2011 International Conference on Computer and Software Modeling
IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore

Regression Model for Software Effort Estimation Based on the Use Case Point Method

Ali Bou Nassif[1,a], Danny Ho[2,b] and Luiz Fernando Capretz[1,c +]

[1] Department of ECE, University of Western Ontario, London, Ontario, Canada

[2]NFA Estimation Inc., Richmond Hill, Ontario, Canada

Abstract. It is very important to conduct software estimation in the early stages of the software life cycle, because it helps managers bid on projects and allocate resources efficiently. This paper presents a novel regression model to estimate the software effort based on the use case point size metric. The use case point model takes use case diagrams as input and gives the software size in use case points as output. The proposed effort equation takes into consideration the non-linear relationship between software size and software effort, as well as the influences of project complexity and productivity. Results show that the software effort estimation accuracy can be improved by 16.5% using PRED(25) and 25% using PRED(35).

Keywords: Use Case Points, Size Estimation, Effort Estimation, Regression Model

1. Introduction

The use case point method was proposed by Karner in 1993 [1]. It is one of the methods that can be used for size and effort estimation which is based on use case diagrams of software projects. The use case point size is calculated by counting the number of use cases and actors, multiplied by their complexity weights. The complexity of each use case is calculated by counting the number of transactions in the success and extension scenarios of the use case scenario. Tables 1 and 2 present the complexity rates of the use cases and actors respectively.

The use case point size is calculated through two stages. This includes the Unadjusted Use Case points (UUCP) and the Adjusted Use Case Points (UCP). The UCCP is expressed in Equation 1 [1].

$$UUCP = \sum_{i=1}^{6} n_i * W_i. \quad (1)$$

where n_i is the number of items of variety i (use case or actor) and W_i is the complexity rate. In order to calculate the UCP, technical and environmental factors (TFs and EFs) of the project should be considered. TF contributes to the complexity of the system whereas EF contributes to the efficiency of the system. TF and EF are shown in tables 3 and 4 respectively. TF is calculated as:

$$TF = C1 + C2 \sum_{i=1}^{13} F_i * W_i. \quad (2)$$

where $C1 = 0.6, C2 = 0.01$ and F_i is a factor that takes values between 0 and 5. The value "0" means the factor is irrelevant while the value "5" is essential. The value "3" means that the factor is not very essential, nor irrelevant. For instance, if all the factors have a value of "3", the TF will be 1.

+ [a]abounass@uwo.ca, [b]danny@nfa-estimation.com, [c]lcapretz@uwo.ca

Table 1: Weighted Use Cases [1]

Use Case Complexity	Number of Transactions	Weight
Simple	3 or fewer	5
Average	4 to 7	10
Complex	More than 7	15

Table 2: Weighted Actors [1]

Actor Complexity	Description	Weight
Simple	Through an API	1
Average	Through a text-based user interface	2
Complex	Through a GUI	3

Table 3: Technical Factors [1]

F_i	Factors contributing to complexity	W_i
F_1	Distributed systems	2
F_2	Application performance objectives	1
F_3	End user efficiency	1
F_4	Complex internal processing	1
F_5	Reusability	1
F_6	Easy installation	0.5
F_7	Usability	0.5
F_8	Portability	2
F_9	Changeability	1
F_{10}	Concurrency	1
F_{11}	Special security features	1
F_{12}	Direct access for third parties	1
F_{13}	Special user training facilities	1

Table 4: Environmental Factors [1]

F_i	Factors contributing to efficiency	W_i
F_1	Familiar with Objectory	1.5
F_2	Stable requirements	2
F_3	Analyst capability	0.5
F_4	Application experience	0.5
F_5	Object oriented experience	1
F_6	Motivation	1
F_7	Difficult programming language	-1
F_8	Part-time workers	-1

On the other hand, the environmental factor EF is calculated as:

$$EF = C1 + C2 \sum_{i=1}^{8} F_i * W_i. \quad (3)$$

where $C1 = 1.4, C2 = -0.03$ and F_i a factor that takes values between 0 and 5, same interpretation as outlined previously. The adjusted use case points (UCP) can be calculated as:

$$\mathrm{UCP} = \mathrm{UUCP} \times \mathrm{TF} \times \mathrm{EF}. \quad (4)$$

The maximum value of TF is 1.3, if the value of all technical factors is 5. Typically, by taking the practical values of TF and EF into consideration, the value of UCP will be 30% more or less than UUCP. According to Karner, the software effort is calculated as:

$$\mathrm{Effort} = \mathrm{Size} \times 20. \quad (5)$$

where size is the calculated software size in UCP and the effort is measured in person-hours.

