaken experiment using Instance-Based (IBK) algorithm generates classification accuracy as high as 74.82%. This indicates that further exploration on the use of structure information in the domain of software classification should be continued.

Key words: Software classification, software metrics, machine learning algorithms, instance-based algorithm, domain, LOC

INTRODUCTION

Software systems consist of a variety of artifacts such as specifications, design diagrams and descriptions, source code, test cases and documentation. Automation of these artifacts became one of the most important topics in software engineering area (Kawaguchi *et al*., 2003). This is because of the new problems occurred upon constructing of software archives. For instance in 2002, the SourceForge.net had over 70,000 registered software (Kawaguchi *et al*., 2003). As this repository receives input (i.e., software files) from various developers whom have various backgrounds, categorizing the packages relies on the text input provided and/or contained in them. One issue which arises from such situation is to find a way to enhance the search process in the software's archive. So there is a need for alternative method in software classification (Kawaguchi *et al*., 2003).

There is a difficulty in organizing the software files/classes into specific categories by just depending on the software file/class name (Kawaguchi *et al*., 2004). The actual challenge will arise if we go through the software code to specify the category type of the software manually because it's not easy to understand other programmer code exactly if it was built in complex way and without code specification (O'Halloran and Smith, 1998). The main cause to this problem is code specification. Sometimes, the specifications are not available or they are incomplete (Ali, 2006). This problem has a significant impact on software repositories and open source web sites; it will be a challenge to manually classify the uploaded files into appropriate category.

Existing approaches that adopts manual classification require more time and high level of software understanding and classification polices (Kawaguchi *et al*., 2003). This is because of the large size code embedded in software and the ambiguous code specification. Hence the developers need to spend their time and efforts to know the specific category that software belongs to (Fuchs, 1992). The process of organizing the files may be carried out by a number of employees and may resulted on unreliable classification.

This research tries to overcome such problem by introducing the use of structure-based descriptors (software metrics) that can be extracted from software files. The undertaken study uses software metrics as feature set of a software file. Existing study have focuses on the use of software metrics to improve software quality (Nagappan, 2004). Furthermore, software metrics play important role in software management by measuring software development process. It is reported that software metrics save the required efforts of software understanding especially for huge size software with high level of complexity (Gray and MacDonell, 1997).

The undertaken research is a preliminary experiment to investigate structure-based descriptors in particular the software metrics in determining the domain of a software file. Furthermore it is to identify

Corresponding Author: Yuhanis Yusof, College of Arts and Sciences Information Technology Building, University Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia

the best machine learning classifier that is to be used for software classification. This is achieved by defining an automatic software classification strategy that operates on Java Games packages and will determine if a file falls into the Puzzle or Board category.

MATERIALS AND METHODS

Software classification helps to order software components in one repository into specific groups. With this, similar components can be grouped in the same category depending on the functionality of these components (Merkl, 1995). Various software metrics can be used to describe the source code characteristics. Software metrics are used to quantitatively map a set of numerical values such as number of code lines and number of the methods per a class. Nevertheless, not all of the metrics provide the same classification power (Vivanco and Pizzi, 2003).

Every day there are many software source code uploaded on the internet, so the web now days contains different sets of source codes which can be reached by using source code web sites such as Source Forge, Plant Source Code and Free code (Korvetz *et al*., 2003). Software classification plays a role in the field of software reusability (Poulin and Yglesias, 1993). For instance 70% of software development budgets are spent on software maintenance, so the need of classifying the software to a particular type became an important topic to help in making accurate decision on code changes (Phillips and Black, 2005).

Code metric histograms and genetic algorithms have been used to develop the Author Identification Software that identifies the original researcher (Lange and Mancoridis, 2007). About 14 variables have been specified such as the way of typing the name of the functions and code specifications. Also software metrics were used to portray specified variables into histograms and later studied the histograms to identify the researcher (Lange and Mancoridis, 2007).

