Use Case Estimation Framework

Gautam Banerjee

Birlasoft Limited
NOIDA, INDIA

Table of Contents

Abstract

Estimates of cost and schedule in software projects are based on a prediction of the size of the future system. Unfortunately, the software profession is notoriously inaccurate when estimating cost and schedule. Preliminary estimates of effort always include many elements of insecurity. Reliable early estimates are difficult to obtain because of the lack of detailed information about the future system at an early stage. However, early estimates are required when bidding for a contract or determining whether a project is feasible in the terms of a cost-benefit analysis. Use case models are used in object-oriented analysis for capturing and describing the functional requirements of a system. This paper reports the mode of usage and benefits of Use Case Estimation Model in the current competitive software market. The paper supports existing claims that use cases can be used successfully in estimating software development effort. However the paper explicitly takes note of the fact that the design of the use case models has a strong impact on the estimates.

Introduction

Birlasoft, founded in 1992, is a major IT player in the globe with development centers at NOIDA, Chennai and Melbourne. The company employs over 1400 professionals worldwide. It is a Six Sigma compliant company assessed at CMMi level 5 in the first attempt.

We at Birlasoft, observed the following challenges in recent years

- Proposal level RFPs/Tenders are using more & more UML artifacts
- Effectiveness of Iterative delivery model involving use cases
- Increasing importance of Micro level project planning involving Use cases
    - Difficulty in monitoring Effort variance for unit level/use case level
    - Need of Defect Prediction at use case level for planning rework effort
- Technical complexity and slow learning curve for prevalent estimation models

In the endeavor to continually improve the software processes and sustain process capability, the need for a new estimation framework for software development effort addressing the above noted points was something inevitable. The Use Case Estimation Model approach was selected as the one, which could provide the framework for such a measure.

Background

Cost models like COCOMO and sizing methods like Function Point Analysis (FPA) are well known and in widespread use in software engineering. But these approaches have some serious limitations. Counting function points requires experts.

In 1993 the 'Use Case Points' method for sizing and estimating projects developed with the object-oriented method was developed by Gustav Karner of Objectory (now Rational Software). The method is an extension of Function Point Analysis and Mk II Function

Point Analysis (an adaption of FPA mainly used in the UK), and is based on the same philosophy as these methods.
A few cost estimation tools apply use case point count as an estimation of size, adapting Karner's method. Karner's work on Use Case Point metrics was written as a diploma thesis at the University of Linköping. It was based on just a few small projects, so more research is needed to establish the general usefulness of the method. The work is now copyright of Rational Software, and is hard to obtain.

The Framework

An early estimate of effort based on use cases can be made when there is some understanding of the problem domain, system size and architecture at the stage at which the estimate is made. The use case points method is a software sizing and estimation method based on use case counts called use case points.

Classifying Actors

Use case points can be counted from the use case analysis of the system. The first step is to classify the actors as simple, average or complex. A simple actor represents another system with a defined Application Programming Interface, API, an average actor is another system interacting through a protocol such as TCP/IP, and a complex actor may be a person interacting through a GUI or a Web page. A weighting factor is assigned to each actor type.

Actor Type	**Weighting Factor**
Simple	1
Average	2
Complex	3

Unadjusted Actor Weights

The total unadjusted actor weights (UAW) is calculated by counting how many actors there are of each kind (by degree of complexity), multiplying each total by its weighting factor, and adding up the products.

Classifying Use Cases

Transaction Based

Each use case is then defined as simple, average or complex, depending on number of transactions in the use case description, including secondary scenarios. A transaction is a set of activities, which is either performed entirely, or not at all. Counting number of transactions can be done by counting the use case steps. Use case complexity is then defined and weighted in the following manner:

Use Case Type	No of Transactions	Weighting Factor
Simple	<=3	10
Average	4 to 7	15
Complex	>=7	20

Analysis Class Based

Another mechanism for measuring use case complexity is counting analysis classes, which can be used in place of transactions once it has been determined which classes implement a specific use case. A simple use case is implemented by 5 or fewer classes, an average use case by 5 to 10 classes, and a complex use case by more than ten classes. The weights are as before.

Flow Based

At Birlasoft, both the above approaches (Transaction and Analysis Class) were applied for measuring the effectivity of classifying Use Cases. However data showed that these two approaches alone at times are not able to give the right measure of a Use Case complexity. So a flow based approach was adopted.

