This book was never published as the market is very small. But in case you need a hard copy of the book mail to bpb@vsnl.com . Depending on the book demand we can go for print on demand model

How to prepare Software Quotation

-- By Shivprasad Koirala

Beta Version 1.0

Pending activities
1) Spell Checks
2) Estimation models for Datawarehousing
3) Estimation model for customization like SAP , BIZTALK etc

More Books written by Shivprasad Koirala
1) .NET Interview Questions
2) SQL Server Interview Questions
3) Java Interview Questions
4) Software Testing Interview questions
5) C# Projects
6) Java Projects

You can buy any of my books from bpb publications mail bpb@vsnl.com for more details.

Contents

Overview of Book

Dedication

This book is dedicated to my kid sanjana, whose dads play time has been stolen and given to this book. I am also thankful to my wife for constantly encouraging me for this assignment. Finally at the top of all thanks to two old eyes my mom and dad for always blessing me.

On professional basis I am thankful to Anish Goenka and AG-TECHNOLOGIES (www.ag-technologies.com) for providing me opportunity in software estimation , i know i made some very big mistakes during estimation but still his constant encouragement has brought me till here in estimation field.I am extremely thankful to all my friends who reviewed my book and gave feedback to make this book stronger.

About the author

Shivprasad.H.Koirala has over 8 years of experience in software industry. He is working presently as project lead in one of the multinational companies and in past has led projects in Banking, travel and financial sectors. Author also provides consultancy in estimation. He can be contacted at shiv_koirala@yahoo.com, adding to chat is also invited.

Readers Feedback

Success in any book is not due to authors but by good reviewers. We expect decent criticism from readers for better of the book.The author of this book does not considers himself a guru in estimation and expects criticism from reader for better understanding of subject. Any mistake in this book either technical or grammatical is by author himself , no other person or the publication company is responsible for it. Mail mistakes and feedback of this book to shiv_koirala@yahoo.com , adding to chat is also invited.

Audience

This book is mainly targeted to the following section of readers:-

- Readers who want to have knowledge of know how of LOC(Lines of code), COCOMO (Constructive Cost Model), FP (Function Points) , UCP (Use Case

points), Maintenance project estimation , SMC (Simple medium complex) approach and WBS (Work break down structure) estimation approach.

- Companies who want to implement scientific estimation techniques in there company.
- Companies who are migrating from Function points to Use Case Points.
- New comers who are very new to estimation this book will go from smaller basics to know how of practical estimation.
- From CEO to programmers who wants to have kick start in estimation.
- Experienced estimators who are looking to improve there estimation skill.
- Professor and students who have software estimation in academics.
- PHD students who are looking for in depth study in different software methodology.
- Readers interested in comparing different estimation techniques.
- Process implementers of the company who are looking for ready made templates for implementing scientific software estimation in company.This book provides excel templatesUse Case templates and Function Points templates) and widely used estimation softwares with CD which can be used for estimation.

How's Book organized

Book is organized in to eight main chapters. Book starts with introductory of quotation and life cycle of quotation.

Then we proceed towards different old ways of estimation and listing down there advantages and disadvantages.

In second chapter we go through one of old methodology of preparing quotation using LOC (Lines of Code). In this we also have a overview of how COCOMO can be used with LOC for estimation.

Third chapter of this book revisits one of the most accepted estimation and quotation preparation technology Function Points. This section defines all the elements of function points (EO, EQ, EI, ILF, and EIF) and then estimates a sample customer project.

Fourth chapter introduces Use Case Points one of the coming up estimation methodology. This chapter goes through all definition of Use Case Point Elements (UAW, UUCW etc) and sample project. The best of the book is that book gives you instant warm up with sample project and excels templates.

Fifth chapter introduces a more adhoc estimation methodology using WBS (Work Breakdown Structure) and SMC (Simple, Medium and Complex). WBS is very popular project management methodology, in this book it is used from estimation point of view. Again this chapter has sample projects for instant practical kick off.

Sixth chapter steps in to practical estimation ground by estimating sample projects. This chapter starts with comparison of two most used estimation methodology Use Case Points and Function Points. Excel templates are shipped with CD for all the sample projects.

From chapter one to six the book mainly concentrates on estimation methodology for early life cycle estimation (i.e. when there is no design document on hand, only small scope text is present). But other than this there are other types of projects like maintenance projects and change request. This chapter focuses on these types of projects. As usual sample projects are estimated from practical aspect.The main purpose of this book to go towards practically implementing software estimation methodology. This chapter focuses from practical implementation what points to be remembered, in short to do and to don'ts list for estimator.

Finally we end with acronym and references for further reading.

What's in CD

"Thousand words are equivalent to one practical implementation".

This is exactly what this book approach will be , so as a initiative we have provided CD with this book. Thanks to all the software companies and individual programmer who have honored this book by giving me copyright to ship there software with CD.

Following are list of estimation software's available in CD:-

- COSTAR (for COCOMO estimation)
- Enterprise Architect (For Use Case Points Estimation)
- Java Open Source(Petalo) (For Use Case Points Estimation)
- EzEstimate (For Use Case Point Estimation)

- Excel Templates (This is provided by author himself).

Note: - Any problem and issues regarding software should be reported directly to the software manufacturer. Any bugs and software problems is beyond the scope of the author.

Chapter 1

Introduction to Software measure

What's a Quotation?

> *"Quotation is a document sent from supplier to customer regarding financial details for the service/product provided. Quotation can also be called as temporary financial value, which is in negotiation stage ".*

All industries fall in to two major categories:

- Physical product selling
- Logical Service selling (Servicing Industry).

Preparing quotation for logical industry is more difficult. Software industry falls in to logical selling or service industry.

> *"Quotation is reflection of effort required to fulfill the client service.It's the first financial document to be sent after scoping phase or during scoping phase."*

"A contractor receives a contract to deliver 10 tons of steel from one place "A" to other place "B". He has 10 trucks with max capacity 1 ton. Place "A" to place "B" is 1 miles away. The contractor analyses cost as 1 truck can carry 1 ton for 1 miles and costing is 500 $.So 500 X 10 = 5000 $."

That's the way for that service a simple contractor concluded his estimation. Let's see how the contractor was able to give quotation so precise, quick and fast.

- He knew his ultimate goal is to deliver 10 tons of steel.
- He knew place "A" and place "B" are 10 miles away.
- He knew he has 10 trucks which can take 1 ton.

He had measures 10 tons, 10 miles etc. So for a better quotation we should be able to measure. This book will in the first half concentrate on measures and then show to convert the measure to quotation.

