www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 7 Issue 9 September 2018, Page No. 24206-24213
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v7i9.02

Software Effort Prediction with Algorithm Based Frameworks

Shrikant Pawar*[1, 2] & Aditya Stanam[3]

[1]Department of Computer Science, Georgia State University, 34 Peachtree Street, 30303, Atlanta, GA, USA.
[2]Department of Biology, Georgia State University, 34 Peachtree Street, 30303, Atlanta, GA, USA.
[3]University of Iowa, College of Public Health, UI Research Park, #219 IREH, Iowa City, Iowa 52242-5000, USA

Abstract
The need for accurate effort predictions for projects is one of the most critical and complex issues in the software industry. Effort estimation is an essential activity in planning and monitoring software project development to deliver the product on time and within budget. The competitiveness of software organizations depends on their ability to predict the effort required for developing software systems accurately. Several algorithmic approaches have been proposed in the literature to support software engineers in improving the accuracy of their estimations. The decision of how to select different techniques considering the characteristics of a specific dataset through genetic algorithms could be viewed as a search-based problem for software engineering. Some groups use well known evolutionary learning algorithms (MMRE, PRED (25), PRED (50) and PRED (75), while others use machine learning methods like artificial neural network and fuzzy logic. Analogy estimation is also being utilized for estimating software effort, which has been shown to predict accurate results consistently. Genetic frameworks are becoming famous for selecting best learning schemes for effort prediction using the elitism techniques, utilizing some well-known parameters like Spearman's rank correlation, the median of the magnitude of relative error (MdMRE), and mean of the absolute residuals (MMAR). The technique of using fuzzy rough sets for the analysis is also becoming popular. This article aims to compare various algorithm based frameworks for predicting software development effort which can generate the best-optimized software product.

Keywords: Machine Learning; Software Effort; Prediction; Algorithm.

Introduction

Effort estimation is a critical activity for planning and monitoring software project development to deliver the product on time and within budget. The competitiveness of software organizations depends on their ability to predict the effort required for developing software systems accurately. Both over or underestimates can negatively affect the outcome of software projects. Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if the input data is incomplete, uncertain, or unstable.[1] Estimation determines how much money, effort, resources, and time it will take to build a specific system. It is based on past data or experience, available documents, assumptions, and identified risks. The four necessary steps in software project estimation are to estimate the size of the development product, determine the effort in person-months or person-hours, expect the schedule in calendar months, and estimate the project cost in agreed currency.[1] Estimation can be done during acquiring a project, planning the project, or the execution of the project as the need arises. Project scope must be understood before the estimation process begins. It is always helpful to have historical project data. Planning requires technical managers and the software team to make an initial commitment leading to responsibility and accountability. Use of at least two estimation techniques to arrive at the estimates and reconciliation of the resulting values is highly recommended. Plans should generally be iteratively allowing adjustments in future.

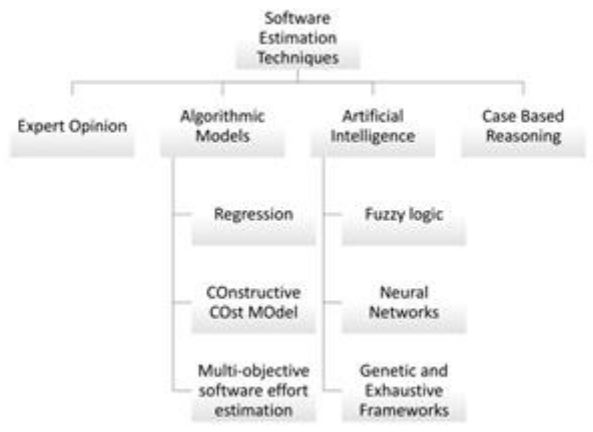

Figure 1: Summary of software effort prediction techniques discussed in the article.

