Cost Estimation Tutorial

Cost is s strategic concept in software development for the following reasons:

1- **Project management**: Estimating cost is extremely crucial in carrying out project management activities such as scheduling, planning and control.
2- **Feasibility Study**: Making investment decisions regarding software projects requires full cost breakdown and analysis .Consequently, identified recurring and one time costs are then incorporated in a financial feasibility study in terms of cost-benefit analysis.
3- **Cost reduction: Since** software engineering aims to provide cost-effective software solutions to business problems, many process and project related activities are designed or re-engineered to achieve the goal of cost minimization.
4- **Evaluating business performance:** Cost is an essential ingredient to calculate many of the financial ratios – explained above- uses to evaluate the financial performance for business firm
5- **Leverage:** Cost plays a significant role in both e the operating and the financial leverage in respect of risk and return. Relying on fixed costs as opposed to variable costs can boost the operating leverage while financing with high percentage on debt-based costs may boost the financial leverage.

Cost Estimation

Every year more projects are doomed by poor cost and schedule estimates than by technical, political or organizational problems. It's no wonder that so few companies realize that software cost estimating can be a science, not just an art. It has been proven that it is quite applicable to accurately and consistently predict development life cycle costs and schedules for a broad array of software projects.

Though a vast body of knowledge exists today in respect to cost estimation techniques, most of these estimation techniques view cost as a function of complexity whether explicitly or implicitly. In early models, complexity means the project size or the program volume, which can be estimated merely via kilo lines of codes KSLOC. In late models, complexity is determined firstly by inputs, outputs, interfaces, files and queries that the software system needs. Then this complexity is further adjusted via up to 14 different added-complexity factors. Eventually, the final result is converted, through a standard conversion table to KLOC.

In basic cost estimation model the calculation is straightforward. By determining the value of only two variables, total efforts in person-months can be easily calculated. These two variables are :

- How many thousands of lines of code (KSLOC) your programmers must develop?
- The effort required per KSLOC (i.e.: Linear Productivity Factor)

Accordingly, multiplying these two variables together will result in the person months of effort required for the project provided that the project is relatively small. Otherwise, another exponential size penalty factor has to be incorporated for larger project sizes. Person-months implies the number of months required to complete the entire project if only one person was to carry out this mission. This underlying concept is the foundation of all of the software cost estimating models especially those originated from Barry Boehm's famous COCOMO models.

COCOMO Sample Example

Suppose it is required to build a Web Development system consisting of 25,000 lines of code. How many person months of effort would this take using just this equation if:

1- The project size was relatively small
2- The project size was considered large

Answer:

1- For a relatively small project:

Efforts = Productivity x KSLOC

= 3.3 x 25 = 82.5 Person-Months

2- For a large project :

Efforts = Productivity x $KSLOC^{Penalty}$

= $3.3 \times 25^{1.030}$ = 90.86 Person-Months

It should be noted , however . that COCOMO formulas have also different modes , models and versions up to COCOMOII and the new COCOTS.

Estimating software costs typically involves the following drivers:

1- Complexity of the software project
2- Size of the software project
3- Efforts needed to complete the project
4- Time needed to complete the project
5- Risks and uncertainties involved .Yet , the risk driver is still not clearly incorporated in the majority of cost estimation models for software systems .

Despite of several differences, most cost estimation models are more or less based on the following rule:

Complexity à size
(Complexity determines software size in terms of KLOC)

Size à efforts
(KLOC determines efforts in person-months with a given level of productivity and exponential size penalty factor)

Efforts à time
(Effort determines time at a given mode and/or model)

Time à Number of people required
(Time determines people “well-trained full time software development team”)

Four standard conversion tables are widely adopted in cost estimation. These tables are shown below.

Table 1. Common Values for the Linear Productivity Factor

Project Type	Linear Productivity Factor
COCOMO II Default	2.94
Embedded Development	2.58
E-commerce Development	3.60
Web Development	3.30
Military Development	2.77

Table 2. Typical Size Penalty Factors for Various Project Types

Project Type	Exponential Size Penalty Factor
COCOMO II Default	1.052
Embedded development	1.110
E-Commerce development	1.030
Web development	1.030
Military development	1.072

Table 3-a Factors for Converting Raw Values to Function Points

Complexity

Description	Low	Medium	High	Total
Inputs	__x 3	__x 4	__x 6	____
Outputs	__x 4	__x 5	__x 7	____
Queries	__x 3	__x 4	__x 6	____
Files	__x 7	__x 10	__x 15	____
Program Interfaces	__x 5	__x 7	__x 10	____
TOTAL UNADJUSTED FUNCTION POINTS				____

Table 3-b Complexity Factors

	Scale of 1 to 5
Data Communications	_____
Heavy Use Configuration	_____
Transaction Rate	_____
End-User efficiency	_____
Complex Processing	_____
Installation Ease	_____
Multiple sites	_____
Performance	_____
Distributed functions	_____
On-line data entry	_____
On-line update	_____
Reusability	_____
Operational Ease	_____
Extensibility	_____
Project Complexity (PC)	_____

Table 4. Lines of Code Per Function Point by Programming Language

Language	SLOC per Function Point
C++ default	53
Cobol default	107
Delphi 5	18
HTML 4	14
Visual Basic 6	24
SQL default	13
Java 2 default	46
C	130
Turbo Pascal	50
Power Builder	15
Packages	10-40

- **Function Points Estimations**

An alternative to direct KSLOC estimating is through function points, then use a the above standard table called "**Lines of Code Per Function Point by Programming Language"** to convert them to KSLOC. Function points was used for the first time by IBM to capture the complexity of the software system in terms of its SRS functionality and the way it interacts with its users.

