An Efficient Approach for Software Maintenance Effort Estimation Using Particle Swarm Optimization Technique

Chamkaur Singh, Neeraj Sharma, Narender Kumar

Abstract: ***The main objective of software engineering community is to develop useful models that are able to calculate the accurate estimating software effort. COCOMO (Constructive Cost Model) is consider as mostly used algorithmic maintenance cost modeling technique among other software maintenance cost estimation techniques. It is mostly used technique due to its simplicity for estimating the effort in person-months for a project at different stages. In this paper we have proposed a new approach that is able to give better results. In proposed approach we have used Tomcat server dataset whose features are extracted using Principle component analysis approach which is further optimized using Particle Swarm Optimization. In previous work most of researchers have used Genetic algorithm but it is a time consuming part. So, in this paper we have used Particle swarm optimization that gives improved results. At the end we have used Linear discriminant analysis for classification that classifies the priority levels and tell how much your system is having the estimations for the cost based on Source lines of the codes or functional points or the efforts required. The proposed approach is tested in terms of functional point, set effort person per month and SLOC that gives best results.***

Keywords: COCOMO, Functional point, Set effort per month, SLOC, Software engineering, PCA, LDA, PSO.

I. INTRODUCTION

The most crucial and essential activity to be consider in software development life cycle (SDLC) is software effort estimation (SEE) that is used for software development activities budgeting, monitoring and planning [1]. It is also used for within and on time software budget delivery that gives most reliable and realistic value of required effort for developing a project. The person-month is the term in which it is calculated. This effort is used for project plans, project budgets, investment analysis, resource allocation schemes, pricing process, etc. It has been seen that for good management decision making software development effort accurate cost estimation is a critical process. For software industry effort estimation reliability and precision is very important because for software companies both underestimates and overestimates software effort are harmful. Thus, from an organizational perspective, an early and accurate cost estimate will reduce the possibility of organizational conflict during the later stages.

So, the main objective of software engineering community is to develop useful models that are able to calculate the accurate estimating software effort. In period 2002-2003, out of 13522 projects only 33% projects were completed within budget and time in UK as per the standard group report1. 82% projects were late, 43% projects were overrun their financial plan and 20% projects were cancelled [2]. This creates a need of making a reliable and realistic software estimation effort model. Many researchers have proposed various effort estimation methods that are categorized into analogy, algorithmic and expert judgment method. Estimation of Resources-Software Estimating Model (SEER-SEM), Function Point (FP) method, Constrictive Cost Model (COCOMO) and Software Life Cycle Management (SLIM) are algorithmic methods. In early phase of SDLC analogy method is applied [3,4,5,6].

Another thing which is very important to be consider is software maintenance. Software maintenance is a very broad activity that includes improvements in capabilities, error correction, optimization and removal of obsolete capabilities. Change is predictable that create a need to develop a mechanism for controlling, evaluating and making changes. So, maintenance work is the work done to modify value of software over the period. The customer base expanding, meeting additional requirements and new more efficient technology can be use to improve the value.

COCOMO (Constructive Cost Model) is consider as mostly used algorithmic cost modeling technique among other software cost estimation techniques. It is mostly used technique due to its simplicity for estimating the effort in person-months for a project at different stages [7]. In this project maintenance cost estimation is predicted by mathematical formulae. Various algorithms and approaches has been used by different researchers for COCOMO model. In this paper, main objective is to propose a cost estimation model based on Particle Swarm Optimization (PSO) and Principal Component Analysis (PCA) is used for feature extraction which is further classified using Linear Discriminant Analysis (LDA). This paper is divided into different section in which first section gives the brief introduction to Software testing and maintenance cost estimation. The brief survey of related work is given in second section and then in third section main focus is given on COCOMO method. The fourth section gives description of used dataset and results are briefly elaborated in section fifth. The paper is ended with conclusion of complete propose work.

Revised Manuscript Received on December 22, 2018.

Chamkaur Singh, Research scholar of I.K. Gujral Punjab Technical University, Jalandhar, Punjab, India.

Dr. Neeraj Sharma, Professor, Gian Jyoti Group of Institutions, Mohali, India.

Dr. Narender Kumar, Assistant Professor, HNB Garhwal University, Srinagar Garhwal Uttarakhand, India.