Each use case is fragmented into number of flows and are defined either Simple or Complex.

Complex flows are such flows which either involves ACID transactions or any other Data repository or any Control Intelligence or Business Logic Computation.

Weights are assigned for each type of flows and then each type of flow is multiplied by the weighting factor, and the products are added up to get Unadjusted Use Case Flow Points (UUCFP) for each Use Case as depicted in the table below.

Flow Type	Weighting Factor
Simple	2
Complex	3

The method also employs a technical factors multiplier for each Use Case as below

Use Case Technical Factor			
Factor	**Description**		**Rating(0 - 5)**
T1	Concurrent		
T2	Security features		
T3	Complex processing		

Ratings (0 means no influence, 3 is average, and 5 means strong influence) are given for each factor.
The Use Case Technical Factor (UCTF) is calculated by applying the following formulae:

UCTF =0.6+(0.01*Rating for each Factor)

The UCTF is multiplied by the Unadjusted Use Case Flow Points (UUCFP) to produce the adjusted Use Case Flow Points, simply called as UCFP.

UCFP= UUCFP x UCTF

So each Use Case will correspond to a specific value of UCFP. Depending on the number of UCFP, Use Cases are defined as Simple, Medium and Complex in nature.

Usecase	**Definition**	**Weighting Factor**
Simple	If Total UCFP is 0-5	10
Average	If Total UCFP is 6-10	15
Complex	If Total UCFP is 10+	20

Unadjusted Use Case Weights

Each type of use case is then multiplied by the weighting factor, and the products are added up to get the unadjusted use case weights (UUCW).

Unadjusted Use Case Points

The UAW is added to the UUCW to get the unadjusted use case points
UAW+UUCW=UUCP

Technical and Environmental Factors

The method also employs a technical factors multiplier corresponding to the Technical Complexity Adjustment factor of the FPA method, and an environmental factors multiplier in order to quantify non-functional requirements such as ease of use and programmer motivation.
Various factors influencing productivity are associated with weights, and values are assigned to each factor, depending on the degree of influence.

0 means no influence, 3 is average, and 5 means strong influence throughout.

See Tables below

Project Technical Factors			
Factor	**Description**	**Weight**	**Rating (0 - 5)**
T1	Distributed System	2	
T2	Response adjectives	1	
T3	End-user efficiency	1	
T4	Reusable code	1	
T5	Easy to install	0.5	
T6	Easy to use	0.5	
T7	Portable	2	
T8	Easy to change	1	
T9	Access for third parties	1	

Project Environment Factors			
Factor	**Description**	**Weight**	**Rating (0 - 5)**
F1	Familiarity With RUP	1.5	
F2	Application Experience	0.5	
F3	Object Oriented Experience	1	
F4	Lead Analyst Capability	0.5	
F5	Motivation	1	
F6	Stable Requirements	2	
F7	Part-time Workers	-1	
F8	Difficult Programming Language	2	

The adjustment factors are multiplied by the unadjusted use case points to produce the adjusted use case points, yielding an estimate of the size of the software.

Technical Complexity Factor

The *Technical Complexity Factor (TCF)* is calculated by multiplying the value of each factor (T1- T9) by its weight and then adding all these numbers to get the sum called the *TFactor.* The following formula is applied:

TCF=0.6+(0.01*TFactor)

Environmental Factor

The *Environmental Factor (EF)* is calculated by multiplying the value of each factor (F1-F8) by its weight and adding the products to get the sum called the *EFactor.* The following formula is applied:

EF= 1.4+(-0.03*EFactor)

Adjusted Use Case Points

The *adjusted use case points (UPC)* are calculated as follows:
UPC= UUCP*TCF*EF

Producing Estimates

Initial pilot run of the Use Case Estimation Framework at Birlasoft has shown that effort can range from 15 to 30 hours per use case point, therefore converting use case points directly to hours at times may be an uncertain measure. The reason behind this is due to the fact that Use Case Specification varies from project to project. UML does not go into details about how the use case model should be structured nor how each use case should be documented. Therefore, use case models can be structured and documented in several alternative ways. Results at Birlasoft show that the structure of the use case model has a strong impact on the precision of the estimates. In particular, we experienced that the following aspects of the structure had an impact:

- The use of generalization between actors
- The use of included and extending use cases
- The level of detail in the use case descriptions

Thus an important prerequisite for applying a use case based estimation method is that the use cases of the system under construction have been identified at a suitable level of detail. At Birlasoft, a standardized Use case Specification document was implemented with proper Guideline document and relevant trainings were conducted so that a uniform Productivity with an accepted tolerance limit can be applied to Use Case Points. Result shows that current productivity at Birlasoft is 12 Person Hours per Use Case Point.

