International Journal of Computer Science and Informatics

Volume 1 | Issue 4 Article 3

April 2012

The Application of Case-Based Reasoning to Estimation of Software Development Effort

Ekbal Rashid
Department of Comp. Sc. & Engg. Siksha 'O' Anusandhan University, Bhubaneshwar, Odisha, India,
ekbalrashid2004@yahoo.com

Vandana Bhattacherjee
Department of Comp. Sc. & Engg. Birla Institute of Technology, Mesra, Ranchi,
vbattacharya@bitmesra.ac.in

Srikanta Patnaik
patnaik_srikanta@yahoo.co.in

Recommended Citation

Rashid, Ekbal; Bhattacherjee, Vandana; and Patnaik, Srikanta (2012) "The Application of Case-Based Reasoning to Estimation of Software Development Effort," *International Journal of Computer Science and Informatics*: Vol. 1 : Iss. 4 , Article 3.
DOI: 10.47893/IJCSI.2012.1045
Available at: https://www.interscience.in/ijcsi/vol1/iss4/3

The Application of Case-Based Reasoning to Estimation of Software Development Effort

Ekbal Rashid[1], Vandana Bhattacherjee[2] & Srikanta Patnaik[3]

[1&3]Department of Comp. Sc. & Engg. Siksha 'O' Anusandhan University, Bhubaneshwar, Odisha, India
[2]Department of Comp. Sc. & Engg. Birla Institute of Technology, Mesra, Ranchi
E-mail : ekbalrashid2004@yahoo.com[1], vbattacharya@bitmesra.ac.in [2], patnaik_srikanta@yahoo.co.in[3]

Abstract **-** Accurate project effort estimation is an important goal for the software engineering community. Till date most work has focused upon building algorithmic models of effort estimation for example COCOMO. We describe an alternative approach to estimation based upon the use of analogy. The objective is to estimate the development effort of student programs based on the values of certain attributes. We have developed a case based reasoning model and have validated it upon student data. Due to the nature of the software engineering domain, it is important that software cost estimation models should be able to deal with imprecision and uncertainty associated with such values. It is to serve this purpose that we propose our case based model for software cost estimation. We feel that case based models are particularly useful when it is difficult to define concrete rules about a problem domain in addition to this, expert advice may be used to supplement the existing stored knowledge.

Keywords **-** *CBR, Effort Estimation, Machine Learning, Analogy.*

I. INTRODUCTION

Estimation models in software engineering are used to predict some important attributes of future entities such as software development effort, software reliability, and productivity of programmers. Among such models, those estimating software effort have motivated considerable research in recent years [21]. Accurate and timely prediction of the development or maintain a software system is one of the most critical activates in managing software project, and has come to be known as 'Software Cost Estimation'. Due to the nature of the software engineering domain, it is important that software cost estimation models should be able to deal with imprecision and uncertainty associated with such values. It is to serve this purpose that we propose our case based model for software cost estimation. We feel that case based models are particularly useful when it is difficult to define concrete rules about a problem domain in addition to this, expert advice may be used to supplement the existing stored knowledge. The rest of the paper is organized as follows: section II gives a brief overview of the various software cost estimation method, section III describes the related work. In section IV we describe the case based reasoning approach in general. Section V gives a brief overview of machine learning, in section VI we present Methodology Overview and prediction technique, section VII presents the development of model. section VIII presents the Results and in section IX conclusion is presented.

II. OVERVIEW OF SOFTWARE COST ESTIMATION METHODS

There are many software cost estimation methods, and we give a brief overview of some of the more popular ones, namely, algorithmic method, analogy based estimation, expert judgment method, top down estimation and bottom up estimation method.

The algorithmic methods are designed to provide some mathematical equation to perform software estimation. The various software cost estimation models based on algorithmic method are the COCOMO & COCOMO II, Putnam, ESTIMACS and SPQR/20 [7]. Even though these methods are repeatable and modifiable, the disadvantages are that they do not deal with interrelationship between management constraints, such as cost versus quality. Poor sizing inputs and inaccurate cost driver rating will result in inaccurate estimation, since some experience and factors cannot be easily quantified. Estimating by analogy means comparing the proposed project to previously completed similar project where the project development

information is known. This method is based on actual experience of projects but it is difficult to ensure the degree of similarity between previous projects and new one. We shall discuss more about this method later in our paper. Expert judgment techniques involve consulting with software cost estimation experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost. Delphi technique and group consensus techniques are the most widely used expert judgment methods. It is an empirical method but also a subjective one. Top-down estimating method is also called Macro Model. Using it, cost estimation is derived from the global properties of the software project, and then the project is partitioned into various low-level components. The advantage of this method are, it is efficient and given system level view but it is too rough. In Bottom-up estimating method, the cost of each software components is estimated and then result are combined to arrive at an estimated cost of overall project. It is a detailed and stable method but it may overlook many of the system-level costs, and may be inaccurate and more time-consuming.

