Volume 8, No. 5, May-June 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Comparison of Maintenance Activity for Effort Estimation in Open Source Software Projects

Avneet Kaur
Centre for Computer Science and Technology
Central University of Punjab
Bathinda, India

Dr. Satwinder Singh
Centre for Computer Science and Technology
Central University of Punjab
Bathinda, India

Abstract:Software estimation accuracy is amongst the biggest challenges for software developers. The most significant activity in software project management is Software development effort prediction. Many modelshave been proposed to make software effort estimations, but still no single model can predict the effort accurately. Thedemand for accurate effort estimation in the software industry is still a challenge. Accurate, precise and reliable estimates of effort at early stages of project development hold great importance for the management to meet the competitive demands of today's world. Software cost estimation is one of the most crucial tasks and predicts the effort and development time needed to develop a software system. It helps the software industries to manage their software development process efficiently. In this paper, we introduce an approach to building an effort estimation model for Open Source Software. Forthis purpose, effort data is mined from the developer's bug fix activities history. Ourapproach determines the actual time spend to fix a bug and considers it as an estimated effort. We propose an artificial neural-network-based approach to predict the amount of effort and development time of developers required for bug resolution. This paper investigates the use of Back-Propagation Neural networks for software effort estimation. The primary purpose of this paper is to estimate the software development effort using Artificial Neural-Network based techniques to improve accuracy of developers for bug resolution.

Keywords: Software effort estimation, Artificial neural networks, Back-propagation neural networks, Accuracy, Prediction.

I.INTRODUCTION

In software engineering, Effort estimation is one of the essential activities in the development of the software. Estimation of software is a difficult task in project planning and management process. [2]Software effort estimation one of the most long-term problems in software engineering. The software is the most expensive component in many computer-based systems. A huge quantity of bugs produces a vast difference between gain and loss during the estimation of effort. [3] Software effort estimation is the process of evaluating effort needed to produce or maintain software based on insufficient, unpredictable input. An effort is used to the total time that takes developers of a software development team to perform a given task. It is usually expressed regarding person-day, person-month, and person-year. This value is important as it helps in estimating other values, like cost or total time required to produce a software product appropriate for software projects. In Open Source Software projects development there is a mutual understanding among open source developers, reporters, and users that capably improves the product quality. However quality and efficiency of Open Source Software depend upon the bugs present in the software, so tracking of bug is important. Bug tracking system plays a significant role in the tracking of bug.

The importance of effort estimation becomes critical during the early stage of the software lifecycle when the software details have not been reported. The effort required in developing a software product plays a significant role in determining the success or failure. Software bugs commonly arise during software development process. Unfound bugs can lead to the loss of billions of dollars. The aim of software maintenance is to not only improve the performance but also fix bugs and enhance features of the software, which lead to better quality software [1]. In recent years, software maintenance has become more challenging due to the increasing number of bugs in large-scale and complex software programs. The earlier studies [2] showed that a significant amount of software development cost is spent on maintenance and development activities. To regulate and avoid overlapping efforts, in large-scale open source software projects, project teams commonly use the bug tracking systems such as Bugzilla to keep track of reported software bugs. An essential component is the bug repository that stores the bug reports, source code and changes history i.e. when a bug is reported and assigned which is produced by users and developers. A bug tracking system is a place that keeps track of software project bugs in the database. The bug report is stored in bug tracking system where assignee of the bug fixes that bug. In open source software environment, the user of open source software often writes a "bug report" when they find a bug or come across a slight mistake. The developers in project teams depend on them to manage and fix the reported bugs. In the software maintenance process for large-scale software programs, especially bug reports, become an important source to help developers to resolve those bugs. Specifically, a user or developer can report the software bug in a specified format (i.e., bug report) and upload it to a bug tracking system such as Bugzilla. Then, a senior developer is assigned to fix the reported bug according to developer specialization and the information shown in the submitted report.