II. RELATED WORK

According to Banker et al. (1987) [15], it is very difficult to estimate software maintenance cost. The problem arises because cost depends upon various technical and non-technical unpredictable factors. Hardware stability, program lifetime, supported application and dependence of the program on its external environment is some of the technical factors that affect software maintenance cost. They have also highlighted technical factors included in modern programming practices and hardware/software change, programming language, software tools and response time like technical factors. The author has further quoted many studies to highlight non-technical factors such as use of duplicate/reusable code, classification of work, and distance of the machine room was also considered in various studies.

The genetic algorithm, neural network or rule induction like machine learning and case based reasoning are getting used in number of applications. In 1997, Samson, B., et.al, [8] has analyzed that in medicines, econometrics and computer vision fields human abilities have proven to be superior to those of computers. Such techniques hold the promise of being able to make sense of a variety of inputs of different types in producing an output. There is low accuracy in terms of prediction in case of statistical methods and hit-or-miss business is software cost modeling. Albus perceptron or CMAC like neural network are used for performing experiment. They have also highlighted some of the problem arise when machine learning techniques are used in software cost modeling. An improved accuracy can be achieved using neural network as compared to conventional regression analysis.

In 2004 [17], H. M. Sneed offered a cost model for software maintenance and evolution based on fixed and variable costs. In order to propose the model, the study has used parameters which are derived from static, dynamic, defect and productivity analysis of the software. Further, the types of tasks covered by this model include error correction, functional enhancement, routine change and technical renovation. The study shows that the proposed model does not give accurate results in case of reengineering and maintenance of web applications.

In 2005, an empirical review on building corrective maintenance effort estimation models has been given by De Lucia et al [18]. The multivariate linear regression techniques are base of this study and cost estimation models should be improve using different tasks types suggested by author. The task types suggested include type A, type B and type C. Type A concerns source code modification. Type B relates to fixing of data misalignments and type C concerns intervention not comprised in the previous categories. It has further been proposed that if task types for maintenance activities are difficult for any project, then other models which are based on coarse-grained metrics should be used. Scope of this study is limited to corrective maintenance.

In the IT industry, precisely evaluate the effort of each software development project to develop cost and development schedule management to the software company in the software are count for much. The manpower cost is the main cost of software project that is directly proportional to schedule of development. Due to it precise effort valuation has become more important. In 2010, Jin-Cherng Lin, et.al, [9] have proposed a approach in which software project clustering is done using K-Means clustering algorithm and one way or pears on product moment correlation coefficient is use for analysis purpose. After that fitness value is evaluated by finding Mean of MRE using PSO and N-1 test method for COCOMO parameters optimization. Then optimized parameters are used to calculate effort of software project that is need to be estimated. In this project 63 history software projects data of COCOMO to use for testing.

It is challenging task for software engineers to make a reliable effort estimation approach. Accurate effort estimation is the state of art of software engineering, effort estimation of software is the preliminary phase between the client and the business enterprise. The software estimation is a beginning of business enterprise and client relationship. Saleem Basha, et.al, (2010), [10] main aim is to study the empirical software effort estimation. From review it has been concluded that for different situation no single technique is prove to be good that is seen from comparison of the results of several approaches is most likely to produce realistic estimates.

In 2012, Syed Ali Abbas, et.al, [11] gives baseline to researchers, software organizations and practitioners to comprehend the enhancements in Cocomo suite models and differences among these derivatives. Authors have stated that over the years various models has been proposed by different researchers for supporting software cost estimation processes and Cocomo (Constructive Cost Model) is considered as one of the best model that gives more accurate schedule estimates. So an overview of Cocomo suite model is discussed in this paper along with it they have also given relevant environment for their use, concept behind their development and the procedures for its development.

In 2014 M. Madheswaran, et.al, [12] have given main focus in constructing artificial neural networks based software effort estimation model . The aim behind this work is to improve the COCOMO model performance. They have used multi layer feed forward neural network to accommodate the model and its parameters to estimate software development effort. The network is trained with back propagation learning algorithm by iteratively processing a set of training samples and comparing the network's prediction with the actual effort. The network is trained and tested using COCOMO dataset and results shows that proposed model gives more estimation accuracy of model as compared to COCOMO model.