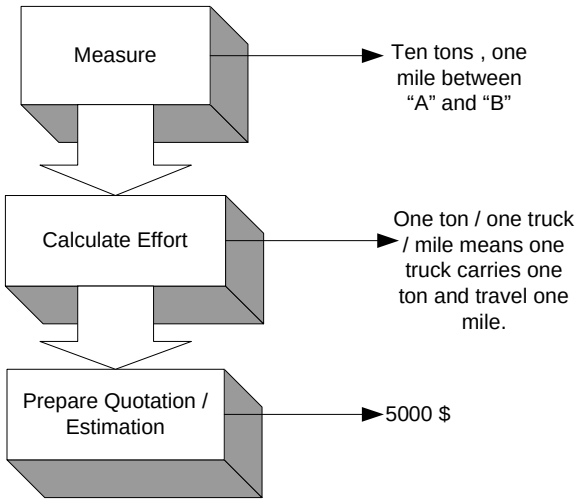


Figure 1.0 Process of estimation / Quotation.

So let's understand what's a software measure and quotation will follow?

Life Cycle of Quotation

This section will look in to how a quotation cycle flows in a company. While reading the steps try to follow Figure 2 give below. But main intention of drawing diagram is as follows

- To show depth of financial effort and time investment done in preparing estimation and quotation documents.
- To highlight success points of successful quotation and estimation documents.
- Step1 :- Company receives project enquiry which can be from following sources.
  - Reference.
  - Marketing person visits the client company.
  - Client approaches himself.

- Step 2:- This enquiry is passed to marketing department. Marketing department evaluates the client from following perspective
  - Is the client serious or is it just a casual enquiry?
  - What's the image and status of the client?
  - Most important, how much is client willing to spend on the project? Clients spending will not be normally known but if known will provide better insight on negotiation table.

This step can be exception in many companies where they directly pass it to the technical department.

- Step 3 :- Once marketing department is sure , they forward it to technical department for following deliverables to prepared:
  - Understanding and Scope document.
  - How much time and effort will it require to complete the project
- Step 4, 5 and 6:- While preparing Understanding document, if they have difficulty understanding the project they get back to the client. The co-ordination with the client is done normally through co-coordinator who is deployed at client side. This can be exception also where the technical team directly co-ordinates with client. Client can be contacted in the following ways :
  - Telephonic conversation
  - Email.
  - Through Business analyst or coordinator appointed at client side.
  - Client himself comes to company to clarify doubts.
- Step 7 and 8:- During this clarification small technical samples can also be prepared to ensure technical feasibility of the project. Example the project has requirement to send reports to mobile. Now is that possible so for this small code samples can be run and seen. This is called as POC [Proof of Concept].If the technical department says it's not possible then probably again coordinate with the online client coordinator or client himself to clarify things.

- Step 9:- Technical department starts prepare quotation which will involve estimating effort using Function Points (FP), Use Case Points (UCP), WBS (Work Break Structure) or ADHOC methodologies [This book will explain all this methodologies in detail later as chapter proceeds].
- Step 10:- After technical department finishes the effort estimate it passes it to the board of directors, or senior members of the company for final approval. They review the quotation amount once before sending to the clients.
- Step 11:- Finally the quotation is sent to the client. If client gives a positive feedback then understanding document is signed off, initial payment given (if in the terms and conditions) and the project is started. If the client is not happy with the quotation pricing amount it moves to negotiation table. So probably client will remove some functionality and ask for a fresh quotation or will stress the company management to minimize the quotation amount.

If the client says "No" then marketing people starts follow up for the reason. If the client response is not too good then either some other software company has put a good quotation amount or some other reason.

In the whole quotation cycle following are the places where company will have financial investment:-

- Technical and marketing manpower engaged to send the quotation.
- Company has to pay salary for these people.
- Client co-ordination investment
- Telephonic conversation. In short telephone bills can be high when its international client.
- Business analyst appointed at client side if any.
- Traveling charges to client location especially can be problem if the client is international.
- Preparing POC and prototype screens to understand project better will also need quiet an effort.

In short after all this investment is waste if there is no client response.

Following are properties of good quotation:

- Quotation amount should not be too less or too more than actual effort required to complete the project. That is if the project is actual of 1000 $ then your estimate should be either 1200 $ or 900 $.Then later director adds his profit is other thing.

Note :- "Software Estimation and quotation with in 10 to 20 percent of actual estimation is considered very excellent estimation."

- A good quotation should at least take the company to a negotiation table. So that company at least does not loose the project.

so if you look at the quotation life cycle diagram its quiet a decent investment from company's financial aspect..

Note :- "In many companies the quotation life cycle can be different but financial investment areas do not change lot."