In SourceForge, it is learned that it contains software categories like Database, Network, Security and Graphics. To classify source code programs into categories, existing software classification approach depends on the following features; Comments and specification, source code variables and Readme files (Korvetz *et al*., 2003). MUDABlue (Kawaguchi *et al*., 2004) is a tool that automatically categorized software system. Categorizing contents of arge software archives is recognized as essential pivot for an effective reusable software archive (Kawaguchi *et al*., 2004). Another research done in software classification is discussed in (Jianhui, 2008). The process of classifying malicious samples into categories are divided into three phases: Analyzing an object, Represent and store the knowledge and self learning from the new objects (Jianhui, 2008).

Software metrics can be used to measure the characteristics of a software system. There are many models of identifying software metrics like functional point analysis, neural networks, fuzzy logic system and others. Examples of software metric outcomes are software size (number of lines of source code) and logical complexity of the system (Chan and Wong, 2005). Component classification is recognized as one of the software reusable procedures.

It is aimed to separate the variety kinds of software components and store them according to software reusable attributes. Some of software metrics can be used to measure the quality characteristics of reusable software component (Lai and Yang, 1998).

Recently, there is a large set of software metrics tools and the majority of these metric tools is to extract software metric form Java, C/C++, UML or other programming languages (Lincke *et al*., 2008). For example, the VizzAlalyzer is a system for reverse engineering which it reads the software source code and other code specifications to perform a set of quality analysis (Panas *et al*., 2007).

OOMeter is another software metrics tool developed by Alghamdi and deals with Java, C# source and UML source code to provide a collection of software metrics such as complexity, cohesion, coupling and Line Of Code (LOC) (Alghamdi *et al*., 2005).

CCCC (C and C+ Code Counter) is software metric which parses various programming language source code and generates accessible HTML reports on various measurements of the code processed (Lincke *et al*., 2008). The generated reports contain various types of tables identifying the module in the submitted source code file and covers: Measure the procedural volume and complexity of each module and its function, Measure the relationship numbers and types related to each module (Inheritance), Identify which part of the submitted file failed to parse and export a summary report over the whole body of code processed.

The undertaken study is based on the CCCC which extracts software metrics from a Java code file. This study focuses on 3 attributes:

Line Of Codes (LOC): This attribute represent total number of counting the non-blank, non-comment lines. Preprocessor lines are treated as blank. Class and function declarations are counted and declarations of global data are ignored.

McCabe's cyclomatic complexity (MVG): Measure the count of linearly independent paths through a flow of control graph this can be found by counting language keywords and operators which affect on source code complexity.

Weighted Methods per Class (WMC1): Represent the total number of the functions (procedure) that have been involved in the parsed file.

This three attributes extracted by the CCCC are statics attributes which approximately will fall in the same range, regardless of the developer or programming language used in developing a software.

Machine learning is one of the main components of intelligent information systems that have the ability to improve the performance in particular domain depending on the experience and enable the compact generalization which inferred from large data sets to be applied as knowledge in different ways such as automatic process in expert systems (Holmes *et al*., 1994; Markov and Russell, 2006).

In this study, WEKA is used as a tool that provides various machine learning algorithms which it is already implemented for predicting nominal and numeric quantities (Witten and Frank, 2005). This study focuses on the following algorithms/classifiers: Multilayer Perceptron, BayesNet, IBK and J48. These classifiers were used to identify and analyze the statistical metrics in order to discover the pattern of the metrics and hence identify the appropriate model that can be used to classify the software files. An example of the use of WEKA is in web document classification that aims to classify new documents according to their topics and intelligent web browser (Markov and Russell, 2006).

There have been many research that use machine learning classifiers to test their classification hypotheses. For example, Reference (Sebastini, 2002) used the classifiers in text categorization. Based on the experiments performed, Reference (Sebastini, 2002) found that the perceptron classifier has produced a good classification compared to decision tree and decision rules classifiers. The use of Multilayer Perceptron can be seen in multimedia document retrieval field. The classifier provide a good experiment results (96% of accuracy) during the training and testing stages to discrimination between speech signals and music signals (Khan *et al*., 2004). Another researchon sponsored search conclude that MultilayerPerceptron learning algorithm based on a set of feature provide a suitable framework for the sponsored search task (Ciaramita *et al*., 2008). From another view, MultilayerPerceptron classifier has been diminished the problem of multiclass classification using a small imbalanced database fluorescence in situ hybridization and contributed to accuracy improvement (Lerner *et al*., 2007).

