Estimating Effort in Global Software Development Projects Using Machine Learning Techniques

Mamoona Humayun and Cui Gang

Abstract—**Effort estimation is crucial for the control, quality and success of any software development project and it become even more crucial in GSD where stakeholders are from different background and interest and there is a huge cultural, linguistic and temporal difference involved between them. Software Effort estimation techniques fall under the categories of Expert judgment, Algorithmic estimation and Machine Learning. In this study a comparative analysis is made between traditional effort estimation techniques and ML techniques. Results show us that ML methods give us more accurate effort estimation as compared to the traditional methods of effort estimation. Moreover the comparisons of different ML techniques are done in this paper to study that which ML method is more suitable in which situation.**

Index Terms—**Global software development (GSD), artificial intelligence (AI), machine learning (ML)**

I. Introduction

Global software development is becoming a common practice in software industry and organizations are now shifting from traditional form of collocated development to global software development rapidly [1,] [2]. GSD is characterized by stakeholders from different locations having different culture and background, collaborating by means of information and communication technologies to develop software systems [2], [3].

With the advent of modern technologies, specially the internet facility, the communication and coordination between remote sites have become a practical option and have aided much in promoting GSD [4]. There are a number of benefits and business reasons that motivate companies to shift from in-house development to GSD; these reasons include latest technologies, availability of resources and methodologies, being closer to emerging markets, low cost etc [5].

Effort estimation is important for successful management and quality of software development projects. It is measured in terms of person months and duration. Both overestimation and underestimation of software effort may lead to risky consequences and sometime even causes the failure of a project. It plays an important role in judging the cost of the project and to avoid budget overrun [10].

Software industry is a major industry contributing toward the world economy. Because of competitive market, there is increasing pressure on companies to deliver good quality software within time and budget control. So software industries need to create a balance between quality and cost. Inaccurate estimation of effort causes a great loss to the software industry. A survey by [11] shows that cost overruns of 30-40 percent is very common in software industries and accurate estimation of cost is necessary to avoid this overrun. Effort estimation in early stages of software development has been in the focus of software engineering research for a long time and it has led to the development of several approaches for effort estimation. However these methods usually focus on collocated software development and do not specifically consider GSD projects. GSD is getting more popularity because of its positive impact on productivity but at the same time it takes more development time as compared to collocated sites. Thus cost analysis in GSD projects is more complex as compared to collocated development. Geographical distance between sites increases the need for reliable effort estimation [6].

Failure rate in GSD projects is higher as compared to the collocated projects [7], and GSD projects in particular suffer from time and budge overrun [8, 9], this shows us that work distribution in GSD projects should be done very carefully. Effort estimation for GSD projects is different from collocated one in two dimensions: Firstly, due to the temporal, linguistic and cultural difference there is a long overhead in GSD projects. Secondly many effort driving factors are site specific and cannot be considered globally [12]. In many GSD organizations development sites have different characteristics due to which the productivity of different sites is not the same. Therefore, we need new cost and effort estimation model that specifically address the characteristics of GSD projects and provide us with accurate estimation methods [6].

Software Effort estimation techniques fall under following three main categories:

A. *Expert Judgment*

In this technique an expert of software development processes estimates software development parameters. The accuracy of such estimates highly depends on the degree in which a new project matches within the experience and the ability of the expert. A recent experiment shows high degree of inconsistency in expert judgment-based estimates of software development effort [11], [15].

B. *Algorithmic Estimation*

It contains mathematical formula to relate independent variables (such as cost drivers) to dependent variable (such as effort, cost), such as regression models, Constructive Cost models etc. One well known model for estimation is COCOMO model developed by Barry Boehm [13].

C. *Machine Learning*

During the last two decades researchers have been focused

Manuscript received March 11, 2012; revised April 13, 2012.

M. Humayun is with HIT Harbin, China (e-mail: mamoona@hit.edu.cn)
Cui Gang is with HIT, Harbin, China (e-mail: cg@hit.edu.cn)

on exploring a new approach using AI based techniques for accurate effort estimation. This approach uses ML a sub field of AI for estimating effort of GSD projects [11]. Commonly used ML techniques are artificial neural networks (ANN), case based reasoning (CBR), Rule Induction (RI), Genetic Algorithm (GA), Classification and regression trees (CART) and Multiple additive regression trees (MART) etc.

