Journal of Engineering and Applied Sciences 2 (10): 1565-1568, 2007
ISSN: 1816-949X

Potentiality of Use Case Point's Method for Estimating the Size and Effort of Software Development Projects

[1]D. Lakshmanan, [2]V. Gunaraj, [3]M. Karnan, [3]R. Sivakumar, [4]G. Deepakkannan
[1]Kumaraguru College of Technology, Coimbatore, Tamil Nadu, India
[2] RVS Engineering College, Coimbatore, Tamil Nadu, India
[3]Tamilnadu College of Engineering, Coimbatore, Tamil Nadu, India
[4]Amrita University, Coimbatore, Tamil Nadu, India

Abstract: The usage of use case models has been progressively used to capture and depict the functional requisites of a software system. There are different approaches and methods to successfully count on effort using use cases. A few researchers tested use case points methods and revealed that it has the potential to be a reliable source of estimation alike the function point method. Estimation of size of software development projects has a strong effect. Use case modeling is increasingly being utilized not only to describe the software and system requirements and a basis of design, development, testing, deployment, configuration management and maintenance but also to have an estimation method that makes the use of them. Perhaps this study aims at the potentiality of use case point's method for estimating the size and effort of software development projects, including the major limitations and offers some possible rectification.

Key words: Functional point analysis, use case point, estimation, complexity factor

INTRODUCTION

A recognized and widespread technique to capture the business processes and requirements of a software application project is use case modeling. It provides the operational scope of the projects and hence the analyses of their contents provide good perception on the effort and size needed to design and implement the project.

Designing and implementation efforts are less for small projects when compared to large projects. The factors affecting project completion time are:

- Number of use case completion steps.
- Number and complexity of actors.
- Technical requisites-concurrency, security and performance.
- Environmental factors-development teams' experience and knowledge.

Estimation using these factors in a projects life cycle produces an estimate within 20% of the actual completion time which will be helpful for project scheduling cost and resource allocation.

USE CASE POINTS (UCP) METHODS

The use case point method has the power to estimate the man-hours a software project requires from its use cases. Based on work by Karner (1993). The UCP method abstracts an equation analyzing the use case actors, scenarios and various technical and environmental factors. UCP is inspired from function point analysis. This can be recognized by readers familiar with Allen Albrecht's Functional Point Analysis (Albrecht, 1979). Variables of UCP equations are:

- Unadjusted Use Case Points (UUCP)
- The Technical Complexity Factor (TCP)
- The Environmental Complexity Factor(ECF)

Each variables is defined and computed separately using the three factors namely,

- Weighted values
- Constraining constants
- Subjective values.

The first two of the above were initially based on Albrecht but later on modified by people at objective systems, LLC, based on their experience with Objectory. Objectory is a methodology created by Ivar Jacobson for developing object oriented applications (Jacobson, 1998).

The subjective values are determined based on the perception of the projects technical complexity and efficiency. In addition to that, the equation is used to estimate the number of man-hours needed to complete a project when productivity is included.

Corresponding Author: D. Lakshmanan, Kumaraguru College of Technology, Coimbatore, Tamil Nadu, India

The complete equation is

$$UCP=UUCP*TCF*ECF*PF$$

Steps necessary to generate the estimate are:

- Determine and compute the UUCPs.
- Determine and compute the TCFs.
- Determine and compute the ECFs.
- Determine the PF.
- Compute the estimated number of hours.

DETERMINATION OF THE VARIABLES

UUCPs:

- The Unadjusted Use Case Weight (UUCW)
- The Unadjusted Actor Weight (UAW)
- UUCW

UUCW is computed based on the total number of activities contained in all the use case scenarios. The number of steps in a scenario affects the estimate. The use case categories are simple, average and complex.

- The larger the number of steps the use case scenario will bias the UUUCW towards complexity and thereby increasing the UCPs.
- The smaller the number of steps the use case scenario will bias the UUCW towards simplicity and thereby decreasing the UCPs.

UUCW is computed by multiplying each total by its specified weighing factor and then adding the products (Table 1).

UAW: Similar to UUCW, the Actor Types are classified as simple, average or complex. The UAW is computed by totaling the number of actors in each category, multiplying each total by its specified weighing factor and then adding the products (Table 2).

The UUCP is computed by adding the UUCW and the UAW. The UUCP is unadjusted because it does not account for the TCFs and ECFs.

TCFs: The impact on productivity that various technical issues have on a project is estimated by 13 standard technical factors. According to relative impact, each factor is weighed (Table 3).

Table 1: UUCW factors

Category	Description	Weighting factor
Simple	*Simple user interface *Single database entity *Three or less steps *Implementation-less than five classes.	5
Average	*More interface design *Two or database entity *Between four and seven steps *Implementation-between five and ten classes	10
Complex	* Complex user interface *Three or database entities *More than seven steps *Implementation-more than ten classes	15

Table 2: UAW factors

Actor type	Description	Weight
Simple	Represents another system with a defined application programming interface.	1
Average	Represents another system interacting through a protocol like Transmission control Protocol/Internet protocol.	2
Complex	The actor is a person interfacing via a graphical user interface.	3

Table 3: TCFs factors

Technical factor	Description	Weight
T1	Distributed system	2
T2	Performance	1
T3	End user eficiency	1
T4	Complex Internal Processing	1
T5	Reusability	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portability	2
T9	Easy to change	1
T10	Concurrency	1
T11	Special security features	1
T12	Provides direct access for Third Parties	1
T13	Special user training Facilities are required	1

PRECEIVED COMPLEXITY

The development team evaluates the technical factor for each project and a perceived complexity value between 0 and 5 are assigned. The perceived complexity factor is subjectively determined by the development team's perception of the projects complexity-concurrent applications (Table 4).

