Software Cost Estimation through Entity Relationship Model

Arshid Ali [1], Salman Qadri [2], Syed Shah Muhammad [2], Jalil Abbas 3, Muhammad TariqPervaiz [2], Sarfaraz Awan [2]

[1.] Department of Computer Science, GCU Faisalabad, Pakistan
[2.] Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan
[3.] Department of Computer Science, University of Central Punjab, Lahore, Pakistan
sayyed_qadri@hotmai.com

Abstract: Software Cost Estimation is essential for efficient control and management of the whole software development process. Today, Constructive Cost Model (COCOMO 11) is very popular for estimating software cost. In Constructive Cost Model lines of code and function, points are used to calculate the software size. Actually, this work represents the implementation stages but in early stages in software development, it was not easy to estimate software cost. The entity relationship model (ER Model) is very useful in requirement analysis for data concentrated systems. This paper highlights the use of Entity Relationship Model for software cost estimation. Pathway Density is ushered in. By using the Pathway Density and other factors, many regression models are built for estimating the software cost. So in this paper, Entity Relationship Model is based on estimated cost of software. [Journal of American Science. 2010;6(11):47-51]. (ISSN: 1545-1003).

Keywords: ER Model, Cost Estimation, Entity

1. Introduction

Software cost estimation is critical process for software development. It is important for efficient control and management of the whole software development process. A number of models, such as Farr and Zagorski Model, Woverton Model have been anticipated [1].Now days Constructive Cost Model (COCOMO11) is popular cost Estimation model. COCOMO 11 is divided into three sub models:

- Applications Composition
- Early design strategy
- post-architecture strategy

They can be blended in various ways to deal with the current and likely future software practices market place [4] it accesses the software effort based on the same model:

$$E = a\,(EDSI)^{b} \times EAF$$

Where E is an effort estimate, expressed in person-months. EDSI refers to the number of Estimated Delivered Source Instructions. The parameters *a* and *b* are determined by the application complexity mode. EAF (Effort Adjustment Factor) is equal to one for the basic sub-model, and equals the product of fifteen cost factors for the intermediate and advanced sub-models [3, 4].

COCOMO II uses Function Points or Lines of Code for estimating the size of a software system. Intuitively, lines of code cannot be obtained or estimated at the early stage of the software development. Function Point appears to be requirements oriented. However, Function Point counts the number of files updated and reports printed, etc, which are actually the result of design. As a result, it also confronts many problems [2]. This research proposes the use of the popular data model, ER model, for the estimation of software cost. ER model is usually constructed in the requirements analysis stage. The organization of this paper is as follows: section 2 shows the background of ER model and the factors we want to use in our research. Section 3 proposes a new term, Path Complexity and how to count the value of Path Complexity. Section 4 illustrates the new estimating model for software cost that is based on the multiple linear regression technique and gives the conclusion.

2. Background

The ER Diagram was brought to limelight by Professor Chen in 1976 [8] and adapted in the Information Engineering approach. The ER Diagram originally used in the database field and now is being used in Object-Oriented Analysis. An ER model is constructed to show the ideal organization of data, independent of the physical organization of the data and where and how data are used. Currently, data-intensive systems constitute a main domain in software. These systems maintain a large amount of structured data in a database built through a database management system (DBMS). Although UML (Unified Modeling Language) has gained its popularity as a standard software modeling methodology, ER model is still used to model the data conceptually in the requirement capturing and

analysis stages. Moreover, most of the design and development activities are based on the ER model. Therefore, ER model seems to have the most readily available information from requirements capturing and analysis stages. After studying the ER Diagram, we find that ER diagram is a non-directional graph[6]. It is easy to sketch an ER Diagram onto a directional graph by considering the entity as the vertex in a graph, and considering the relationship as the edge in a graph. From our observation, software effort is effected by the number of entities, relationships and attributes. In addition, it can be seen that more complex structure of ER Diagram is, mere effort that will be spent on the software system. Thus, we want to find out some relationships with software effort. In this research, we use the following metrics in the estimating model [7, 8]:

NOE: the number of entities in an ER Diagram.