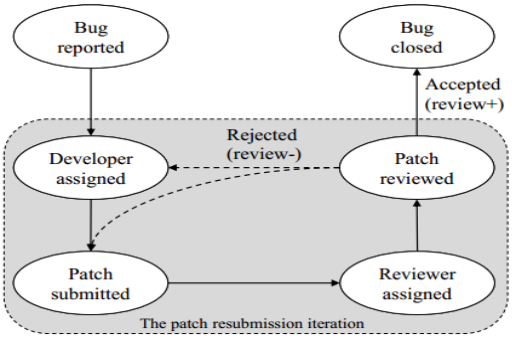

Figure 1: Bug Lifecycle

During the development of any large software system, developers are frequently changing the source code lines of program files. The changes in source code are presented to improve the product quality, i.e., to introduce new features, or to fix detected faults in the software. Version control systems contain source code changes from which effort estimation systems get the data that allows for resolving product costs issues and release management issues. Besides, effort estimation also provides knowledge about the complexity of the product. An additional important feature of actual effort estimation systems is the consistent and exact schedule of release dates for the product. Accurate scheduling of release dates of product always increase the demand of the product and reduce the costs of the project.

In this paper, a method is present to build an effort estimation model for OSS projects. For this purpose, make a developer activity log-book to manage the development activities of developers and other contributors who are involved in the overall process. To mine effort data we have to answer questions:

1) How is effort data mined from the history of developer's bug-fix-activity?

2) How is effort estimation model build using mined effort data?

3) How bug-fix-activity data is use to construct the developer's activity log-book?

4) How bug-fix-activity data is used to estimate developer's contribution and visualize collaborations?

To tackle questions our paper helps in:

1) Develop a novel approach to mine the effort done by developers from software bug repository.

2) Building a developer's log-book that helps in maintains the developer's bug fix activity records.

3) The log-book is maintained to get the actual time given by each developer f
or bug fixing.

Moreover, we introduce a way of visualizing the different aspects of the collaboration among the developers.

4) We present empirical results obtained when applying the method for effort estimation on the Eclipse project. To perform, we downloaded program file bug-fix-activity data of developers from Eclipse's Bugzilla repository. Bugzilla is a bug tracking system and very commonly used in OSS development.

So far, the method which is use in the neural network for predicting software effort estimation is back propagation trained multilayered feed-forward networks amidst sigmoidal activation function. Artificial Neural Network (ANN) uses machine learning and pattern recognition methodology [23] to find accurate estimates for software development effort. It is found that ANN improves the performance of effort estimation by mean absolute error [24]. ANN (Artificial Neural Network) can discover relationships between the dependent and independent variables. There are some flaws that restrict it from being accepted as the standard manner in effort estimation. Slow convergence is the major drawbacks of the back propagation learning algorithm. The sigmoid activation function used in its hidden and output layer units is the foremost reason for slow convergence.

II. PREDICTION MODELS USED

1. Neural Networks: A great challenge for project managers and developers is predicting software development effort with higher accuracy. A large number of different prediction models (estimation models) have been proposed in past years. Many issues are there that should be covered in the choice of a prediction model, and it is likely that there is a need to made trade-offs in the process. The most common objective is to maximize the prediction accuracy. ANNs are massively parallel systems encouraged by the architecture of biological neural networks, which comprised of simple interconnected units. These interconnected units are known as artificial neurons. The neuron computes a weighted sum of its inputs and produces an output if the sum exceeds a certain threshold. This output then becomes a positive or negative input to other neurons in the network which are connected. The process lasts until one or more outputs are generated. The ANN is modified with random weights and learns the relationships contained in a training data set by adjusting its weights when performed with these data. Among the number of available training algorithms, the error back propagation is the most used by software metrics researchers. In general, the use of ANNs is to predict software development effort have focused mostly on the accuracy comparison.

Back-propagation trained feed-forward neural networks are developed by first selecting an appropriate design of neurons. This includes how many layers of neurons will be used, the number of neurons in each layer, and how the neurons will be interconnected to each other and the precise nature of neurons, such as their transfer function and parameters for training the algorithm. Once the architecture has been formed, the network is trained by giving it with a series of inputs and the correct output from the training data. As with all empirically based modeling techniques, data should be withheld for verification and validation purposes. The network learns by adjusting its weights to decrease the error between its predicted output and actual output. This process of training continues until the network's ability to generalize as measured by its predictive performance on new data is ideal.