A comparative study of selected classifiers with classification accuracy in user profiling have been proposed (Cufoglu *et al*., 2009). The comparative results noted that the NBTree classifier is the best classifier on related information. Also, they noted that performance of the SimpleCart and J48 classifiers are approximately similar to NBTree but is different from NB, SMO, IB1 and Id3 classifiers (Cufoglu *et al*., 2009).

Figure 1 shown the proposed architecture for the undertaken study. The architecture represents a road map that includes the requirements (data and tools) to achieve the study objectives.

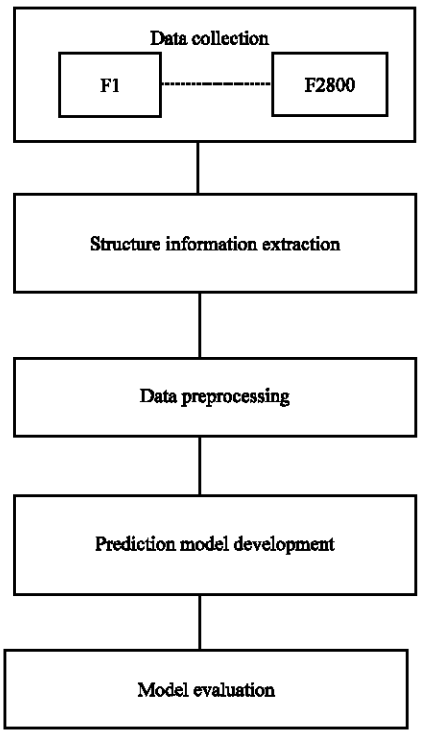

Fig. 1: Automatic software artifact classification architecture

Data collection: This study uses Java source code files as currently there is a huge collection of Java programs that can be downloading from the internet. Many research have been performed on Java files to prove the identified hypotheses. A total of 2800 game files have been downloaded from the Games and Entertainment category in Sourceforge.net. These files have been pre-categorized into two application domains; Puzzle and Board.

Structure information extraction: Software metric extractor (i.e., CCCC) is later used to identify structure information of the source code files. The identified metrics is stored in an Excel sheet. CCCC helps in analyzing Java source files and extracting the attributes that the study focuses on (Lincke *et al*., 2008). Table 1 shown examples of extracted metrics.

Data preprocessing: Upon obtaining software metrics of the source code files, data preprocessing is performed. This includes removing interface classes-files that contain 0 for LOC, MVG and WMC. Data in Table 2 are examples of records that are discarded. The preprocess data is later learned to have the average values of the extracted metric as in Fig. 2. Based on Fig. 2 it is found that the average values for puzzle files are LOC (85), MVG (9) and WMC1 (9). On the other hand, average values for Board files are LOC (175), MVG (26) and WMC1 (15). Comparing between the results it is learned that average values for Board files are greater than average values of the puzzle.

Prediction model development: In this research, the extracted metrics are then fed into a machine learning tool (i.e., WEKA (Menkovski *et al*., 2008) in order to discover patterns of structure information in the games source files. Two types of model development method are performed; percentage split and cross validation. The percentage split experiment would require a division of data into two sets; training and testing, while the latter (cross validation) method would use all of the data for training purposes.