Figure 1 summarizes the software effort prediction techniques discussed in the article. The most widely used project estimation approach is the decomposition technique, which mostly is a divide and conquer approach. Size, effort, and cost estimation are performed in a stepwise manner by breaking down a project into significant functions or related software engineering activities. Before an evaluation is completed, we need to understand the scope of software to estimate the software size.[2] It generally starts with the statement of scope and decomposes the software into functions that can each be evaluated individually. The size of each service is then calculated. Combine function estimates are used to produce an overall rating for the entire project.

Estimation is usually done with effort (in person hours/days) required to complete each task, and the combined effort to produce an estimate for the activity. The cost units (i.e., cost/unit effort) are then determined for each event from the database to compute the total attempt.[2] Combine effort and cost estimates for each activity to produce an overall effort, and cost estimate for the entire project can then be calculated. Next, reconciliation of estimates is performed by comparing the resulting values from previous steps to define if the scope of the project is adequately understood. The range can also be misinterpreted, so the function breakdown should be accurate. The historical data used for the estimation techniques are appropriate for the application, or is obsolete is needed to be verified before estimation.[2]

Software estimation requires accuracy, which is an indication of how close something is to reality. Some important factors that can affect the accuracy of estimates are the accuracy of input data, calculation accuracy, the accuracy of historical data used to calibrate the model matching. The predictability of organization's software development process, the stability of both the product requirements and the environment that supports the software engineering efforts are also significant.[3] Reliable estimates can be achieved by base estimates from similar projects that have already been completed, using relatively simple decomposition techniques to generate project cost and effort estimates. To ensure accuracy, it's always advisable to perform estimation using at least two methods for comparing the results.

There can be many estimation issues; one example is an estimation of schedules by project managers with neglecting project size. This may arise because of the timelines set by the top management or the marketing team. It then becomes difficult to estimate the schedules to accommodate the scope changes. It is also important to note all the assumptions in the estimation sheet. Reasonable estimates may have inherent assumptions, risks, and uncertainty, and may still be treated as accurate. The best way of expressing views is as a range of possible outcomes. For example, the project will take 5 to 7 months instead of stating a particular date for completion.[3] An uncertainty as an accompanying probability value can also be included. For example, a 90% probability that the project will be completed on or before a definite date is also a good deliverance strategy.[3] If there exists a schedule constraint by management or client, negotiation on the scope and functionality to be delivered can be adopted. Failure in accommodating contingency in the final estimate is likely to cause issues. It’s been observed that when resource assignment is more than 80% utilization, there are frequent occurrences in deliverance slippages.[3]

During estimation, it is recommended to ask other people's experiences and put users own experiences at the task. Resources working on multiple projects take longer to complete tasks because of the time lost switching between these projects, so it is always recommended to include management time in any estimate.[3] Contingency building for problem-solving, meetings and other unexpected events to allow enough time to do proper project estimation is advisable. Rushed estimates can be inaccurate and are considered as high-risk estimates. For large development projects, the estimation step is usually regarded as a mini-project.

Usage of documented data from organization’s similar past projects can result in the most accurate estimation. If the organization has not kept historical data, immediate documentation is advisable. Developer-based opinions can also be used for evaluations. Further, utilizing several different people to estimate and use several different estimation techniques will improve estimation accuracies. Reconciliation of the estimates can then be performed to observe the convergence or spread amongst the forecast. Confluence states a reasonable view. Wideband-Delphi technique can also be utilized to gather and discuss estimates using a group of people to produce an accurate and unbiased assessment.[4] The final step is to perform a re-estimation of the project throughout its lifecycle. Software effort models and effort estimates help project managers to allocate resources, control costs and improve current practices, leading to projects that are finished on time and within budget. In the context of software development and maintenance, these issues are crucial, and very challenging, given that projects have short schedules and a highly fluidic scope.