It is difficult to determine which technique gives more accurate result on which dataset. However, a lot of research has been done in Machine learning techniques of estimation and Literature suggests that ML methods are capable of providing adequate estimation models as compared to the traditional models especially in GSD projects [11]-[22].

ML algorithms offer a practical alternative to the existing approaches to many SE issues. Fig. 1 shows the relation between ML and Software Engineering.

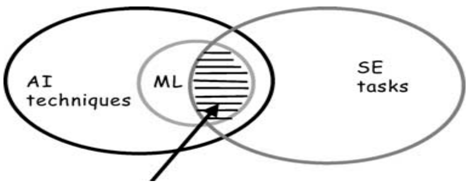

Fig. 1. Relation between ml and software engineering

During last two decades Artificial Intelligence based models are attracting researcher's attention for the estimation of software parameters. In 1995 [23] have compared AI based techniques with traditional COCOMO, Function Point Analysis and Software Lifecycle Management (SLIM) and concluded that AI models are viable to traditional methods. Authors of the paper [24] have concluded that AI based models are capable of providing acceptable estimation models. Below we will describe some commonly used methods of ML for measuring effort and in the next section we will compare these methods, so that this paper may help the practitioners and researchers in the selection of suitable effort estimation methods.

Commonly used ML methods for measuring effort in GSD projects are as follows

D. Artificial Neural Network (ANN)

ANN is a computational or mathematical model that is stimulated by the biological human brain. Through learning process ANN can be configured for a specific application, such as pattern recognition or data classification. ANNs include the two basic components of biological neural networks that are Neurons (nodes) and Synapses (weights). A neuron has a set of *n* (number neurons in previous layer) synapses (inputs), which are characterized by *n* different weight (free parameters).

E. Feed-Forward Neural Network (FFNN)

Many neurons are used in the construction of an FFNN; these neurons are connected with each other through specific network architecture. The primary goal of the FFNN is to transform the inputs into meaningful outputs. There is no self loop or backward feed in this network [11].

F. Radial Basis Neural Network (RBNN)

There is no main difference in the network architecture of RBNN compare to FFNN. The only difference is that the radial basis layer of RBNN is located like Hidden layer in FFNN architecture. Radial basis Layer contains different type of neurons, which contains Radial Basis Function (RBF) as an activation function [11].

G. Case-Based Reasoning (CBR)

It is a cyclic procedure that is composed of four stages:

1. Retrieval of similar cases,
2. Reuse of the retrieved cases to find a solution to the problem,
3. Revision of the proposed solution if necessary
4. Retention of the solution to form a new case.

When a new problem arises, a possible solution can be found by retrieving similar cases from the case repository. The solution may be revised based upon experience of reusing previous cases and the outcome retained to supplement the case repository. Consequently, issues concerning case characterization, similarity and solution revision must be addressed prior to CBR system deployment.

H. Rule Induction (RI)

In RI general concepts can be obtained from specific examples. To derive these general concepts many existing examples are analyzed, and this general concept defines the production conditions. The process of producing a set of rules is done randomly or sometime algorithmically and the subset of examples that are selected for this purpose are often referred to as the training set. These rules can be tested on the rest of the examples (the validation or test set) to assess how well they represent the data. RI can be used for such kind of problems where there exists a set of suitable examples. Rules can be seen as decision trees where the leaf node contains the predicted value or range of values. Numeric decision trees are generated by calculating the average output for the set of cases being considered at each node [25].

I. Regression Trees

It is a prediction model that can be used in almost every field. It analyses the relationship between a dependent variable and one or more independent variable to find the best fit. It helps in understanding that how dependent variable change its value as a result of change in independent variable. The standard equation of regression analysis is $Y= f(x)$ where many kind of function can be used in $f(x)$ such as exponential, logarithmic and linear function etc.

J. Classification and Regression Tree Algorithm (CART)

CART is a binary decision tree algorithm that is used in data mining problems involving classification and regression. The CART constructs a binary decision tree by splitting a data set in such a way that the data in the child subsets are more unadulterated than the data in the parent set. The tree continues to grow until a node is reached such that no significant decrease in re-substitution estimate is possible. This node is the terminal node.

K. Multiple additive Regression Tree (MART)

MART extends CART using boosting. Boosting is a process that is used to increase the accuracy of any learning algorithm by fitting a series of model with low error rate and then aggregate them into a model that may perform better. MART models posses all the advantages of tree based-model and overcome the disadvantage of inaccuracy. The main features of MART are Selection of automatic variable subset, Ability to handle data without pre-processing, Resistance to outliers, handling missing values automatically, Ability to detect dirty and partially inaccurate data, High speed and less training requirement.

