Journal of Theoretical and Applied Information Technology

www.jatit.org

COMPLEXITY METRIC FOR ANALOGY BASED EFFORT ESTIMATION

[1]VANDANA BHATTACHERJEE [2]PRABHAT KUMAR MAHANTI [3]SANJAY KUMAR

[1]Department of Cs & E, Birla Institute Of Technology, Ranchi

[2]Department of Csas, Saint John University Of New Brunswick, Canada

[3]Department of Cs & E, Birla Institute of Technology, Ranchi
E-mail: vbhattacharya@bitmesra.ac.in , pmahanti@gmail.com, k_sanjay71@yahoo.com

ABSTRACT

The problem of cost estimation in software engineering has been addressed by several researchers. Research shows that among many factors that affect the development cost, size of the product plays an important role. In addition to the size, product properties like complexity, cohesion and coupling are also mentioned as cost factors. We focus on the relationship between complexity of a product to its development cost. The aim of this research is to identify the use of complexity metrics as important parameters in similarity matching for retrieval of cases in case based effort estimation. The objective is to estimate the development effort of student programs based on the values of certain attributes using case based reasoning model. We have developed a case based estimation model using complexity metrics and have validated it upon student data. Our focus is on object oriented programs.

Keywords: *Effort estimation, Case based reasoning, Analogy*

1. INTRODUCTION

Software cost estimation is one of the most important activities for any software development project. Before the development of the product, estimation of cost is very difficult and without cost estimation we cannot proceed further. Developers in large projects use measurements to help them understand their progress towards completion. Metrics gathered from historical data provide an estimate of future similar projects. Estimation models for predicting software effort have motivated considerable research in recent years [1 - 5] [7] [8] [13].

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 analogy based reasoning method for software cost estimation [6]. We feel that such 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. Software metrics gather information about a product based on its measurable characteristics and hence provide good criteria for measuring the similarity of two software products [11] [15 – 18] [24 - 25] [31 – 32]. In this paper we have proposed a complexity metric and used it for retrieving similar cases. The rest of the paper is organized as follows: Section 2 gives a brief overview of the various complexity metrics and the ones used for the study, Section 3 describes the related work and approach in general. In Section 4 we present Methodology Overview, Section 5 presents the Results and in Section 6 Conclusion is presented. The Appendix provides the details of partial data set.

2. ROLE OF COMPLEXITY METRICS IN PREDICTING DEVELOPMENT EFFORT

Among the most commonly known and earliest proposed complexity metrics is the Weighted Method per Class (WMC) Metric of Chidamber and Kemerer [31 – 32]. Consider a class C_1 with methods M_1, M_2, M_3,….M_n that are defined in the class. Let c_1, c_2, c_3,…..c_n be the complexity of the methods.
Then, WMC (Weighted Method per Class) =

Vol6. No1. (pp 001 - 008)

$$\sum_{i=1}^{n} c_i$$

If all method complexities are considered to be unity, then WMC = *n*, the number of methods. Chidamber and Kemerer are of the view that the number of methods and the complexity of methods involved is a predictor of how much time and effort is required to develop and maintain the class.

Another metric to be mentioned is the Class Method Complexity (CMC) Metric. Li's [38] CMC (Class Method Complexity) is the summation of the internal structural complexity of all local methods, regardless whether they are visible outside the class or not (e.g. all public and private methods in C++). This definition is essentially the same as the first definition of the WMC metric in [13]. However, the CMC metric's theoretical basis and viewpoints are significantly different from the WMC metric. The CMC metric is directly linked to the effort needed to design, implement, test, and maintain a class. The more complex a class methods are, the more effort is needed to design, implement, test, and maintain the methods.

In defining the Class Complexity (CC) Metric Balasubramanian [25] states, that class complexity is the sum of the number of instance variables in a class and the sum of the weighted static complexity of a local method in the class. To measure the static complexity Balasubramanian uses McCabe's Cyclomatic Complexity where the weighted result is the number of nodes subtracted from the sum of the number of edges in a program flow graph and the number of connected components. The CMOOD metric of Rajnish and Bhattacherjee (Complexity Metric for Object Oriented Design) for measuring the complexity of class in object-oriented design is defined as the sum of all private, public and protected variables, sum of the formal parameters used in all local methods of a class and the sum of the weighted static complexity of all local methods in the class [18] [33].