Discussion

Several algorithmic approaches have been proposed to support software engineers in improving the accuracy of their estimations. These methods often produce a point estimate of the effort required to develop a new project.

i. **Early effort estimation techniques:** Prediction is a necessary part of an effective process, whether it be authoring, design, testing, or development as a whole. A prediction process involves the identification of measures that are believed to influence the effort required to develop a new application.[4] The formulation of theories about the relationship between the selected measures and effort is needed. The capturing of historical data (e.g. size and actual effort) about past projects historical data or even past development phases within the same project helps immensely in prediction. The purpose of estimating effort is to predict the amount of effort to accomplish a given task, based on knowledge of other project characteristics that are believed to be related to effort. Project characteristics (independent variables) are the inputs, while its effort (dependent variable) is the output we wish to predict. For example, a web company may find that to predict the effort necessary to implement a new web application will require following inputs: estimated number of new web pages, total number of developers who will help develop the new web application, developers, average number of years of experience with the development tools employed, and the number of functions/features to be offered by the new web application. Figure no 2 illustrates the explained effort prediction process.

ii. **Current methods for effort estimation:** Several techniques for effort estimation have been proposed, we have summarized the most important and effective techniques in this article.

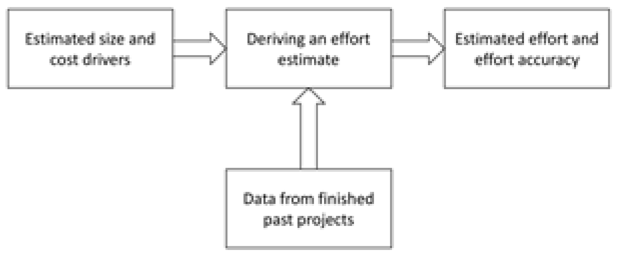
Estimated size and cost drivers
Deriving an effort estimate
Estimated effort and effort accuracy
Data from finished past projects

Figure 2: Illustration of software effort prediction process.

1. **Expert opinion.**
2. **Algorithmic models:** Regression, COnstructive COst MOdel (COCOMO), Multi-objective software effort estimation.
3. **Artificial intelligence:** Fuzzy logic.
4. Case-based reasoning (CBR).

1. Expert opinion: Expert opinion represents the process of estimating effort by subjective means and is often based on previous experience from developing/managing similar projects. It has been and still is widely used in software and web development. The drawback of this technique is that it is challenging to quantify and to determine factors that have been used to derive an estimate. However, studies show that this technique can be a useful estimating tool when used in combination with other less subjective methods (e.g., algorithmic models). An expert looks at the estimated size and cost drivers related to a new project for which effort needs to be determined. Based on the data obtained data retrieval on past finished projects for which actual attempt is known is received and a subjective effort estimation of the new plan is performed. Deriving accurate effort estimation is more likely to occur when there are completed projects similar to the one having its estimated size and cost drivers.

2. Algorithmic models: There are several well-known models, the most effective being regression, COCOMO, and Multi-objective software effort estimation technique.

A. COCOMO model: An example of an algorithmic model that uses equation 1 is the COCOMO model, where parameters and b are based on the type of project under construction, and the *'Effort Adjustment Factor'* (EAF) is based on 15 cost drivers that are calculated and then summed. Parameters and b are chosen based on criteria's like the type of software project being developed. *'EstSizeNewproj'* is the estimated size for the new project.[4] The COCOMO model is an example of a generic algorithmic model, and is believed to be applicable to any type of software project with suitable calibration or adjustment to local circumstances.

Equation 1[4]:

$$EstimatedEffort = a \cdot EstSizeNewproj \cdot b \cdot EAF$$

This technique attempts to formalize the relationship between effort and one or more project characteristics. The central project characteristic used in such a model is usually taken to be some notion of software size (example the number of lines of source code, number of web pages, or the number of links).[5] This formalization is often translated as an equation such as that shown by equation 1. Equation 1 shows that size is the primary factor contributing to effort and can be adjusted according to an EAF, calculated from cost drivers (example developers, experience, tools).[4]