L. Genetic Algorithm (GA)

The technique of GA was developed for handling general optimisation problems with large search apace.GA need no prior knowledge, expertise or logic related to the problem under study. The basic process of GA is as follows

1) Generate a set of solutions(family of chromosomes) randomly
2) Applying GA to the fittest chromosomes to create a new population from the previous one
3) Repeat step 2 until the required number of generation has been produced

The best solution in the final round is taken as a best approximation for that problem [26].

II. Comparison of ML methods of Effort Estimation

The traditional approaches of effort estimation do not address the characteristics of distributed development in great detail, e.g. COCOMO has only one effort multiplier for multi-site development but as this is only a single number, it cannot reflect the inherent complexity and various overhead drivers of global software development. The approach selected for GSD projects should consider the characteristics of different sites involved in GSD. Effort estimation approaches should predict the effort with respect to the site-specific characteristics.

In [11] Neural network approach is compared with regression analysis for software development effort estimation. And result of this study matches with some existing studies [17]-[29] that neural network approach is better than regression analysis.

In [14] models for predicting effort using neural networks and Case Based Reasoning (CBR) are proposed. They compared analogy based method using CBR with different versions of FP-based Regression models and NN. The data used consisted of 299 projects from 17 different organizations and concluded that NN performed better than analogy followed by regression models.

According to Gray and McDonnell, ANNs are the most common software estimation model-building technique used as an alternative to mean least squares regression [30]. Capability of MART in estimating software project effort has been empirically evaluated in [15]. In this paper the technique of MART is applied on a well-known NASA software project dataset and its estimation accuracy was compared with existing models. The result shows that improved estimation accuracy is achieved by applying MART when compared with linear regression, radial basis function neural networks, and support vector regression models.

In this paper three commonly uses ML methods (ANN, CBR, RI) were compared, the dataset used for this comparison was comprised of 81 software projects derived from a Canadian software house in the late 1980s. ANN technique was considered as best however the author of the paper suggests that it may be other characteristics of these techniques that will have an equal, if not greater, impact upon their adoption [25].

In this paper the regression models, ANN and CBR techniques are applied on the data of 299 projects from 17 different organizations. The results show that Regression models do not perform very effectively in modeling the complexities involved in software development projects. Author claims that artificial intelligence models are capable of providing adequate estimation models. While both ANN and CBR provide better and somewhat accurate estimate of software effort [24].

III. Results

On the basis of above discussion it can be concluded that ML methods provide us good effort estimation results as compared to the traditional methods of effort estimation. However there are many factors that impact software effort estimates, these factors include team size, concurrency, intensity, fragmentation, software complexity, computer platform and different site characteristics in case of GSD. As there are different ML techniques for predicting effort and we cannot say that which technique is better from other rather their performance is to a large degree dependent on the data on which they are trained, and the extent to which suitable project data is available will determine the extent to which adequate effort estimation models can be developed.

References

[1] J. Herbsleb, “Global software engineering: the future of socio-technical coordination,” *Future of software engineering* (FOSE’07), pp. 23-25, 2007.

[2] E. Conchuir, H. Holmstrom, P. Agerfalk and Brian. “Fitzgerland. Exploring the Assumed Benefits of Global Software Development,” ICGSE’06, pp. 159-168, 2006.

[3] A. Gabriela, “Of Deadlocks and People ware - Collaborative Work Practices in Global Software Development. International Conference on Global Software Engineering (ICGSE’07),” pp. 91-102, 2007.

[4] S. Bikram, C. Satish and S. Vibha, “A research agenda for distributed software development,” *International conference on software engineering (ICSE’06)*, pp. 731-740, 2006.

[5] D. Damian and D. Moitra, “Global Software Development: How Far Have We Come?” *IEEE software*, vol. 23, no. 5, pp.17-19, 2006.

[6] L. Ansgar, M.. Jurgen, “Estimating the Effort Overhead in Global Software Development,” *International conference on Global Software Engineering*, 2010.

[7] M. Fabriek, M. van de Brand, S. Brinkkemper, F. Harmsen, and R. W. Helms, “Reasons for success and failure in offshore software development projects,” *European Conference on Information System*, pp. 446-457, 2008.

