International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 3, June 2019, pp. 2041~2048
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i3.pp2041-2048

Reengineering framework for open source software using decision tree approach

Jaswinder Singh[1], Kanwalvir Singh[2], Jaiteg Singh[3]
[1]Department of Computer Application, IKG Punjab Technical University, India
[2]Depatment of Computer Science & Engineering, Baba Banda Singh Bahadur Engineering College, India
[3]Departement of Computer Application, Chitkara University, India

Article Info

Article history:

Received Jan 9, 2018
Revised Nov 27, 2018
Accepted Dec 20, 2018

Keywords:

Complexity metric
Decision tree
Maintainance
Reengineering
Software engineering

ABSTRACT

A Software engineering is an approach to software development. Once software gets developed and delivered, it needs maintenance. Changes in software incur due to new requirements of the end-user, identification of bug in software or failure to achieve system objective. It has been observed that successive maintenance in the developed software reduces software quality and degrades the performance of software system. Reengineering is an approach of retaining the software quality and improving maintainability of the software system. But the question arises "when to reengineer the software". The paper proposed a framework for software reengineering process using decision tree approach which helps decision makers to decide whether to maintain or reengineer the software systems.

Corresponding Author:

Jaswinder Singh
Department of Computer Application,
IKG Punjab Technical University,
Kapurthala, India.
Email: jaswinder_luthra@yahoo.co.in

1. INTRODUCTION

Changes in software due to user requirements, faults or due to technology change are very frequent. The need of customer or client may get changed as per business requirements. New changes may introduce new challenges to the developer. To adapt these changing requirements software needs to maintain again and again. According to Lehman [1] frequent changes increase the complexity of software. Summerville [2] stated that aging of program results in increase in maintenance cost which further degrade the program structure and it becomes harder to understand and change. There are various complications in upgrading the legacy systems also [3]. Reengineering is defined as reconstitution of an existing system [4], system changing activity [5], reconstruction or reworking on part or all of the legacy system [6] and aimed to improve the quality of software [7].

Very important and crucial decision is whether a software must be further maintained or whether it must be reengineered. Reengineering aimed to enhance the quality of software and the increasing the software maintainability [7]. Another question is what metric or parameter can be used for this decision making.

This framework describes a generic process to identify reengineering requirements based on the software metrics. As reengineering is making changes of software at design level, so making right metric choice is very important. Reengineering is also important for many popular software Metrics exist that aimed for understanding and improving the quality and reducing the complexity of the software [8]-[12]. Ck Metric [13] is well known complexity metric suit in the domain of object-oriented paradigms for measuring software quality. The basic set of Metrics includes Weighted Methods per Class (WMC), coupling Between Object

classes (CBO), Depth of the Inheritance Tree (DIT), Number of Children (NOC) and Response for a Class (RFC). In this paper, complexity of software is analyzed statistically using CK metric and then decision tree is used to for deciding the reengineering requirements of software systems.

2. RELATED WORK

The concept of reengineering is having very strong research base but still, the use of decision tree approach to support the decision of reengineering and maintenance is limited. The existing studies provide software reengineering framework but in different domains. Rehma *et al*. [14] have suggested REXDES, an Expert based Decision support framework for software Reengineering. This framework uses RASIC components that use expert based decision system to assist decision makers in reengineering related decisions. But this framework is having theoretical existence only, it lacks practical implementations. Sood [15] proposed a metric framework for making the decision among reengineering and maintenance. He calculates Reengineering requirement cost with 'defect cost', 'Fault cost' as reengineering metrics, but his approach ignores use of functional independence in reengineering decision making. Many reengineering models have also been given by various authors [16]-[22].