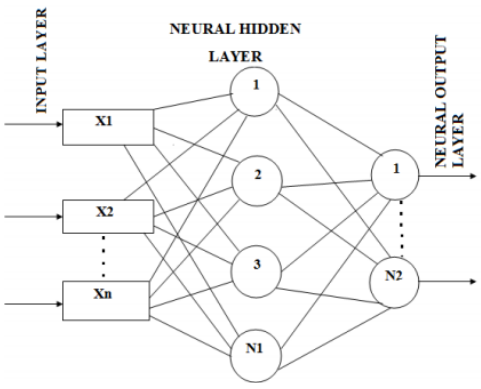

Figure 2: Structure of Multilayer ANN feed forward network

2.Logistic Regression: Logistic regression follows a statistical approach for examining a dataset. The result is determined by one or more independent variables. The outcome measure with a dichotomous variable (two desirable results). The dependent variable is dichotomous or

in binary form, i.e. it only contains data coded as 1 (TRUE, success) or 0 (FALSE, failure) in logistic regression. Logistic regression is an alteration of simple regression. Logistic regression is used when the dependent or response variable is a dichotomous variable, and the independent or input variables are continuous, categorical, or both (Hair et. al. 1998). Logistic regression in software cost estimation provides realistic interval predictions where the cost will lie. Log-likelihood function is more convenient to work with as it is a monotonically increasing function, and the logarithm of function reaches its maximum value at the same points as the function itself. The statistical techniques like decision tree and logistic regression are much similar in their process to analyze the dataset. The Logistic Regression Equation is

$$\text{Log}(p/1\text{-}p) = \beta_{0+}\beta_1X_{1+}\beta_2X_2\ldots\ldots\ldots\ldots\ldots\ldots\ \beta_kX_k$$

In this model, p is the probability that the dependent variable Y=1 and X1,X2,.. ., Xk are the independent variables (predictors). β0 is a constant and β1,β2,.... βk are known as the regression coefficients, which have to estimate from the data. Logistic regression calculates the probability of a specific event occurring. Logistic regression thus forms a predictor variable (log (p/(1-p)) which is a linear combination of the explanatory variables. The values of this predictor variable transform into probabilities by a logistic function. Such a function has the shape of an S.

III.LITERATURE REVIEW

Software effort estimation is a fundamental component to software cost estimation. Software effort estimation methods divided into four categories—empirical, regression, theory-based, and machine learning techniques. Empirical techniques comprise of analogy, function points (FP), and rules of thumb [7]. Regression techniques use parametric and nonparametric models [6]. The theory-based methods use the underlying theoretical considerations characterizing some aspects of software development processes [4]. COCOMO and the SLIM model are the examples of theory-based techniques. Machine Learning (ML) techniques for predicting software effort involve Artificial Neural Networks (ANNs), Classification and Regression Tree, Case-based Reasoning, Genetic Algorithm (GA), Genetic Programming (GP), and Rule Induction (RI) [5]. Jorgensen [8] presents a comprehensive study on the software development effort. Many software cost estimation models have been evolved over the years. The advantage of Neural networks have its ability of learning and are good for modeling complicated nonlinear relationships; delivers more flexibility to corporate expert knowledge into the model. Neural networks technique is applied by many researchers to measure software development effort [9, 10, 13,14 17, and 18]. Many models of NN have been introduced [11,15,18]. Models may be grouped into two broad categories. First one is feed-forward neural networks where there is no loops in the network path take place. Another one is feedback neural networks that have recursive loops in the network path. The most common method used in the area of effort estimation is the feed-forward multilayer perceptron with back propagation learning algorithm. An investigation by Samson et al. [14] practices an Albus multilayer perceptron to predict software effort. They work on Boehm's COCOMO dataset. Srinivasan and Fisher [17] consider the use of a neural network with a back propagation learning algorithm. They observed that the neural network overtook other techniques. Karunanithi et al. [12] work in the use of the neural network in estimating software effort produced very precise results, but the drawback is due to resultant accuracy massively depends on the size of the training set.In [19], Gavin R. Finnie and Gerhard E. Witting proposed models for predicting effort using neural networks and Case-Based Reasoning (CBR) by comparing with various versions of FP-based Regression Models and Neural Networks(NN). The data used comprised of 299 projects from 17 different organizations and concluded that NN is done better than analogy followed by regression models. Their performance to a great extent dependent on the dataset on which they are trained. According to Gray and McDonell [20], NNs are the most common software effort estimation model-building practice used as an alternative to mean least squares regression. These are estimation models that can be "trained" using historical data to produce positive outcomes by automatically modifying their algorithmic parameter values to reduce the error between known actual and model predictions results. Anita Lee et. al [21] combined NN with cluster analysis for software development cost estimation and determined that it proved to be a capable approach to providing more accurate results on the forecasting of software development costs or effort. Additionally, it shows that the combination of NNs with cluster analysis increases the training efficiency and performance of the network, which in turn results in more accurate cost estimation than using just NNs. K.K. Shukla[22] presented a genetically trained NN on historical data. It established significant development in prediction accuracy as compared to both a regression-tree-based approach, as well as back propagation trained NN technique.