NOR: the number of relationships in an ER Diagram.

NOA: the number of attributes in an ER Diagram.

NOP: the number of Path Complexity of an ER Diagram.

3. Complexity Metrics:

The ER Diagram shows the whole structure of the Database, where one entity can access the other entities through the relationship and get the related data. More ways one entity can access the other entities, more things we should consider about it. This can reflect the whole system complexity. Thus, we want to quantify the data that can be used in the software system. It is a problem of complexity metrics of the software product. Path Complexity is proposed as a complexity metrics to measure software effort [5]. In order to quantify the whole system data, we should first know through how many paths one entity could influence the other entities, and the length of each path. Because an ER Diagram can be converted into a graph, we can calculate these data by using graph theory path.

Definition 1:

Path Complexity of a vertex is:

$$Pi = \sum_{j \neq i} 1/n \sum_{j \neq i} l_{ij}$$

Where

The *i*th vertex is not the same to the *j*th vertex;

Pi is the Path Complexity of the *i*th vertex;

n is the number of the paths through which *i*th vertex can access the *j*th vertex;

l is the length of each path through which *i*th vertex can access the *j*th vertex.

Definition 2

Path Complexity of an ER Diagram is:

$$P = \sum_{v \in G} \mathbf{P_V}$$

Where

P is the Path Complexity of the whole ER Diagram;

Pv is the Path Complexity of the *v*th vertex in the ER Diagram. Algorithm *Search Path (G,s)* given by following is used to count how many paths we can get from a fixed vertex in a connected graph to the other vertices in the same graph, and the length of each. We assume that the input graph *G= (V,E)* is a connected graph and it can be represented using adjacency matrix. Each vertex u in the connected graph has a timestamp *time*[*u*], it records how many edges it has in one path from *s* (the beginning vertex) to *u* (the destination vertex). We use *P*[*u*] to record a set of vertices that are ahead of the vertex *u* through a path. *N*(*G*) is a vertex set to present the vertices left in a connected graph

Search Path
1 **for** each component *G*
2 **while** ( ) (*G N)*
3 do select *s* *N* (*G*) as the Beginning Vertex
4 for each *t* } {) (s *G N*
5 set *t* = EndVertex
6 for each vertex *u*) (*G V*
7 *time*[*u*] = 0
8 *P*[*u*] NIL
9) (*s Search*)
(*s Search*

1 for each vertex *u* Adj[*s*]

2 *P*[*u*] *P*[*s*] + *s*

3 *time*[*u*] *time*[*s*] +1

4 if *u* = *t*

5 print *time*[*u*]

6 else if *u* *P*[*u*]

7 break

8 else) (*s Search)*

3.1 Proposed Model

We proposed a model to estimate software effort (shown in Figure 1). It contains an Adjacency Matrix Generator, A Path Complexity Generator, a Metric Generating Tool and a Statistical Module.

Among the existing techniques used in software estimation, regression-based techniques are the most popular ways of building models. After comparing four techniques (regression, neural networks, Case-Based Reasoning, Rule induction)[3] shows a result that regression and Case-Based Reasoning perform better than the other techniques [4] also compares the methods used in regression, neural networks and genetic programming. We winds up that although neural networks and genetic programming can improve the estimations of regression, the results are not very impressive. Thus in its multiple linear regression model was adopted. All system data used in this project are actual industry data. Several software development companies in Singapore and Pakistan were considered, and provided twelve software systems' data. These projects cover multiple application domains including freight management, quotation, billing or order processing. We got data of NOE, NOR, NOA and NOP from those software systems. And we use Man-Day to measure software cost. (Shown in Figure1).