3. METHODOLOGY

Dataset consist of open source JAVA projects of different sizes. Diomidis [23] derived CKJM (Chidamber and Kemerer Java Metrics) tool to apply CK complexity metric. Proposed research used CKJM ver-1.9 [24] for java code analysis. CK metrics are measured for all the java projects. For classes in the java project, statistical methods (measuring average complexity) are applied for basic set of CK metrics. For all the Java projects, average of six set of complexities for each class has been computed and the sum of all average complexities computer per class is the Total Average Complexity of Modules of the project (TACMP) [25]. Total mean complexity (TMC) is average complexity of all project's TACMP. Decision making is based on the prediction with the help of decision tree. Rapid Miner studio ver- 7.1 [26] is used to model the decision tree for data sets. TACMP and size (LOC) are used in Decision Tree as attributes to predict the required outcome. Decision Trees are predictive in nature [27], [28]. Prediction is based on the rules by dividing data into groups. A training data set is generated consisting of 15 Java projects. Training data set contains SIZE (LOC), Total Average Complexity of Modules of the project and category as attributes. The category attribute is a label and is measured on the basis of TMC and size. Another Model dataset contains 5 Java projects on which predictions will be applied.

Here proposed algorithm is presented which calculate the complexity metric for open source software and use Rapid Miner tool to predict the maintenance and reengineering requirement.

Input: Open source JAVA projects,

Output: Statistical Complexity Measures and Reengineering predictions

1) Consider 20 Open source JAVA projects
2) Apply CKJM Metric tool to calculate Basic Metric set of CK metric for each module of every single JAVA project
3) Perform statistical analysis to calculate total average complexity of modules of project [TACMP].
4) Calculate total Mean complexity [TMC] for all projects
5) Analyze the correlation between size and total average complexity
6) Use Rapid Miner studio to import Data Sets
7) Apply 'select attribute 'operator on imported data to select Size and TMC as attributes.
8) Design Roles to apply predictions
9) After Designing Roles, apply Decision tree operator on Training data stream
10) Use Apply Model operator to apply prediction to Model Dataset
11) Execute the designed Process to get the predicted result

4. DISCUSSION

Our dataset consist of 20 open source Java projects. For every project, the basic set of CK metrics are generated using CKJM. Table 1 present measure for one of the Java Project. For every class of the project, we get the numeric count of six metrics. To get one central or typical value for all six different metric measures we used the average of the numeric count of six metrics. To get the overall complexity measure for a complete project, the total average complexity of modules of the project (TACMP) is measured which is sum of all the averages of modules.

Table 1. Metric Measure for Point of Sale JAVA Project [25]

Sr No	Metrics & Classes	WMC	DIT	NOC	CBO	RFC	LCOM	Avg Complexity
1.	Customer by Id	8	5	0	3	66	16	16.3
2.	Remove Product	8	5	0	3	68	8	15.3
3.	Pro By Type	4	5	0	2	44	4	9.8
4.	Purchase	16	5	0	7	86	76	31.7
5.	Main Frame	65	6	0	47	187	1324	271.5
6.	Update Customer	10	5	0	4	70	25	19
7.	Pro By Name	4	5	0	2	45	4	10
8.	Product Report	4	5	0	2	43	4	9.7
9.	Cust By Name	4	5	0	2	43	4	9.7
10.	New Product	10	5	0	4	83	29	21.8
11.	Product By Id	6	5	0	2	64	3	13.3
12.	Payment	4	5	0	2	65	0	12.7
13.	Update Product	10	5	0	4	78	25	20.3
14.	Cutomer Report	4	5	0	2	43	4	9.6
15.	Print	4	5	0	9	38	6	10.3
16.	Remove Customer	10	5	0	4	70	25	19
17.	New Customer	12	5	0	5	80	56	26.3
		Total average complexity of modules of projects						526.5

Similarly, TACMP for all the 20 JAVA projects have been calculated. The dataset is further divided into two subsets to be used in the decision tree. One data set is training data set and other is model data set. Projects of varying sizes are chosen in two sets as swon in Table 2.