For the purpose of this study, we propose the following complexity metric Difficulty Level Metric (DLM). It is obtained by multiplying the size of a program/ class with its difficulty level. Any size measure such as Lines of Code or function points (or a combination of the two) could be chosen. The difficulty level of software is considered to be the difficulty the programmer/ developer faces in designing and implementing the program. Since the difficulty level is an ordinal variable (low, medium, high), we scale it as per the formula given in section 4. For example, if a class is of 250 lines of code and the difficulty level for the programmer is 2 (on a scale of 1 to 3 ranging from least to most difficult), then DLM for the class is 500.

In our present work we build two models, one based on product attributes only and the other incorporating the above mentioned complexity metrics (CMOOD and DLM). A comparative analysis has been done for their predictive capability. The study has been conducted at Birla Institute of Technology, Ranchi. Post graduate students participated in the study. The model was developed using Visual Basic at the front end and Microsoft Access at the back end. To make the database secure Oracle could be used as the database.

3. RELATED WORK

Several methods have been proposed to estimate software cost estimation using soft computing approaches, for example, COCOMO [38], neural networks [39], and expert judgement [40]. The basis for these methods is that similar projects will have similar costs. This has been presented by several researchers [5] [10] [12] [14] [19 – 20] [22 - 23] [28] [30]. A. Abran et al have proposed several approaches for estimating software cost, Fuzzy Analogy being one of them [1 – 4]. V Bhattacherjee et al have proposed a framework for expert – case based model [29 - 30] [34 – 35]. V. Bhattacherjee has proposed and established a soft computing approach to development time estimation, developing a neural network model based on program and programmer attributes [36]. More recently, a case based model has been proposed considering several attributes [37] however, research papers correlating software metrics with development effort are few [17 – 18] [21][41].

For the cost estimation of a new project by analogy based method, the following steps must be done [23]:

- Measure the features of the new project
- Identify projects with similar features from database
- get the new estimate using the known costs of the retrieved projects

A project P is described by a set of features (f_1, f_2,…….f_n, c), where f_1, f_2,…….f_n denote the features which are measured at estimation time and c is the development cost at completion.

The similarity of two projects P_1 and P_2 is defined as a weighted MinKowski distance:

$$\delta (P_1, P_2) = \left[\sum_{i=1}^{n} w_i \, abs \left(f_{i1} - f_{i2} \right)^p \right]^{1/p}$$

when p = 2, this formula gives the Euclidean distance and when p = 1, it gives the Manhattan distance.

4. METHODOLOGY OVERVIEW

For each program, effort was collected and accounted for designing and coding the class, documenting, testing and correcting it. For the effort data collection, the students were instructed to record the effort spent on each program. This includes all effort for designing, implementing and testing/ debugging the programs.

The parameters chosen for the model were based upon the following assumptions [24]:

- 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

The case base was created by the data collected from the student programs. 25% of the data was set aside for the validation of the models. Two models were created based on the collected data. The Model 1 considered the above mentioned parameters for retrieval of cases. Model 2 considered the CMOOD and DLM metrics alongwith the parameters of Model 1.

To compute the dissimilarity between objects described by variables of mixed type (interval – scaled, categorical, ordinal, symmetric/ asymmetric, binary or ratio – scaled), the general procedure is to combine the different variables into a single dissimilarity matrix bringing all of the meaningful variables onto a common scale of the interval [0.0, 1.0]. Suppose that the data set contains p variables of mixed type. The dissimilarity ***d (i, j)*** between objects ***i*** and ***j*** is defined as:

$$\boldsymbol{d(i,j)} = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

where, $\delta_{ij}^{(f)} = 0$, if there is no measurement of variable ***f*** for object ***i*** or ***j***; or $x_{if} = x_{jf} = 0$ and variable ***f*** is asymmetric binary; otherwise $\delta_{ij}^{(f)} = 1$,

The contribution of variable ***f*** to the dissimilarity between ***i*** and ***j***, i.e., $d_{ij}^{(f)}$ is computed dependent on its type. All interval – scaled variables were normalized by the formula:

$$x_{if} = \frac{x_{if} - \min_f}{\max_f - \min_f}$$

Each ordinal variable was scaled as:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

where, r_{if} is the rank of the variable and M_f is the number of ordered states. A small distance indicates a high degree of similarity. To determine the effort estimate of a new project, its distances to each of the stored projects are calculated. The costs of the most similar projects are used to estimate the new project.

Data collected from students included the following:

- number of lines of code (statements)
- number of functions
- number of variables
- difficulty level of program (low, medium, high)
- number of formal parameters in each function
- exposure to programming language (low, medium, high)
- familiarity of problem area (low, medium, high)

- development time

5. DEVELOPMENT OF MODEL

In this section we briefly describe the development steps for the prediction model. As a first step towards development of the model, the high level requirements were identified as follows:

- Computation of complexity metric for input cases.
- Prediction based on analogy that uses various similarity measures.
- Updation mechanism to update the database as new cases are generated.
- Modifiaction or removal of a particular record.
- Updation mechanism for adding new similarity measures and additional parameters.