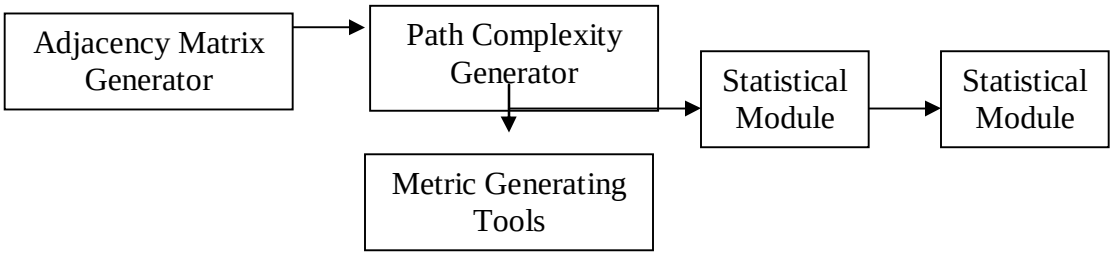

Figure 1: Complexity path definition

Factors and software cost, we use the coefficient of correlation *r*. The closer *r* is to 1, the stronger the positive linear relationship is.the values of r to NOE, NOR, NOA and NOP are respectively 0.9395, 0.9729, 0.9474 and 0.8842. According to these values, all these factors have the strong linear relationships with software cost. We got the multiple regression equation are Y = 23.01 + 3.80 NOV + 1.03NOR – 0.01 NOA – 0.49 NOP to check the accuracy of this multiple linear regression model, we use the multiple coefficient of determination R2 and the F-statistic. R2 can evaluate the strength of the multiple regression relationship. In this project, the value of R2 is 0.8766. It shows this model has a high regression relationship. In order to use the F-statistic, the hypotheses were set as below, and

alpha =0.05:

H0: all the regression coefficients are zero

H1: not all the regression coefficients are zero

After *F*-statistic, it gives a p-value of 0.0171; this value is much smaller than, so H0 was rejected. It indicates that it is highly unlikely that all of the regression coefficients are zero. Therefore, we can jump to a conclusion that this multiple regression model is reasonable. The comparison of estimating cost and actual cost is shown in figure 2.Here, the authors are grateful to the support and help of AKEMCO Technology Pvt Ltd, IPACS e-Solutions(S) Pvt Ltd and Singapore Computer Systems Ltd. to provide the system data. However, only twelve projects are not enough, more system data will be collected in the future work.

	NOE	**NOR**	**NOA**	**ODP**	**Predictive Cost**	**Actual Cost**
1	6	5	112	9.33	45	48
2	6	5	75	8.33	46	38
3	21	22	512	63.17	90	81
4	3	2	86	2.67	34	29
5	3	2	86	2.67	34	34
6	25	33	656	118.03	88	92

7	4	3	66	5.00	43	33
8	8	7	212	19.00	49	70
9	14	14	126	31.43	74	80
10	16	17	441	33.93	80	85
11	64	69	1524	204.15	324	322
12	38	38	779	84.36	225	235