Table 2. Complexity Measure for Training Data Set

Sr No	Project Name	Size (LOC)	TACMP
1	Battle City	563	77.2
2	Bounce Ball	160	12.1
3	Chess Game	150	29
4	Fifo	637	75
5	Pong Game	713	31.3
6	TicTacToe	276	12.7
7	Parser	143	13.8
8	Cricket Analyzer	234	16.7
9	CustomerInfoSystem	1139	120.3
10	DiaryApp	431	26.3
11	Dictionary	337	24.7
12	Trignomatric Function	634	362.7
13	Scheduling and dispatch	203	82.7
14	Mynotepad	290	2
15	Chat Server	284	24.3
		TMC	62.68

Statistical measure for the calculating complexity for each module of the java project is given in (1) [25].

$$\mathrm{TACMP} = \mathrm{Avm}_1 + \mathrm{Avm}_2 + \mathrm{Avm}_3 + \cdots + \mathrm{Avm}_n \quad (1)$$

Total Mean complexity (TMC) of training data set having different size is calculated as 62.68. The formulation is given in (2) [25].

$$\mathrm{TMC} = (\mathrm{TACMP}_1 + \mathrm{TACMP}_2 + \cdots + \mathrm{TACMP}_N)/N \quad (2)$$

In general, size (LOC) alone is considered as one of the metric to determine the complexity and as the size increases, complexity of software also increases. In our experiment, Size and TACMP have also shared a moderate relationship and correlated to each other as swon in Table 3. Figure 1 shows the relationship between TACMP and Size.

Table 3. Complexity Measure for the Model Data Set

Sr No	Project Name	Size(LOC)	TACMP
1	CodeLevelSecurity	201	144.5
2	PointofSale	1082	526.5
3	e-library	323	55
4	Smart File Convertor	440	39.7
5	Shopping Cart	154	24.7
	Total	440	158.08

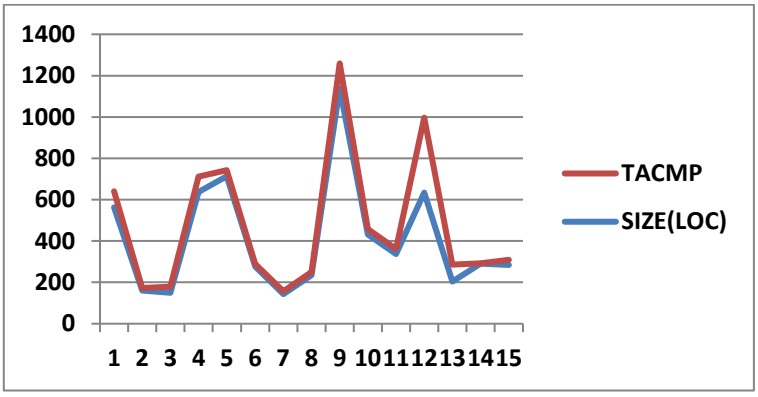

Figure 1. Correlation between TACMP and SIZE

Both size and TACMP attributes are used in training and model data set. Another attribute 'Category' is generated in training dataset. This attribute will serve as a label when training data will be applied to model data. This attribute is providing decision of whether java project will undergo maintenance or reengineering. Import both the data sets named training data and data model in Rapid Miner repository using retrieve operator. To remove the empty columns or selecting required attributes 'select attributes' operator is connected with the Training data set. In design, predictions are applied to the Roles. First Role is connected and in parameters, select category as the label. Project names are not required to be a part of analysis so another role is designed to create project name attribute as ID that is an identification that means it will not be a part of the analysis. After designing the roles, decision tree operator is selected and added to the training data stream. This decision tree operator generates a multiway decision tree. Rapid Miner uses the C4.5 algorithm in order to obtain multiway decision tree. The input to the decision tree is our training data set consisting of 15 projects. The output of the decision tree is classification model that can be applied to the new dataset (model data in our case) for prediction. In Decision tree parameters, we selected criterion as Gini index. Retrieve operator of data model is connected with the role. The parameter of this role 'category' is set to be predicted. As shown in Figure 2, the output of decision operator is connected to the model input of the Apply model operator. The model data is connected to the unlabelled data input of Apply model operator. The model as an input to the Apply model is applied to the model data at an unlabelled input.

