On the Value of Parameters of Use Case Points Method

Tomas Urbanek, Zdenka Prokopova, and Radek Silhavy

Faculty of Applied Informatics
Tomas Bata University in Zlin
Nad Stranemi 4511
Czech Republic
turbanek@fai.utb.cz

Abstract. Accurate effort estimates plays crucial role in software development process. These estimates are used for planning, controlling and managing resources. This paper deals with the statistical value of Use Case Points method parameters, while analytical programming for effort estimation is used. The main question of this paper is : Are there any parameters in Use Case Points method, which can be omitted from the calculation and the results will be better? The experimental results show that this method improving accuracy of Use Case Points method if and only if UUCW parameter is present in the calculation.

Keywords: analytical programming, differential evolution, use case points, effort estimation.

1 Introduction

Effort estimation is the activity of predicting the amount of effort that is required to complete a software development project [1]. Accurate and consistent prediction of effort estimation is a crucial point in project management for effective planning, monitoring and controlling of software projects. Better management decisions could be made with more accurate effort estimates. It is also very important to predict these estimates in the early stages of software development [2]. Atkinson et al. [3] claims that regression analysis does not provide enough accuracy. Therefore, the use of artificial intelligence may be a promising way to improve accuracy of effort estimations.

Attarzadeh et al. [4] declare that effort estimation in software engineering is divided into two categories.

- Algorithmic methods
- Non-algorithmic methods

Algorithmic methods are based on a mathematical formula and relies on historical datasets. The most famous methods are COCOMO [5], FP [3] and UCP method [2]. To the second category belong methods like expert judgement and analogy based methods. The most famous method of this group is Delphi [6].

R. Silhavy et al. (eds.), *Artificial Intelligence Perspectives and Applications*,
Advances in Intelligent Systems and Computing 347, DOI: 10.1007/978-3-319-18476-0_31

The quality of software could be reduced, when we underestimate the effort. This could cause dysfunctional software and raise the cost for testing and maintenance [7]. With more accurate effort estimations the management of software projects could be less challenging [2]. Therefore, we need more accurate predictions to effectively manage software projects [8].

This study offers some important insights into the statistical value of Use Case Points parameter. No previous study has investigated the importance of parameters of the Use Case Points method, while analytical programming method is used. What is not yet clear is the impact of UUCW parameter of this method. Therefore, this study makes a major contribution to research on the statistical value of Use Case Points method parameters, while analytical programming method is used.

1.1 Related Work

Despite of a lot of effort of scientists, there is no optimal and effective method for every software project. Very promising way is a research of Kocuganeli et al. [9], this paper shows, that ensemble of effort estimation methods could provide better results then a single estimator. Because we need an effort estimate as soon as possible, there is a strong pressure to predict these estimates after requirements are built. There is a work of Silhavy et al. [10] which proposed a method for automatic complexity estimation based on requirements. The work of Kaushik et al. [8] and Attarzadeh et al. [4] uses neural networks and COCOMO [5] method for prediction. COCOMO method is widely used for testing and calibrating in cooperation with artificial intelligence or with fuzzy logic [11]. Neural networks in these cases search parameters of the regression function. Unlike presented method, which search for the regression function itself. Differential evolution and analytical programming are used for this task. Because it is very difficulty to obtain a reliable dataset in case of Use Case Points method, this paper shows results on a dataset from Poznan University of Technology [12] and from this paper [13]. Presented approach is an evolution of my previous work [14], in this research is used differential evolution instead of self-organizing migration algorithms.

1.2 Use Case Points Method

This method was presented in 1993 by Gustav Karner. This method is based on the similar principle as function point method. The project manager have to estimate project parameters to four tables. These tables are following

- Unadjusted Use Case Weight (UUCW) can be seen in Table 1
- Unadjusted Actor Weight (UAW)can be seen in Table 2
- Technical Complexity Factor (TCF) can be seen in Table 3
- Environmental Complexity Factor (ECF) can be seen in Table 4

Table 1. UCP table for estimation unadjusted use case weight

Use Case Classification	No. of Transactions	Weight
Simple	1 to 3 transactions	5
Average	4 to 7 transactions	10
Complex	8 or more transactions	15

Table 2. UCP table for actor classification

Actor Classification	Weight
Simple	1
Average	2
Complex	3

1.3 Differential Evolution

Differential evolution is a optimization algorithm introduced by Storn and Price in 1995 [15]. This optimization technique is evolutionary algorithm based on population, mutation and recombination. Differential evolution is simple to implement and have only four parameters to set. These parameters are Generations, NP, F and Cr. Parameter Generations determines the number of generations, NP is population size, parameter F is weighting factor and parameter CR is crossover probability.[16]