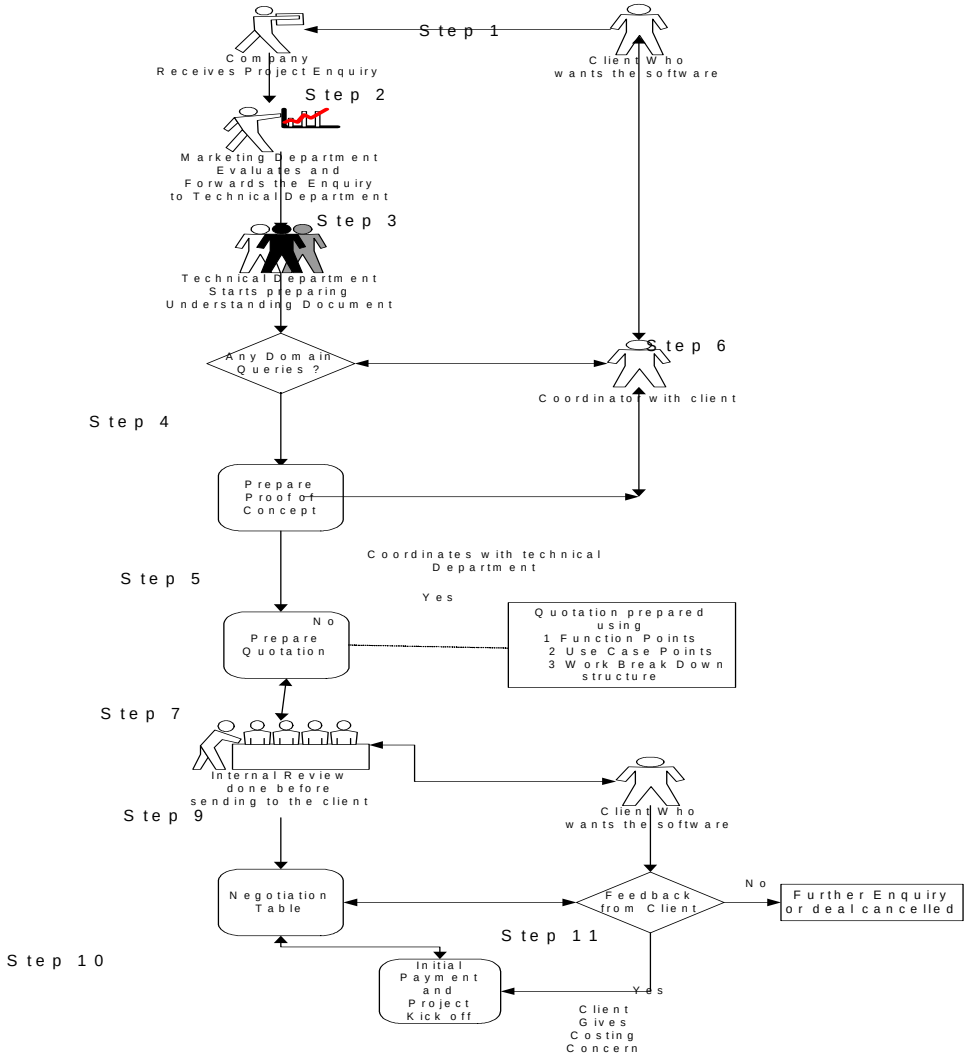


Figure 2 Estimation / Quotation Life Cycle

Word Before we start

Software Industry is divided in to two groups on who believe that scientific way can be applied to measure software and the other group who do not believe. It would be bad on my part if I do not address the group who do not believe. Many times my programmer friends use to smile and say mathematical equations for preparing quotation you are kidding? But well even I was a part of that category so no comments. So will this book give a math's equation or formulae which can give 100 % ball point quote for software?

The answer is – NO.

Many scientific way of estimation has been proposed for last 35 years (Around 1000 ways of estimation way proposed and 6000 plus white papers written).Large international bodies have been setup so that these models become mature. But every model had there own advantages and disadvantages. This book will give overview of widely used costing model in software industry.

Ok now back to addressing the non-believers. Non-believers follow one of the below ways to come to a quotation / estimation:

Price-to-Win

In this approach the quotation amount is exactly or probably little than what customer pockets are or competitor submitted quotations. So the basis of preparing quotation is the customer and the competitor. So if the project is 1000 dollars and customer can afford only 400 dollars. Estimator is asked to modify the quotation. So in price-to-win either programmer works overnight or the company goes under loss. Normally upcoming software companies do have this approach to increase there client list. This approach is not recommended as probably you will end with large quantity, projects but serving them will end up in to loss. Many software companies during recession have followed this approach and have ended no where. History tells any business follows this type of approach had tough time to survive.

Freelancer approach

Normally 90 % of freelancers follow this approach. The basic equation they follow is

- Salaries + Overhead = Annual Costs.
- Divide the annual cost by annual hours worked.

That's how freelancers come to a fixed charge and for maintenance they charge per visit charges. Which is normally calculated by traveling charges plus how much they want the

profit in this visit? Again this approach is not recommended as the quotation is independent of project. But rather is dependent on running cost of company.

Figure 3 Freelancer Calculation Table

Salaries	$10,000
Overhead	$5000
Annual hours	2080
Total calculated hourly rate	10,000 + 5000 /2080 = $ 7.21 = approx $ 7.50

Expert Judgment or Sixth sense approach

This is the most organized way non-believer works. In this approach we divide the whole project in to small subsystems and there components. Every components of the subsystem is assigned how many man/days will be required. After the allotment is completed all the man/days for the component is added up to come out with total man/days of the project.

Below is typical example of Expert judgment:

Functionality	Man/Days
Accounting	
Creating Chart of Accounts	5
Journal Master maintenance	5
Transaction Entry screen	10
Reports	7
Closing of accounts	4
Administrator module	
User maintenance	2
User Rights	2
User Roles	2
Help Files	2
Total Project in Man/Days	**39**

Figure 4 Expert Judgement table

The above sheet is taken from real project. The above quotation was for simple accounting project. I have tried best not to modify any values but only tabularized it.

When asked on what basis the quotation was prepared. Answer was:

- Past Experience
- If no past experience is available then visualizing by sixth sense the complexity and putting the figure (i.e. Man/Days).

The above approach is followed by most small scale companies and even large scale. Many times estimators do use this approach when measuring models can not be applied to some projects.

Following are drawbacks of expert judgment:

- It's difficult that past project will be exact replica of the current project. As one of the criteria for measurement is past experience. The above quotation is for accounting module, but if the company again gets project of accounting module. They can not put in the same quotation as one companies accounting approach is very different from other companies accounting approach.
- Difficulty to find expertise in all software sectors. A software company which is more in to database projects, suddenly lands on to a embedded system project. On what basis will they give estimate?
- Quotation is prepared more on imaginative basis; customer can question authenticity of the quote.
- Whole quotation is fully relied on estimators forecasting ability.
- Quotation can vary from one estimator to other estimator with a huge difference.
- Large possibility of either over quote or under quote.

Scientific way to approach to cost estimation has always got heavy criticism, because it applies uniformly across all scenarios. But definitely estimation should be unbiased and should not change with time, technology and project.

As said before software industry is still not matured and unified on common way of estimation. So definitely every good estimator will use the expert judgment when costing model formulae fails for scenarios.

So let's define first what's a measure? Then let's see what are we trying to measure in software?

> *"We should no longer ask if we should measure the question is how" [Dieter Rombach at Euro metrics].*

Introduction

> *"You can not control what you can not measure "[DeMarco 87].*

Right from childhood in school, we are taught different unit measurements and its conversion. These measuring units have been accepted by human kind across countries and language boundaries. Example:-

1 Mile = 1.6093 Kilometers

This Conversion is true in USA, CANADA, EUROPE or any other country. Even the conversion from one unit to other unit is consistent.

1 Mile = 1.6093 Kilometers

Conversion back

1 Kilometer = 0.62137Mile

Complex mathematical calculations can be performed on these units of measurement. These units of measurement an also be compared if targeted towards a common objective.

- If measurement objective is length, we can use Kilometer, Meter, Centimeter etc.
- If measurement objective is weight we can use Kilogram , Gram etc
- If measurement objective is software????? (Hold your breath)

All the above human accepted measurement is objective by nature. That's it can be physically seen or felt. But software is logical can not be felt. So what do we measure? , How do we measure?

Definition of software measure

Software Measure is numerical value assigned to a software project depending on the complexity of project. The complexity of the software can be known from "Data Collected" and "Client Artifacts".

Data Collected can be any of the documents mentioned below:-

- Project or Software Specification Document :-

These types of documents are gathered before the implementation stage. Normally software project quotations have to be given at very initial stage probably when you have no data at all. Initial stage means probably in first or in second meeting with client. That's really a challenging part because just on basis of rough idea you have to give quotation.