Classification of game files into categories is performed by applying the following algorithms/ classifiers: Multilayer Perceptron, BayesNet, IBK and J48. Performance of the classifiers is evaluated using precision and recall measurement. Precision and recall are important to consider because it is used as a measurement to evaluate classification accuracy. Precision is the proportion of relevant instances in the results returned. For instance, if the precision is 0.72

Table 1: Examples of extracted metrics

File	LOC	MVG	WMC1	Target
1	8	0	0	Puzzle
547	22	1	1	Puzzle
1660	96	0	21	Board
2798	10252	1024	225	Board
2800	11186	182	85	Board

Table 2: Examples of metrics in removed files

LOC	MVG	WMC1	Target
3	0	3	Puzzle
17	0	2	Puzzle
19	2	5	Board
8	0	8	Board

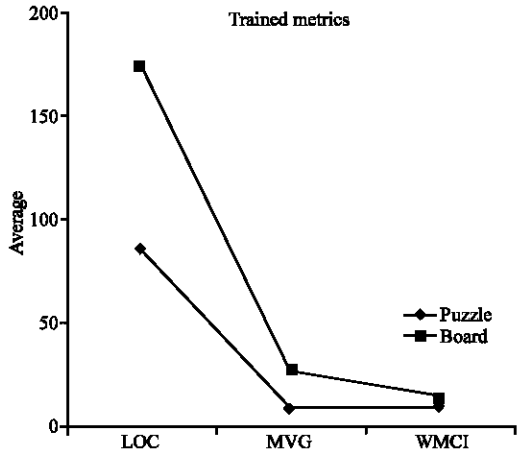

Fig. 2: Average values of software metrics for puzzle and board

then it means that 72% of returned instances were relevant. Recall values represent the ratio of relevant instances found to the total of relevant instances.

RESULTS AND DISCUSSION

Training of each classifier was implemented for 3 times using different number of folds; 5, 10 and 15. Number of folds represents how many instances will be taken from the dataset at each time to train the data. For example, if the fold is equal to 10 this mean that the algorithm will choose 10 by 10 instances until achieve the last instance. Given the size of the dataset is 2800 and then if fold is set to 10, the chosen algorithm will repeat the training operation for 280 times.

Data shown in Table 3 shows the outcomes of classifying data using cross validation. The table consists of 6 columns; the first column represents the algorithm that has been applied, the second column shows the folds values, the third and the fourth columns represent the correctly and the incorrectly predicted instances while the fifth and the sixth columns are the precision and recall. As shown in Table 4, the training for each classifier was

Table 3: Prediction accuracy using cross validation

Classifier	Folds	Correctly ins (%)	Incorrectly ins (%)	Precision P	Precision B	Recall P	Recall B
Multilayer perceptron	5	1640 (58.5714)	1160 (41.4286)	0.568	0.615	0.714	0.458
	10	1622 (57.9286)	1178 (42.0714)	0.563	0.608	0.711	0.448
	15	1630 (58.2143)	1170 (41.7857)	0.565	0.613	0.719	0.446
Bayes net	5	1721 (61.4643)	1079 (38.5357)	0.591	0.655	0.744	0.485
	10	1734 (61.9286)	1066 (38.0714)	0.596	0.659	0.744	0.495
	15	1734 (61.9286)	1066 (38.0714)	0.596	0.658	0.742	0.496
IBK							
KNN 1	5	2086 (74.5)	714 (25.5)	0.719	0.778	0.805	0.685
	10	2095 (74.8214)	705 (25.1786)	0.723	0.780	0.806	0.691
	15	2090 (74.6429)	710 (25.3571)	0.720	0.780	0.807	0.686
KNN 4	5	1945 (69.4643)	855 (30.5357)	0.651	0.774	0.840	0.549
	10	1987 (70.9643)	813 (29.0357)	0.664	0.791	0.849	0.570
	15	1988 (71)	812 (29)	0.664	0.791	0.849	0.571
KNN 7	5	1846 (65.9286)	954 (34.0714)	0.643	0.679	0.715	0.604
	10	1840 (65.7143)	960 (34.2857)	0.638	0.682	0.726	0.588
	15	1834 (65.5)	966 (34.5)	0.633	0.686	0.739	0.570
J48	5	1780 (63.5714)	1020 (36.4286)	0.618	0.659	0.710	0.561
	10	1855 (66.25)	945 (33.75)	0.655	0.671	0.686	0.639
	15	1812 (64.7143)	988 (35.2857)	0.642	0.653	0.665	0.629