III. RELATED WORK

Several researchers have used soft computing approaches to estimate software cost. Idri et al have implemented the COCOMO cost model using fuzzy logic in [2] and also a fuzzy logic based analogy estimation approach in [3-5]. Case based reasoning has also been used by Kadoda et al in[9] They examine the impact of the choice of number of analogies when making predictions: They also look at different adaptation strategies. The analysis is based on a dataset of software projects collected by a Canadian software house. Their results show that choosing analogies is important but adaptation strategy appears to be less so. For this reason they urge some degree of caution when comparing competing prediction systems and only modest numbers of cases. Myrtveit et al in [10] and Ganesan et al in [12] have also studied case based approach to development effort prediction. Bhattacharjee et al have proposed Expert Case Based Models in [20-26].

IV. CASE BASED REASONING

Case-based reasoning (CBR) is a problem solving paradigm that is fundamentally different from other major AI approaches, in that instead of relying solely on general knowledge of a problem domain it uses specific cases [1]. Instead of making association along generalized relationships between problem descriptors and conclusion, CBR utilizes the specific knowledge of previously experienced, concrete problem situation (cases). Finding a similar past case, and reusing it in the new problem situation: this is the technique of CBR to solve a new problem. A second important difference is that CBR is also an approach to incremental, sustained learning since a new experience is retained each time a problem has been solved, making it available for future problems.

Thus, the notion of case-based reasoning does not only denote a particular reasoning method, irrespective of how the cases are acquired, it also denotes a machine learning paradigm that enables sustained learning by updating the case base after a problem has been solved. Learning in CBR occurs as a natural by-product of problem solving. When a problem is successfully solved, the experience is retained in order to solve similar problems in the future. When an attempt to solve a problem fails, the reason for the failure is identified and remembered in order to avoid the same mistake in the future. Case based reasoning prefers learning from experience, since it is usually easier to learn by retaining a concrete problem solving experience than to generalize from it. Case-based estimation is one of the more attractive techniques in the software effort estimation field. Central tasks that all case-based reasoning methods have to deal with are to identify the current problem situation, find a past case similar to the new one, use that case to suggest a solution to the current problem. Evaluate the proposed solution, and update the system by learning from this experience. We can thus broadly categorize the four primary steps comprising a CBR estimation system as:

- Retrieve the most similar case or cases, i.e., previously developed projects.
- Reuse the information and knowledge represented by the case (s) to solve the estimation problems.
- Revise the proposed solution.
- Retain the parts of this experience likely to be useful for future problem solving.

Case based estimation comes in handy when limited, domain knowledge is available and optimum solution is difficult to define. In cost estimation we use analogy by stating, “Similar Projects will have similar costs”. An advantage of case- based cost estimation is that it is easy to comprehend and explains its process to practitioners. In addition, it can model a complex set of relationship between the dependent variable (such as, cost or effort) and the independent variables or cost drivers. However, its deployment in software cost estimation needs improvements. The best working example of case-based reasoning is the complex human intelligence. However, our (human) reasoning by analogy is more than always approximate and vague rather than precise and certain.

A. THE APPLICATION OF CBR TO EFFORT ESTIMATION

CBR offers enormous advantages over the other effort estimations [27]. Attempts to quantify the casual dependencies within the domain have led to the development of the various algorithmic models. However, these models do not efficiently solve the problem indicating the importance of good understanding of elements that contribute to the effort estimation. The clear-cut advantage that CBR has over use of algorithmic models is that the use of CBR evades the need to model the domain and also possess the capability to explain its reasoning. In CBR it is possible to view such cases which are retrieved as similar to the target case and to view the adaptation strategies that operate on the retrieved cases which results in the particular prediction. CBR also allows manual adaptation so that an expert working in this can extrapolate from the similar retrieved cases and thus adjust the recommended solution if felt necessary. In recent years some tangible research in the application of CBR to effort estimation suggests that CBR can provide a practical support to software development managers [24-26], [28].