Below mentioned are some documents

- MOM (Minutes of meeting) with clients
- Scope document
- Documentation done at client side.
- Tele-conversation with clients.
- Email received from client etc
- Source code (If it's a migration project)

Any type of client received documents should be studied properly to arrive to a good estimate. Now that we are familiar with definition of software measure. Let's go to the next section of what are characteristics of a good software measure.

Characteristic of a software measure

Till today Human accepted measurement has following characteristics:

- Measure should not change with environment.100 centimeter does not change with climate, country etc
- Measures from same objective should be comparable. Example 100 Centimeter > 50 Centimeter when the objective is length.60 Square meter > 20 Square meter when objective is area.
- Measure should be able to convert from one form to other in same objective. 1 Mile = 1.6093 Kilometers when the objective is length.

In order that software industry to have such accepted measure many people have tried to define characteristic of a good software measure. Elaine Weyuker has brought out 9 properties that good software measure should have:-

"All the measures considered depend only on the syntactic features of the program "

All the nine properties which we will list below depend on the syntax (Grammar) of the programming language used.

1. UN-EQUALITY: Given any program "P" there exists another program "Q" such that M (P) ! = M (Q).Where "M" is the software measure of the program. It also means that software measures which assign all programs the same value is not a measure.

2. EQUALITY: For any program "P" there exists program "Q" such that M(P) = M(Q).

3. RE-ARRANGEMENT UN-EQUALITY: There exists program “P” and “Q” such that “Q” is re-arrangement of “P” and M(P) != M(Q).This equation can be valid even if “P” and “Q” are delivering same functionality.

4. IMPLEMENTATION UN-EQUALITY: There exists program “P” and “Q” such that they produce same output given the same input and M(P) != M(Q).That is same type of logic written by two programmers can have different approach.

5. MERGING UN-EUQALITY : For all programs “P” and “Q” then

M (P) < = M (P+Q)

M(Q) < = M(P+Q)

P+Q here mean concatenation.

6. EXTERNAL PROGRAMS MERGING UNEQUALITY: There exists program P,Q and R such that M(P) = M(Q) and M(P+R) != M(Q+R).This equation just says that way one program interacts with other program is quiet different.

7. INTERACTION INCREASES COMPLEXITY: There exists program P and Q such that M (P) + M (Q) < = M (P+Q).

8. FINITENESS: Given a non-Negative number C, there are only finite number of programs entities such that M(P) = C.

9. ROSE PRINCIPLE: If P is a renaming of Q then M (P) = M (Q).

Last Words on Weyuker properties

Weyuker properties if read first time by any reader would think that its is comparison of size i.e. size of executable code or number of lines of code. But definitely any software project can not be measured by its size only, but also how complex it is.Weuyker nine properties also points towards complexity. Example property number (6) M (P) = M (Q) and M (P + R)! = M (Q+R).This property says that a good software measurement should not only take in to account size of the program, but also its complexity. If the comparison was only size of code or number of lines of code this property would not have existed.

The above 9 properties will help us gauge a software measure. Before introducing a software measure in a company, we can apply litmus test of these 9 properties. Please note this book does not stress Weyuker properties are golden rules. But definitely they can be used as a good base to test a software measure methodology.

What's a perfect unit of software measurement?

After going through all nine properties of Weyuker, what's the perfect unit of software measurement?

- Is it Complexity? Can we say Dear customer, the software is of 120 complexities?
- Is it Size? Dear Customer the software is of 120 MB so we charge you this much invoice?

That's our next question what's the unit of software which is globally accepted? For past many years software industry has been trying to search for the globally accepted software measure but none accepted globally.

Some widely used software measures:

- Lines of Code
- Halstead complexity
- Function points
- Use Case Points.
- Object Points
- Full Function Points

And lot more. I am sure many of the readers must not have even heard of the above measure. So definitely if you include any of above measure in quotation, it would lead to blank faces.

The right way would be to give Software Cost estimation rather than actual software measures. Software Cost estimation can be given in any of the following ways:

- Effort (Usually in Man-Days or Man-Months)
- Project Duration (Calendar Time)
- Cost (In Any currency normally dollar is taken as standard)

Software Cost estimation is the process of predicting the effort required to develop a software system. Software Cost estimation rely on software measure as input.

Note: - Many refer Software measure as Software size but many also disagree to it. This book assumes that they are same for simplicity purpose. As we can get in to complexity of difference between them rather than concentrating on main topic.

Dependency of software measure

A measure should be independent of external variations. Like example 1 Kilometer is independent of environment. But when things come to software measure it's really difficult to answer the question. Different tools, compilers, architecture makes it difficult to come to a consistent way. The first assumption of Weyuker properties says that it's syntactic dependent. The effort required to complete a module in Assembly will be different from C++.Even though many Software measure approach claim to language and Compiler independent but still considerable differences where noticed. Even architecture of the project makes a big difference on the estimation.

What's so Complex in Software?

"Human beings in order to make there life easier have introduced complexity, software is the biggest complexity what human beings have made "

Definition of Complexity

"It is a measure of the resources which must be expended in developing and maintaining or using a software product".

As discussed before we are looking at finding a way to measure complexity of the software and give quotation accordingly to the customer. Different people have different way of looking at the complexity:-

- From System Architecture point of view it's the architecture of how the software will be designed.
- From programmer point of view its source code and the logic to implement.
- From testers point of view it's simulating different environments and trying to make the software bug free.

Software complexity can be classified broadly as Follows:-

Logical Complexity

This is code complexity. Writing complex Coding logic (Example encryption logic, compression logic), Complex search screens, Complex business validations etc .In short it refers the complexity of algorithm to solve the problem.Theorotically during analysis and design phase logical algorithm can be roughly gauged. But practically algorithmic complexity is crystal clear when your code is ready.

“In one of my projects customer has a requirement that all search screen should have sorting logic. During design phase bubble sort algorithm was decided. But at actual implementation stage (Coding) we had to alter the algorithm as some sorting logic just did not fit in bubble sort. Alteration of algorithm costed us 2 % extra of projects total quotation”.

Logical complexity forms lowest unit level of complexity. Identifying logical complexity before design phase, i.e. in scope phase is really difficult.

Structural Complexity

It stands at the top level of logical complexity. How do the logical complexity interact, how will be the structure? In broader level they are also called as “Architecture of the Project”.

Some sample listings which fall in Structure Complexity:

- What level of reusability is the project? As higher the reusability, more we get in to generalized interfaces, deeper inheritance classes, Complex aggregations (Interaction of objects) between objects.
- Deployment Complexity: Projects where we are looking at sharing data across geographical boundaries, this complexity will come in. If client is also looking for auto-deployment that is as version changes all clients should be updated with the new version on server is again a type of structural complexity. N-Tier architecture, Object pooling etc all will come in structural complexity.

Structural complexity can be visualized at the inception stage to a good limit. But it becomes clearer as the project enters the design phase.

