American Journal of Software Engineering and Applications
2013; 2(2): 49-53
Published online April 2, 2013 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j. ajsea.20130202.14

Analogy-based software quality prediction with project feature weights

Ekbal Rashid[1], Srikanta Patnaik[2], Vandana Bhattacharya[3]

[1]Department of CS & E CIT, Tatisilwai,Ranchi, India
[2]Department of CS & E SOA University, Bhubaneswar, Orissa,India
[3]Department of CS & E BIT Mesra, Ranchi, India

Email address:

ekbalrashid2004@yahoo.com (E. Rashid), patnaik_srikanta@yahoo.co.in (S. Patnaik),
vbhattacharya@bitmesra.ac.in (V. Bhattacharya)

To cite this article:

Ekbal Rashid, Srikanta Patnaik, Vandana Bhattacharya. Analogy-based Software Quality Prediction with Project Feature Weights, *American Journal of Software Engineering and Applications*. Vol. 2, No. 2, 2013, pp. 49-53. doi: 10.11648/j.ajsea.20130202.14

Abstract: This paper presents analogy-based software quality estimation with project feature weights. The objective of this research is to predict the quality of project accurately and use the results in future predictions. The focus includes identifying parameters on which the quality of software depends. Estimation of rate of improvement of software quality chiefly depends on the development time. Assigning weights to these parameters to improve upon the results is also in the area of interest. In this paper two different similarity measures namely, Euclidian and Manhattan were the measures used for retrieving the matching cases from the knowledgebase to increases estimation accuracy & reliability. Expert judgment, weights and rating levels were used to assign weights and quality rating levels. The results show that assigning weights to software metrics increases the prediction performance considerably. In order to obtain the results, we have used indigenous tools.

Keywords: Analogy, CBR, Effort Estimation, Software Quality Prediction, Similarity Function

1. Introduction

Software Quality estimation is an important and hard management task. This is due to the lack of information on making decisions in the early phases of the project development. Most of today's software quality estimation models are built on using data from projects of single organization. Using such data has well known benefits such as ease of understanding and controlling of collected data. But different researchers have reported contradictory results using different software quality estimation modeling techniques. It is still difficult to generalize many of the obtain results. This is due to the characteristics of the datasets being used and dataset's small size. In Case-Based Reasoning (CBR) problem solving is seen as a process, which involves the retrieval of similar prior cases from case bases using mobile agent methodology and the adaptation of retrieved cases' solutions to fit the new problem's requirements. Estimation models in software engineering are used to predict some important attributes of future entities such as software development effort, software reliability, software quality, and productivity of programmers. Among such models, those estimating software effort have motivated considerable research in recent years [11]. Correct prediction of the software quality or maintain a software system is one of the most critical activities in managing software project. Due to the nature of the software engineering domain, it is important that software quality estimation models should be able to deal with ambiguity and indistinctness associated with such values. To serve this purpose, we propose our case-based estimation model for software quality 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. A case-based reasoning model was developed in [13] for estimating software development effort.

In this paper, we have used features with weights, which are based on expert judgments. For example, if the difficulty level of a program increases then there is also increase in the efforts and development time. For our experiment, we have assumed weights based on expert judgment and by empirical study. We displayed the software quality relative to the lines of code retrieved from

the knowledgebase in Table 4. The rest of the paper is organized as follows: Section 2 gives a brief overview of the various related work. Section 3 describes the methods. In section 4, we present research methodology. Sections 5 and 6 present the production rule and the development of models. Section 7 presents the results and analysis. Finally, section 8 concludes the paper and presents some future trends.

