ISSN:0975-9646

Estimating Software Development Effort using UML Use Case Point (UCP) Method with a Modified set of Environmental Factors

Pragya Jha[1], Preetam Pratap Jena[2], Rajani Kanta Malu[3]
School of Computer Engineering, KIIT University,
Bhubaneswar, Odisha[1,2,3]

***Abstract* - Software developers frequently depend on use cases to describe the business processes of object- oriented projects. Since use cases consists of the strategic goals and scenarios that provide value to a business domain, they can also provide insight to a projects complexity and required resources. The traditional models for cost estimation are no longer used effectively due rapid changes in technology. UML based Use case point is found to be robust one for the cost estimation having no influence of complexity of language, complexity of development platforms etc. Calculating the software development effort using Use Case Point method is found to be a suitable method. Many of the authors have done remarkable work on effort estimation using use case points by considering the external technical factors and environmental factors. In this paper the authors attempted to modify the environmental complexity factors to obtain precise and accurate result.**

Keywords: Use Case Point, Software Cost Estimation, Environmental Factors

I. INTRODUCTION

Software estimates are very important in the context of software engineering as a means to derive effort, cost and size of a project, thus helping with the decision of whether the project is worth developing or not. Several cost estimation models are proposed by various researchers, but many of them became outdated because of the rapid changes in technology. An accurate and reliable estimation is difficult to obtain because of the lack of detailed information about the system to be developed, in the early stages of software development. Cost estimation models like COCOMO [1,2] and size estimation models like Function Point analysis are well known and frequently used in literature. These models were applicable only to procedural paradigm, and are not directly applicable to software products developed using the object oriented methodology or real time systems. With object orientation, use cases emerged as a dominant technique for structuring requirements. This technique was integrated into the Unified Modeling Language (UML) and Unified Process and became the de facto standard for Software requirements modeling [3]. It is this idea that gave birth to the creation of Use Case Point (UCP) metrics, originally developed by Gustav Karner[4]. In this method, Gustav Karner attempted to estimate the project size by assigning points to use cases, like in the same way, FPA assigns points to functions [4]. Use Case Point is simple, and it has more accuracy than lines of code or DELPHI (expert experience method), it avoid quite a difference results in the same project caused by different estimating personnel[5].

II. AIMS AND OBJECTIVES

The main objective of this research is to make the result of the UCP method more accurate and precise by adding the following environmental factors.:

- Client Type
- New Technology
- Team Co-ordination
- Organisation library availability
- Team Composition
- Growth Rate of Organization

III. USE CASE POINT METHOD

The UCP method is the extension of Function Point method with the benefit of requirement analysis in the object-oriented process. It starts with measuring the functionality of the system based on the use case model in a count called *Unadjusted Use Case Point* (UUCP). The same technical factors are used as of Function Points. A new factor called *Environmental Factor* is proposed by the author[4]. The author defined the Use Case Points (UCP) as the product of these three factors (i.e. UUCP, Technical Factors and Environmental Factors). The UCPs shows an estimation of the size of the system which can further mapped to man hours in order to calculate the effort required to develop the system[4].To estimate the size of software using this method, several rules should be applied. These rules include :

1. Identify the complexity of each use case.
2. Assign a weight factor for each level of Complexity for use cases .

TABLE I. USE CASE CLASSIFICATION [4]

Use Case Type	Description	Weight
Simple	Less than 3 transactions	5
Average	4 to 7 transactions	10
Complex	More than 7 transactions	15

3. Identify the complexity of each actor .
4. Assign a weight factor for each level of complexity for actors .

TABLE II. ACTOR CLASSIFICATION [4]

Actor Type	Description	Weight
Simple	System Application Programming Interface (API)	1
Average	Interactive or Protocol-Driven Interface	2
Complex	Graphical Interface (GUI)	3