Psychological Complexity

It measures the effort required to understand the software. This complexity has more to do with psychological perception or relative difficulty of undertaking or completing a system. We are not considering this complexity as it is more to do with Social scientist and psychologists.

“Complexity is strongly connected to amount of resource to be allocated to complete it. “

In short in various software life cycles, various complexities are involved in order to complete the software. At this moment there is no complexity model which can encompass all the complexity in one total. In 1970’s McCabe introduced a way to calculate complexity from actual sourcecode.Let’s have a deeper look at this methodology.

> *Note for Readers:-The next sections discuss different complexity measures. It also involves mathematical equations. So probably during reading of the below section you will start feeling is this book meant for preparing quotations or a math’s book. We are looking at preparing quotations in scientific way so we do have to look at different methodology of doing measures. So do not panic keep reading as most the estimation techniques are outdated and is only for theoretical understanding. At this moment we will just try to understand the various ways of measurement and its advantages and disadvantages.*

McCabe Complexity (Logical Complexity)

McCabe (1976) proposed that complexity can be measured by program’s flow graph.

The equation is as follows

V (F) = E-N +2

V: - McCabe’s cyclomatic number

F: - Flow graph

E: - Number of directed edges

N: - Number of nodes in flow graph.

Node: - It corresponds to a program statement.

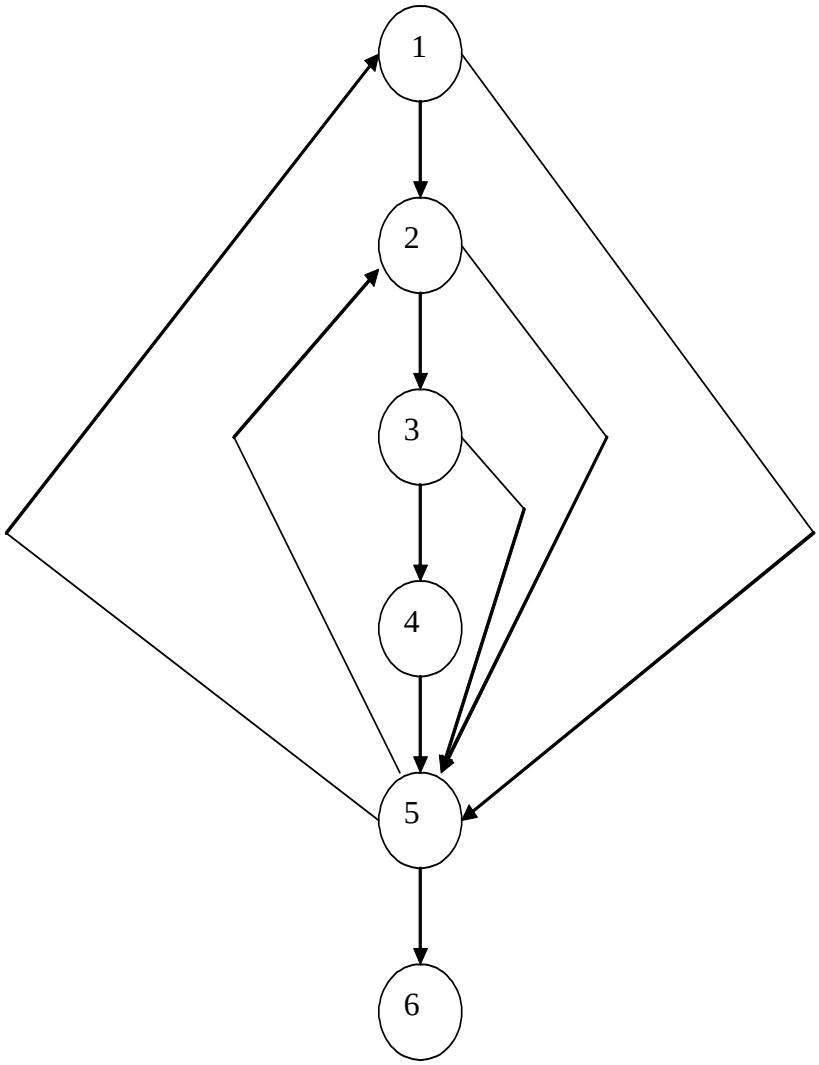
Directed Edge: - It indicates the flow of control from Node to Node.

Directed Graph: - They are depicted with set of nodes and each directed edge connects pair of node. Arrow of directed edge indicates the data coming in node and an arrow out

indicates data coming out of node.Heres below a sample code for which we will apply McCabe's formulae:

5.Figure

```
Procedure Xyz ()
        Var i, var j, var c
        Begin
            For I = 1 to 10                          Node 1
                For j= 1 to 5                        Node 2
                    If I=j                           Node 3
                        Begin                            Node 4
                            c++                          Node 4
                            c=c*10.35                    Node 4
                        End                              Node 5
        End                                              Node 6
```



6.Figure

The diagram represents the flowgraph.So substituting

$V(f) = 10 - 6 + 2 = 6$

Modules with high values of "V" have high complexity and definitely should be quoted more. McCabe proposed value greater than "10" should be redesigned to reduce errors.

This is useful tool and provides very decent information about the complexity of software, but definitely not completely. But it does not take in to account

- Portability
- Flexibility
- Complex algorithms
- Linkage between different modules

(Example inheritance, polymorphism, calling of sub-routines etc, Interaction between object etc)

This is an old technique of measuring complexity for structural approaches. But in today's world of object oriented approach it does not fit in. It also does not include complexity of the whole software development cycle i.e. Analysis, coding, design etc. So using McCabe Complexity in today's changed software industry can not be used to quotation. As quotations have to be submitted during first or second meet of the client and the paths etc can only be clear during implementation stage.

Definitely this below given code according to McCabe's has high complexity, but it's actually not.

```
Procedure ShowColor (color)
        Begin
            Var currentcolor,
            Select case color

                If color=red
                    Begin
                    Currentcolor=1
                    End
```

```
If color=Blue
        Begin
        Currentcolor = 2
        End
If color=White
        Begin
        Currentcolor=3
        End
If color=Yellow
        Begin
        Currentcolor=4
        End
If color=Black
        Begin
        Currentcolor=5
        End
If color=Black
        Begin
        Currentcolor=6
        End
If color=LightYellow
        Begin
        Currentcolor=7
        End
```

```
        End select
    End
```

7.Simple source code having high McCabe's complexity

Inter-Modular Complexity (Henry and Kafura)

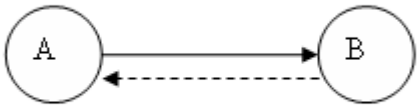
McCabe's complexity concentrates more on control-flow complexity.Inter-modular complexity stresses more on complexity between modules [Separately compilable modules]. In order to understand Inter-modular complexity we have to capture relationship between modules. Researchers have attempted to quantify several aspects of inter-modular complexity. If the inter-modular complexity is more then quotation has to be charges accordingly.