2. Problem Definition

In Equation 5, Karner proposed a method to calculate the software effort from software size. However, this method has several shortcomings. These shortcomings include:

- Karner assumed that the relationship between the effort and size is linear. This assumption is not true in real life. For instance, if the effort required to build a software project of size 250 UCP is 5,000 person-hours, the effort needed to build the same project type of size 500 UCP would be more than 10,000 person-hours.
- In Equation 5, the effort is a function of UCP size. As explained in the previous section, the UCP encompasses the non-functional requirements of the system and it may increase the original UUCP by 30%. However, IBM states that the non-functional requirements might represent more than 50% of the total effort [2]. This means that the non-functional requirements may virtually increase the unadjusted size by 100%.

The proposed model tackles these shortcomings.

3. Related Work

Some work has been done to enhance the effort estimation of UCP model. These include:

Sparks et al. [3] reported that the effort to develop one UCP should be between 15 and 30 person-hours. However, the authors did not state in details how this range should be applied.

Schneider et al. [4] mentioned that although the environmental factors are included in UCP, they should also be considered while calculating the effort. The authors suggested counting the number of factor ratings of

F1-F6 in Table 4 (EF) that are below 3 and the number of factor ratings of F7-F8 that are above 3. If the total is less than 3, then 20 person-hours per UCP should be used. If the total is 3 or 4, then 28 person-hours per UCP should be used. If the total is 5 or more, then the project team should be reconstructed so that the numbers fall at least below 5. A value of 5 indicates that this project is at significant risk of failure with this team. The main limitation of this method is that the effort required to develop one UCP is either 20 or 28 person-hours.

Nassif et al. [5] applied a fuzzy logic approach to enhance UCP model. However, the original software effort equation has been used in this work (Equation 5).

4. Research Methodology and Implementation

The general equation of software effort can be represented as [6]:

$$\text{Effort} = \frac{Complexity}{Productivity} \times Size. \quad (6)$$

where *Complexity* is the complexity factor of a project and *Productivity* is the productivity factor of the team that is developing this project. The first step of the proposed model is to discover the non-linear relationship between software size and software effort. For this purpose, regression analysis was applied on several projects that have similar project complexity and team productivity. Regression analysis assumes that data should be normally distributed [7]. If the histograms of software size and effort were normally distributed, the regression equation would be:

$$\text{Effort} = a \times \text{Size} + b. \quad (7)$$

where a and b are constants.

Several experiments were conducted using Minitab to determine how data were distributed. The histograms of software size and effort showed that data were not normally distributed. After normalizing data using logarithmic transformation (ln size and ln effort), data became normally distributed. The regression equation after logarithmic transformation using Minitab version 16 is:

$$\text{Effort} = 8.16 \times Size^{1.17} . \quad (8)$$

where *Size* is the software size in UCP and *Effort* is the effort in person-hours. For instance, Equation 8 shows the non-linear relationship between Effort and Size, and ignores the Complexity and Productivity factors. The main equation of software effort is expressed in Equation 9.

To measure the accuracy of the regression equation (equation 8), we measured the value of R^2. R^2 is the percentage of variation in Effort explained by the variable Size. The value R^2 reported for the regression model in Equation 8 is 0.972. Approximately 97 % of the variation in Effort can be explained by the variable Size. This shows a strong relation between Size and Effort.

Karner only took the influence of complexity and productivity factors when calculating the adjusted use case size (UCP). As seen in Equation 5, the software estimation based on Karner's model is a function of UCP. Schneider [4] reported that the software effort is a function of the UCP and team productivity. The non-productive team might raise the estimation effort by 40% (28 person-hours for 1 UCP versus 20 person-hours for productive team). Francalanci et al. [8] argue that project complexity should be taken into consideration and this can increase the software effort by 27%. We believe that the software effort should be a function of UCP, complexity and productivity. The main equation for software effort in the proposed model is expressed as:

$$\text{Effort} = 8.16 \times \frac{Project\_\text{Complexity}}{Productivity} \times (Size)^{1.17}. \quad (9)$$

Karner's technical factor TF (see Equation 2) increases or decreases the unadjusted size by 30%. Although the technical factors in Table 3 do not include all complexity factors, yet, Karner's technical factor TF can represent the Project_Complexity factor during the estimation of UCP and consequently, the Project_Complexity factor in Equation 9 can be ignored.

With respect to productivity, Table 4 lists some productivity attributes. Better approach than Schneider has been taken to calculate the Productivity factor. Based on Table 4, the highest Productivity factor is achieved when the value of the factors F1 to F6 is 5 and the value of the factors F7 and F8 is 0. This implies that the value of $(\sum_{i=1}^{8} F_i * W_i)$ is 32.5. On the other hand, the lowest productivity is achieved when the value of F1 to F6 is set to 0 and the value of F7 and F8 is set to 5. This implies that the value of $(\sum_{i=1}^{8} F_i * W_i)$ is -10. The Productivity factor is determined based on the value of $(\sum_{i=1}^{8} F_i * W_i)$. Table 5 shows the proposed values of Productivity:

Table 5: Productivity Factor

$(\sum_{i=1}^{8} F_i * W_i)$	**Productivity Description**	**Productivity Factor**
Less than 0	Very Low	0.4
Between 1 and 10	Low	0.7
Between 11 and 20	Average	1
Greater than 20	High	1.3

If the productivity is very low, there is a big risk that the project might not be completed. A Productivity Factor of value 0.4 indicates that the effort estimation should be increased by 150%. If the productivity is high, the effort estimation will be decreased. If the project complexity is high and the productivity is low, the effort estimation will increase between 80% and 100% which reflects the IBM's assumption.