V. OVERVIEW OF MACHINE LEARNING

Machine learning deals with the problem of building computer programs that improve their performance at some task through experience [29]. Machine learning have been utilized in various problem domain. Some typical applications of machine learning are: data mining problems in which large databases contains valuable inherent regularities that can be discovered automatically.

VI. METHODOLOGY OVERVIEW

The environment of our study is the university campus and students of computer science and engineering are our target group. All students are provided with same level of guidance by instructors, support by the laboratory staffs, resources like computers, software etc., so for the students, development effort is the development time.

The parameter chosen for the model were based upon certain assumptions [23]. These are as follows:

- The mental discrimination required to design and code a program depends upon the numbers of methods and number of variable names.
- The final lines of code produced affects the development time.
- The number of methods is a predictor of how much effort is required to develop a program.
- The programming language exposure / experience of a programmer affects the development time.
- The inherent program difficulty level (as experienced by the programmers) also affects the development time.

Data collected from students included the following:

- Number of lines of code
- Number of functions
- Number of variables
- Difficulty level of program (low , medium , high)
- Number of formal parameters in each function
- Exposure to programming language
- Programmers experience

A. SIMILARITY MEASURES USED

Suppose a record set R1 of n fields has following values f1, f2,…, fn for the n fields respectively. Similarly a record set R2 of same type as R1 with field values g1, g2, … , gn. Then

The unweighted Eucledian distance (UWED) of R1 from R2 is:

$$UWED=\sqrt{(f_1 - g_1)^2 + (f_2 - g_2)^2 + \ldots + (fn - gn)^2}$$

The unweighted Manhattan distance (UWMD) of R1 from R2 is:

$$UWMD = |abs(f_1 - g1) + abs(f2 - g2) + \ldots + abs(fn - gn)|$$

B. CALCULATION OF ERROR IN PREDICTION

The actual development time is input by the user along with other parameters like lines of code, number of function, difficulty level etc, and based on the matching case (which is retrieved from knowledge base) predicted development time and the error in prediction is calculated. Furthermore, the relative error is calculated using the formula

Actual development time: ta

Predicted development time: tp

Error in prediction = ta – tp

Relative error = abs (ta - tp)/ ta

Percentage error = abs (ta - tp) * 100 / ta

and a result is derived where acceptable range is within 25%.

VII. DEVELOPMENT OF MODEL

In this section we predicts the development time of the software for which the parameters have been given as input. The development time is calculated using different similarity measures. In this case two similarity measures are used like Euclidean method and Manhattan method. These measures use the knowledge base to find the matching cases for the input parameters. Once the matching cases are generated and an exact solution is not found then the user is given an option to modify the input, and in case a solution is found after modification, then the new case is added to the knowledge base. The context level diagram for the software prediction is given in figure 1and the diagram for the proposed system is given in Figure2.

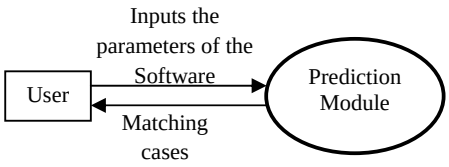

Fig. 1 : Context Level diagram for Software Prediction.

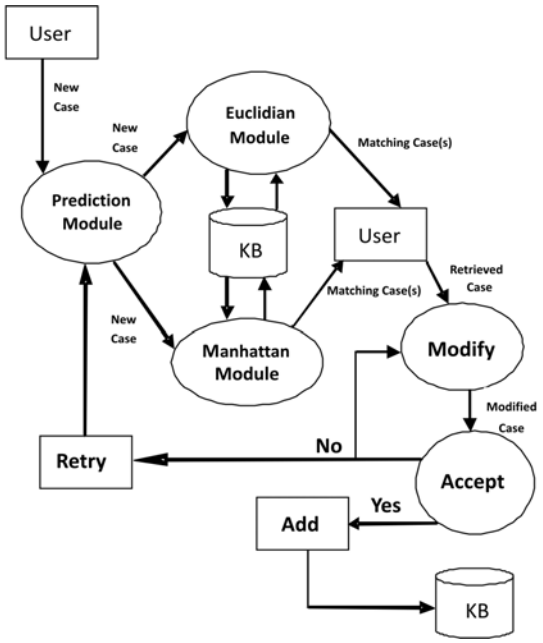

Fig. 2 : The Diagram for prediction of development time of the software.