The COCOMO II model extension is offered by Vu. Nguyen (2014), [19] after evaluation of various popular software maintenance cost estimation models. He has revealed that the various existing models suffer from weaknesses concerning limited choice of input metrics and limited scope of maintenance activities. The author has evaluated and presented the extended version of COCOMO II model for effort and size estimation of software maintenance projects. The author has used regression approach to build the estimation model. It has been emphasized that the proposed model can be used for the organizations where data is not sufficient to calibrate various estimation models. This extended version of COCOMO II model also considers SLOC metric of deleted code in its size metric.

However, the author has cautioned that this model is limited to functional enhancement and fault correction activities of maintenance. Hence, the model needs further improvement to support reengineering, language and data migration, performance improvement and other cost effective activities.

It has been seen that software effort is an essential and crucial activity for the software development life cycle. In this paper Rohit Kumar Sachan, et.al, (2016) [13] have proposed a new model based on genetic algorithm that is used for basic COCOMO model parameters optimization. The NASA software project dataset is applied in proposed approach that shows better results in terms of realistic estimation as compared to basic COCOMO.

III. COCOMO

In 1981, Barry Boehm has introduced the COCOMO model that is stands for Constructive Cost Model that is used for estimating software projects cost, effort and schedule in his classic text comes under software Engineering Economics. On the basis of 63 completed projects from diverse domains this model is defined [4]. The basic, intermediate and detailed are three increasingly detailed and accurate forms hierarchy makes a COCOMO 81. In case of quick, early, rough order of magnitude estimates of software maintenance

costs basic COCOMO or first level is good. There is need of 15 different effort multipliers into account and cost drivers in case of intermediate COCOMO [14]. Detailed COCOMO additionally accounts for the influence of individual project phases. COCOMO additionally accounts for the influence of individual project phases.

The effort required to make project in general form is given below:

$PM = \alpha * (KLOC)^{b} * \left(\pi_{j}\, EM_{j}\right)$ …..(1)

Where effort estimate in person months is denote by PM, productivity coefficient is denote by α, b denotes economies of scale coefficient. Kilo lines of code is denoted by KLOC, effort multipliers or cost drives is denoted by EM in the above given equation.

Simple Model

$C = A \times (S)^{B}$ …..(2)

The above given second equation gives the estimate software maintenance cost for basic pattern or simple model of estimation where estimated software effort is denoted by C, A, B are estimated parameters, thousand of lines of code (KLOC) is represented by S. The adjustment factor is not considered in this model.

Intermediate Model

$C = A\, S^{B} \times \prod_{i=1}^{15} x_i$ ……(3)

The above equation gives the software effort estimation of intermediate model where C, A, B, S is same as in equation 2. The only addition is multiplication by the product of 15 factors and adjustment factor value is represented by x_i. The difference between basic and intermediate mode is the addition of 15 more important factor in the software effort estimation. Further it is divided into four categories of project property, personal, computer and product attributes.

Detail Model

In this the same equation of intermediate model is used. The only difference is that in this more detail classification of 15 factors for the project is there as compared to intermediate model.

Project modes COCOMO

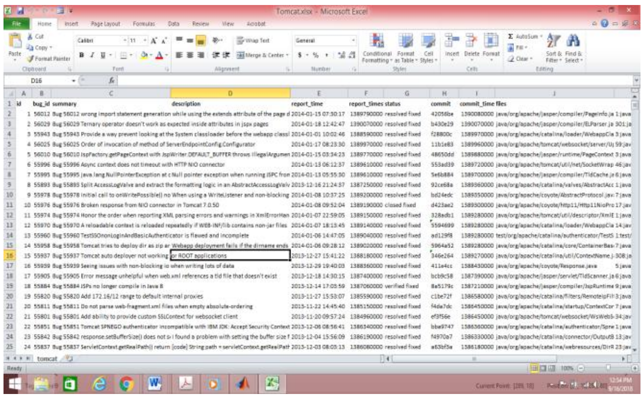
Organic, Semidetached and Embedded are three project modes of COCOMO.

IV. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization is a technique developed by Dr. Eberhart and Dr. Kennedy in 1995[20]. It is population based stochastic optimization technique inspired by inspired by social behavior of bird flocking or fish schooling. In a population of solutions, each solution is considered as a particle and the main objective is to find an optimal solution through the movement of a particle in solution space. Each time particle moves from one position to another it keeps track of coordinate values of a particle with best solutions through some fitness function. Best values achieved by the individual particle from its all movement is consider lbest . The value of the solution is achieved best among all the solutions is consider gbest. Each time velocity is changing towards it lbest [21].