Table 4: Prediction accuracy using percentage split

Classifier	Correctly ins (%)	Incorrectly ins (%)	Precision P	Precision B	Recall P	Recall B
Multilayer perceptron	499 (59.4048)	341 (40.5952)	0.599	0.591	0.489	0.693
Bayes net	521 (62.0238)	319 (37.9762)	0.609	0.630	0.602	0.637
IBK						
KNN 1	621 (73.9286)	219 (26.0714)	0.698	0.792	0.813	0.670
KNN 4	576 (68.5714)	264 (31.4286)	0.635	0.773	0.828	0.552
KNN 7	562 (66.9048)	278 (33.0952)	0.638	0.709	0.735	0.607
J48	524 (62.3810)	316 (37.6190)	0.689	0.598	0.408	0.827

implemented for 3 times 3-3 different fold values. The highest classification accuracy was obtained using the IBK (i.e., KNN 1) classifier. By using 10 as the numbers of fold, the accuracy reached 74.82%.

The next experiment is undertaken by splitting a dataset into training and testing sets. We use 70% of data as training, while the balance (30%) as testing. Prediction model is obtained by training 70% of the data and is later verified using the testing dataset. Prediction accuracy of the testing dataset is shown in Table 4.

Upon completing the training process, the obtained classifier model must be verified and validated. The evaluation process is undertaken on an independent testing dataset. The testing dataset consists of 28 Java files of Puzzle and Board games. The metrics for these files were also extracted using CCCC tool. Using the obtained IBK model (i.e. IBK- KNN 1), prediction of the files in this independent dataset is made using WEKA.

Identifying the relations of software metrics and the software categories is one of the study issues. For this purpose, the average value for each attribute in the unclassified dataset is shown Fig. 3. Based on Fig. 3 it is found that the average values for Puzzle programs are 33 for LOC and 2 for MVG and WMC1, respectively. As for the Board programs, the average values are 115, 16 and 13 for the LOC, MVG and WM1, respectively.

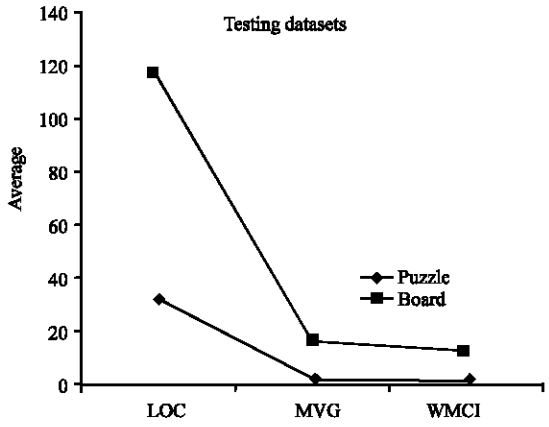

Fig. 3: Average values of software metrics for puzzle and board-independent dataset

Comparing between the two results, it is noted that the average values for Board programs are greater than average values of Puzzle programs. Such a result is similar to the one obtained during training. Refer to Fig. 2.

The IBK model is later used to predict the category for the new unclassified instances (testing dataset). The results shows that 64% of the Puzzle instances and 57%

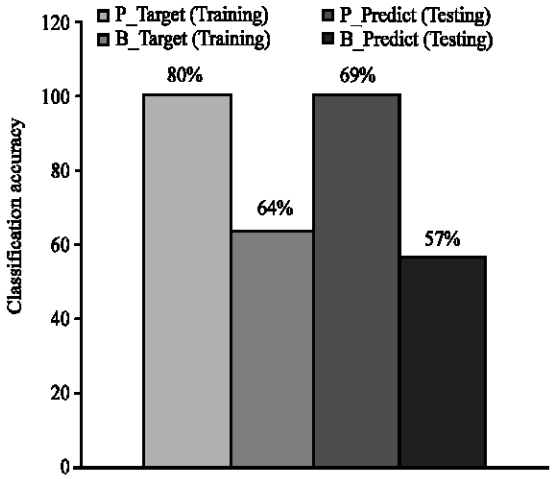

Fig. 4: Classification accuracy of training and independent testing dataset

of the Board instances have been classified correctly; shows Fig. 4 which illustrates the classification accuracy on training dataset and testing dataset. Comparing between column 1 and column 2 in Fig. 4, it is noted that the classification accuracy for Puzzle files on training dataset is 80% while on the tested dataset is 64%. On the other hand comparing between column 3 and column 4 it is found that the classification accuracy for Board files on the training dataset is 69% while on the tested dataset is 57%. As instances included in the testing dataset have not been seen by the classifier, an accuracy that is greater than 50% is promising.