- Through prediction module we accept the values of various parameters from the user end.
- Euclidean module accepts the record set as new case and finds the matching cases from the knowledge base. The similarity measure is the Euclidean method.
- Manhattan module generates the matching case based upon the Manhattan method.
- New case is Modified by the user and if the case is found within acceptable limits then the knowledge base is updated with modified case.

VIII. RESULTS

We present the results obtained when applying the Case Based Reasoning model to the data set. The accuracy of estimates is evaluated by using the magnitude of relative error MRE defined as:

$$\text{MRE} = \frac{abs(ta - tp)}{ta}$$

Prediction level Pred is also used to test the performance of the model. It is defined as:

$$\text{Pred (p)} = K/N$$

Where, N is the total size of the data set and K is the number of programs with MRE less than or equal to p. We calculate Pred (0.25), mean of MRE called MMRE and minimum of MRE called minMRE.

Table 1 : Prediction Error Analysis (Manhattan Distance)

Parameters	Development time (minutes)
MMRE	1.02
MinMRE	0.00
Pred (0.25)%	46.66

Table 2 : Prediction Error Analysis (Euclidean Distance)

Parameters	Development time (minutes)
MMRE	0.198
MinMRE	0.00
Pred (0.25)%	95.0

IX. CONCLUSION

In this research paper we developed the case based reasoning model using two similarity measures: Manhattan Distance and Euclidean Distance. The prediction in 46.66% of the cases is within 25% error for the Manhattan Distance based model while the prediction in 95% of the cases was within 25% error for the Euclidean Distance based model. The results may be considered good considering the fact that the

development time is a very complex attribute since it is dependent on human behavior. As part of our ongoing work, increasing the volume of knowledge base is another objective. The larger the database more likely the results are to be accurate. we are collecting data from different categories of students and with more parameters. In this research , students programs were the target of study. We are also planning to incorporate weight measures with the various attributes.

REFERENCES

[1] A. Abran and P.N.. Robillard. (1996), "Function Points Analysis: An Empirical Study of its Measurement Processes", IEEE Transactions on Software Engineering, 22(12): pp. 895-909.

[2] A. Idri, L.Kjiri, and A Abran. (2000), "COCOMO Cost Model Using Fuzzy Logic", In Proceedings of the 7th International Conference on Fuzzytheory and Technology, pp.219-223. Atlantic City, NJ, USA.

[3] A. Idri and A Abran. (2000b), "Towards A Fuzzy Logic Based Measures for Software Project Similarity", In Proceedings of the 6th Maghrebian Conference on Computer Sciences, pp. 9-18, Fes Morroco.

[4] A. Idri and A. Abran. (2001), "A Fuzzy Logic Based Measures For Software Project similarity: Validation and Possible Improvements", In Proceedings of the 7th International Symposium on Software Metrics, pp. 85-96, England, UK, IEEE.

[5] A. Idri, A. Abran and T.M. Khoshgoftaar .(2001c), " Fuzzy Analogy: Anew Approach for Software Cost Estimation", In Proceedings of the 11th International workshop on software Measurements, pp.93-101, Montreal, Canada.

[6] A. Aamodt and E. Plaza. (1994), "Case-Based Reasoning : Foundational Issues, Methodological Variations. and System Approaches", AI Communication, IOS Press, vol. 7:1, pp.39-59.

[7] B.W.Boehm.(1981), "Software Engineering Economic", Prentice –Hall.

[8] B.W.Boehm et al. (1995), "Cost Models for Future Software Life Cycle Processes: COCOMO II". Annals of Software Engineering: Software Process and Product Measurement, Amsterdam.

[9] G.Kadoda, M Cartwright, L Chen, and M.shepperd.(2000), "Experiences Using Case-Based Reasoning to Predict Software Project Effort", In Proceeding of EASE, p.23-28, Keele,UK.

[10] I. Myrtveit and E. Stensrud. (1999), "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models", IEEE transactions on software Engineering, vol 25,no. 4, pp. 510-525.