The apply model applies the training model to the model data to predict the value of the attribute that is with 'category' attribute. The prediction is applied to the model data. Apply model produces two outputs. One is labeled data that is when training data is applied to model data, the attribute with prediction role is added to the model data. This attribute stores the predicted values of the labeled attribute using the given trained model. Another output of apply model operator is model. The training model that was input is passed without changes to the output port. Finally, execute the designed scenario.

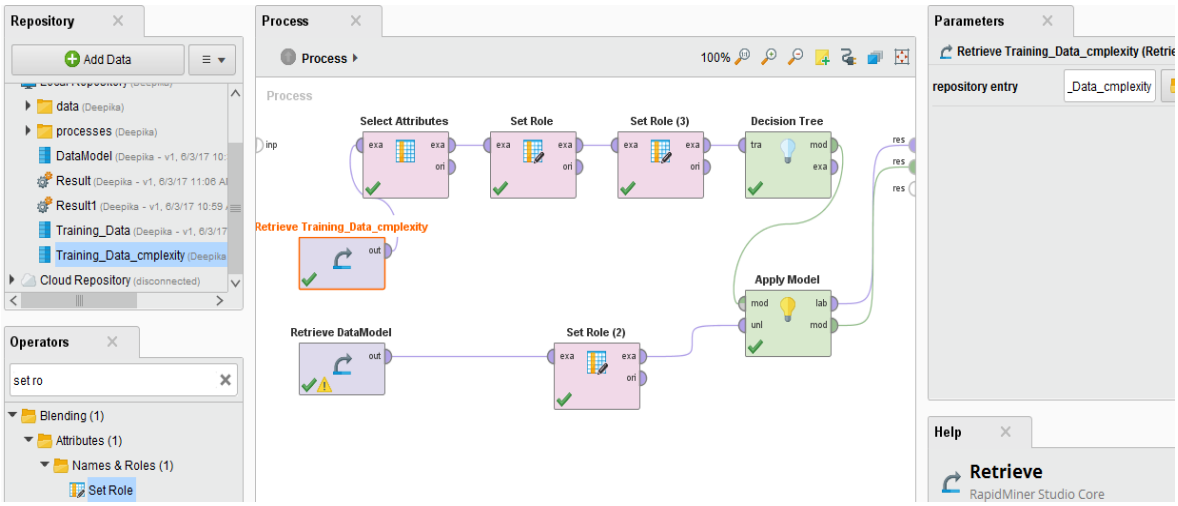

Figure 2. Decision tree modelling in rapid miner

5. RESULT

Executing the design, decision tree model as shown in Figure 3 will be generated and can be viewed by clicking on the tree tab. A decision tree is a collection of nodes and leaves. Nodes are represented as gray oval shapes. Nodes are the attributes that serve as good predictor. On the root is the average complexity (TMC). The root node represents the prominent predictor. So TMC is our best predictor of whether or not the java project is reengineered. The predicted value for TMC is 25.5. Size (LOC) is the second best predictor. The leaves are represented with multicolored endpoints that show us the distribution of category attribute. Each leaf node represents value of a labeled attribute that is category in our case. Leaf R represent reengineering and leaf M represents maintenance. The number of edges of the node is equal to the number of possible values of label attribute. The edges are labeled with disjoint ranges as per the prediction made by the decision tree. Tree from root to leaf can be interpreted as if TMC>25.5 and size (LOC)>176.5 the software undergoes reengineering.

