ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

The Efficient Way of Estimating the Cost of Software by Exploiting the Assumptions Using Fuzzy Analogy

Disha
Shadan Engineering College, India
bdishacse@gmail.com

Abstract

Software effort estimation research shows that there is no universal agreement on the "best" effort estimation approach. This is largely due to the "ranking instability" problem, which is highly contingent on the evaluation criteria and the subset of the data used in the investigation. There are a large number of different method combination exists for software effort estimation, selecting the most suitable combination becomes the subject of research in this paper. Unless we can reasonably determine stable rankings of different estimators, we cannot determine the most suitable estimator for effort estimation. This paper reports an empirical study using 90 estimation methods applied to 20 datasets as an attempt to address this question. One of the commonly used machine learning techniques is the analogy method that cannot handle the categorical variables efficiently. In general, project attributes of cost estimation are often measured in terms of linguistic values. These imprecise values leads to analogous while explaining the process. The proposed fuzzy analogy method is a new approach based on reasoning by analogy using fuzzy logic for handling both numerical and categorical variables where the uncertainty and imprecision solution is also identified by the behavior of linguistic values utilized in the software projects. The performance of this method validates the results based on historical NASA dataset. The outcome of fuzzy analogy method is analyzed which indicates its improvement over the existing fuzzy logic methods. Estimation by analogy can be significantly improved by a dynamic selection of nearest neighbors, using only the project data from regions with small variance.

Keywords: Effort estimation, Analogy, Cost estimation, Fuzzy logic, Linguistic.

Introduction

Many software managers struggle with estimating projects. As can be seen from the following chart, the inherent problem with estimating is that small projects can be very easy to estimate, but the required accuracy is not important. On the other hand, large projects are very difficult to estimate, but the required accuracy is very important. Estimating software video is at the bottom of this page. I would encourage you to read the article and watch the video. According to a market research performed on information technology (IT) projects, the three main issues related to projects within the software development industry are time overruns, budget overruns and more than expected costs involved when maintaining software. These three issues converge to one activity related to the project management: software effort estimation. Software effort estimation is one of the first stages in a software project and helps to foresee the work that a specific project will entail. This helps the project leader to identify the amount of time and resources that are needed in order to complete the project in a timely manner. The challenge involved in this task is quite a difficult one, since it is very hard to predict the challenges that certain tasks will involve, especially if these types of tasks are being performed for the first time. Furthermore there is a certain degree of psychological pressure involved in this estimation exercise, since the project success or failure may very well depend on it. This estimation is a much-debated topic to this date, due to the fact that accurate estimation still eludes most methods in use within the industry, even more due to the fact that most project managers depend on expert estimation methods, which involve the estimation of tasks based on personal experience in the field of work. Therefore, expert estimation can be either somewhat accurate or totally off target, mostly depending on the individual or individuals performing the estimation. Another method for estimation involves a fixed mathematical equation to which variable parameters are applied depending on the project's specifications; the formula is then worked out to obtain a value for the effort. This method is also a very imprecise one, since it lacks the ability to predict factors like code reusability or the methodology used to develop the software. Ranking stability in software effort estimation

should be the primary research focus, being able to correctly classify the characteristics of each method allows the most suitable estimators to be used in the estimation process. This paper presents a method which can be used to determine the best effort estimators to use at different situations. Estimation by analogy is simple and flexible, compared to algorithmic models. Analogy technique is applied effectively even for local data which is not supported by algorithmic models (Keung, 2008; Ekrem Kocaguneli et al., 2010). It can be used for both qualitative and quantitative data, reflecting closer types of datasets found in real life. Analogy based estimation has the potential to mitigate the effect of outliers in a historical data set, since estimation by analogy does not rely on calibrating a single model to suit all the project .Unfortunately, it is difficult to assess the preliminary estimation as the available information about the historic project data during early stages is not sufficient (Hasan Al-Sakran, 2006). The proposed method effectively estimates the software effort using analogy technique with the classical fuzzy approach.

Table 1. Comparison of proposed effort with the existing and actual effort

Pjt.ID	Actual Effort	Estimated Effort	
		Fuzzy Method	Fuzzy Aralogy Method
1	36	45.31	34.14
2	42	32.37	36.48
3	42	43.83	36.60
4	50	60.74	43.625
5	60	80.99	52.125
6	120	113.77	103.122
7	72	94.04	61.794
8	192	172.59	163.201
9	239	268.04	203.150
10	300	354.73	254.512
11	352.8	354.73	324.538
12	420	483.07	355.993

Searching for the Best Estimator

A result of a classification/ranking procedure is a list of performance indicators, ranked according to their relevance to the target problem. Unlike dataset feature subset selection, there is no consolidated theory exists in literature for estimator selection stability. Ranked estimator lists are highly unstable in the sense that different method combining with different preprocessors may yield very different rankings, and that a small change of the data set usually affects the obtained estimator list considerably. The estimator ranking stability issue 3 has not been considered for its importance in the literature, but unfortunately, the issue has not grown into the main focus of research in the last few years, perhaps as a consequence of immediate benefits of individual development of estimators claimed to be more superior than the others, but limited to a very specific circumstance. Without being able to understand the ranking stability, it is unlikely to progress the research in the area of software cost estimation, as a consequence there is not convincing evidence to support the practical usage of the developed methods and tools available in the literature. To derive stable rankings about which estimator is “best”, there have been attempts in trying to compare model prediction performance of different approaches. For example, Shepperd and Kododa [32] compared regression, rule induction, nearest neighbor and neural nets, in an attempt to explore the relationship between accuracy, choice of prediction system, and different dataset characteristics by using a simulation study based on artificial datasets. They also reported a number of conflicting results exist in the literature as to which method provides superior prediction accuracy, and offered possible explanations including the use of an evaluation criteria such as MMRE or the underlying characteristics of the dataset being used can have a strong influence upon the relative effectiveness of different prediction models. Their work as a *simulation study* that took a single dataset, then generated very large artificial datasets using the distributions seen on that data. They concluded that: *None* of these existing estimators were consistently “best”;The accuracy of an estimate depends on the dataset characteristic and a suitable prediction model for the dataset. They conclude that it is generally *infeasible* to determine which prediction technique is the “best”. Recent results suggest that it is appropriate to revisit the ranking instability hypothesis. Menzies et al.applied 158 estimators to various subsets of two COCOMO datasets. In a result consistent with Shepperd and Kododa, they found the precise ranking of the 158 estimators changed according to the random number seeds used to generate train/test sets; the performance evaluation criteria used; and which subset of the data was used. However, they also found that four methods consistently outperformed the other 154 across all datasets, across 5 different random number seeds, and across three different evaluation criteria.