How Function Points Work?

1- Estimate the number of external inputs, external interface files, external outputs, external queries and logical internal tables (files).

2- Use the **Function Point Conversion Factor** table to get total initial function points .

3- Initial function points are adjusted via 14 complexity factors to obtain final (adjusted) function points.

4- Use adjusted function points to obtain KSLOC.

5- Use KSLOC to estimate efforts as explained in COCOMO examples above

FP Sample Example

Suppose the requirement specification for the **Blood Bank Website Development** of the blood bank project has been carefully analyzed and the following estimates have been obtained. There is a need **for 11 inputs, 11 outputs, 7 inquiries, 22 files, and 6 external interfaces**. Also, assume **outputs, queries , files** function point attributes are of **low complexity** and all other function points attributes are of **medium complexity**.

The complexity adjustment value for factor 1 is set to 3 because the SRS requires that the software product has only a good degree of data communication; factor 2 is set to 0 because the SRS emphasizes no need for heavy use configuration; factor 5 is set to 3 because the order-web-based order fulfillment module has medium level of complex processing; factors 10 and 11 are set to 4 and 2 respectively because the module is always on-line but needs only few updates ; factor 3, 4 6,7,8,9,12,13,14 are set to 4, 3,2, 3,4,3,4,3,2 respectively based on their estimated level of complexity or demand..

Make the following calculations showing the full procedure in details:

1- What is the FUNCTION POINTS (FP) for the blood bank project
2- What is the ADJUSTED FUNCTION POINTS (AFP) for the blood bank project?
3- What is the approximate number of LOC in the following languages:
   - "C++" programming language
   - "Java" Programming language
4- Calculate the estimated schedule time in person-months assuming that Java is used as the implementation language
5- Use COCOMO Model to directly estimate efforts and time without using function points ?

Answer

1- Calculating Function Points

FUNCTION POINTS ESTIMATION (1)				
DESCRIPTION	LOW	MEDIUM	HIGH	TOTAL
INPUTS	X3	11X4	X6	44
OUTPUTS	11X4	X5	X7	44
QUERIES	7X3	X4	X6	21
FILES	22X7	X10	X15	154
PROGRAM INTERFACES	X5	6X7	X10	42
	Total Unadjusted Function Points			305

2- Calculating Adjusted Function Points

FUNCTION POINTS ESTIMATION (2)	
DATA COMMUNICATIONS	3
HEAVY USE CONFIGURATION	0
TRANSACTION RATE	4
END-USER EFFICIENCY	3
COMPLEX PROCESSING	3
INSTALLATIOIN EASE	2
MULTIPLE SITES	3
PERFORMANCE	4
DISTRIBUTED FUNCTIONS	3
ON-LINE DATA ENTRY	4
ON-LINE UPDATE	2
REUSABILITY	4
OPERATIONAL EASE	3
EXTENSIBILITY	2
PROJECT COMPLEXITY (PC)	40

FUNCTION POINT ESTIMATION (3)	
PROCESSING COMPLEXITY(PC):	40
ADJUSTED PROCESSING COMPLEXITY (PCA)	0.65+(0.01 *40)= 1.05
TOTAL ADJUSTED FUNCTION POINTS	305 * 1.05 = 320.25

3- Approximate number of LOC for the following languages:

- **"C++" programming language :**

LOC = 320.25 x 53 = 16973.25 ~ 17 KSLOC

- **"Java" Programming language**

LOC = 320.25 x 46 = 14731.50 ~ 14.7 KSLOC

4- Estimated efforts calculation

$$\text{Efforts} = \text{Productivity} \times \text{KSLOC}^{\text{Penalty}}$$

$$= 3.3 \times 14.7^{1.030} = 52.58 \text{ Person-Months}$$

5- Using COCOMO Model to directly estimate efforts and time without using function points

We are going to use the Intermediate COCOMO and Semi-Detached mode for this example

Estimated KSLOC	New KSLOC	Function/Module name
5.5	5.5	General/Donor Website Development
8.5	8.5	Internal/Blood Bank Website Development
3.0	3.0	Blood Requestor Website Development
4.5	4.5	Web security
4.5	4.5	Reporting tools
3.0	3.0	Fault – tolerance
29.0	29.0	Total Project for Web Development

Comments = (1/5) * New KSLOC * 20%
= (1/5) * 29000 * 0.2
= 1160

Total KSLOC = New KSLOC + Comments
= 29000 + 1160
=30160
≈30 KSLOC

LM = 3.0 * (KSLOC)^1.12
= 3.0 * (30)^1.12
≈135 Labor Month

DT = 2.5 * (LM)^0.35
= 2.5 * (135)^0.35
≈ 14 Calendar Month

LM: Labor Month
DT: Development Time