ESTIMATING SOFTWARE FOR SPIRAL MODEL BASED ON USE CASE POINTS

Nirupma Pathak*

ABSTRACT

Several estimating models have been developed over the years. Those preceding Use Case Point (UCP) and forming the basis for the UCP model include Function Point Analysis and the Constructive Cost Model. Function Point Analysis (FPA) was a valuable technique developed by A. J. Albrecht, in 1977. FPA assigns a point to each function in an application. Various modifiers then act upon the function points in order to adjust for the product environmental. Modifiers typically included applying weighted percentages or multipliers that would simply increase or decrease the point values. Environment factors included modifiers for complexity of technical issues, developer skill level, and risk. One problem organizations attempting to use this method would run into was consistent definition of a function and consistent definition of environmental factors across multiple projects and multiple development languages.

*M.Tech(CS&Egg.) ,B.Tech(CS&Egg.),HOD,Department of Computer Science, LBSGCM , Lucknow

1. INTRODUCTION

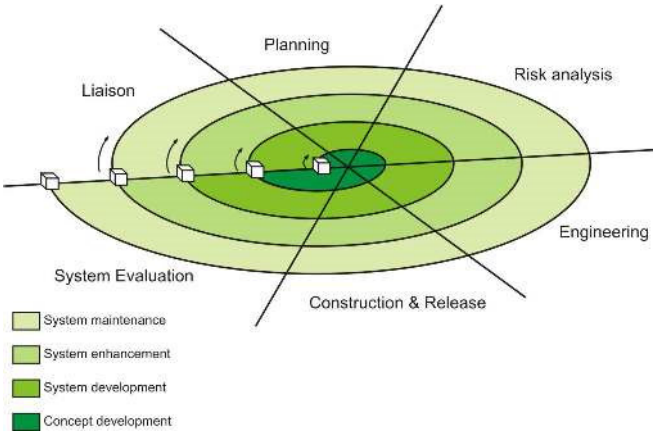
The spiral model was defined by Barry Boehm in his 1988 article A Spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration matters. As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with the client (who may be internal) reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

Spiral model is an evolutionary version of incremental prototyping. Each iteration of the prototype represented as a cycle in the spiral. The Spiral software development model is a risk-oriented.

Spiral software development model may be applicable to projects where:

The projects requirements are very difficult

Where new technologies are used



The aim of **customer communication** is to establish effective communication between developer and customer.

The **planning** objectives are to define resources, project alternatives, time lines and other project related information.

The purpose of the **risk analysis** phase is to assess both technical and management risks.

The **engineering** task is to build one or more representations of the application.

The **construction and release** task **–** to construct, test, install and provide user support (e.g.,

documentation and training).

The **customer evaluation** task - to obtain customer feedback based on the evaluation of the software representation created during the engineering stage and implemented during the install stage.

1.1 Estimating Use Case Points-

This section will describe the Use Case Point estimating model. The model is described here for clarity in illuminating our continuous process improvement progress.

1.1.1Actors-

The process starts by considering the Actors. For each actor, determine whether the actor is a simple, average or complex actor. A simple actor represents another system with a defined Application Programming Interface (API). An average actor is either another system that interacts through a protocol such asTCP/IP, or it is a person interacting through a text-based interface. A complex actor is a person interacting through a graphical user interface (GUI).Count the number of simple, average and complex actors and lists the quantity of each in the table. Multiply the quantity of each type of actor times the weighting factor and sum the total.

Actor Type	Description	Qnty	Weight Factor	Sub total
Simple	Defined API	3	1	3
Average	Interactive or Protocol driven interface	2	2	4
Complex	Graphical User Interface	1	3	3
Total Actor Points				10

Table – 1 Weighting Actors for Complexity

It should be apparent that the weighted values used to modify the quantity of actors (in this section, as in subsequent sections) are only valid if the impact they have on the result is backed up by feedback from historical data. In the case of Actors, use the feedback from historical data to adjust the definition of simple, average or complex.

1.2 Weighting Use Cases-

For each use case, determine whether it is simple, average or complex based on the number of transactions in a use case, including secondary scenarios. For this purpose, a transaction is defined as an atomic set of activities which is either performed entirely or not at all. A simple use case has 3 or fewer transactions, an average use case has 4 to 7 transactions, and a complex use case has more than 7 transactions.

If analysis classes have been defined for the system, and it has also been identified as to which ones are used to implement a particular use case, use this information in place of transactions to determine the use case complexity. Note: Used use cases or extended existing use cases do not need to be considered. Count the number of simple, average and complex use cases and list the quantity of each in the table. Multiply the quantities of each type of use case times the weighting factor and sum the total.

Use Case Type	Description	Qnty	Weight Factor	Sub total
Simple	3 or fewer transactions	3	5	15
Average	4 to 7 transactions	2	10	20
Complex	Greater than 7 transactions	1	15	15
Total Use Cases				50

Table -2 Weighting Use Cases for Complexity

Add the total for Actors to the total for use cases to determine the Unadjusted Use Case Points (UUCP).