Henry and kafura's Information flow

Henry and Kafura concentrated more on how the information flows. Below are the different types of information flow. In diagrams the dark arrow lines indicate the main caller and the bashed arrow lines indicate that information flows back.

Direct Flow (Direct Module)

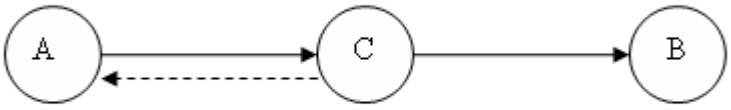
Module "A" invokes "B" and "B" passes information back to Module "A".



8.Henry Kafura's direct flow

Indirect Flow

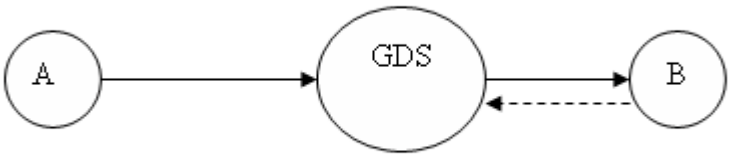
Module “C” invokes “A” and passes information to “B”



9Henry and Kafura’s indirect flow

Global

If information flow from module “A” to module “B” through global shared data.



10.Henry and Kafura’s global flow.

There are two basic fundamentals in Henry and kafuras information flow

- Fan-in(FI)

- Fan-Out(FO)

Fan-in is the number of modules which are capable of causing information flow in to the system. Basically Fan-in is external modules which cause change in module's state.

Fan-out is total count of other modules which cause information to Flow-out of the module. McCabe's complexity encompasses lower level complexity, while Henry and kafura give a higher level of complexity. As Henry and kafura complexity higher level of complexity it can be used during initial stage by drawing rough block diagrams depicting information flow.

$$\textit{Information Flow complexity (M)} = \textit{Length (M)} \; X \; [\textit{Fan-in (M)} \; X \; \textit{Fan-out (M)}]^2$$

Length factor in Henry and Kafuras equation can be measured in different ways; one is to use "Lines of Code". The above equation is non-linear as complexity nature is non-linear. According to Raps and Weyuker the highest information flow complexity should be looked in to.

Advantages of Henry and Kafura's Complexity

1. They look at higher level complexity, while McCabe looks at lower level complexity. McCabe's complexity can only be applied when you have detail source code or pseudo-code. But Henry and Kafura's complexity can be applied during design phase and also at scoping phase if proper information flow block diagrams are drawn.

Disadvantages of Henry and Kafura's Complexity

1. If fan-in or fan-out is zero the whole complexity will become zero. Just imagine you are using Henry and Kafura's complexity and fan-in or fan-out is zero the quotation is zero Dollars.

2. The length factor is completely different attribute. Multiplication of the factor with complexity has been questioned by expert.Arthimetic calculation can be performed on measurement of similar types. Example adding 1 litre + 10 volts will yield nothing sensible.

Shepperd (1990) refined it by excluding the length factor.

Shepperd complexity (M) = [Fan-in (M) X Fan-out (M)] 2

He also proposed some definition changes with respect to local, global flow data structure. Development time and shepperd complexity is very much co-related.

Halstead's Measure of Complexity

In 1977 Maurice Halstead introduced alternative method to LOC (Lines of Code) for measuring size. Note: - Hence forth in this book you will come across acronym do look at the acronym section for further details. McCabe's complexity was introduced in 1975 and halstead just followed (1977) after that.Halstead used attributes to determine the complexity of the software. According to Halstead operators and operand determine the complexity measure. According to halstead Define a program "P" as collection of tokens, classified as either operators or operands.

Here are basic definitions of tokens:-

n1 = Number of unique operators.

n2 = Number of unique operands.

N1 = Total occurrences of operators.

N2 = Total occurrences of operand.

Every language have there own way of defining operator and operand.

Operator		Operand	
Array	1	Save	7
String	2	Add	2
If then	2	X	3
Else	1	n2=3	N2=12
<	2		
=	4		
n1=6	N1=12		

11.Table Sample of counting operators and operands

Please note the small and capital letters "n1","N1"/"n2","N2". Counting the operator and operand in big projects manually will be huge task. So for this measurement you will need some automated tool. Below is the table which has all the equations related to halstead measures.

Size of vocabulary $n=n1+n2$

Program Length $N=N1+N2$

Program Volume $V=N \log 2(n)$

Programming Effort $E=(n1*n2*N1*\log2(n))/(2*n2)$

Programming Time(Seconds) $T=E/18$

Approximation for N $N = n1*\log(n1)*n2*\log(n2)$

12.Halstead equation's

Volume is number of mental comparisons needed to write a program of length "N".Halstead complexity gives 3 different views of size(Program length, Volume and effort).The number "18" use for calculating programming time is element discrimination per second a human mind can do."E" is the element of discrimination needed to understand the program.

Advantages of using Halstead measure

- To define operator and operand is the most difficult and tedious thing in this measure.
- For different languages you have to define operator and operand again and again.
- Halstead assumption that human mind is capable of 18 mental discrimination per second is too exact. It does not have any actual real time evidence.
- Attributes like volume and effort relationship is unclear with real world.
- Halstead measurement can only be done by automated system tool. Nobody would sit counting operator and operand in huge projects which has 1000 of modules.
- From preparing quotation and estimation using halstead source code or pseudo-code should be at place. Halstead measure will fail for initial phase of software cycle. As in initial stage you will have only explanatory documents which will not give picture of operator and operands.

Logical complexity/Algorithm Complexity

McCabe's, Halstead and Henry kafura deal with structural complexity. But in software programs structure complexity is not every thing. You write complex logic, algorithm etc in order to achieve functionality. More complex the algorithm the more the quotation.

Definition of algorithm: - A set of rules which must be expressed in order to solve a significant computational problem (Jones 1996).Sorting, Searching, Complex financial calculation, compression logic are typical example of algorithm complexity. There are no rules of how to determine algorithm complexity. Here are some guide lines given by caper Jones (1996):-

- Number of calculation steps or rules required by algorithm.

- Number of factors or data element in the algorithm.

The above two things determine complexity of algorithm.

Disadvantages of using Algorithm complexity in preparing quotation

- Algorithm complexity can not be measured at initial stage.
- In today's world of ready made plug and play components, algorithm complexities have become negligible. In days of COBOL to write a sorting algorithm was all from scratch. Today every data display has by default sort algorithm.
- Again way of writing algorithm varies from programmer to programmer.
- In a software project algorithm complexity is very small aspect of project. So preparing quotation purely on basis of the project complexity can be quiet misleading.