IV.RESULTS AND DISCUSSION

Effort Estimation is a complex activity that requires knowledge of many key attributes. At the beginning of a project, there is risk about these project attributes. The NNs have been used in various phases of software development right from planning phase for effort estimation to software testing, software quality assurance as well as reliability prediction. This paper presented background information on software project models and software metrics to be used for effort and cost estimation. We have predicted the software project effort to solve bug using Multilayer Perceptron using Backpropagation Algorithm and logistics regression. The aim is to construct a prediction model with the estimation accuracy of both the prediction model so that the estimated effort and the actual effort nearly equal. Implementation is done using WEKA tool. The results obtained from the trained neural network are compared with that of the Logistic Regression model. The obtained results suggest that the recommended architecture of Neural Network can precisely forecast the software development effort. The evaluation criteria are used to assess and compare the performance of the neural network model, and logistic regression is the magnitude of relative error (MRE). Results obtained are given below.

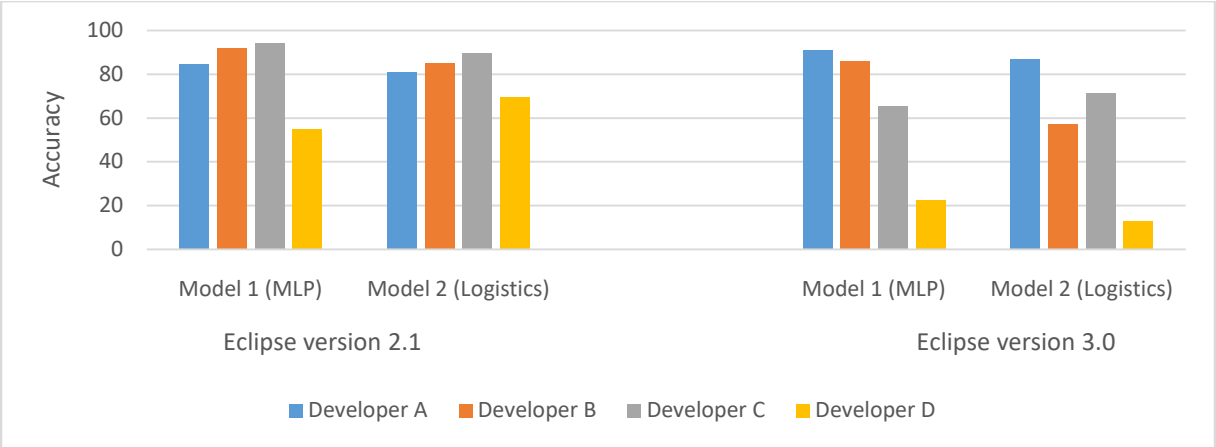

Figure 3: Accuracy Graphs for Eclipse Version 2.0 applied on Eclipse Version 2.1 and 3.0