1.4 Analytical Programming

Analytical programming (AP) is a tool for symbolic regression. The core of analytical programming is a set of functions and operands. These mathematical objects are used for synthesis a new function. Every function in the set of analytical programming core has various numbers of parameters. Functions are sorted by these parameters into general function sets (GFS). For example GFS_{1par} contains functions that have only 1 parameter like $sin()$, $cos()$ and other functions. AP must be used with any evolutionary algorithm that consists of a population of individuals for its run [17] [18]. In this paper is used differential evolution (DE) as the evolutionary algorithm for analytical programming [15]. The function of AP is following:

A new individual is generated by evolutionary algorithms. Then this individual (the list of integer numbers) is passed to the function of analytical programming. These integer numbers serves as an index into the general function set, where the functions are defined. Then the algorithm of analytical programming creates a new function from these indexes. After that this new function is evaluated by cost function. The evolutionary algorithm decides either this new equation is suited or not for the next evolution.

Table 3. UCP table for technical factor specification

Factor	Description	Weight
T1	Distributed system	2.0
T2	Response time/performance objectives	1.0
T3	End-user efficiency	1.0
T4	Internal processing complexity	1.0
T5	Code re-usability	1.0
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portability to other platforms	2.0
T9	System maintenance	1.0
T10	Concurrent/parallel processing	1.0
T11	Security features	1.0
T12	Access for third parties	1.0
T13	End user training	1.0

Table 4. UCP table for environmental factor specification

Factor	Description	Weight
E1	Familiarity with development process used	1.5
E2	Application experience	0.5
E3	Object-oriented experience of team	1.0
E4	Lead analyst capability	0.5
E5	Motivation of the team	1.0
E6	Stability of requirements	2.0
E7	Part-time staff	-1.0
E8	Difficult programming language	-1.0

2 Problem Definition

Dataset with values of Use Case Points method was obtained from Poznan University of Technology [12] and from this paper [13]. The Table 5 shows Use Case Points method data from 24 projects. Only data of Use Case Points method with transitions have been used in this paper in case of Poznan University of Technology dataset. There are 4 values for each software project UUCW, UAW, TCF and ECF.

Gustav Karner in his work [2] derived nominal value for calculation of staff/hours from Use Case Points method. This value was set to 20. Thus, effort estimate in staff/hours is calculated as

$$estimate = UCP * 20$$

Table 5 shows calculated differences. The equation for calculation of MRE is Equation 1.

Table 5. Data used for effort estimation

ID	Act. Effort [h]	$UCP * 20$	MRE [%]
1	3037	2971	2
2	1917	1094	43
3	1173	1531	31
4	742	2103	183
5	614	1257	105
6	492	883	79
7	277	446	61
8	3593	6117	70
9	1681	1599	5
10	1344	1472	10
11	1220	1776	46
12	720	1011	40
13	514	627	22
14	397	1884	375
15	3684	6410	74
16	1980	2711	37
17	3950	6901	75
18	1925	2125	10
19	2175	2692	24
20	2226	2862	29
21	2640	3901	48
22	2568	3216	25
23	3042	5444	79
24	1696	2127	25
MMRE			62

$$MRE = \frac{|ActualEffort - (UCP * 20)|}{ActualEffort} , \tag{1}$$

where MRE is calculated error for each project in Table 5. Results from MRE calculation and Equation 2 were used for calculation of MMRE.

$$MMRE = \frac{1}{n} \sum_{i=1}^{n} MRE , \tag{2}$$

where $MMRE$ is mean magnitude of relative error through all project in Table 5.

$$MMRE = 62\%$$

The question is, there is a parameter in this dataset which can be omitted from calculation and the accuracy of analytical programming method for effort estimation will show better results.