V.DATASET

Software cost estimation deals with the implementation of any kind of software which is based on some data sets. So you have to estimate the cost of the implementation at the end of the process and you have to evaluate the performance in terms of functional points, the source lines of codes and the efforts that are required to implement the software. Here we have taken the Tomcat server process[23][24][25]. In Tomcat server process we have deal with some big ids and some bug report and their priority levels and after that we have implemented that process on Matlab software[22]. Further we have classified their priority levels that which bug will come, which bug is having high priority to implement and to solve or to resolve.



id	bug_id	summary	description	report_time	report_times	status	commit	commit_time	files	
1	56012	Bug 56012 wrong import statement generation while using the extends attribute of the page d		2014-01-15 07:30:17	1389790000	resolved fixed	42056be	1390080000	java/org/apache/jasper/compiler/PageInfo.ja	1:java
2	56029	Bug 56029 Ternary operator doesn't work as expected inside attributes in jspx pages		2014-01-18 12:42:47	1390070000	resolved fixed	b430e29	1390070000	java/org/apache/jasper/compiler/ELParser.ja	301:ja
3	55943	Bug 55943 Provide a way prevent looking at the System classloader before the webapp classl		2014-01-01 10:02:46	1388590000	resolved fixed	f28800c	1389970000	java/org/apache/catalina/loader/WebappCla	3:java
4	56025	Bug 56025 Order of invocation of method of ServerEndpointConfig.Configurator		2014-01-17 08:23:30	1389970000	resolved fixed	11b1e83	1389960000	java/org/apache/tomcat/websocket/server/U	59:jav
5	56010	Bug 56010 JspFactory.getPageContext with JspWriter.DEFAULT_BUFFER throws IllegalArgumen		2014-01-15 03:34:23	1389770000	resolved fixed	48650dd	1389680000	java/org/apache/jasper/runtime/PageContext	3:java
6	55996	Bug 55996 Async context does not timeout with HTTP NIO connector		2014-01-13 06:12:37	1389610000	resolved fixed	553ad39	1389720000	java/org/apache/tomcat/util/net/SocketWrap	46:jav
7	55995	Bug 55995 java.lang.NullPointerException at c	Null pointer exception when running JSPC fron	2014-01-13 05:55:30	1389610000	resolved fixed	5e6b884	1389700000	java/org/apache/jasper/compiler/TldCache.ja	6:java
8	55893	Bug 55893 Split AccessLogValve and extract the formatting logic in an AbstractAccessLogValv		2013-12-16 21:24:37	1387250000	resolved fixed	92ce68a	1389360000	java/org/apache/catalina/valves/AbstractAcc	1:java
9	55978	Bug 55978 Initial call to onWritePossible() no	When using a WriteListener and non-blocking	2014-01-08 10:37:25	1389200000	resolved fixed	bd24edc	1389350000	java/org/apache/coyote/AbstractProtocol.jav	7:java
10	55976	Bug 55976 Broken response from NIO connector in Tomcat 7.0.50		2014-01-08 09:52:04	1389190000	closed fixed	d423ae2	1389300000	java/org/apache/coyote/http11/Http11NioPro	17:jav