Figure 2 Predictive Cost and Actual Cost

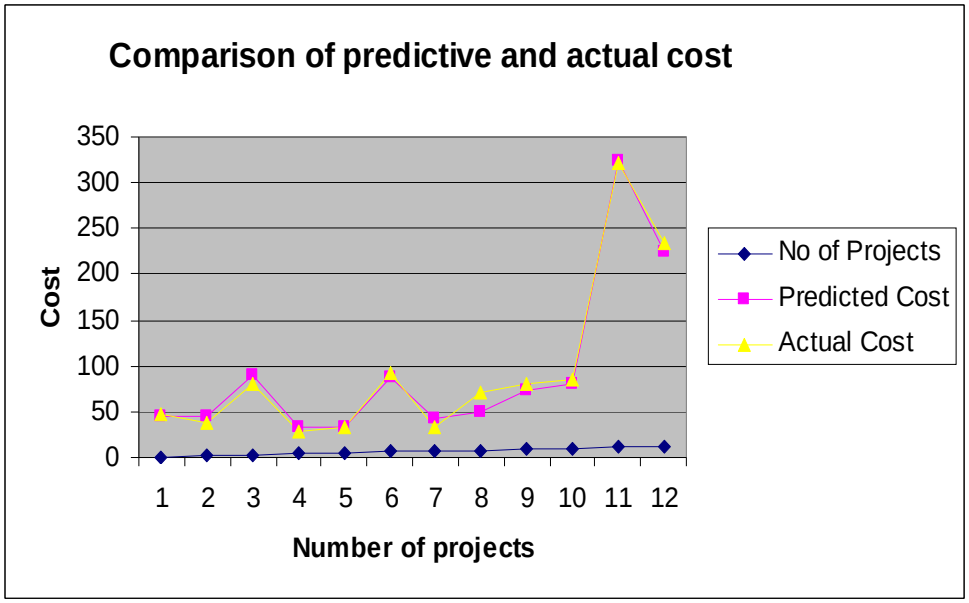

Figure 3. Comparison of Predictive and Actual Software Cost

This work is exploring a software cost estimation model, especially for software development industries [10] and it is best software cost estimation model for large-scale development and in house development. It is very plain model for software costing rather than the difficult techniques like COCOMO MODEL, which is not efficient in the environment of Pakistan. We can estimate a cost of a project from scratch by using this simple model. our aim to provide a model and a user-friendly tool to do those estimations in order to assist managers assessing the worthiness of the investment they are going to undertake. We believe that massively collected software project data present an interesting aspect of cost modeling, providing a unique opportunity to design helpful tools for software managers that wish to benchmark their projects and are interested in developing knowledge concerning software measurement and estimation [9].

4. Future Work:

This research focuses on the cost estimation technique and finds the best way cost estimation through ER Diagram for large-scale project and small projects. For future Extension, this area requires the improvement by using agile software cost methodology implementation technique to finds out the software cost estimation for better and more efficient way.

Acknowledgements:

Authors are grateful to the Department of Computer Science, Virtual University of Pakistan, for support to carry out this work.

Corresponding Author:
Syed Shah Muhammad
Department of Computer Science,
Virtual University of Paksitan,
Lahore, Pakistan.
E-mail: sayyed_qadri@hotmail.com

5. References:

1. Boehm B, Software Cost Estimation with COCOMO II. Prentice Hall PTR, 2000, pp 436
1. Stirilng G, Wilsey B. Emprical relationships between species richness, eveness and proporational diversity. Am Nat 2001;158(3):286-99.
2. Chen P. P, The Entity-Relationship model – towards a Unified View of Data. ACM Trans, Database Syst, 1(1), Mar,1976: pp. 9-36.
3. Dolado J, Limits to the Methods in Software Cost Estimation, SCASE 99, Soft Computing Applied to Software Engineering, Limerick University Press. 1999 pp 63-68,
4. Londeix B, Cost Estimation for Software Development, STC Telecommunications,UK, Addison-Wesley Publishing Company. 1987, pp386
5. Magne Jorgeson & Martin January, 2007 ShepperdIEEE Transactions on Software Engineering, , 1987 Volume 33, Issue No. 1
6. Mair C, "An Investigation of Machine Learning Based Prediction Systems", pp102
7. Nikolaos Mittas & Lefteris Angelis May, Comparing cost prediction models by resampling techniques, Journal of Systems and Software, 2008 Volume 81 , Issue 5 PP: 616-632,ISSN:0164-1212
8. Ruhe G, Attribute Selection and Weighting Using Rough Sets for Effort Estimation by Analogy—Initial Results, Software Engineering Decision Support Laboratory at the University of Calgary, Canada, 2007, PP:46-102.
9. Stefan Gueorguiev, Mark Harman and Giuliano Antonio Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering, Montreal, Québec, Canada .SESSION: Track 14: search based software engineering Pages: 1673-1680, 2009 ISBN:978-1-60558-325-9
10. Zhong, N, and Dong, J. Using Rough Sets with Heuristics for Feature Selection, Journal of Intelligent Information Systems 2001 16 PP:199-214.73.

4/4/2010