B. Regression based model: Regression-based algorithmic models are most suitable to local circumstances such as "in-house" analysis as they are derived from past data that often represents projects from the company itself. Regression analysis used to generate regression-based algorithmic models provides a procedure for determining the 'best' straight-line fit to a set of project data that represents the relationship between effort (dependent variable) and the project characteristics (example size, experience, tools, the predictor or independent variables).[6] Two of the most widely used techniques are multiple regression (MR) and stepwise regression (SWR).[4] The difference between both is that MR obtains a regression line using all the independent variables at the same time, whereas SWR is a technique that examines different combinations of independent variables, looking for the best grouping to explain the highest amount of variation in the effort. The past data is used to generate a cost model, which then receives input as the values for the new project characteristics. Eventually, this model generates estimated effort. Equation 2 explains a simple linear regression equation, where Yi is the dependent variable, B0, B1, Xi and Ei are population Y-intercept, slope coefficient, independent variable and random error respectively.

Equation 2[4]:

$$Y_i = \beta_0 \dotplus \beta_1 \chi_i \dotplus \varepsilon_i$$

C. Multi-objective software effort estimation algorithm: It combines confidence interval analysis and assessment of mean absolute error. Equations 3-7 explains these thresholds for algorithm implementation.

Equations 3-7[1]:

$$EstimatedEffort = c_1 \; op_1 \; f_1 \ldots c_n \; op_{2n-1} \; f_n \; op_{2n} \; C$$

$$MRE = |RealEffort - EstimatedEffort| / RealEffort$$

$$Pred(25) = k/N$$

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |RE_i - EE_i|$$

$$SA = (1 - MAE_{P_j} / MAE_{r_{guess}}) \cdot 100$$

In the above equations, *fi* represents the value of the *ith* project feature with *ci* as its coefficient, *C* represents a constant, while *opi* represents the *ith* mathematical operator of the model. To compare the performance of prediction models, usage of Mean of Magnitude of Relative Error (MMRE) or Mean Absolute Error (MAE).[4] MMRE is used to have a cumulative measure of the error. *pred(25)* measures the percentage of the estimates whose error is less than l% and it is usually set at 25. For calculating MAE and Standardized Accuracy (SA), *N* is the number of projects used for evaluating the performance, *REi* and *EEi* are the actual and estimated efforts, *MAEPj* is the MAE of the approach being evaluated and *MAErguess* is the MAE of a large number of random guesses. This algorithm is tested on three different alternative formulations, baseline comparators and current state-of-the-art effort estimators. It has been applied to five real-world datasets from the PROMISE repository, involving 724 different software projects. This algorithm outperforms the baseline, state-of-the-art algorithms used for large scale datasets. The accuracy results from this algorithm are promising, significant and with improved accuracy (Table 1).

Table1: Depicts the accuracy results from this multi-objective software effort estimation algorithm (CoGEE) (Reproduced from Wang *et. al. 2016*).

Class	*SA*	*Desharnais*	*SA*	*Finnish*	*SA*	*Maxwell*	*SA*	*Miyazaki*	*SA*
CoGEE	0.48	CoGEE	0.47	CART	0.52	CoGEE	0.56	CoGEE	0.90
GA-SAE	0.48	LR	0.46	CoGEE	0.45	GA-SAE	0.56	LR	0.76
GA-Ct	0.45	GA-SAE	0.45	GA-Ct	0.45	CART	0.51	GA-Ct	0.66
CART	0.40	CART	0.38	LR	0.42	CBB3	0.51	GA-SAE	0.66
CBB3	0.40	CBB3	0.34	CBB3	0.41	CBB1	0.48	N8GAII-UO	0.60
Median	0.38	Median	0.33	GA-SAE	0.41	CBB2	0.47	CBB2	0.56
CBB2	0.35	CBB2	0.32	CBB2	0.38	N8GAII-UO	0.41	CBB3	0.56
CBB1	0.29	CBB1	0.27	CBB1	0.31	LR	0.38	CBB1	0.55
Mean	0.25	Mean	0.26	N8GAII-UO	0.19	Median	0.33	Media	0.49
LR	0.23	GA-Ct	0.09	Mean	0.18	Mean	0.28	CART	0.46
N8GAII-UO	-1.7	N8GAII-UO	0.08	Median	0.14	GA-Ct	0.26	Mean	0.30

3. Artificial intelligence techniques: Artificial intelligence techniques have been used as a complement to, or as an alternative to the previous categories. Some important of them include fuzzy logic, regression trees and neural networks.[7,11] It starts with selecting datasets from the database according the procedure and characteristics, followed by executing the three frameworks, genetic, exhaustive and random guessing for collecting the performance measures.