11	55974	Bug 55974 Honor the order when reporting XML parsing errors and warnings in XmlErrorHan		2014-01-07 22:59:05	1389150000	resolved fixed	328adb1	1389280000	java/org/apache/tomcat/util/descriptor/XmlE	1:java
12	55970	Bug 55970 A reloadable context is reloaded repeatedly if WEB-INF/lib contains non-jar files		2014-01-07 18:13:45	1389140000	resolved fixed	5594699	1389280000	java/org/apache/catalina/loader/WebappCla	14:jav
13	55960	Bug 55960 TestSSOnonLoginAndBasicAuthenticator is flawed and incomplete		2014-01-06 14:47:05	1389040000	resolved fixed	ad129f8	1389280000	test/org/apache/catalina/authenticator/TestS	1:test/
14	55958	Bug 55958 Tomcat tries to deploy dir as zip ar	Webapp deployment fails if the dirname ends	2014-01-06 09:28:12	1389020000	resolved fixed	5964a52	1389280000	java/org/apache/catalina/core/ContainerBas	7:java
15	55937	Bug 55937 Tomcat auto deployer not working	or ROOT applications	2013-12-27 15:41:22	1388180000	resolved fixed	346e264	1389270000	java/org/apache/catalina/util/ContextName.j	308:ja
16	55939	Bug 55939 Seeing issues with non-blocking io when writing lots of data		2013-12-29 19:40:03	1388360000	resolved fixed	411e4cc	1388430000	java/org/apache/coyote/Response.java	5:java
17	55905	Bug 55905 Error message unhelpful when web.xml references a tld file that doesn't exist		2013-12-18 14:30:15	1387400000	resolved fixed	bcb9c58	1387390000	java/org/apache/jasper/servlet/TldScanner.ja	6:java
18	55884	Bug 55884 JSPs no longer compile in Java 8		2013-12-14 17:03:59	1387060000	verified fixed	8a5179c	1387210000	java/org/apache/jasper/compiler/JspRuntime	9:java
19	55820	Bug 55820 Add 172.16/12 range to default internal proxies		2013-11-27 15:53:07	1385590000	resolved fixed	c1be72f	1386580000	java/org/apache/catalina/filters/RemoteIpFil	3:java
20	55811	Bug 55811 Do not parse web-fragment.xml files when empty absolute-ordering		2013-11-22 14:45:40	1385150000	resolved fixed	f4da7dc	1386450000	java/org/apache/catalina/startup/ContextCor	7:java
21	55801	Bug 55801 Add ability to provide custom SSLContext for websocket client		2013-11-20 09:57:24	1384960000	resolved fixed	ef3f56e	1386450000	java/org/apache/tomcat/websocket/WsWebS	34:jav
22	55851	Bug 55851 Tomcat SPNEGO authenticator incompatible with IBM JDK: Accept Security Context		2013-12-06 08:56:41	1386340000	resolved fixed	bba9747	1386360000	java/org/apache/catalina/authenticator/Spne	1:java
23	55842	Bug 55842 response.setBufferSize() does not s	I found a problem with setting the buffer size f	2013-12-04 15:56:09	1386190000	resolved fixed	f4970a7	1386330000	java/org/apache/catalina/connector/OutputB	13:jav
24	55837	Bug 55837 ServletContext.getRealPath() return	[code] String path = servletContext.getRealPath	2013-12-03 08:03:13	1386080000	resolved fixed	a63bf3a	1386180000	java/org/apache/catalina/webresources/DirR	23:jav