• Weighted Actors + Weighted Use Cases = UUCP

10 + 50 = 60

The UUCP will be further modified to reflect the complexity of the project and experience level of the development team.

1.3 Weighting Technical Factors-

Weighting technical factors is an exercise to calculate a Use CasePoint modifier which will modify the UUCP by the weight of the technical factors. Start by calculating the technical complexity of the project. This is called the technical complexity factor (TCF).To calculate the TCF go through the following table and rate each factor from 0 to 5. A rating of 0 means the factor is irrelevant for this project, 5 means it is essential. For each factor multiply its rating by its weight from the table.

Technical Factor	Factor Description	Weight Factor	Project Rating	Sub total
T1	Must have a distributed solution	2	5	10
T2	Must respond to specific performance objectives	1	3	3
T3	Must meet end-user efficiency desires	1	5	5
T4	Complex internal processing	1	5	5
T5	Code must be reusable	1	3	3
T6	Must be easy to install	.5	3	1.5
T7	Must be easy to use	.5	3	1.5
T8	Must be portable	2	0	0
T9	Must be easy to change	1	5	5
T10	Must allow concurrent users	1	0	0
T11	Includes special security features	1	5	5
T12	Must provide direct access for 3rd parties	1	0	0
T13	Requires special user training facilities	1	3	3
Total TFactor				42

Table -3 Weighting Technical Factors Technical

Add the resultant values together to get the total T factor.

• (Weighting Factor) * Σ(Tlevel) = TFactor

The TFactor does not directly modify the UUCP. TocalculateTechnicalComplexity Factor (TCF), multiply TFactor by 0.01and then add 0.6.

(0.01 * Tfactor) + 0.6 = TCF

(0.01 * 42) + 0.6 = 1.02 TCF

Calculate the size of the software (use case) project by multiplying

UUCP times TCF.

UUCP * TCF = SzUC

60 * 1.02 = 61.2

Note on Reusable Components: Reusable software components should not be included in this estimate. Identify the UUCP associated with there usable components and Adjust the size of SzUC accordingly.

1.4 Weighting Experience Factors-

The level of experience for each team member can have a great affect on the accuracy of an estimate. Consider the experience level for each team member, called the Experience factor (EF).

Experience Factor	Factor Description	Weight Factor	Project Rating	Sub total
E1	Familiar with FPT software process	1	4	4
E2	Application experience	0.5	2	1
E3	Paradigm experience (OO)	1	4	4
E4	Lead analyst capability	0.5	4	2
E5	Motivation	0	4	0
E6	Stable Requirements	2	2	4
E7	Part-time workers	-1	0	0
E8	Difficulty of programming language	-1	3	-3
Total EFactor				12

Table -4 Weighting Experience Factors

To calculate EF, go through the table above and rate each factor from 0 to 5. For factors E1-E4, 0 means no experience in thes ubject, 3 means average, and 5 means expert. For E5, 0 means no motivation on the project, 3 means average, and 5 means high motivation. For E6, 0 means unchanging requirements, 3 means average amount of change expected, and 5 means extremely unstable requirements. For E7, 0 means no part-time technical staff, 3 means on average half of the team is part-time, and 5means all of the team is part-time. For E8, 0 means an easy to use programming language is planned, 3 means the

language is of average difficulty, and 5 means a very difficult language is planned for the project .For each factor, multiply its rating by its weight from the table above. Add together all of these factors to get the total E factor.

Σ(Elevel) * (Weighting Factor) * = Efactor

Calculate the Experience Factor (EF) by multiplying Efactor times

–0.03 and adding 1.4.

(-0.03 *Efactor) + 1.4 = EF

(-0.03 * 12) + 1.4 = 1.04

To calculate Use Case Points (UCP), multiply SzUC by EF

SzUC * EF = UCP

61.2 * 1.04 = 63.648

o or UUCP * TCF * EF = UCP

60 1.02 * 1.04 = 63.648

2. REFERENCES

[01]. [BOE88] Boehm: "A Spiral Model for Software Development and Enhancement", Computer, vol.21, no.5, May 1988, pp. 61-72.

[02]. [IEE93] IEEE Standards Collection: Software Engineering, IEEE standard 610.12-1990, IEEE, 1993.

[03]. Stein Grimstad , Magne Jrgensan, " A Framework for the analysis of software cost estimation accuracy," ISESE'06,ACM,September 21-222 , 2006.

[04]. J.P.Lewis, " Large Limits to Software Estimation," ACM , Vol26, No.4, July 2001 , p.54-59.

[05]. Chris F. Kemer, "Software Cost Estimation Models",IEEE,1991.

[06]. Chris F.Kemer, " An Empirical Validation of Software Cost Estimation Models",Communications of the ACM,V.30 no.5 ,pp.416-429,1987.