Analyzing Results

The results of the implementation of the Framework is monitored and compared with implementations of previous frameworks within Birlasoft.

Before Implementation of UCP Model

Effort Variance with FP Estimation Model

Sl No.	Project	Effort Variance (FP)	Target goal
1	A	-27%	5%
2	B	5%	5%
3	C	10%	5%
4	D	9%	5%
5	E	10%	5%
6	F	7%	5%
7	G	99%	5%

Process Sigma Calculation

No. Of units processed	N	7
No of defects opportunity per unit	O	1
Total no. Of defects	D	6
DPO	D/(N*O)	0.857143
DPMO		857142.9
Process Sigma Short Term	**0.4**	

After Implementation of UCP Model

Effort Variance with UCP Estimation Model

Sl No.	Project	Effort Variance (UCP)	Target goal
1	H	5.5%	5%
2	I	2%	5%
3	J	-5%	5%
4	K	16%	5%

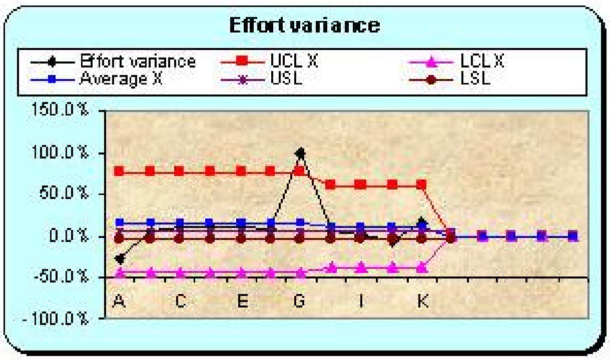
Process Sigma Calculation

No. Of units processed	N	4
No of defects opportunity per unit	O	1
Total no. Of defects	D	1
DPO	D/(N*O)	0.25
DPMO		250000
Process Sigma Short Term	**2.1**	

Effort Variance Control Chart for FP and UCP

Data collected for Projects A to G used FP estimation framework. Projects H to K used UCP Estimation framework. Effort variance for all the projects was captured and control chart (Figure 1) was drawn to analyze the trend. The trend shows that mean and UCL have improved which gives an affirmation for increased capability level.

Project	Mode of Estimation	Effort variance	UCL X	Average X	LCL X	USL	LSL
A	FP	-27.03%	75.27%	15.97%	-43.33%	5.00%	-5.00%
B	FP	5.00%	75.27%	15.97%	-43.33%	5.00%	-5.00%
C	FP	9.54%	75.27%	15.97%	-43.33%	5.00%	-5.00%
D	FP	8.59%	75.27%	15.97%	-43.33%	5.00%	-5.00%
E	FP	10.02%	75.27%	15.97%	-43.33%	5.00%	-5.00%
F	FP	6.95%	75.27%	15.97%	-43.33%	5.00%	-5.00%
G	FP	98.71%	75.27%	15.97%	-43.33%	5.00%	-5.00%
H	UCP	5.50%	60.87%	11.77%	-37.34%	5.00%	-5.00%
I	UCP	1.57%	60.87%	11.77%	-37.34%	5.00%	-5.00%
J	UCP	-5.44%	60.87%	11.77%	-37.34%	5.00%	-5.00%
K	UCP	16.01%	60.87%	11.77%	-37.34%	5.00%	-5.00%



Conclusion

The implementation of Use Case Estimation framework within Birlasoft shows that it is as effective as conventional estimation frameworks if not better. However this does not undermine the potential of other existing and popular frameworks. However considering the current market competition in technical trends and upgrades (emergence of design language like UML and popularity of RUP and its variants), Use Case Estimation Model is bound to make its presence in this market with further calibration and refinement. The common concerns and roadblocks for any change management is also valid for this estimation framework due to its budding nature which can be achieved by imparting proper training, mentoring and guidance.