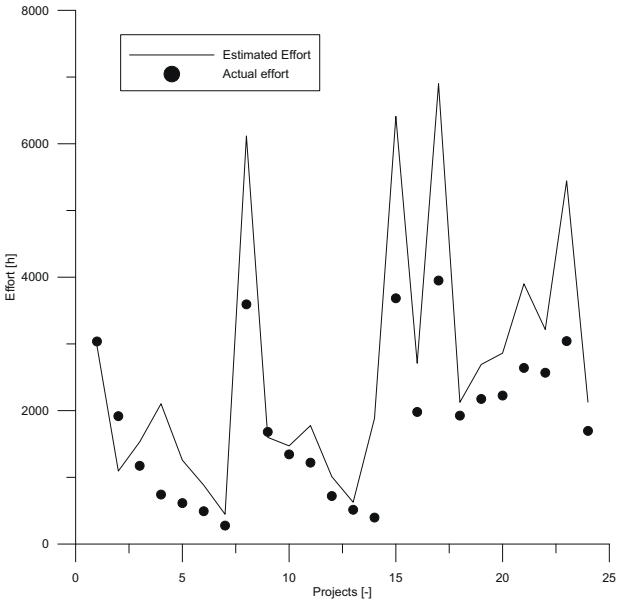

Fig. 1. Difference between estimated and real effort

3 Method

From dataset was constructed matrix A. This matrix has size MxN, where $M = 5$ and $N = 24$. Every row of this matrix A contains a calculation of Use Case Points method and actual effort.

The columns of the matrix A from beginning to end were UUCW, UAW, TCF, ECF and actual effort. Whole dataset could not be optimized by the evolutionary algorithm, because no data was remained for testing purposes. Because of this problem, the matrix A was divided into two matrices. Matrix B is training dataset and matrix C is testing dataset.

The Matrix B contains 12 rows of data for training purposes and the matrix C contains 12 rows for testing purposes. The matrix B was processed by analytical programming with the differential evolution algorithm. The Result of this process was a new equation. This new equation describes relationships between variables in training dataset, moreover in testing dataset.

Table 6 shows the set-up of differential evolution. The set-up of differential evolution is subject of further research.

Table 7 shows the set-up of analytical programming. Number of used function was set to 30, because we want to find function, which can perform better and also function complexity is not a problem. There is no need to generate short and easily memorable equations, but equations, that will be more accurate and predict effort estimation better. There was chosen linear functions like plus(), multiply() because there was a possibility that final estimation will be linear,

Table 6. Set-up of differential evolution

Parameter	Value
NP	20
Generations	60
F	0.7
Cr	0.7

Table 7. Set-up of analytical programming

Parameter	Value
Function number	30
Functions	Plus, Subtract, Divide, Multiply, Tan, Sin, Cos, Exp

Table 8. Parameter groups

One parameter	Two parameters	Three parameters	Four parameters
UUCW	UUCW,UAW	UUCW,UAW,TCF	UUCW,UAW,TCF,ECF
UAW	UUCW,TCF	UUCW,TCF,ECF	
TCF	UUCW,ECF	UUCW,UAW,ECF	
ECF	UAW,TCF	UAW,TCF,ECF	
	UAW,ECF		
	TCF,ECF		

on the other hand, there was being chosen also functions non-linear because the data from Poznan university contains strong non-linear behaviour.

3.1 Parameter Groups

Parameters of Use Case Points method were divided into parameter groups. There were 15 parameter groups and these can be seen in the Table 9. Each of these groups were calculated by an analytical programming method.

3.2 Cost Function

The new equation that is generated by the method of analytical programming contains these parameters UUCW, UAW, TCF and ECF. There is no force applied to analytical programming, that equations generated by this method have to contain all of these parameters. Cost function that is used for this task is following:

$$CF = \sum_{i=1}^{n} |B_{n,5} - f(B_{n,1}, B_{n,2}, \ldots, B_{n,4})| \,, \tag{3}$$