Figure 1 Tomcat server process Dataset

The above figure shows some of the taken Tomcat server process dataset. This dataset contains some bug IDs, summary of that bugs, report time, their status of report time, status report where it is resolved. It also includes commit processes commit ID that is committed for that particular bug implemented in Java software. This dataset will deal with whole process and it is consider as house server implementations or Organic project mode.

VI. RESULTS

In this work we have used Tomcat server process dataset as given in above section. This section will give the brief about implementation of proposed approach using MATLAB Graphical user interface (GUI). The complete work is divided into different parts. In order to implementation of the COCOMO model the principal component analysis is used to extract the feature vector as the fetching and processing of the data initially is the raw vector which is not understandable and having no hierarchy so we have to extract the characteristic values as feature values in terms of Eigen vectors using principal component analysis and to optimize the performance of the system and to reduce the complexity of the vector processing which reduces the redundancies of the data then we have performed the optimization process using particle swarm optimization which increases the execution of the processes in the system to reduce its complexity in an effectual manner.

We have to deal with the priority levels that which bug-id is having high priority. The Figure 2 shows the GUI panel of proposed work. In this GUI there is first pushbutton on Upload database which is selected to upload the dataset then will get the bug ID, description and report timestamps on the GUI as shown in figure.

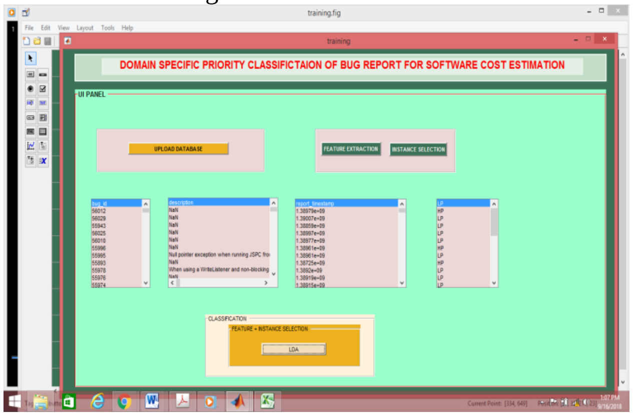

training.fig
training
DOMAIN SPECIFIC PRIORITY CLASSIFICTAION OF BUG REPORT FOR SOFTWARE COST ESTIMATION
UI PANEL
UPLOAD DATABASE
FEATURE EXTRACTION
INSTANCE SELECTION
CLASSIFICATION
FEATURE + INSTANCE SELECTION
LDA

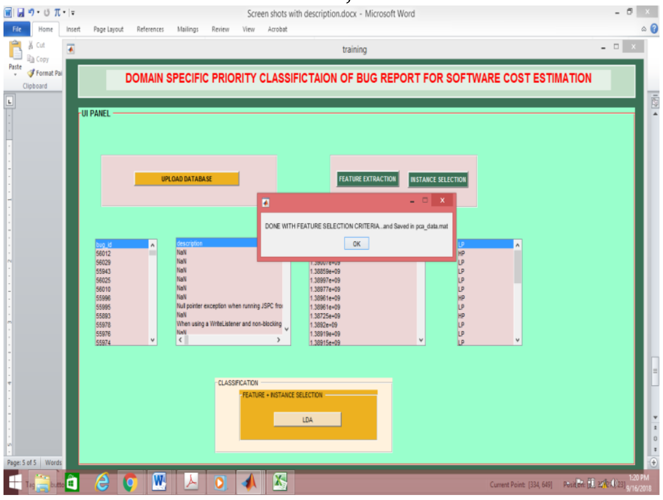
Figure 2 GUI panel

Then the next step will come after when you will click on the LDA section. So we have implemented some algorithms for that. First of all, we have upload data set and then we have to extract the feature vectors from data set and then we have implemented the instance selection process. It deals the selection of some relevant features from the existing feature vector and then we have to apply the classification process which is LDA that is linear discriminant analysis, and we have to click the button for Classification of their priority levels that which bug ids having high priority to be resolved.

The below figure 3 shows the Feature selection Criteria. For feature extraction we have used PCA. Mainly PCA is used for feature extraction, classification and compression of images. PCA stands for principle components for transformation purpose. First principle component only monitors the variations in data and after this all the related component examine the residual changes if possible. The maximum changes are monitored by the first principle component. The second principal component implemented to measure the variation in the subspace which is perpendicular to the first. After this the third principal component implemented to measure the maximum changes in the subspace which is perpendicular to first and second.

In this technique the correlated variables are transformed into various non-correlated variables. PCA method is usually implemented for classification as well as compression of images. From the name it is cleared that it uses principle components for transformation purpose. In this technique the efficient and brief details of data set are computed. It mainly deals with some icon vectors and then we have two eigen Vector deals with a characteristics values and instance elections deals with the characteristics value or relevant feature vectors from that existing feature extraction process.

So after that we have to click on that feature section process, then you will see that the feature selection criteria is done and it's saved in the data set, which is .mat file.


Screen shots with description.docx - Microsoft Word
training
DOMAIN SPECIFIC PRIORITY CLASSIFICTAION OF BUG REPORT FOR SOFTWARE COST ESTIMATION
UI PANEL
UPLOAD DATABASE
FEATURE EXTRACTION
INSTANCE SELECTION
OK
CLASSIFICATION
FEATURE + INSTANCE SELECTION
LDA

DONE WITH FEATURE SELECTION CRITERIA...and Saved in pca_data.mat

Figure 3 Feature selection Criteria

After that we have to tracked the instance selection using particle size or particle swarm optimization process. So, we have to keep in that instance election, then you will get some optimize feature values for the instance selections. PSO attracts the high fitness space positions of particle. It works by considering velocity and positions two factors which are attained locally and globally. Firstly local best position is calculated using specified equation which is further used for comparison purpose to get global best position value.