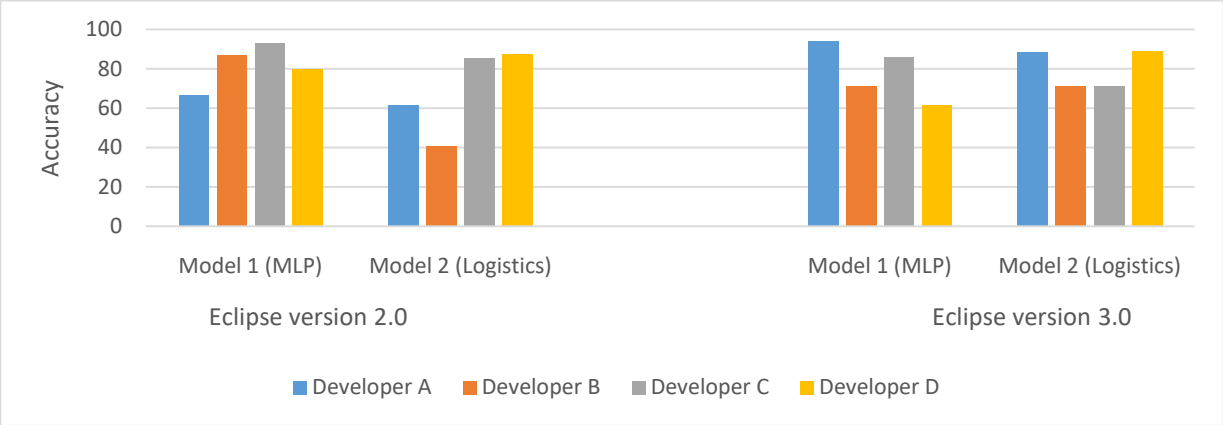

Figure 4: Accuracy Graphs for Eclipse Version 2.1 applied on Eclipse Version 2.0 and 3.0

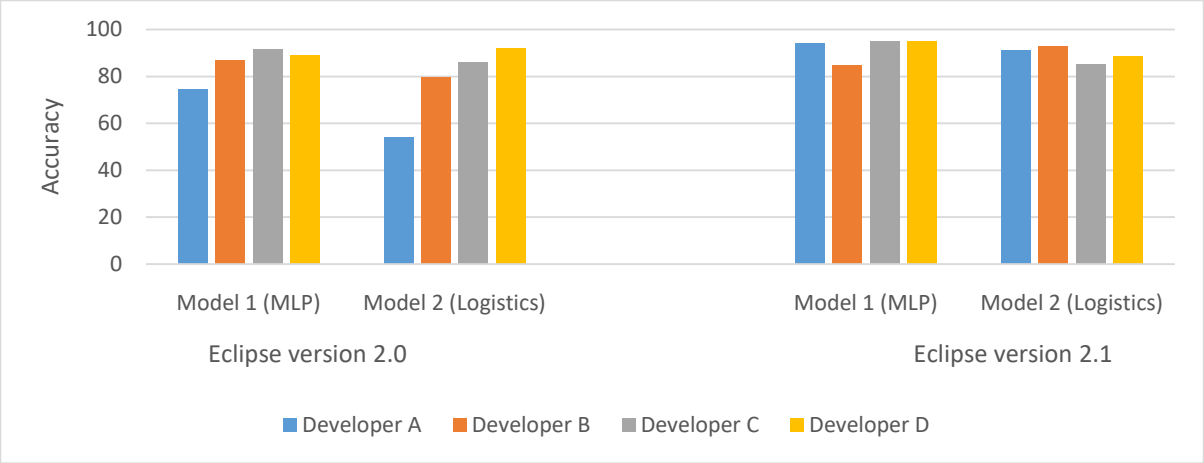

Figure 5: Accuracy Graphs for Eclipse Version 3.0 applied on Eclipse Version 2.0 and 2.1

V. CONCLUSION

From the obtained results it is concluded that Neural Networks results in higher accuracy than logistics regression for effort made by developers for bug resolution.

VI.REFERENCES

[1] R. Charette, "Why Software fails [software failure]," in IEEESpectrum, **42(9)**, 2005, pp. 42–49.