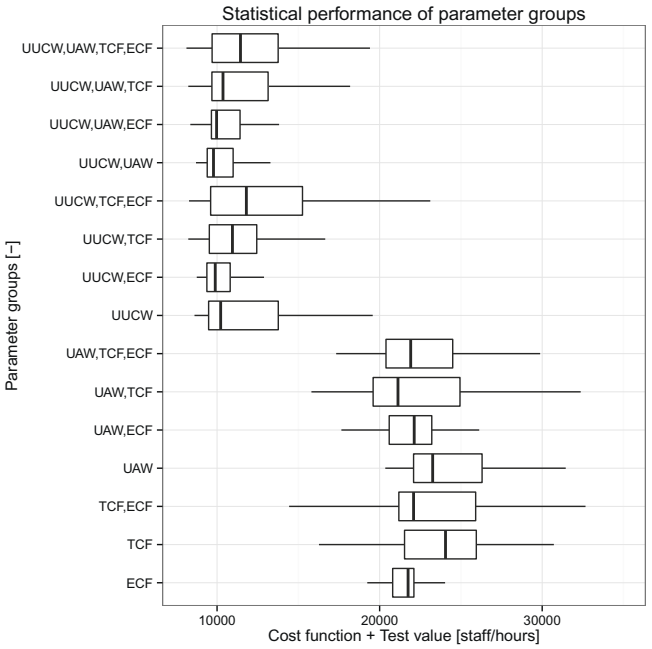

Fig. 2. Statistical performance of each parameter group

where n is equal to the number of projects in training dataset, $B_{n,5}$ is actual effort, $B_{n,1}$ is UUCW, $B_{n,2}$ is UAW, $B_{n,3}$ is TCF, $B_{n,4}$ is ECF.

4 Results

It was calculated 100 equations for each parameter group. Each calculation was generated in approximately 22 seconds. Simple statistical analysis was used to evaluate these calculations.

The Figure 2 shows the statistical performance of each parameter group. What is interesting in this data is that the performance of these calculations can be divided into two groups. The parameter groups which contain UUCW parameter, perform better than parameter groups without UUCW parameter. Further analysis showed that the best results give parameter groups UUCW,ECF and UUCW,UAW. On the other hand the worst result gives parameter group TCF,ECF.

The Figure 3 shows generated data for 15000 equations after removing extreme values. These extreme values have not been calculated in training data, because these values have been removed by natural selection of the differential evolution algorithm. On this figure can be seen a lot of calculations, which have estimation

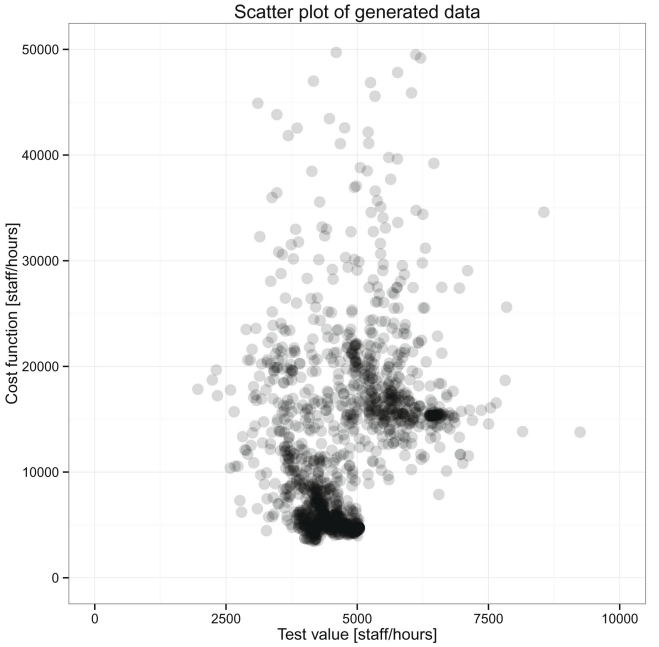

Fig. 3. Scatter plot of generated data

Table 9. Simple statistical analysis of parameter groups

Parameter group	N	Minimum	Quartile 25	Median	Quartile 75	Maximum
UUCW,UAW	100	8704	9439.25	9916.5	13362.00	1000000
UUCW,ECF	100	8750	9474.00	10072.0	12906.00	1000000
UUCW,UAW,ECF	100	8355	9731.00	10115.5	12296.25	1000000
UUCW,UAW,TCF	100	7642	9632.50	10440.5	14325.25	1000000
UUCW,TCF	100	7620	9371.50	10946.0	13262.25	1000000
UUCW,UAW,TCF,ECF	100	7722	9621.50	11630.5	15158.00	1000000
UUCW,TCF,ECF	100	7678	9604.25	12068.5	15525.00	1000000
UUCW	100	8611	9647.50	12551.5	27409.25	1000000
ECF	100	14693	20932.00	21801.5	22644.50	1000000
UAW,ECF	100	14142	21071.75	22919.0	42457.50	1000000
TCF,ECF	100	14433	21632.00	23308.5	27956.75	1000000
UAW,TCF,ECF	100	17327	21009.25	23345.0	55521.25	1000000
TCF	100	14142	21953.00	24351.5	26871.25	1000000
UAW,TCF	100	15804	20498.00	24886.0	48705.00	1000000
UAW	100	20348	22621.00	25537.0	40015.50	1000000

improving parameters. The cluster of calculations near point [5000, 5000] were the calculation of parameter groups which contained parameter UUCW. And the cluster of calculation near point [5000,16000] were calculated without UUCW parameter.