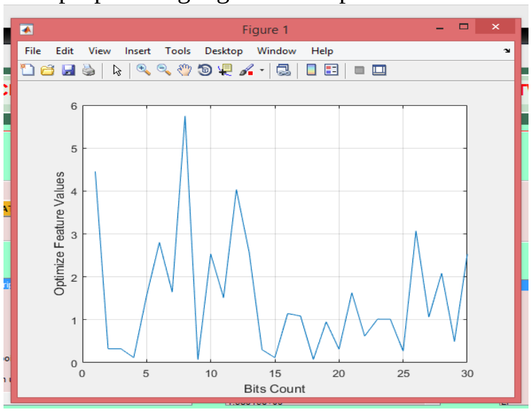


Figure 4 Optimized features

So once we clicked on LDA pushbutton we will get some high priorities low protein levels for the bug IDs for the bugs which are absorbed and after working on that LDA selection process that is the classification process which classifies the priority levels and that how much your system is having the estimations for the cost based on Source lines of the codes or functional points or the efforts required.

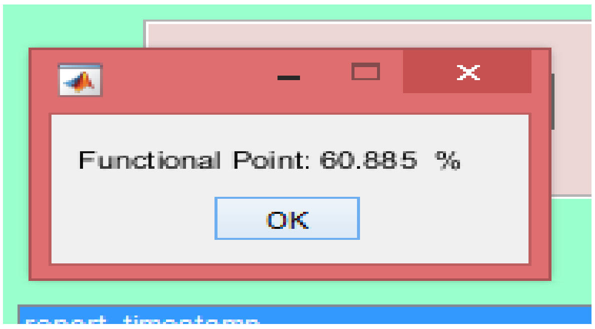

Functional Point: 60.885 %

Figure 5 Functional point using Proposed approach

After evaluation of those parameters we have got the functional point set as 60.885%. It will tell how much our system is reliable to provide the services for the appropriate functions. And it also tells how much system is able to provide some you can say that estimations for these services to the user or the product provides to a user which is used to compute the functional size measurement of the software.

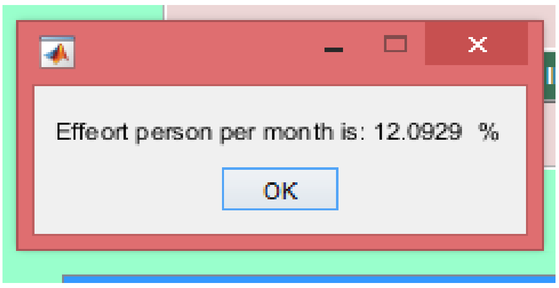

Effeort person per month is: 12.0929 %

Figure 6 Effect person per month of proposed approach

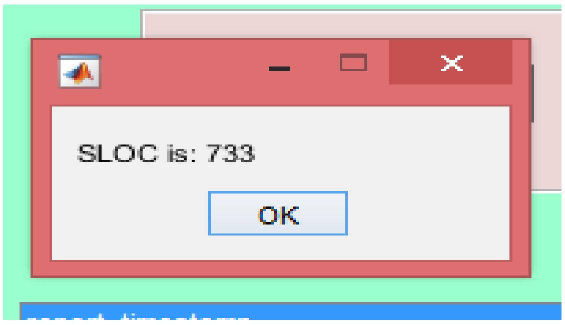

SLOC is: 733

Figure 7 SLOC of proposed approach

The above given Figure 6 and figure 7 gives the results of effort person per month and SLOC using proposed approach. The efforts persons per month are 12.0929 in terms of percentage and SLOC is 733.

Functional points analysis is a major techniques among all the techniques used in this paper. Large part of the empirical research is based on this technique but should not be taken as reason of exclusion of other techniques.

VII. CONCLUSION

Accurate and reliable Software maintenance effort estimation is a very challenging for both academic communities and software industrial. There are several software effort forecasting models that can be used in forecasting future software development effort. In this paper, we have proposed a maintenance cost estimation model based on Particle swarm optimization. In our approach we use a model that maps COCOMO model to a PSO. The tomcat server dataset is use whose features are extracted using PCA which are further optimized using PSO. The last section of this work is classification of priority levels and that how much your system is having the estimations for the cost based on Source lines of the codes or functional points or the efforts required that is performed using LDA. The proposed approach is tested in terms of functional point, set effort person per month and SLOC. The functional point achieve is 60.885%, set effort person per month and SLOC is 12.0929 and 733 respectively.