CONCLUSION

In this study, software metrics representing structure information contained in software artifacts is utilized in classifying artifacts into application domains. Experimental results show that classification accuracy as high as 74.82% is obtained using instance-based machine learning algorithm. As structure information may benefit future software development, the utilization of software metrics in such classification is promising and may aid existing classification approaches of semantic-based. We have also identified relations between software metrics and application domains which is one of the study issues. Using existing datasets, it is learned that the average values for metrics extracted from Board programs are greater than the ones in puzzle programs.

Further research needs to be done to improve the classification accuracy of IBK model and to prepare the mechanism for more real world use. There are various ways to improve the classification accuracy and one of them is to expand the size and diversity of the data. CCCC tool can measure over than 10 attributes that may be used in software classification such as Depth of Inheritance Tree, Number of Children and Coupling between objects. The undertaken study focuses on three attributes (LOC, MVG and WMC1) and it is recommended to increase the number of measured attributes. Also in this study, the number of trained classifiers is limited so it is important to implement other classification strategies such as C4.5 decision tree.

REFERENCES

Alghamdi, J.S., R.A. Rufai and S.M. Khan, 2005. OOMeter: A software quality assurance tool. Proceedings of the Ninth European Conference on Software Maintenance and Reengineering, March 21-23, King Fahd University of Petroleum and Minerals, pp: 190-191.

Ali, S., 2006. AutoAbstract: Problem statement and hypothetical solutions. Testing: Academic and Industrial Conference-Practice and Research Techniques, 29-31 Aug. Windsor, pp: 75-80.

Chan, V.K.Y. and W.E. Wong, 2005. Optimizing and simplifying software metric models constructed using maximum likelihood methods. 29th Annual International Computer Software and Applications Conference, July 26-28, Edinburgh, pp: 65-70.

Ciaramita, M., V. Murdock and V. Plachouras, 2008. Online learning from click data for sponsored search Proceedings of the 17th International Conference on World Wide Web, (ICWWW'08), Beijing, China, pp: 227-236.

Cufoglu, A., M. Lohi and K. Madani, 2009. A comparative study of selected classifiers with classification accuracy in user profiling. Proceedings of the WRI World Congress on Computer Science and Information Engineering, March 31-April 2, Los Angeles, CA, pp: 708-712.

Fuchs, N.E., 1992. Specifications are (preferably) executable. Software Eng. J., 7: 323-334.

Gray, A.R. and S.G. MacDonell, 1997. Applications of fuzzy logic to software metric models for development effort estimation. Proceerings of the Annual Meeting of the North American Fuzzy Information Processing Society, Sept. 21- 24 Syracuse, New York, USA., pp: 394-399.

Holmes, G., A. Donkin and I.H. Witten, 1994. WEKA: A machine learning workbench. Proceedings of the 1994 2nd Australian and New Zealand Conference on Intelligent Information Systems, Nov. 29-Dec. 2, IEEE Computer Society Press, pp: 357-361.

Jianhui, L., 2008. On malicious software classification. International Symposium on Intelligent Information Technology Application Workshops, 21-22 Dec, Inf. Technol., Hubei Univ. of Police, Wuhan, pp: 368-371.

Kawaguchi, S., P.K. Garg, M. Matsushita and K. Inoue, 2003. Automatic categorization algorithm for evolvable software archive. Proceedings of the 6th International Workshop on Principles of Software Evolution, (IWPSE'03), IEEE Computer Society Washington, DC, USA., pp: 195-195.