Table 4. *Classification used for Software Quality prediction.*

Program Scenario	Value of Q	Class		
Sl.No		Excellent	Good	Poor
1	0.195	√	×	×
2	0.028	√	×	×
3	0.030	√	×	×
4	0.037	√	×	×
5	0.088	√	×	×
6	0.600	√	×	×
7	0.090	√	×	×
8	2.99	×	√	×
9	4.067	×	√	×
10	7.036	×	×	√
11	0.014	√	×	×
12	0.092	√	×	×
13	0.142	√	×	×
14	0.037	√	×	×
15	0.074	√	×	×

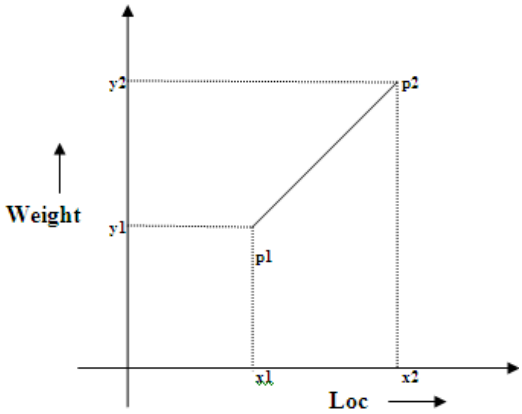
2. Background and Motivation

Many researchers have used soft computing approaches for software quality estimation. Zhong et. Al in [16] have used unsupervised Learning techniques to build a software quality estimation system. Idri et. Al have implemented the COCOMO cost model using fuzzy logic in [1] and also a fuzzy logic based analogy estimation approach in [2-4]. Case based reasoning has also been used by Kadoda et. Al in [5].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. For this reason they urge some degree of caution when comparing competing prediction systems and only modest numbers of cases. Myrtveit et. Al in [6] and Ganesan et. Al in [7] have also studied case based approach to development effort prediction. Bhattacherjee et. Al have proposed Expert Case Based Models in [9-14]. Rashid et. Al emphasized on the importance of software quality estimation [15].

3. Methods

3.1. Hypothesis

Distance between the status of two programs p1 and p2.



We can consider a particular parameter weight which may be dependent on LOC.

Let us take LOC =X and the hypothetical parameter weight that depends upon LOC be = Y

Then we can express the relation between the two as an ordered pair as:

$$f:NR : y=f(x)$$

Now, we can represent the two different program states by two points on the Cartesian plane. Let that be $p_1(x_1,y_1)$ and $p_2(x_2,y_2)$, then the distance between the two can be calculated using the Euclidean distance formula:ED=

$$dist\ (p_1,p_2)=[(x_2-x_1)^2 + (y_2-y_1)^2]^{1/2}$$

The Manhattan distance (MD) of p_1 from p_2 is:

$$MD = |abs\ (x_2-x_1) + abs\ (y_2-y_1)|$$

A small distance indicates a high degree of similarity. When a new project is estimated, its distances to each project in the historical feature database are calculated.

A fundamental question in this model is how to set the feature weights:

w_i since individual features should influence project similarity to a different degree [8]. Various approaches have been proposed:

- Set all project feature weights to identical values: $w_i = 1$, i = 1,….l.
- Set each project feature weight to a value determined by human judgment.
- Set each project feature weight to a value obtained by

statistical analysis.

•Set each project feature weight to either 0 or 1 so that an estimation quality metric is maximized. This brute-force approach proposed by Shepperd and Schofield tries to identify a subset of important features. Once these features are identified, they are all given the same weight. We have adopted the combined approach of expert judgment and empirical study. We now present the methodology adopted for software quality estimation based on Case-Based Reasoning using project feature weights.

4. Research Methodology

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, supported by the laboratory staffs, resources like computers, software etc.

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 and
•Programmers Experience

5. Production Rule

We have used production rules for quality rating levels. See table 2

Table 2. *Quality rating levels.*

Quality Score	Rating
If Q<2	Excellent
If Q >= 2 and Q<=5	Good
If Q> = 6 and Q<=10	Poor

RULE: R1 IF Q<2 THEN
EXCELLENT
RULE: R2 ELSE
IF Q>=2 AND Q<=5 THEN
GOOD
RULE: R3 ELSE
IF Q>=6 AND Q<=10 THENPOOR

6. Model Development

A knowledgebase is created and maintained to store the cases against which the matching process has to be performed. Parameters with weight related to the software are given as input and the quality is predicted by finding the best match from the knowledgebase. See Figure 1.