5. Calculate the total use case weight factor (UseCase_WeightFactor):
∑ Simple use cases * WF + ∑ Average use cases * WF + ∑ Complex use cases * WF
6. Calculate the total actor weight factor (Actor_WeightFactor) :
∑ Simple actor * WF + ∑ Average actor * WF + ∑ Complex actor * WF
7. Calculate the Unadjusted Use Case Points (UUCP):
UUCP = UseCase_WeightFactor + Actor_WeightFactor
8. Calculate the Adjustment Use Case points (AUCP):
AUCP = UUCP * TF * EF

where

TF is the Technical Factor.
TF = 0.6 (0.01 * TWF)
EF is the Environment Factor.
EF = 1.4 + (-0.03 * EWF)

TABLE III. TECHNICAL COMPLEXITY FACTORS [4]

F_i	Factors Contributing to Complexity	W_i
F_1	Distributed systems.	2
F_2	Application performance objectives, in either response or throughput.	1
F_3	End user efficiency (on-line).	1
F_4	Complex internal processing.	1
F_5	Reusability, the code must be able to reuse in other applications.	1
F_6	Installation ease.	0.5
F_7	Operational ease, usability.	0.5
F_8	Portability.	2
F_9	Changeability.	1
F_{10}	Concurrency.	1
F_{11}	Special security features.	1
F_{12}	Provide direct access for third parties	1
F_{13}	Special user training facilities	1

TABLE IV. ENVIRONMENTAL COMPLEXITY FACTORS [4]

F_i	Factors Contributing to Efficiency	W_i
F_1	Familiar with RUP	1.5
F_2	Part time workers	-1
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	Stable requirements	2

IV. PROPOSED WORK

In our research we have introduced six new external factors, which we believe are strongly related to the development environment and can be helpful for getting more accurate and precise result. We have proposed the equation for calculating the ECF by adding our proposed factors with the existing factors proposed by Karner. So the number of factors became 14 and the equation is ;

$$\mathbf{ECF=1.4 + (-0.03)} \sum_{i=1}^{14} Fi * Wi$$

TABLE V. NEW ENVIRONMENTAL FACTORS

F_i	Suggested Environmental factors	Weight (W_i)
F_9	Client Type	1.2
F_{10}	New Technology	1
F_{11}	Team Co-ordination	1.5
F_{12}	Growth Rate of Organization	0.5
F_{13}	Team Composition	1.5
F_{14}	Organization Library Availability	-0.5

The steps for calculating Use case point (UCP) remains same as of Karner.

V. DISCUSSION

This research is exploratory in nature. We have assigned weights to each factor in a similar way to existing factors based on historical data. Traditionally, each factor is rated from zero to five, a rating of zero means the factor is irrelevant for this project, three means it is average and five means it is essential. Another point to be addressed is how to weight these new factors. It should be apparent that the weighted values used to modify the influence of each factor, so deriving the weight value must go through a well-defined process. A slight variation in the value of weight will dramatically increase use case points and therefore the total project effort. Even small adjustment, for instance half point, will vary the final result by 40%[6]. However, this means that inappropriate environmental weight factors will lead to catastrophic results.

VI. CONCLUSION

This paper exemplified and highlighted the importance of UCP in software effort estimation. The authors addressed new modified sets of environment factors, which they believe will be able to give more accurate and precise result for the use case point method. The evaluation of this model is left for future work because this research is still at the data capturing stage.

REFERENCES

[1] B. Boehm, "Software Engineering Economics",Prentice Hall, 1981.
[2] B. Boehm et al.,"Software Cost Estimation with COCOMO II " , Prentice Hall Englewood Cliffs, NJ, 2000.
[3] N. J. Nunes et al., "iUCP: Estimating Interactive- Software Project Size with Enhanced Use-Case Points", IEEE Software, 2011.
[4] G. Karner, "Metrics for Objectory", Diploma thesis,University of Linköping, Sweden. No. LiTHIDA- Ex- 9344:21.December 1993.
[5] Q. Yu et al., " Application of Estimating Based on Use Cases in Software Industry ", 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2011.
[6] K. Ribu, " Estimating Object-Oriented Software Projects with Use Cases", in Department of Informatics, University of Oslo, 2001.