REFERENCES

1. Ch. Satyananda , KVSVN Raju, “An Improved Fuzzy Approach for COCOMO’s Effort Estimation Using Gaussian Membership Function”, Journal of Software, vol 4, pp 452-459, 2009.
2. The Standish Group Internaction “EXTREME CHAOS Report 2001” , 2001.
3. P. Suri, P. Ranjan, “Article: Comparative Analysis of Software Effort Estimation Techniques”, International Journal of Computer Applications, vol. 48, pp. 12-19, 2012.
4. B. W. Boehm, et al., “Software Engineering Economics”, Prentice-hall Englewood Cliffs (NJ), vol. 197, 1981.
5. J. E. Matson, B. E. Barrett, J. M. Mellichamp, “Software Development Cost Estimation Using Function Points”, IEEE Transactions on Software Engineering, vol. 20, pp. 275-287, 1994.
6. L. Putnam, W. Myers, “Measures for Excellence”, 1992.
7. Yucheng Kao, Jin-Cherng Lin , Jian-Kuan Wu “A Differential Evolution Approach for Machine Cell Formation”, IEEE Industrial Engineering and Engineering Management, 2008., pp. 772-775, 2008
8. Samson, B., David Ellison., Pat Dugard., “Software cost estimation using an albus perceptron”, journal of Info & Softw., vol.39, pp.55-60, 1997.
9. Jin-Cherng Lin, Han-Yuan Tzeng, “Applying Particle Swarm Optimization to Estimate Software Effort by Multiple Factors Software Project Clustering”, IEEE, pp. 1039-1044, 2010.
10. Saleem Basha, Dhavachelvan P, “Analysis of Empirical Software Effort Estimation Models”, (IJCSIS) International Journal of Computer Science and Information Security, vol. 7, pp. 68-77, 2010.
11. Syed Ali Abbas, Saleem Ullah Lar, Xiaofeng Liao, Raja Aftab Naseem, “Software Models, Extensions and Independent Models in Cocomo Suite: A Review”, Journal of Emerging Trends in Computing and Information Sciences, vol. 3, pp. 683-693, 2012.
12. M. Madheswaran, D. Sivakumar, “Enhancement of prediction accuracy in COCOMO model for software project using neural network”, IEEE, pp. 1-5, 2014.
13. Rohit Kumar Sachan, Ayush Nigam, Avinash Singh, Sharad Singh, Manjeet Choudhary, Avinash Tiwari and Dharmender Singh Kushwaha, “Optimizing Basic COCOMO Model using Simplified Genetic Algorithm”, Twelfth International Multi-Conference on Information Processing-2016, vol. 89, pp. 492-498, 2016.
14. B.W. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts, “Software cost estimation with COCOMO II”, Prentice Hall, 2000.
15. R. D. Banker, S. M. Datar, C. F. Kemerer, “Factors affecting software maintenance productivity: an exploratory study”, Proceedings of 8th international conference on Information system. Pittsburgh S, pp. 160–175, 1987.
16. M. M. Lehman, J.F. Ramil, P.D. Wernick, D.E. Perry, W.M. Turski, “Metrics and laws of software evolution - the nineties view", IEEE International Symposium on Software Metrics (METRICS'97), Los Alamitos, CA, pp.20-32, 1997.
17. H.M. Sneed, “A cost model for software maintenance and evolution", Proceedings of IEEE 20th International Conference on Software Maintenance, Chicago, USA pp. 264 – 273, 2004.
18. De Lucia, E. Pompella, S. Stefanucci, “Assessing effort estimation models for corrective maintenance through empirical studies”, Journal of Information and Software Technology, vol. 47, pp. 3–15, 2005.
19. Vu. Nguyen, “Improved size and effort estimation models for software maintenance”, An Unpublished Ph.D. Dissertation, University of Southern California, Los Angeles, CA,viewed 20 December 2014
20. Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948.

21. Chen, Wei-neng; Zhang, Jun (2010). "A novel set-based particle swarm optimization method for discrete optimization problem". IEEE Transactions on Evolutionary Computation. 14 (2): 278–300. doi:10.1109/tevc.2009.203033
22. PROMISE (PRedictOr Models in Software Engineering) http://code.google.com/p/promisedata/
23. Ultimate Debian Database (UDD) http://udd.debian.org/
24. Bug Prediction Dataset (BPD): http://bug.inf.usi.ch/
25. Eclipse Bug Data (EBD): http://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/