Conclusion

In this chapter we looked in to various ways of measuring complexity. The basic intention of looking in to different complexity ways was to come out with a measure. This measure can later be used to prepare quotation. But the problem with all the complexity discussed above is that they demand source code or pseudo-code at the first place. Normally when a software company gets projects enquiry they get client artifacts like

- Documents (Probably world documents, Excel spread sheet etc)
- MOM(Minutes of Meeting)
- Telephonic conversation
- Emails

On any of the above documents applying Halstead, Henry and Kafura or McCabe measure is difficult.

But there are situation where a company gets migration projects. For migration project do get source code at initial stage itself. But the following are the problems identified in migration project old source code:-

- The learning curve of understanding the old source code and applying one of the above complexity measures is time consuming.

- All above complexity measure depend heavily on program structure. Let's say company receives an old code in COBOL (The project is of compressing files) which has to be migrated to Java or .NET. So by using automated tools you run on the COBOL project files and the tool gives you complexity measure. But in JAVA for the same functionality you have ready made component. So the readings can be completely wrong. It would be injustice to client also.

- Normally in migration projects the client does not expect the same functionality. They expect the existing functionality and some thing extra. So even if you are successful applying one of the complexity measures, for estimating extra functionality it will be again difficult.

The above discussed complexity can be used very effectively in the following ways:-

- To see which code can be more error prone. So that the testing department an effectively work on that section.

- To measure the quality of code. Example if you see a section of program has high McCabe's complexity. That section can be again revisited to improve quality.

From all above measures McCabe's complexity is most preferred for error proneness. Moreover it is always supported in all automated tools.

This book will not use any of the methodologies defined in the previous chapters it was only for education purpose. Many of the counting methodologies were defined for old type of development environment. Today software industry development is more using RAD and code generators which can lead to weird estimations. So lets go ahead and explore other methodologies.

Chapter 2

Lines of Code and COCOMO

"Any estimation +- 10 % is a good estimation/quotation.Estimation and Quotation preparation for software services is not a bullet proof science and so termed as estimation."

Complexity and Size Measure

In previous chapter we looked at popular ways of measuring complexity. The problem with complexity approach is following:-

- It's difficult to come out with size at initial stage. Complexity depth of software can only be better known after actual implementation.
- Definition of complexity has changed with software community becoming more matured. In today's Object Oriented Programming and RDBMS(Relational Database Management System) approach Structure or Algorithm complexity is not the only aspect. But rather new complexity have been introduced :-
  - Object Oriented Complexity (Depth of inheritance , Dynamism of polymorphism etc)
  - Distributed Architecture (2-tier,3-tier,N-tier etc).
  - Database design complexity etc.

So in order to measure size of software by software complexity approach, you have to take in to consideration total of all type of complexity. This approach will be time consuming and the correctness of the results will not be known.

- Complexity changes with language and implementation. Complexity of COBOL program is different from .NET or JAVA language. So it's not independent of language. So if a new language is adopted by organization, you have to reinvent the estimation wheel from scratch.

- Complexity solving approach depends on programmer logical capability and programmer's knowledge about the programming language. Example for simple sorting to be incorporated on display grid. Following can be approaches :-
  - Newbie programmer will think writing from scratch.
  - Medium programmer will think of bubble sort.
  - Experienced programmer will use some existing feature provided by the programming language.

If given a quotation to prepare for the sorting logic following can be results:-

- Newbie programmer – 2 days
- Medium programmer – 1 day
- Experienced programmer – 1 hour.

So depending on programming and experience capabilities your estimation and quotation can very to very decent extent.

Algorithm complexity or structural complexity or any type of complexity form Basic/ Smallest unit of software. To judge this smallest unit in a software process at initial stage is very difficult..
So the best way experts thought is rather than getting in to smaller view, take broader view and come out with cost. The other approach would be look at size rather than complexity. Size means to look at quantity, now that quantity can be from perspective of :-

- How many lines of code?
- How many files of database to be maintained?
- How many modules to be delivered?
- How much functionality to be completed? Etc

To prepare quotation using size will lead us to language independent and technology independent measure.

"Difference between complexity and size is, complexity is more related to programming deliverables and size is more related to end customer deliverables ".Lets start with a popular and most accepted size measure "Lines of Code".

Lines of Code

This is the most criticized method, but is one of the earliest attempts towards scientific measurement of software. Its history goes back to 1974 (That's before my birth) when Wolverton made attempt to measure programmers productivity by LOC. Even though it's most criticized measure, we should not forget its one of the initial steps to make software measurement scientifically.

LOC can not be used to prepare early software life cycle quotation. To make a judgment at initial stage about the quantity of number of Lines of code is difficult.

Software industry is matured of what it was ten years back. Automated code generation tools, high use of third party components, free open source code make project size huge. But these types of things can not be counted in estimation. Example your project is of hundred lines of code and you have used a open source component in your system. Open source system coding lines can not be counted in estimation as it's not coded by your company. Comment lines written by programmers can not be counted as it's written for understandability of code for third person and from customer perspective it's of least importance. In short the counting should be logical. In short we should have a check list of what should be counted in effort estimation and what should not be counted in effort estimation.

SEI (Software Engineering Institute) released check list table to improve LOC called as "Logical Source Statement of Code". The table below is refined and simplified from CMU-SEI-92-TR-20 and should be referred for details to get more insight. The rules for counting logical source statements are used to refine your logical source statement definition.

Litmus Test	Include in Counting or Not
Depending on Statement Type	
Executable	Yes
Non-Executable	
Declarations	Yes
Compiler Directives	Yes
Comments	
On their own lines	No
On lines with Source Code	No
Banners and nonblank spacers	No
Blank (Empty) Comments	No
Blank Lines	No
How produced	
Programmed==	Yes
Generated with Source Code Generators	Yes
Converted with automated translators	Yes
Copied or reused without change	Yes
Modified	Yes
Removed	No
Origin	
New Work:no Prior Existence	Yes

Check List of SEI

Litmus Test	Include in Counting or Not
New Work: no Prior Existence	Yes
Prior work: Taken or adapted from	
A previous version, build or release	Yes
COTS Commercial, off-the-shelf software, other than reuse libraries	Yes
Another product	Yes
A vendor-supplied language support library (unmodified)	No
A vendor-supplied operating system or utility (unmodified)	No
A local or modified language support library or operating system	Yes
Other Commercial library	Yes
A reuse library (Software designed for reuse)	Yes
Other Software component or library	Yes
Usage	
In or as part of primary product	Yes
External to or in support of the primary product	Yes
Delivery	
Delivered	
Delivered as Source	Yes
Delivered in compiled or executable form, but not as Source	Yes

Check List of SEI

Litmus Test	Include in Counting or Not
Not delivered	
Under configuration control	No
Not under configuration control	No
Functionality	
Operative	Yes
Inoperative (Dead, bypassed, unused, unreferenced or unaccessed)	
Functional (Intentional Dead Code, reactivated for special purpose)	Yes
Nonfunctional (unintentionally present)	No
Replications	
Master Source Statements (originals)	Yes
Physical replicates of master statements, stored in master code	Yes
Copies inserted, instantiated, or expanded when compiling or linking	No
Postproduction replicates – as in distributed, redundant or reparameterized systems	No

Check List of SEI

Litmus Test	Include in Counting or Not
Development Status	
Estimated or planned	No
Designed	No
Coded	No
Unit tests Completed	No
Integrated into components	No
Test readiness review completed	No
Software (CI) test completed	No
System test completed	Yes

Check List of SEI

If you count line of code with out using SEI check list that means it is DLOC (Delivered Lines of code).