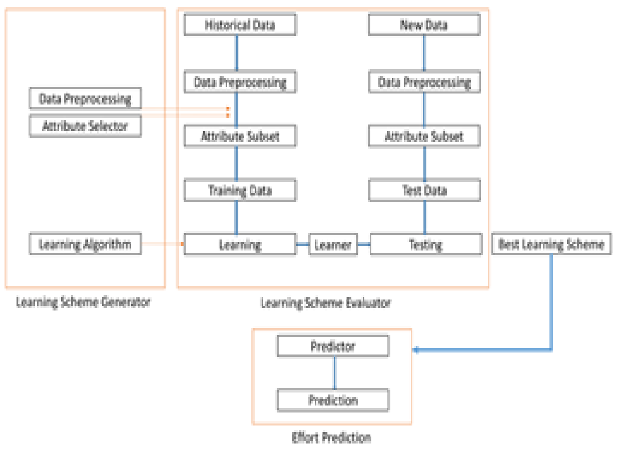

Figure 3: Depicts the workflow of applying machine learning technique study (Reproduced from Murillo *et. al. 2017*).

The genetic and exhaustive frameworks should consist of attribute selectors and learning algorithms. Execution of the genetic framework should be done with different configurations including 3 levels of generations, population, mutation and crossover.[8] This technique have shown to have a prediction accuracy of 45-80%. Figure 3 depicts the workflow of applying artificial intelligence technique study. Some groups have utilized evolutionary algorithms for software effort estimation. K-nearest neighbor (KNN) is used to clean noisy data followed by training and validation.[9] This is done to obtain a clean dataset as most of the effort estimation datasets consists of missing values.[10] Fuzzy learning is based on genetic programming and simulated annealing. Fuzzy algorithms, FRSBM, GFSGPG-R, GFS-GSP-R, and GFS-SAP-Sym-R are applied to the tested datasets.[5] Adaptive Neuro Fuzzy Inference System (ANFIS) can be calculated with equation 8.

Equation 8[5]:

$$RE = (A - E)/A. RE = (RE1 + ... + REn)/n. MRE = (MRE1 + ... + MREn)/n PRED(q) = k/n$$

In equation 8, *RE* is the relative error, *A* is estimate of preset attractiveness and *E* is the error factor. By comparing MRE, PRED and RE values, GFS-SAP-Sym-R is found to be an accurate estimator for dataset of any size (small, medium and large). Ensemble-R has shown worst performance for all 3 datasets (Table 2). Budget of a project can be estimated correctly if applicability of a particular evolutionary technique under various circumstances is known. If the dataset is of small, medium or large size, selection of evolutionary technique can be done accordingly.[5]

Table2: Characteristics of datasets used with MMRE and PRED values of fuzzy logic algorithm application (Reproduced from Wang *et. al. 2016*).

Dataset	*No. of observations*	*No. of features*	*No. of numeric features*	*No. of categorical features*
Desharnais	**82**	**7**	**7**	**0**
Maxwell	**62**	**25**	**3**	**22**
Muyazaki	**48**	**8**	**7**	**1**

Dataset	*Method*	*MMRE*	*PRED(25)*
Desharnais	Ensemble-R	0.59	0.16
	FRSBM	0.38	0.28
	GFS-GPG-R	0.28	0.34
	GFS-GSP-R	0.29	0.33
	GFS-SAP-Sym-R	0.29	0.33
Maxwell	Ensemble-R	1.17	0.12
	FRSBM	0.58	0.25
	GFS-GPG-R	0.57	0.11
	GFS-GSP-R	0.27	0.30
	GFS-SAP-Sym-R	0.29	0.30
Muyazaki	Ensemble-R	0.75	0.12