Based on the assumptions above, the Project_Complexity factor can be ignored while the productivity factor remains. The final equation of the proposed model is:

$$\text{Effort} = \frac{8.16}{Productivity} \times (Size)^{1.17}. \quad (10)$$

where Effort is in person-hours, Size is in UCP and Productivity is a value between 0.4 and 1.3.

5. Evaluation

The evaluation of the proposed model was performed using 24 projects that were not included among the projects used in the regression analysis. Software estimation was conducted using Karner's, Schneider's, and the proposed model. In software estimation, most practitioners use MMRE and PRED(x) to calculate the error percentage. MMRE is the Mean of the Magnitude of Relative Error and it is a very common criterion used to evaluate software cost estimation models [9]. The Magnitude of Relative Error (MRE) for each observation i can be obtained as:

$$MRE_i = \frac{|\,Actual\ Effort_i - Predicted\ Effort_i\,|}{Actual\ Effort_i} \quad (11)$$

MMRE can be achieved through averaging the summation of MRE over *N* observations:

$$MMRE = \frac{1}{N} \sum_1^N MRE_i. \quad (12)$$

On the other hand, PRED(x) is the percentage of projects for which the estimate falls within x% of the actual value. For instance, if PRED(30) = 60, this indicates that 60% of the projects fall within 30% error range. Table 6 shows the evaluation results of Karner's, Schneider's and the proposed model. The columns Kar, Sch, Pro, Imp_Sch_Kar, Imp_Pro_Kar and Imp_Pro_Sch represent Karner, Schneider, the proposed model, improvement of Schneider over Karner, improvement of the proposed model over Karner and improvement of the proposed model over Schneider. In table 6, the results show that the proposed model improves the MMRE of original use case point model by 6%. Moreover, the results also show substantial improvements in PRED(x) over Karner. PRED(25) and PRED(35) were improved by 16.5% and 25% respectively.

Table 6: Comparison Between Karner, Schneider and the Proposed Model

Criteria	**Kar (%)**	**Sch (%)**	**Pro (%)**	**Imp_Sch_Kar (%)**	**Imp_Pro_Kar (%)**	**Imp_Pro_Sch (%)**
MMRE	34.0	30.0	28.0	4.0	6.0	2.0
PRED (25)	37.5	62.5	54.0	25.0	16.5	-8.3
PRED (35)	50.0	66.6	75.0	16.6	25.0	8.3
PRED (50)	87.5	83.3	91.6	-4.2	4.1	8.3
PRED (75)	95.8	91.7	95.8	-4.1	0	4.1
PRED (100)	100	95.8	95.8	-4.2	-4.2	0

6. Conclusions

The use case point model has been used to conduct early software effort estimations. The main advantage of this model is that it is simple and can be easily automated. However, the original model ignores the non-linear relationship between software effort and size and it lacks accuracy in the effort estimation. A new regression model has been proposed to tackle these drawbacks.

Future work will focus on two main concerns. First, the proposed model should be tested with projects of larger sizes (greater than 5,000 person-hours) when data are available. Secondly, the productivity factors proposed in Table 5 should be calibrated using a fuzzy logic approach.

7. References

[1] G. Karner. Resource Estimation for Objectory Projects. *Objective Systems,* 1993.

[2]Y. Ossia. (2011), IBM haifa research lab. *IBM Haifa Research Lab* [Online]. Available: https://www.research.ibm.com/haifa/projects/software/nfr/index.html.

[3] S. Sparks and K. Kaspczynski. *The art of sizing projects* . Sun Works, 1999.

[4] G. Schneider and J. P. Winters, *Applied use Cases, Second Edition, A Practical Guide.* Addison-Wesley, 2001.

[5] A. B. Nassif, L. F. Capretz and D. Ho, "Enhancing Use Case Points Estimation Method using Soft Computing Techniques," *Journal of Global Research in Computer Science,* vol. 1, no. 4, pp. 12-21, November, 2010.

[6] D. D. Galorath and M. W. Evans, *Software Sizing, Estimation, and Risk Management.* Boston, MA, USA: Auerbach Publications, 2006.

[7] A. C. Cameron and P. K. Trivedi. *Regression Analysis of Count Data.* Cambridge, UK: Cambridge University Press, 1998.

[8] C. Francalanci and M. Francesco. The impact of complexity on software design quality and costs. in *European Conference on Information Systems,* 2008, pp. 1-13.

[9] A. B. Nassif, L. F. Capretz and D. Ho. Software estimation in the early stages of the software life cycle. in *International Conference on Emerging Trends in Computer Science, Communication and Information Technology,* 2010, pp. 5-13.