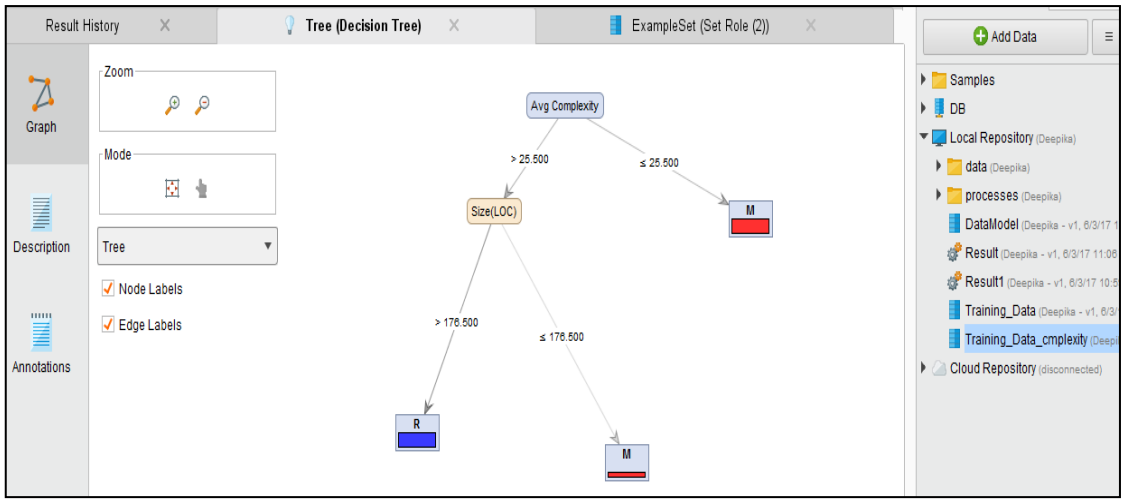

Figure 3. Decision tree stucture for model data set

Decision tree not only provides predictions but also determine how reliable the predictions are. Decision tree consists of nodes and leaves that can generate predictions based on percentage of confidence using actual attributes in the training data set and can then be applied to similarly structured data (data model in our case) to generate predictions. Decision trees not only tell us about prediction but also about confidence percentage on prediction and how to arrived on these predictions. As shown in Figure 4, level of confidence

in prediction is 100% for both Reengineering and maintenance. Four projects (Row 1 to 4 in Figure 4) need to be reengineered with 100% of confidence and one project (Row 5th) need to be maintained again with 100% of confidence. As stated earlier in this paper, Rehma et al. [14] work is having theoretical existence only, it lacks practical implementations. Comparision of proposed framework and Sood [15] framework is given in Table 4.