Calculated factor = Each factors weight* perceived complexity factor

Technical total factor = ΣCalculated factors

Table 4: The perceived complexity factors

Perceived complexity	Technical factor
0	Irrelevant for project
3	Average
5	Strong Influence

Two constants are computed with the Technical Total Factor to produce the TCF. The constants constraint the effect the TCF has on the UCP equation from a range of 0.60 (perceived complexities all zero) to a maximum of 1.30 (perceived complexities all five)

TCF value < 1 – Reduce UCP, since any +ve value * +ve fraction
It reduces the magnitude.
100*0.60 = 60 Reduction by 40%
TCF value>1 - Increase UCP, since any +ve value * +ve mixed number
It increases the magnitude.
100*1.30 = 130 increases by 30%

Since the constants constrain the TCF from a range of 0.60 to 1.30, the TCF can impact the UCP equation from 40% to a maximum of 30%.

Complete formula to compute TCF,

TCF = C1 + C2∑Wi * Fi
Constant1 (C1) = 0.6
Constant2 (C2) = 0.01
W = Weight
F = Perceived Complexity Factor

ECFs: The ECF provides a concession for the development team's experience. More experienced team will have a greater impact on the UCP computation than less experienced teams (Table 5).

The development team determines each factors perceived impact based on its perception, the factor has on the projects success (Table 6).
For instance, team members with little or no motivation will have strong negative impact (1).
Team members with strong object-oriented experience will have strong positive impact (5).

Calculated factor = Each factors weight* perceived impact
Environmental total factor = ΣCalculated factors

Two constants are computed with Environmental Total Factor to produce the Final ECF. The constants, based on interviews with experienced Objectory users at

Table 5: ECFs factors

Environmental factor	Description	Weight
E1	Familiarity with UML	1.5
E2	Part-time workers	-1
E3	Analyst capability	0.5
E4	Application experience	0.5
E5	Object-oriented experience	1
E6	Motivation	1
E7	Difficult programming language	-1
E8	Stable requirements	2

Table 6: Perceived impact factors

Perception impact	Perception
0	No negative
1	Strong negative
3	Average
5	Strong positive

Objective Systems (Karner, 1993), constrain the impact the ECF has on the UCP equation from 0.425 to 1.4. Therefore, the ECF can reduce the UCP by 57.5 percent and increase the UCP y 40%. The ECF has a greater potential impact on the UCP count than the TCF.
The formal equation is

$$TCF = C1 + C2 \Sigma Wi * Fi$$

Constant1 (C1) = 1.4
Constant2 (C2) = -0.01
W = Weight
F = Perceived impact

Calculating the UCP: The UCP equation is

$$UCP = UUCP * TCF * ECF$$

Estimated hours: Additional factor is required to estimate the number of hours to complete the project is PF.

Total Estimate = UCP * PF
PF = Development man-hours needed / use case point

Initial PF can be estimated from the past projects. For instance, if a past project with a UCP of 120 took 2200 hours to complete, PF is 18 man-hours per use case point. If no historical data has been collected, consider two possibilities:

- Establish a baseline by computing the UCP for previously completed projects.
- Use a value between 15 and 30 depending on the development teams total experience and past accomplishments. If it is a brand new team, use a value of 20 for the first project.

After the completion of the project, divide the number of actual hours it took to complete the project by the UCP count. The product becomes new PF.

CONCLUSION

Previously, the project estimate was a supporting tool for the managers, developers and testing professionals plan in the process of seeking resources with regard to the project requirement. The case studies enunciated in this study overtly edify that the UCP strategies may produce an early estimate ranging from 1 to 20% of the actual effort and perhaps closes to the actual effort when compared to the experts and other estimation strategies. As a matter of fact, in many conventional estimation methods, influential technical and environmental factors are not given due consideration. The UCP method quantifies these subjective factors into equation variables that may be tweaked over time in the production of several precise estimates. Ultimately, the UCP method possessed a versatile extensibility to a variety of developments and testing projects. In short, it is comprehensive for learning and intelligible for applying.

REFERENCES

Albrecht, A.J., 1997. Measuring Application Development Productivity. Proc. Of IBM Applications Development Symposium, Moneterey, CA, 14: 17 9-183.

Anda, Bente, 2005. Improving Estimation Practices By Applying Use case models. <www.Cognizant.com/cog/community/presentations/Test_Effort_Estimation.pdf.>

Anda, Bente *et al*., 2005. Effort Estimation of Use cases for Incremental Large-scale Software Development. 27th International Conference on Software Engineering, St. Louis, pp: 303-311.

Carroll, Edward R., 2005. Estimation Software Based on Use Case Points.Object-oriented, Programming, Systems, Languages and Applications (OOPSLA) Conference, San Diego, CA.

Jacobson, I., G. Booch and J. Rumbaugh, 1998. The Objectoru Development Process. Addison-Wesley.

Karner, Gustav, 1993. Resource Estimation for Objectory Projects. Objective Systems SFAB.

Nageswaran, Suresh, 2001. Test Effort Estimation Using Use Case Points. June <www.Cognizant.com/cog/community/presentations/Test_Effort_Estimation.pdf.>