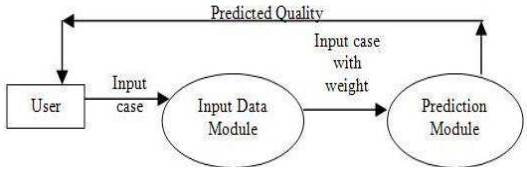


Figure 1. *cuntest level diagram.*

6.1. Input Data Module

This module accepts the values of various parameters from the user. It also has the provision of assigning weights to the parameters.

6.2. Prediction Module

This module predicts the quality of software for which the parameters have been given as input. The quality is calculated using different similarity measures. These measures use the knowledgebase to find the matching cases for the input parameters. Once the matching cases are generated the new results are added to the database. Depending upon the dissimilarity between two projects it calculates the quality (Q) of the module with respect to lines of code. Only those results are added that give an error of 5% or less.

The inputs to the proposed model are as follows:

•Lines of code
•Number of Functions or Procedures
•Experience of the Programmer in years
• Difficulty level of software.

7. Results and Analysis

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:

$$\text{MRE} = \text{abs}\left|\frac{AP - TP}{AP}\right|$$

Where AP = Actual parameter

TP = Targeted parameter

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

Pred(p)= E/R

Where, R is the total size of the data set and E is the number of programs. We calculate Pred (0.10) values for the various values of weights. We have used feature

weights from wi ranging between 0 to 1. These were the values as suggested by experts, in our case, they comprised of a team of faculty members from various colleges/ institutes actively involved in software engineering research. The results of applying the different combinations of weights are displayed in Table 1, quality rating levels are shown in Table 2 and results for software quality prediction are shown in Table 3.

Table 1. *Results of applying different weight measures to the analogy-based model.*

S. No.	W1	W2	W3	W4	Pred(0.1) Euclidean	Pred(0.10) Manhattan
1	0.5	1	1	1	0.97	0.883
2	1	0.5	0.5	0.5	0.833	0.316
3	0.5	1	0.5	0.5	0.816	0.3
4	0.5	0.5	0.5	1	0.833	0.316
5	1	0.25	0.3	0.3	0.616	0.316
6	0.25	1	0.3	0.3	0.65	0.25
7	0.25	0.25	1	0.3	0.66	0.26
8	1	0.5	0.5	0.5	0.816	0.33
9	1	1	1	1	0.9	0.66
10	0.5	1	1	1	0.97	0.883
11	0.5	0.5	1	1	0.8	0.633
12	1	0.5	0.5	1	0.9	0.633
13	0.4	0.25	1	0.5	0.622	0.61
14	0.25	0.25	0.25	0.5	0.80	0.360
15	0.5	0.25	1	0.25	0.77	0.522

Table 3. *Quality of Software.*

Rules	Quality Level of Software	Class	%
R1	High	Excellent	80.1%
R2	Medium	Good	13.3%
R3	Low	Poor	6.6

8. Conclusions and Future Trends

The aim of paper is to improve in accuracy of predictions and increasing the reliability of the knowledgebase was our priority. For each program we have evaluated the students four times. Value of Q (Quality of software) is calculated by indigenous tools written in c language and graph was plotted through MATLAB 7.10.0 version which is shown in Figure 2. It can be seen in Table 3 the results are very good where quality of software is concerned because 80.1% are excellent, 13.3% are good and 6.6% are poor (As per quality rating levels). In this research paper we have used two similarity measures Manhattan Distance and Euclidean Distance. The combination (0.5, 1, 1, 1) works the best where prediction is concerned because 97% data are within 10% error for Euclidean distance and 88% data are within 10% error for Manhattan distance measure. Apart from predicting quality of software, this system can also be used to predict the development time. As part of our ongoing work, increasing the volume of knowledgebase is another objective.