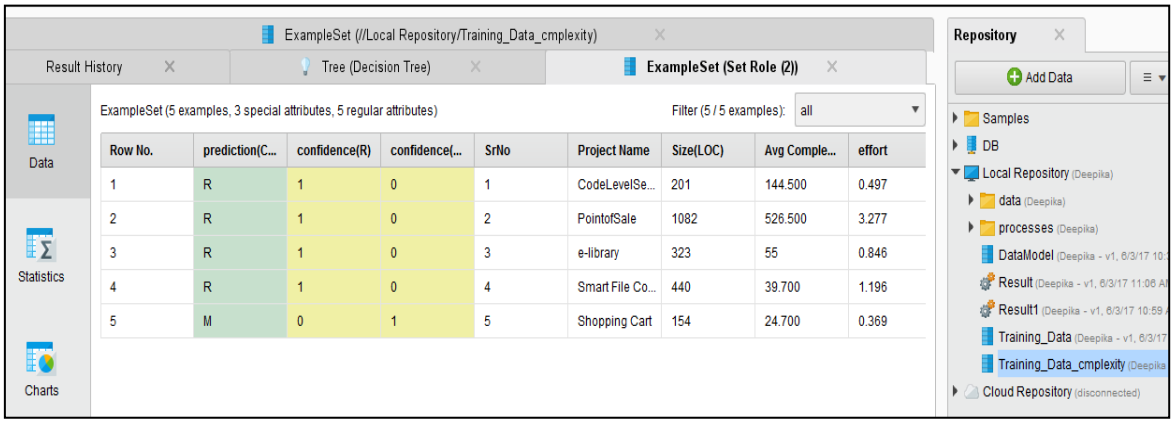

Figure 4. Level of confidence in predictions

Table 4. Comparison of Existing and Proposed Framework

	Proposed Framework	Existing Framework
Approach used	Prediction based on decision tree using Ck metric suit.Complexity and size as attributes.	Measuring reengineering requirements by knowing the changes in numbers of lines of code
Data Set	Twenty open source Java based software	Three C,C++ based software
Metric Used for reengineering decision making	Well known CK metric suit- Weighted Methods per Class (WMC), coupling Between Object classes (CBO), Depth of the Inheritance Tree (DIT), Number of Children (NOC) and Response for a Class (RFC)	Defect cost and Fault cost are used to calculate Reengineering Requirement Cost.
Methodology differences	Framework is based on data set of heterogeneous (Different size and measured CK complexity) Java Projects. Decision tree based prediction approach is used for reengineering and maintenance decision making.	Framework is based on three projects developed in C and C++.
Results Analysis	Statistical Measure of Complexity of CK metric suit for twenty Java based Projects. Reengineering is prediction based using decision tree prediction approach supported by confidence measure. Results can surely be further improved with improvement of data set.	Calculation of Reengineering Requirement Cost metric is dependent on defect cost metric which needs to calculate total lines affected by defect. Method of calculating total lines affected by defect is ambiguous.

6. CONCLUSION

To improve the quality of the software, it is important to analyze the changes at design level without changing the functionality of the software. By calculating the complexity using six basic CK metrics and with the help of decision tree using rapid miner, predictions have been made regarding the requirements of reengineering or maintenance for the software. This framework can become a basis for deciding whether the project should undergo reengineering or maintenance. Further, the results can be generalized by considering more projects of different size and complexity.

REFERENCES

[1] Lehman G. and Meir M., "Programs Life Cycles, and Laws of Software Evolution," *Proc. IEEE*, vol/issue: 68(9), pp. 1060–1076, 1980.
[2] Ian S., "Software Engineering," 9th edition, Pearson Publication, 2014.
[3] Srinivas M., *et al.*, "Analysis of Legacy System In Software Application Development: A Comparative Study," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 6(1), pp. 292-297, 2016.