Kawaguchi, S., P.K. Garg, M. Matsushita and K. Inoue, 2004. MUDABlue: An automatic categorization system for open source repositories. Proceedings of the 11th Asia-Pacific Software Engineering Conference, Nov. 30-Dec. 03, Busan, Korea, pp: 184-193.

Khan, M.K.S., Al-Khatib, W.G. and M. Moinuddin, 2004. Automatic classification of speech and music using neural networks. Proceedings of the 2nd ACM International Workshop on Multimedia Databases, (IWMD'04), Washington, DC, USA., pp: 94-99.

Korvetz, R., S. Ugurel and C.L. Giles, 2003. Classification of source code archive. Proceedings of 26th Annual International ACM SIGIR Conference On Research and Development in Information Retrieval, (RDIR'2003), ACM, pp: 425-426.

Lai, S. and C.C. Yang, 1998. A software metric combination model for software reuse. Procedings of the Asia Pacific Software Engineering Conference, Dec. 2-4, Taipei, Taiwan, pp: 70-77.

Lange, R.C. and S. Mancoridis, 2007. Using code metric histograms and genetic algorithms to perform author identification for software forensics. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, (GEC'2007), ACM, pp: 2082-2089.

Lerner, B., J. Yeshaya and L. Koushnir, 2007. On the classification of a small imbalanced cytogenetic image database. IEEE/ACM Trans. Computational Biol. Bioinformatics, 4: 204-215.

Lincke, R., J. Lundberg and W. Lowe, 2008. Comparing software metrics tools. 2008 international symposium on Software Testing and Analysis, (SSTA'2008), ACM, pp: 131-142.

Markov, Z. and I. Russell, 2006. An introduction to the weka data mining system. 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education, (ITCSE'2006), ACM, pp: 367-368.

Menkovski, V., I.T. Christou and S. Efremidis, 2008. Oblique decision trees using embedded support vector machines in classifier ensembles. Proceedings of the Conference on Cybernetic Intelligent Systems, Sept. 9-10, London, pp: 1-6.

Merkl, D., 1995. Content-based software classification by self-organization. Proceedings of the IEEE International Conference on Neural Networks, Nov/Dec 1995, Perth, WA, Australia, pp: 1086-1091.

Nagappan, N., 2004. Toward a software testing and reliability early warning metric suite. Proceedings of the 26th International Conference on Software Engineering, (ICSE'2004), IEEE Computer Society, pp: 60-62.

O'Halloran, C. and A. Smith, 1998. Don't verify, abstract!. Proceedings of the 13th IEEE International Conference on Automated Software Engineering, Oct. 13-16, Honolulu, Hawaii, pp: 53-62.

Panas, T., D. Quinlan, R. Vuduc and Lawrence Livermore Nat. Lab., Livermore, 2007. Tool support for inspecting the code quality of HPC applications. Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing Applications, May 20-26, Minneapolis, MN, pp: 2-2.

Phillips, N. and S. Black, 2005. Distinguish between learning, growth and evolution. Proceedings of the IEEE International Workshop on Software Evolution, Sept. 26, London, South Bank Univeersity, UK., pp: 49-52.

Poulin, J.S. and K.P. Yglesias, 1993. Experiences with a faceted classification scheme in a large reusable software library (RSL). Proceedings of the 17th Annual International Computer Software and Applications Conference, Nov. 1-5, Phoenix, AZ, USA., pp: 90-99.

Sebastini, F., 2002. Machine learning in automated text categorization. ACM Comput. Surveys, 34: 1-47.

Vivanco, R.A. and N.J. Pizzi, 2003. Identifying effective software metrics using genetic algorithms. International Conference on Electrical and Computer Engineering, 4-7 May Biodiagnostics, Nat. Res. Council of Canada, Winnipeg, Man., Canada, pp: 1305-1308.

Witten, I.H. and E. Frank, 2005. Data Mining: Practical Machine Learning Tools and Techniques. 2nd Edn., Morgan Kaufmann, San Francisco, ISBN: 0120884070, pp: 525.