The Table 9 contains simple statistical analysis for each parameter group. As can be seen the minimum can be found in UUCW,TCF group. Nevertheless the median with minimal value can be found in UUCW,UAW parameter group. The same maximum value in each parameter group means that at least one equation of 100 calculations of each parameter group contains pathological equation and value 1000000 is penalization.

5 Conclusion

The main goal of the current study was to determine the statistical value of Use Case Points method parameters, while the analytical programming method is used. This study shown that calculation without UUCW parameter results in significant statistical differences. This finding suggests that in general the UUCW parameter is the most important parameter in Use Case Points method. It was also shown that this method could improve estimation with error of 7620 staff/hours in this dataset of 24 software projects. This research extends our knowledge about significance of the UUCW parameter in Use Case Points method. Further research might explore the significance of these parameters, whether some of the projects will be omitted from the calculations.This research has thrown up many questions in need of further investigation.

Acknowledgement. This study was supported by the internal grant of TBU in Zlin No. IGA/FAI/2014/019 funded from the resources of specific university research.

References

1. Keung, J.W.: Theoretical Maximum Prediction Accuracy for Analogy-Based Software Cost Estimation. In: 2008 15th Asia-Pacific Software Engineering Conference, pp. 495–502 (2008)
2. Karner, G.: Resource estimation for objectory projects. Objective Systems SF AB (1993)
3. Atkinson, K., Shepperd, M.: Using Function Points to Find Cost Analogies. In: 5th European Software Cost Modelling Meeting, Ivrea, Italy, pp. 1–5 (1994)
4. Attarzadeh, I., Ow, S.H.: Software development cost and time forecasting using a high performance artificial neural network model. In: Chen, R. (ed.) ICICIS 2011 Part I. CCIS, vol. 134, pp. 18–26. Springer, Heidelberg (2011)
5. Boehm, B.W.: Software Engineering Economics. IEEE Transactions on Software Engineering SE-10, 4–21 (1984)
6. Rowe, G., Wright, G.: The Delphi technique as a forecasting tool: issues and analysis. International Journal of Forecasting 15, 353–375 (1999)
7. Jiang, Z., Naudé, P., Jiang, B.: The effects of software size on development effort and software quality. Journal of Computer and Information Science, 492–496 (2007)
8. Kaushik, A., Soni, K., Soni, R.: An adaptive learning approach to software cost estimation. In: 2012 National Conference on Computing and Communication Systems, pp. 1–6 (November 2012)

9. Kocaguneli, E., Menzies, T., Keung, J.W.: On the value of ensemble effort estimation. IEEE Transactions on Software Engineering 38(6), 1403–1416 (2011)
10. Silhavy, R., Silhavy, P., Prokopova, Z.: Automatic complexity estimation based on requirements. In: Latest Trends on Systems, Santorini, Greece, vol. II, p. 4 (2014)
11. Reddy, C., Raju, K.: Improving the accuracy of effort estimation through fuzzy set combination of size and cost drivers. WSEAS Transactions on Computers 8(6), 926–936 (2009)
12. Ochodek, M., Nawrocki, J., Kwarciak, K.: Simplifying effort estimation based on Use Case Points. Information and Software Technology 53, 200–213 (2011)
13. Subriadi, A.P., Ningrum, P.A.: Critical review of the effort rate value in use case point method for estimating software development effort. Journal of Theroretical and Applied Information Technology 59(3), 735–744 (2014)
14. Urbanek, T., Prokopova, Z., Silhavy, R., Sehnalek, S.: Using Analytical Programming and UCP Method for Effort Estimation. In: Modern Trends and Techniques in Computer Science. Springer International Publishing (2014)
15. Storn, R., Price, K.: Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012 (1995)
16. Storn, R.: On the usage of differential evolution for function optimization. In: Fuzzy Information Processing Society, NAFIPS (1996)
17. Zelinka, I., Davendra, D., Senkerik, R., Jasek, R., Oplatkova, Z.: Analytical programming-a novel approach for evolutionary synthesis of symbolic structures. InTech, Rijeka (2011)
18. Zelinka, I., Oplatkova, Z., Nolle, L.: Analytic programming-symbolic regression by means of arbitrary evolutionary algorithms. Int. J. of Simulation, Systems, ... 6 (2005)