[4] Chikofsky and Cross J. H., "Reverse Engineering and Design Recovery: A Taxono," *IEEE Software Engineering journal*, pp. 13-17, 1990.
[5] IEEE Std 1219-1998, "IEEE Standards Software Engineering," 1999 Edition, Volume Two, Process Standards, IEEE Press, 1999.
[6] Ian S., "Software Engineering," 8th edition, Addison Wesley, 2008.
[7] Sneed M., "20 Years of Software-Reengineering: A Resume," *10th Workshop software reengineering (WSR'08)*, pp. 115-124, 2008.
[8] Basili V. R. and Perricone B. T., "Software errors and complexity: An empirical investigation." *ACM*, vol. 27, pp. 42-52, 1984. http://portal.acm.org/citation.cfm?id=2085.
[9] McCabe T. J., "A complexity measure," *IEEE Trans. Software Eng*., vol. 2, pp. 308-320, 1976.
[10] Halstead M. H., "Elements of Software Science," 1st Edn, Elsevier North Holland, New York, pp. 127, 1977.
[11] Li W. and Henry S., "Maintenance Metrics for the Object Oriented Paradigm," *Proc. of IEEE 1st Int. Sw. Metrics Symposium*, pp. 52-60, 1993.
[12] Bhardwaj M. and Rana A., "Key software Metrics and its Impact on each other for Software Development Projects," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 6(1), pp. 242-248, 2016.
[13] Chidamber S. R. and Kemerer C. F., "A metrics suite for object-oriented design," *IEEE Trans. Software Eng.*, vol. 20, pp. 476-493, 1994.
[14] Rahma A. K., *et al.*, "Expert-based decision support framework for software reengineering," *Malaysian Conference in Software Engineering*, 2011.
[15] Sood S., "Software reengineering: a metric set based approach," PhD Thesis, Himachal Pradesh University, 2012. http://hdl.handle.net/10603/129186.
[16] Wood S., *et al.*, "Semantic Foundation for Architectural Reengineering and Interchange," *P*roc. *International Conference on Software Maintenance (ICSM-99), Oxford, England, Los Alamitos, Ca: IEEE Computer Society*, pp. 391-398, 1999.
[17] R. Kazman, *et al.*, "Requirements for Integrating Software Architecture and Reengineering Modes: CORUM II," *Proc. 5th Working Conference on Reverse Engineering (WCRE-98), Honolulu, Hi, Los Alamitos, Ca: IEEE Computer Society*, pp. 154-163, 1998.
[18] E. J. Byrne, "A conceptual foundation for software reengineering," *Proc of conference on Software Maintenance, Orlando, Florida*, pp. 226-235, 1992.
[19] Yang X., *et al.*, "Dual-Spiral Reengineering Model for Legacy System," *TENCON 2005, IEEE Region 10*, pp. 1-5, 2005.
[20] Su X., *et al.*, "Parallel iterative reengineering model of legacy systems," *Proc. of IEEE International Conference on Systems, Man and Cybernetics, USA*, pp. 4054-4058, 2009.
[21] Murphy G. C. and Notkin D., "Reengineering with reflexion models: a case study," *IEEE Computer*, vol/issue: 30(8), pp. 26-36, 1997.
[22] Chung S., "Service-Oriented Software Reengineering: SoSR," *Proc. HICSS 2007 - 40th Hawaii International Conference on Systems Science*, Waikoloa, Big Island, HI, USA. pp. 172, 2007.
[23] Spinellis D., "Tool writing: A forgotten art?" *IEEE Software*, vol/issue: 22(4), pp. 9-11, 2005.
[24] CKJM Tool Website, Available: http://www.spinellis.gr/sw/ckjm/doc/ver.html
[25] Singh J., *et al.*, "Identification of requirements of software reengineering for JAVA projects," *2017 International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, pp. 931-934, 2017.
[26] Rapid Miner Tool Website Available : http://rapid-i.com/content/view/30/82/lang,en/
[27] North M., "Data Mining for the masses," A Global Text Project Book, 2012.
[28] Niswatin R. K. and Wulanningrum R., "Prediction of College Student Achievement Based on Educational Background Using Decision Tree Methods," *Indonesian Journal of Electrical Engineering and Computer Science*, vol/issue: 4(2), pp. 429-438, 2016.

BIOGRAPHIES OF AUTHORS

Jaswinder Singh holds Masters in Computer Application and persuing PhD in computer application. He is having 10 years of experience in Acedemics and his expertise includes Software Maintenanace, Reengineering, Algorithm generations and Data Mning.Current work includes Identification of Metrics for reengineering Java projects, study of significance of reengineering in todays scenario for Sofftware development industry.

Dr. Kanwalvir Singh Hold PhD in computer engineering .His area of interest includes Cloud Computing, Big Data, IoT, Mobile Computing, Database & Security, Web Engineering.

Dr. Jaiteg holds a PhD in Computer Science and Engineering with 12 years of experience in Research, Development, Training, Academics at Institutes of Higher Technical Education. His areas of expertise are Software Engineering, Business Intelligence, Data and Opinion mining, Cartography, Curriculum design, Pedagogical Innovation & Management. Areas of interest include Sustainable Software Engineering, Education Technology, Offline Navigation Systems and Cloud Computing.