[11] J.Matson, E.B.E.Barrett, J.M. Mellichamp. (1994), Software Development Cost Estimation Using Function Points", Transaction on Software Engineering, vol. 20, no 4, pp. 275-287, IEEE Computer Society.

[12] K. Ganeasn, T.M. Khoshgoftaar, and E. Allen. (2002), "Case-based Software Quality Prediction", International journal of Software Engineering and Knowledge Engineering, 10 (2), pp. 139-152.

[13] L. Angelis and I Stamelos. (2000), "A Simulation Tool for Efficient Analogy Based Cost Estimation", Empirical software Engineering, vol. 5, no. 1, pp.25-68.

[14] L. Angelis, I. Stamelos, and M. Morisio. (2001), "Building a Software Cost Estimation Model Based on Categorical Data", in Proceedings of the 7th international software Metrics Symposium, pp. 4-15, London, UK, IEEE Computer Society.

[15] M. Shepperd C. Schofield, and B. Kitchenham. (1996), "Effort Estimation using Analogy", In Proceeding of the 18th International Conference on Software Engineering, pp.170-178, Berlin.

[16] M. Shepperd and C. Schofield. (1997), "Estimating Software Project Effort Using Analogies", IEEE Transactions on Software Engineering, vol. 23, no 12, pp. 736-743, November 1997.

[17] R. Gulezian. (1991), "Reformulating and Calibrating COCOMO", Journal Systems Software, vol. 16, pp.235-242.

[18] S. Vicinanza and M.J. Prietulla. (1990), "Case-Based Reasoning in Software Effort Estimation", In Proceeding of the 11th International Conference on Information System.

[19] S. Kumar and V.Bhattacharjee,(2005),"Fuzz logic based Model for Software cost Estimation ",In Proceedings of the international Conference on information Technology, Nov'05, PCTE, Ludhiana India.

[20] S. Kumar and V.Bhattacharjee,(2007),"Analogy and Expert Judgment: A Hybrid Approach to Software Cost Estimation", In Proceedings of the National Conference on information Technology: Present practice and Challenge, Sep'07, New-Delhi, India.

[21] V. Bhattacherjee and S. Kumar,(2004),"Software cost estimation and its relevance in the Indian software Industry", In Proceedings of the International Conference on Emerging Technologies IT Industry, Nov'05, PCTE, Ludhiana India..

[22] V. Bhattacherjee and S .Kumar,(2006),"An Expert- Case Based Frame work for Software Cost Estimation", In Proceedings of the National Conference on Soft Computing Techniques for Engineering Application (SCT-2006), NIT Rourkela.

[23] V. Bhattacherjee,(2006),"The Soft Computing Approach to Program Development Time Estimation In Proceeding of the International Conference on Information Technology, ICIT 06, Dec'06, Bhubneshwar, India, IEEE Computer Society.

[24] V. Bhattacherjee, S .Kumar and Ekbal Rashid (2008), "Estimation of Software Development Effort in University Setting: A Case Study", Accepted for Presentation at National Conference on Architecturing Future IT Systems (NCAFIS '08), Indore, October 2008.

[25] V. Bhattacherjee, S .Kumar and Ekbal Rashid (2008), "Case Based Estimation Model using Project Feature Weights", In Proceedings of The National Seminar on Recent Advances on Information Technology (RAIT-2009) ISMU, Dhanbad.

[26] V. Bhattacherjee, S. Kumar and E. Rashid, (2008),"A Case Study on Estimation of Software Development Effort.", Accepted for presentation and publication in ICACT-2008 Conference proceedings, Gokaraju Rangaraju Institute of Engineering & Technology, Hydrabad. (A.P.), India.

[27] Mukhopadhyay T. S. Vicinanza & M. Prietula 1992 "Examining the feasibility of a case-based reasoning model for software effort estimation" MIS Quarterly 16(2) p.155-171.

[28] Prietula M., S. Vicinanza & T. Mukhopadhyay 1996 "Software effort estimation with a case-based reasoning" Journal of Experimental and Theoretical Artificial Intelligence 8(3-4) p.341-363.

[29] R. S. Michalski, I. Bratko and M. Kubat (ed.), Machine Learning and Data Mining: Methods and Applications, John Wiley & Sons Ltd., 1998.

◈◈◈