Effort Estimation

Fuzzy logic is based on human behavior and reasoning. It has an affinity with fuzzy set theory and applied in situations where decision making is difficult. A fuzzy set can be defined as an extension of classical set theory by assigning a value for an individual in the universe between the two boundaries that is represented by a membership function.

Analogy based estimation is another technique for early life cycle macro-estimation. Analogy based estimation involves selecting one or two completed projects that most closely match the characteristics of

ISSN: 2277-9655
Impact Factor: 1.852

your planned project. The chosen project(s), or analogues, are then used as the base for your new estimate. The ISBSG Data Portal can be used to select a suitable analogue. Analogy based estimation differs from the comparison based estimation above, in that comparison based estimation uses the medians from a group of similar projects. Analogy operates with one, or perhaps two past projects selected on the basis of their close similarity to the proposed project. Comparing a planned project to a past project is commonly used in an informal way when "guesstimating", consequently it is a familiar technique to the practitioner. Estimating software project effort by analogy involves a number of steps: Establish the attributes of your planned project, (eg size, language type, etc.), Measure or estimate the values of those project attributes, Search the ISBSG repository for a project that closely matches the attributes of your planned project. (There is a tool on the Toolkit CD that does this search for you), Use the known development effort from the selected project, (analogue), as an initial estimate for the target project, Compare each of the chosen attributes, (size, platform etc.,)., Establish or adjust the initial effort estimate in light of the differences between the analogue and your planned project.

$$A = \int_{x} \mu_A(x) / x$$

Where x is an element in X and □*x*□□*A* □ is a membership function. A Fuzzy set is characterized by a membership function that has grades between the interval [0, 1] called grade membership function. There are different types of membership function, namely, triangular, trapezoidal, Gaussian etc.

A Fuzzy analogy

Fuzzification of classical analogy procedure is Fuzzy analogy. It comprises of three steps, 1) Identification of cases, 2) Retrieval of similar cases and 3) Case adaptation. Each step is the fuzzification of its equivalent classical analogy procedure. The key activities for estimating software project effort by analogy are the identification of a candidate software project as a new case, the retrieval of similar software projects from a repository, the reuse of knowledge derived from previous software projects to generate an estimate for the candidate software project.estimation by analogy has motivated considerable research in recent years.howevwer, none has yet dealt with data. we present here a new approach based on reasoning by analogy and fuzzy logic which extends the classical analogy in the sense that it can be used when the software projects are described.

Conclusion

Prior reports of ranking instability about effort estimation may have been overly pessimistic. Given relatively large number of publicly available effort estimation datasets, vit is now possible to make stable rankings about the relative value of different effort estimators. The effectiveness of a learner used for effort estimation (e.g. regression or analogy methods) can be significantly altered by data preprocessing (e.g. logging all numbers or normalizing them zero to one).Neural nets and simple linear regression perform much worse than other learners such as analogy learners. Stepwise regression was a very useful preprocessor, but surprisingly a poor learner. Non-simple regression methods such as regression trees and partial linear regression are relatively strong performers. Regression trees that use tree pruning performed best of all in this study (with a preprocessor that normalized the numerics into the range zero to one) Very simple methods (e.g. *K*=1 nearest neighbor on the log of the numeric's) performed nearly as well as regression trees.

References

[1] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.

[2] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl. Optimal project feature weights I n analogy-based cost estimation: Improvement and limitations. *IEEE Transactions on Software Engineering*, 32:83–92, 2006.

[3] D. Baker. A hybrid approach to expert and model-based effort estimation. Master's thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, 2007. Available from https://eidr.wvu.edu/etd/documentdata.eTD?documentid=5443.

[4] A. Bakir, B. Turhan, and A. Bener. A new perspective on data homogeneity in software cost estimation: study in the embedded systems domain. *Software Quality Journal*, 2009.

[5] B. W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, Upper Saddle River, NJ, USA 1981.

[6] A. Brady and T. Menzies. Case-based reasoning vs parametric models for software quality optimization In *International Conference on Predictive Models in Software Engineering PROMISE'10*. IEEE, Sept. 2010.

[7] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[8] C. Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Trans. on Computers*, pages 1179– 1185, 1974.

[9] U. M. Fayyad and I. H. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

[10] Ahmeda MA and Muzaffar Z (2009) Handling imprecision and uncertainty in software development effort prediction: a type-2 fuzzy logic based framework. *Info. & Software Technol.* 51, 640–654.

[11] Ch. Satyananda Reddy and KVSVN Raju (2009) An improved fuzzy approach for COCOMO's effort estimation using gaussian membership function. *J. Software.* 4(5), 452-459.