	FRSBM	0.41	0.20
	GFS-GPG-R	0.61	0.16
	GFS-GSP-R	0.29	0.31
	GFS-SAP-Sym-R	0.27	0.33

4. Case-based reasoning (CBR): Case-based reasoning (CBR) provides estimates by comparing the current problem to be estimated against a library of historical information from completed projects with a known effort (case base).[9] Characterizing a new project *p*, for which an estimate is required, with attributes (features) common to those completed projects stored in the case base. In terms of software cost estimation, features represent size measures and cost drivers. Use of this characterization as a basis by finding similar (analogous) completed projects, for which effort is known. This process can be achieved by measuring the 'distance' between two projects, based on the values of the number of features (*k*) for these projects.[12] Some important features of CBR are, feature subset selection determining the optimum subset of features that yield the most accurate estimation is utilized. Similarity measures the level of similarity between different cases, with several similarity measures proposed in the literature. Scaling also known as standardization, represents the transformation of attribute values according to a defined rule, such that all attributes present values within the same range and hence have the same degree of influence on the results. Number of analogies refers to the number of most similar cases that will be used to generate the estimation. Analogy adaptation, once the similar cases have been selected the next step is to decide how to generate the estimation for project. Adaptation rules are used to adapt estimated effort, according to a given criterion such that it reflects the characteristics of the target project more closely.

Conclusion:

Software project estimation is important to manage cost and resources of organization. There are different ways of performing estimation (expert opinion, algorithmic models, artificial intelligence, case-based reasoning (CBR). By comparing all the mentioned techniques, the artificial intelligence approach seems to be the most accurate and upcoming mode of prediction.

Supporting Information

No external funding has been utilized for this article.

Author Information

Corresponding Author:

Shrikant Pawar

Author Contributions:

SP and AS did the background research, writing, and critical revisions of this manuscript.

Acknowledgments

Not applicable.

References

[1] Sarro F, Petrozziello A, Harman M (2016). Multi-objective Software Effort Estimation. IEEE/ACM 38th IEEE International Conference on Software Engineering.

[2] Huang J, Li YF, Xie M (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. Inf Softw Technol 67: 108–127.

[3] Murillo-Morera J, Quesada C, López C, Herrera Jenkins. (2017). A genetic algorithm based framework for software effort prediction. Journal of Software Engineering Research and Development. 5:4.

[4] Gabrani G, Saini N (2016). Effort Estimation Models Using Evolutionary Learning Algorithms for Software Development. Symposium on Colossal Data Analysis and Networking (CDAN).

[5] Rizkiana R, Riyanarto S, Rochimah S (2017). Accuracy Improvement of the Estimations Effort in Constructive Cost Model II Based on Logic Model of Fuzzy. Advanced Science Letters. 23. 2478-2480.

[6] Mudunuru M, Karra S, Viswanathan H (2017). Regression-based reduced-order models to predict transient thermal output for enhanced geothermal systems. Geothermics. 70. 192-205.

[7] Schmidhuber J (2015). Deep Learning in Neural Networks: An Overview. Technical Report IDSIA-03-14 / arXiv: 1404.7828 v4.

[8] Wang R, Peng P, Xu L (2016). A Novel Algorithm for Software Development Cost Estimation Based on Fuzzy Rough Set. Journal of Engineering Science and Technology Review.

[9] Wang Z, Hamza W, Song L (2017). k-Nearest Neighbor Augmented Neural Networks for Text Classification. Computation and Language. arXiv:1708.07863v1.

[10] Sangwan OP (2017). Software effort estimation using machine learning techniques. 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, Noida. 92-98.

[11] Rao PS, Reddi KK, Rani R (2017). Optimization of neural network for software effort estimation. International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), Chennai. 1-7.

[12] Althoff K (2001). Case Based Reasoning. Handbook of Software Engineering and Knowledge Engineering. 549-587.