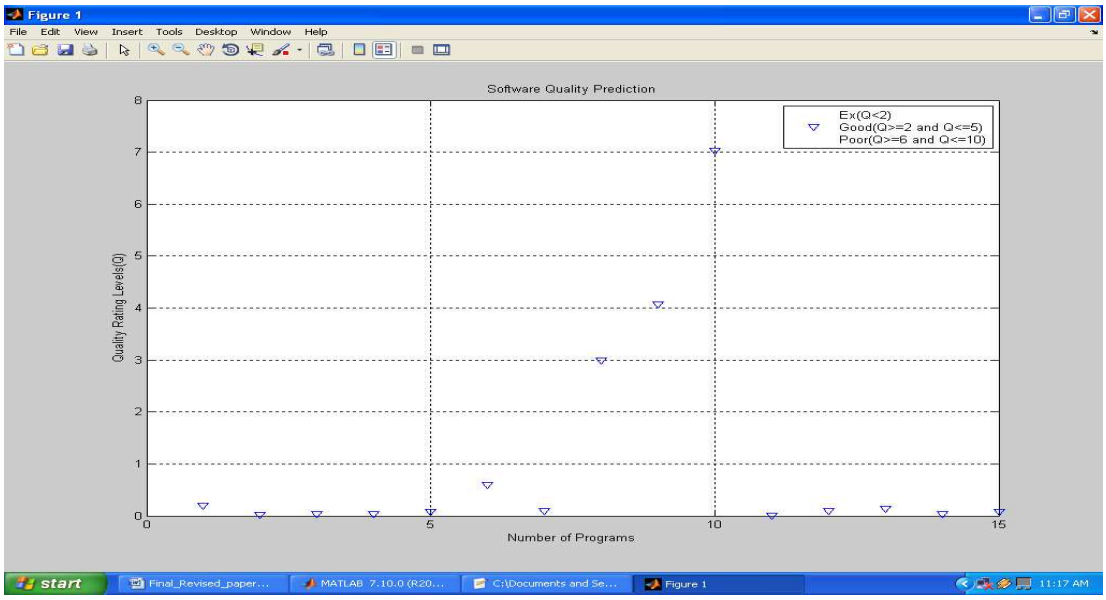


Figure 2. *Classification used for Software Quality prediction.*

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] M. Auer, A. trendowicz, B. Graser, E. Haunschmid and S. Biffl, "Optimal Project Feature Weights in analogy-based Cost Estimation: Improvement and Limitations", IEEE, TSE, Vol.32, No.2, Feb 2006, pp 83-92.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] E. Rashid, V. Bhattacherjee, S. Patnaik, "The Application of Case-Based Reasoning to Estimation of Software Development Effort". International Journal of Computer Science and Informatics (IJCSI) ISSN 2231 –5292, Vol 1 Issue 3 pp 29-34 Feb 2012.

[14] V. Bhattacherjee, S. Kumar and E. Rashid ,A Case Study on Estimation of Software Development Effort" In Proceedings on International Conference on Advanced Computing Technologies(ICACT-2008), Gokaraju Rangaraju Institute of Engg & Technology, Hyderabad, India,p.no.161-164.

[15] Ekbal Rashid, Srikanta Patnaik, Vandana Bhattacherjee "A Survey in the Area of Machine Learning and Its Application for Software Quality Estimation" has been published in ACM SigSoft ISSN 0163-5948, volume 37, number 5, September 2012, http://doi.acm.org/10.1145/2347696.2347709 New York, NY, USA.

[16] Shi Zhong,Taghi M.Khoshgoftaar and Naeem Selvia "Unsupervised Learning for Expert-Based Software Quality Estimation".Proceeding of the Eighth IEEE International Symposium on High Assurance Systems Engineering (HASE'04).