Acknowledgements

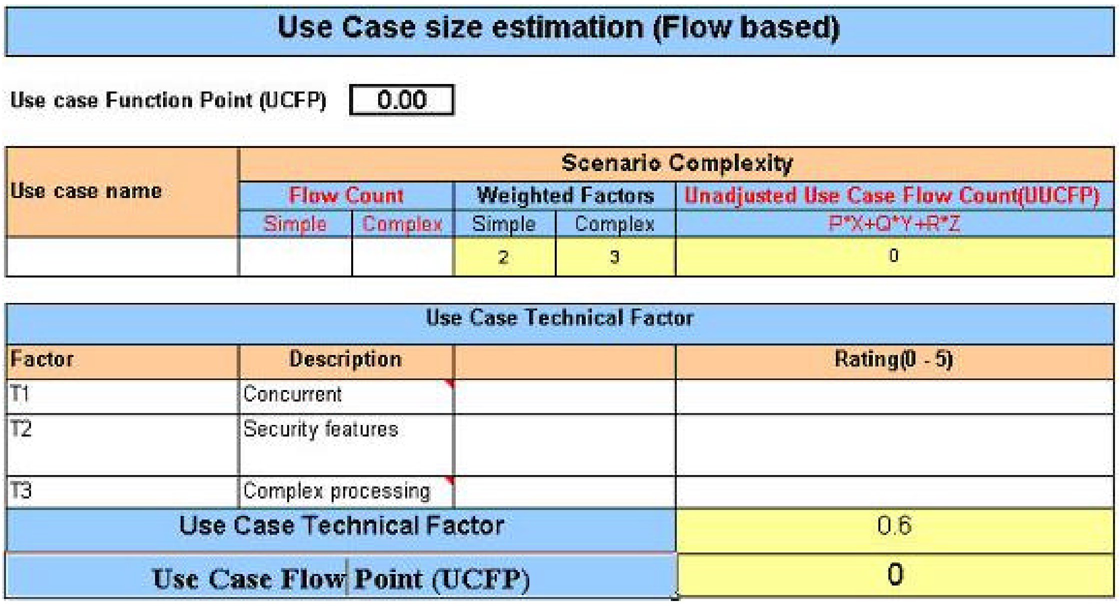
I would like to thank people who have contributed and extended their support to make this framework successful at Birlasoft. Firstly I would like to thank Mrs. Sapna Sharma who took great initiative in implementing the framework inside the organization. I would also like to thank Mr. Rajib Das for his constant support in providing me the inputs and feedback.

References

a) Boehm, B.W. *Software Engineering Economics*. Prentice-Hall. 1981.
b) Cockburn, A. Writing Effective Use Cases. Addison-Wesley. 2000.
c) Estimating use-case driven iterative development for "fixed-cost" projects by Amit Bhagwat (http://www.therationaledge.com/content/oct_03/f_estimate_b.jsp)
d) Karner, G. Metrics for Objectory. Diploma thesis, University of Linköping, Sweden. No. LiTHIDA- Ex-9344:21. December 1993.
e) OMG Unified Modeling Language Specification, Version 1.3. June 1999. (http://www.rational.com/media/uml/post.pdf).
f) Schneider, G. and Winters, J. *Applying Use Cases – A Practical Guide*. Addison-Wesley. 1998.
g) Use Cases -- Yesterday, Today, and Tomorrow by Ivar Jacobson (http://www.therationaledge.com/content/mar_03/f_useCases_ij.jsp)
h) Smith, J. The Estimation of Effort Based on Use Cases. Rational Software, White paper. 1999.
i) The Object Factory. Estimating Software Projects using ObjectMetrix, White paper. April 2000.