The context level diagram for the proposed system is given in Figure 1.

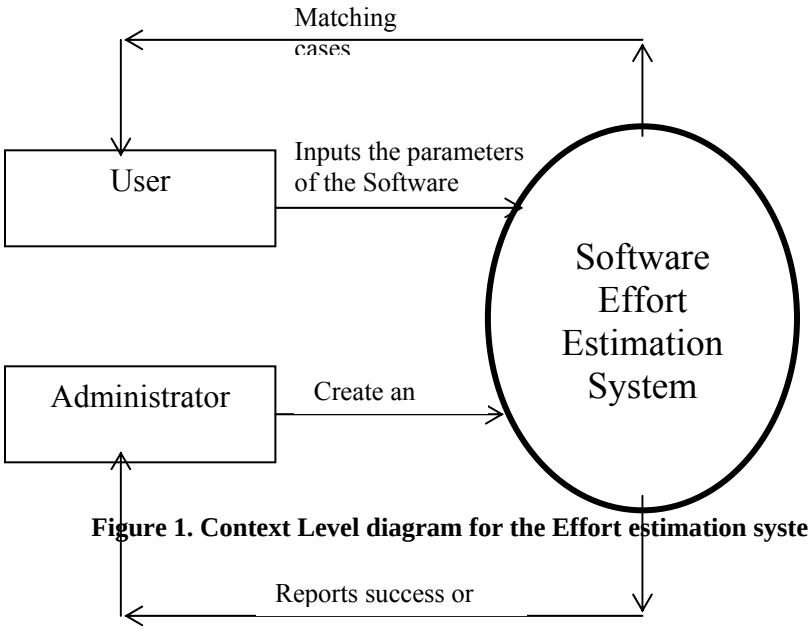

Figure 1. Context Level diagram for the Effort estimation system

A master database is created and maintained to store the cases against which the matching process has to be performed. Parameters related to the software are given as input and the development time is predicted by finding the best match from the database. The matching is done using varying similarity measures like Manhattan and Euclidean. Once the result is predicted it is added to the database to enhance the accuracy of future predictions. The objective is to predict the cost of project accurately and use the results in future predictions. Increasing the volume of database is yet another objective. The larger the database more likely the results are to be accurate

6. 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:

$$MRE = \frac{abs(time\_actual - time\_predicted)}{time\_actual}$$

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

$$Pred(p) = K/N$$

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 (Model - 1: Manhattan Distance)

Parameters	Development time (minutes)
MMRE	**0.157**

MinMRE	**0.00**
Pred (0.25) %	**66.667**
R^2	**0.2125**

Table – 2: Prediction Error Analysis (Model - 1: Euclidean Distance)

Parameters	Development time (minutes)
MMRE	**1.571**
MinMRE	**0.00**
Pred (0.25)%	**13.33**
R^2	**0.0026**

Table – 3: Prediction Error Analysis (Model - 2: Manhattan Distance)

Parameters	Development time (minutes)
MMRE	**0.167**
MinMRE	**0.00**
Pred (0.25)%	**73.33**
R^2	**0.9612**

Table – 4: Prediction Error Analysis (Model – 2: Euclidean Distance)

Parameters	Development time (minutes)
MMRE	**0.315**
MinMRE	**0.00**
Pred (0.25)%	**46.67**
R^2	**0.211**

7. CONCLUSION

The aim of this research was to identify the use of complexity metrics as important parameters in similarity matching for retrieval of cases in case based effort estimation. The objective was to estimate the development effort of student programs based on the values of certain attributes using case based reasoning model. We have used complexity metrics for retrieval of similar cases in case based estimation. Our focus is on object oriented programs. In this research, student programs were the target of study. Data regarding seven parameters were collected and the associated complexity metrics were also calculated. Two case based models were developed, one with the measured attributes and the other with the metrics as well. The improvement in prediction was 250 % for Euclidean Distance and 10 % for Manhattan Distance. The MRE improved by 79 % for Euclidean Distance but decreased by 6 % for Manhattan Distance. The Coefficient of Determination improved in both the cases. The effect of complexity metrics in similarity matching for retrieval of cases is to improve the overall performance of the models. As part of our ongoing work on metrics, we are in the process of developing coupling and inheritance based metrics and study their effect on other program parameters. We also aim to carry on the validation on more realistic data.