[8] T. Carter, Cheaper's not always better. *Dr. Dobb's Journal*, March 1, available on: http://www.ddj.com/184415486 (accessed at Feb 25 2010),2006.

[9] G. Seshagiri, GSD: Not a business necessity, but a march of folly. *IEEE Software*, vol. 23, no.5, pp. 63-64,2006.

[10] M. Ruchika and J. Ankita, “Software Effort Prediction using Statistical Machine Learning Methods,” (IJACSA) *International Journal of Advanced Computer Science and Applications*, vol. 2, no.1, 2011

[11] S. Vachik and S. Dave, “vachik and Dutta. Kamlesh. Comparison of Regression model,” Feed-forward Neural Network and Radial Basis

Neural Network for Software Development Effort Estimation. ACM SIGSOFT Software Engineering Notes, vol. 36, no. 5, 2011.

[12] R. Madachy, "Distributed global development parametric cost modeling. International Conference on Software Process," (ICSP '07), pp. 159-168, 2007.

[13] W. B. Boehm, Chris. A. Abts, "Winsor brown, Sunita. Chulani," Bradford k. Clark, Ellis. Horowitz, Ray. Madachy, Donald J. Reifer and Bert. Steece. Software Cost Estimation with COCOMO II. Englewood Cliffs, NJ, USA: Prentice-Hall.2007.

[14] G. R. Finnie and G. E. Witti. "AI Tools for Software Development Effort Estimation," *International conference on Software Engineering: Education and Practice*, 1996.

[15] M. O. Elish, "Improved estimation of software project effort using multiple additive regression trees," *Expert Systems with Applications* pp. 10774–10778, 2009.

[16] B. Baskeles, B. Turhan, and A. Bener, "Software effort estimation using machine learning methods," In *Proc. of 22nd international symposium on computer and information sciences*, 2007.

[17] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," *Information and Software Technology*,pp 911–922, 2004.

[18] E. Jun. and J. Lee, "Quasi-optimal case-selective neural network model for software effort estimation," *Expert Systems with Applications*, pp. 1–14,2001.

[19] A. Oliveira, "Estimation of software project effort with support vector regression," Neurocomputing, pp.1749–1753, 2006.

[20] H. Park. and S. Baek, "An empirical validation of a neural network model for software effort estimation," *Expert Systems with Applications*, vol. 35, no. 3, pp. 929–937, 2008.

[21] M. Shin, and A. Goel, "Empirical data modeling in software engineering using radial basis functions," *IEEE Transactions on Software Engineering*, vol. 26, no. 6, pp. 567–576, 2000.

[22] A. Idri, A. Abran, and T. Khoshgoftaar, "Estimating software project effort by analogy based on linguistic values," In *Proc. of 8th IEEE symposium on software metrics*, pp. 21–30, 2002.

[23] K. Srinivasan, and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort," *IEEE Transaction on Software Engineering*, vol. 21, pp. 126-137, 1995.

[24] G. R. Finnie and G. E. Wittig, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks," *Case-Based Reasoning and Regression Models. J.SYSTEMS SOFTWARE*, pp.281-289, 1997.

[25] M. caroly, K. Gada, E. Le, Martin, and P. Keith, "Schofield. Chris, Shepperd. Martin and Webster. Steve. An investigation of machine learning based prediction systems," *The Journal of Systems and Software*, pp. 23-29, 2000.

[26] J. B. Colin and L. MartiCan, "Genetic programming improves software effort estimation? A comparative evaluation. Information and software technology, pp. 863-873, 2001.

[27] J. Kaur, S. Singh, K. S, Kahlon, and P. Bassi, "Neural Network-A Novel Technique for Software Effort estimation," *International Journal of Computer Theory and Engineering*, vol. 2, no. 1. pp. 17-19, 2010.

[28] R. Bhatnagar, V. Bhattacharjee, and M. K. Ghose, "Software Development Effort Estimation - Neural Network vs," *Regression Modeling Approach. International Journal of Engineering Science and Technology*, vol. 2, pp.2950- 2956, 2010.

[29] P. C. Pendharkar, "Probabilistic estimation of software size and effort," *Expert Systems with Applications*, pp. 4435–4440, 2010.

[30] A. R. Gray, and S. G. MacDonnell, "A Comparison of Techniques for Developing Predictive Models of Software Metrics," *Information and software technology*, pp. 425–437, 1997.