[2] J. Lee, W. Lee, J-Y Kuo, "Fuzzy Logic as a Basic for Use Case Point Estimation," in IEEE International Conference on Fuzzy Systems, Taipei, Taiwan, June 27-30, 2011, pp. 2707-2707.

[3] K. Hamdan, M. Madi, "Software Project Effort: Different Methods of Estimation," in International Conference on Communications and Information Technology (ICCIT), Aqaba., 2011, pp. 15-18.

[4] R.E. Fairley, "Recent Advances in Software Estimation Techniques," in Proceedings of the 14th International Conference on Software Engineering., ACM, 1992, pp. 382-391.

[5] C.J. Burgess and L. Lefley, "Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation," Information and Software Technology, vol. 43, no. 14, 2001, pp. 863-873.

[6] A.R. Gray, S.G. MacDonnel, and M.J. Shepperd, "Factors Systematically Associated with Errors in Subjective Estimates of Software Development Effort: the Stability of Expert Judgment," in Proceedings of Sixth International Software Metrics Symposium, IEEE, 1999, pp. 216-227.

[7] C. Jones, "By Popular Demand: Software Estimating Rules of Thumb," Computer, vol. 29, no. 3, March 1996, p. 116.

[8] M. Jørgensen, "A Review of Studies on Expert Estimation of Software Development Effort," in Journal of Systems and Software, Volume 70 (1), 2004, pp. 37-60.

[9] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," in Journal of Information and Software Technology, Volume 44 (15), 2002, pp. 911–922.

[10] R.T. Hughes, "An evaluation of machine learning techniques for software effort estimation," University of Brighton, 1996.

[11] M. Jorgerson, "Experience with accuracy of software maintenance task effort prediction models," IEEE Transactions on Software Engineering, Volume 21 (8), 1995, pp. 674–681.

[12] N. Karunanithi, D. Whitley, Y.K. Malaiya, "Using neural networks in reliability prediction," IEEE Software, Volume 9 (4), 1992, pp. 53-59.

[13] C.F. Kemerer, "An empirical validation of software cost estimation models," Communications of the ACM, Volume 30 (5), 1987, pp. 416–429.

[14] B. Samson, D. Ellison, P. Dugard, "Software cost estimation using an Albus perceptron (CMAC)," in Journal of Information and Software Technology, Volume 39 (1), 1997, pp. 55–60.

[15] C. Schofield, "Non-algorithmic effort estimation techniques," Technical Report TR98-01, 1998.

[16] C. Seluca, "An investigation into software effort estimation using a back propagation neural network," M.Sc.Thesis, Bournemouth University, UK, 1995.

[17] K. Srinivasan, D. Fisher, "Machine learning approaches to estimating software development effort," IEEE Transactions on Software Engineering, Volume 21 (2), 1995, pp. 126–137.

[18] G. Wittig, G. Finnie, "Estimating software development effort with connectionist models," in Journal of Information and Software Technology, Volume 39 (7), 1997, pp. 469–476.

[19] G.R. Finnie, G.E. Wittig, "AI tools for software development effort estimation," in Proceedings of Conference on Software Engineering and Education and Practice, IEEE Computer Society Press, Los Alamitos, 1996, pp. 346–353.

[20] A.R. Gray, S.G. MacDonnell, "A Comparison of Techniques for Developing Predictive Models of Software Metrics," Information and Software Technology, Volume 39(6), 1997, pp. 425–437.

[21] A. Lee, C.H. Cheng, J. Balakrishan, "Software Development Cost Estimation: Integrating Neural Network with Cluster Analysis," Information and Management Volume 34(1), 1998, pp. 1–9.

[22] K.K. Shukla, "Neuro-Genetic prediction of software development effort," in International Journal of Information a3nd Software Technology, Volume 42(10), 2000, pp. 701–703.

[23] G. Wittig and G. Finnie, "Estimating software development effort with connectionist models," Information and Software Technology, Volume 39(7), 1997, pp. 469-476.

[24] M. Sharif, M. Yasmin, S. Mohsin, "Neural Networks in Medical Imaging Applications: A Survey," World Applied Sciences Journal, Volume 22(1), 2013, pp. 85-96