8. ACKNOWLEDGEMENT

This research work has been partially funded by UGC [F. No.: 33 – 61/ 2007 (SR)] under financial grants for Major Research Project.

9. REFERENCES

[1] A. Abran and N. P. 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 Fuzzy theory 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 Morocco.

[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, Software Engineering Economics, Englewood Cliffs, NJ: Prentice Hall, 1981.

[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] B. Boehm, S. Chulani, et. al., "Software Cost Estimation with COCOMO II', Prentice Hall, 2000.

[10] 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.

[11] G. Booch, *Object Oriented Design with Applications*, Benjamin/ Cummings, Menlo Park, CA, 1995.

[12] 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.

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

[14] K. Ganesan, 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.

[15] K. Rajnish and V.Bhattacherjee, "Maintenance of Metrics through class Inheritance hierarchy", in P*roceedings of International conference on Challenges and Opportunities in IT Industry",* PCTE, Ludhiana, 2005, pp. 83.

[16] K.Rajnish and V.Bhattacherjee," A New Metric for Class Inheritance Hierarchy: An Illustration*",* in *Proceedings of National Conference on Emerging Principles and Practices of Computer Science & Information Technology"*, GNDEC, Ludhiana, 2006, pp. 321-325.

[17] K. Rajnish, V. Bhattacherjee, "Class Cohesion and Development Time: A Study", in *Proceedings of National Conference on Methods and Models in Computing (NCM2C-2006),* 18-19 December, 2006, JNU, New-Delhi, India, 2006, pp. 26-34.

[18] K. Rajnish and V. Bhattacherjee, "Complexity of Class and Development Time: A Study", *Journal of Theoretical and Applied Information Technology (JATIT)*, Vol. 3, No. 1, Dresden, Germany, January 2007, pp. 63-70.

[19] 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.

[20] 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.

[21] L. C. Briand and J. K. Wust, "Modeling Development Effort in Object – Oriented Systems Using Design Properties", *IEEE Trans. Software Eng.,* 27, 11 (2001), 963 – 986.

[22] 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.

[23] 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.

[24] M. Alshayeb and W. Li, "An Empirical Validation of Object – Oriented Metrics in Two Different Iterative Software Processes", IEEE Trans. on Software Engineering, 29, 11 (2003), 1043 – 1049.

[25] N. V. Balasubramanian, "Object - Oriented Metrics", *Asian Pacific Software Engineering Conference (APSEC-96),* December-1996, pp. 30-34.

[26] R. Pratap, *Getting Started with MATLAB – V,* Oxford University Press, 1998.

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

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

[29] S. Kumar and V. Bhattacherjee, (2005),"Fuzzy logic based Model for Software cost Estimation", in *Proceedings of the International Conference on*

Information Technology, Nov'05, PCTE, Ludhiana, India.

[30] S. Kumar and V.Bhattacherjee, (2007), "Analogy and Expert Judgment: A Hybrid Approach to Software Cost Estimation" in *Proceedings of the National Conference on information Technology: Present Practices and Challenges,* Sep'07, New Delhi, India.

[31] S. R. Chidamber and C. F. Kemerer, "Towards a Metrics Suite for Object Oriented Design", in *Proceedings of Sixth OOPSLA Conference,* 1991, 197 – 211.

[32] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering,* 20, 6(1994), 476 – 493.

[33] V. Bhattacherjee and K.Rajnish, "Class Complexity: A Case Study", *Proceedings of First International Conference on Emerging Applications of Information Technology (EAIT-2006)*, Kolkata, India, 2006, pp. 253-258.

[34] 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.

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

[36] 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, Bhubaneswar, India, IEEE Computer Society.

[37] V. Bhattacherjee, S. Kumar and E. Rashid (2008), "Estimation of Software development Effort in University Setting: A Case Study", in *Proceeding of the National Conference on Architecturing Future IT Systems* NCAFIS – 08, Indore, pp. 40 - 43

[38] Wei Li, (1998)"Another Metric Suite for Object-Oriented Programming", *The Journal of System and Software*", 44, pp. 155-162.

[39] J. Bode, (1998)"Decision Support with Neural Networks in the Management of Research and Development: Concepts and Application to Cost Estimation", Information and Management, no. 34, pp. 33 – 40

[40] R. Hughes, (1996), "Expert Judgement as an Estimating Method", Information and Software Technology, Vol. 38, no. 2, pp. 67 – 75, 1996

[41] V. Bhattacherjee, (2008), "Use of Complexity Metric for Similarity Matching in Software Development Effort Estimation", Accepted for presentation at NCM2C, JNU, New Delhi, December 8 – 9, 2008

APPENDIX