Appendix

Screen Shots of Biralsoft's Use Case Estimation Framework Template



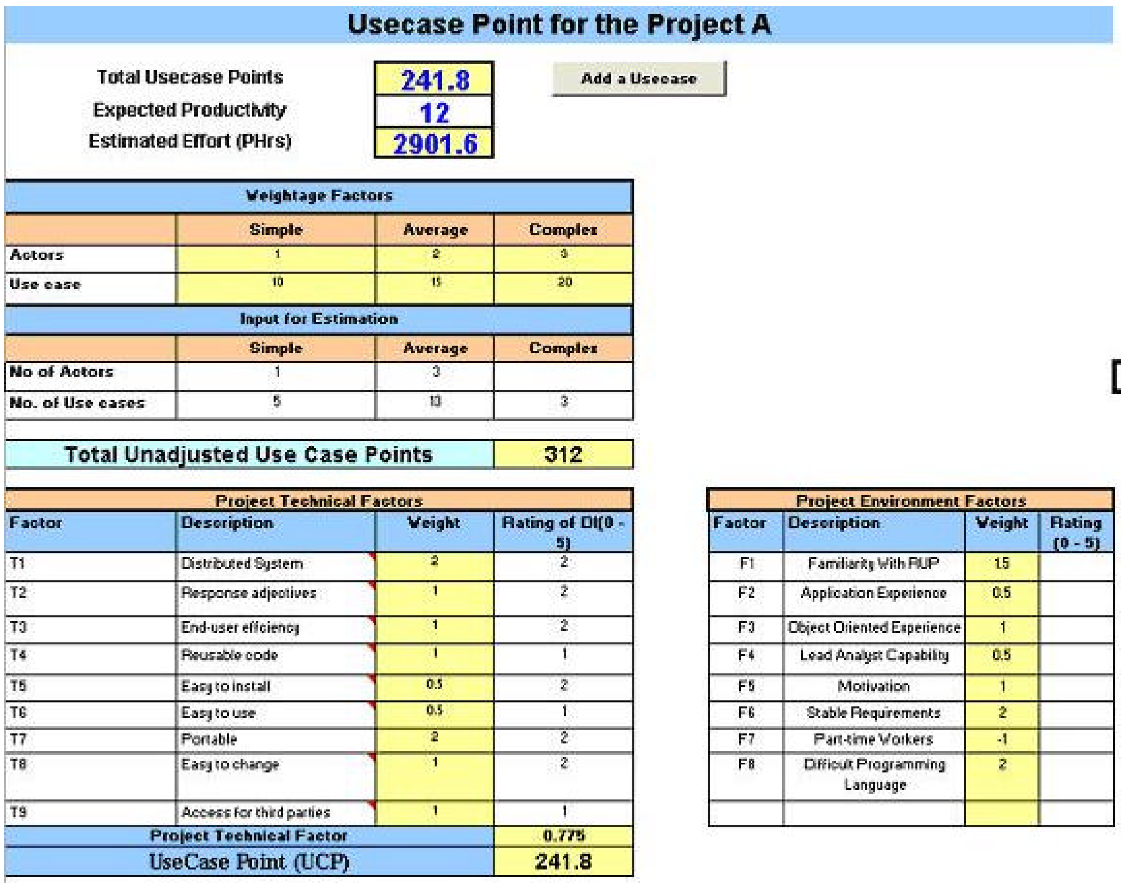
Use Case size estimation (Flow based)

Use case Function Point (UCFP) 0.00

Use case name	Scenario Complexity				
	Flow Count		Weighted Factors		Unadjusted Use Case Flow Count(UUCFP)
	Simple	Complex	Simple	Complex	P*X+Q*Y+R*Z
			2	3	0

Use Case Technical Factor

Factor	Description		Rating(0 - 5)
T1	Concurrent		
T2	Security features		
T3	Complex processing		
Use Case Technical Factor			0.6
Use Case Flow Point (UCFP)			0



Usecase Point for the Project A

Total Usecase Points	241.8
Expected Productivity	12
Estimated Effort (PHrs)	2901.6

Weightage Factors			
	Simple	Average	Complex
Actors	1	2	3
Use case	10	15	20

Input for Estimation			
	Simple	Average	Complex
No of Actors	1	3	
No. of Use cases	5	13	3

Total Unadjusted Use Case Points	312

Project Technical Factors

Factor	Description	Weight	Rating of DI(0 - 5)
T1	Distributed System	2	2
T2	Response adjectives	1	2
T3	End-user efficiency	1	2
T4	Reusable code	1	1
T5	Easy to install	0.5	2
T6	Easy to use	0.5	1
T7	Portable	2	2
T8	Easy to change	1	2
T9	Access for third parties	1	1
Project Technical Factor			0.775
UseCase Point (UCP)			241.8

Project Environment Factors

Factor	Description	Weight	Rating (0 - 5)
F1	Familiarity With RUP	1.5	
F2	Application Experience	0.5	
F3	Object Oriented Experience	1	
F4	Lead Analyst Capability	0.5	
F5	Motivation	1	
F6	Stable Requirements	2	
F7	Part-time Workers	-1	
F8	Difficult Programming Language	2	