Advantages of LOC

- Counting LOC is Simple.
- As they are final deliverables they can be used as Base Line to define companies' productivity.

Disadvantages of LOC

- They do not consider algorithmic complexity.
- Earlier estimation is difficult.
- Difficult to convince the end customer saying that project is 1000 lines of code so the cost is "XYZ" dollars.

- Need of some tool to count. No one will count huge project with 100000 lines of code.

Converting LOC to Effort

From the above SEI recommended table you get SLOC.So lets says your company has either bought or made your own tool which applies the above SEI rules and does the counting. But getting SLOC or DLOC is not enough.

> *Note: Difference between DLOC (Delivered Lines of Code) and SLOC (Source Lines of Code) is that DLOC is everything that's there in source code, it does not consider SEI table, but SLOC is after applying the SEI table rules.*

Only getting LOC count is not a meaningful thing, effort evaluation should be in one of the following form:

- Man days
- Man-Month
- Calendar Days
- Total Financial amount in dollars or local currency.

In order that SLOC or DLOC is converted to one of the following forms there are two methods:

- Base Line Technique
- COCOMO [Constructive Cost Module]

In Base Line technique we take company history. That means company can deliver consistently 600 SLOC or DLOC in one week with ten programmers. In case your company has no base lines best is with use of automated tools check your companies past project LOCs. Then take a average and use that as base line. If this is your companies first project then best way is give one of your average programmer a small project see his performance and take that as a base line.

COCOMO 81

In the upper section we have seen how to use base line technique to estimate for LOC. But still it needs history and does not take in to account if company starts getting mature with every project completed. That is in the first project it took ten days but later as reusable component was in place the same type of project is taking now eight days. The solution to this is to have some kind of parametric estimation model which will take in to account these parameters – Welcome to the world of parametric estimation (COCOMO – Constructive Cost Model).COCOMO is not a size measurement methodology. But rather it takes input as size and then applies non-functional characteristic of a project (Means "X" factor of project which we will discuss shortly). LOC (Lines of code), FP (Function points), UCP (Use Case Points) and any other counting methodology measure application size, but they do not take in to account "X" factor (Non-Functional specification) of the projects.

Some sample "X" factor of projects:

- Does the project demand innovativeness?
- Is the team interaction good or bad?
- Is it a prototype model?
- Is the company following CMMI specification?
- What is the software cycle process project is following?

For these "X" factors COCOMO has come in to picture. COCOMO is set of matured formulae with cost drivers where you can feed in your size and adjust cost drivers to account for your "x" factor.

Most of the software projects failure emerges from these "X" factors. Project of one Man / Month stretch to five man / months because programmers resigned rapidly, customer does not communicate well etc. COCOMO provides cost drivers and scale drivers to adjust the "x" factor. In this section we will see the basic COCOMO equation and the cost drivers to fine tune the "x" factor according to projects climate.

COCOMO 81 models have basically two equations:-

1. $Development\ Effort\ MM = a * KDSI^B$

If you look closely at the equation above "a" coefficient is the multiplicative complexity of project and "b" coefficient is the exponential complexity of the project.

MM (Man Month):- One month of effort by one person or one staff.

> *Note: In COCOMO there is 152 hours per person month means 19 working days with 8 working hours every day. That's excluding Saturdays and Sundays.*

KDSI: - Delivered Source Instruction (Expressed in thousand). Note this is Delivered source instruction that means with out applying SEI table check list.

In the above equation "a" and "b" depends on project development mode.

2. Effort and Development Time (TDEV) : $TDEV = 2.5 * MM^{C}$

In the above equation "c" depends on project development mode.

		Project Characteristics		
Development Mode	**Size**	**Innovation**	**Deadline/-Constraints**	**Development Environment**
Organic	Small	Little	Not Tight	Stable
Semi-Detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex Hardware /Customer Interfaced

Development Mode Table

The above two equations and table 14 (Development mode table) forms the base of COCOMO. So keep these three things and mind while reading the next section.

Basic COCOMO

Using the above given basic equation Boehm proposed three levels of COCOMO 81 mode:-

- Basic COCOMO 81
- Intermediate COCOMO 81
- Advanced COCOMO

Basic COCOMO applies table 15 to the above equation with out any details consideration. Note the basic equations discussed above section (COCOMO 81) did not change. Table 15 has the actual measure and the details of how a project is judged as organic, semi-detached or embedded are shown in table 14.

*Development Effort MM = a * KDSI^B*

*Effort and Development Time (TDEV): TDEV = 2.5 * MM ^C*

Basic COCOMO	a	b	c
Organic	2.4	1.05	0.38
Semi-Detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

Basic COCOMO project development mode table

So count the KDSI that means actual source code with out applying SEI check list and feed in the first equation and also feed in “a”,”b” and “c” depending on table 15.0 and you have the development time.

Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is necessarily limited because of its lack of factors to account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other project attributes known to have a significant influence on software costs.

Intermediate COCOMO

Basic COCOMO considered the “X” factor but is limited only to type of project. Basic COCOMO gives you flexibility to adjust the “a”, “b” and “c” coefficient depending on the project mode table. But as COCOMO started getting used in projects companies

started realizing the need for more parameterization of the formulae. So intermediate COCOMO was introduced and for fine tuning the “X” factors fifteen cost drivers where introduced. So the basic COCOMO equation remained same as basic COCOMO ($a*KDSI^B$), but in addition to it fifteen cost drivers are rated on scale of “Very Low” to “Very High” to calculate the specific effort multiplier. Second coefficient “a” is different in intermediate COCOMO .Coefficient “b” and “c” is same as Basic COCOMO.Adjustment factor is “1” for cost driver which is normal.

Basic COCOMO	a	b	c
Organic	3.2	1.05	0.38
Semi-Detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

Ratings of Development Mode Table for intermediate COCOMO

Equations remain same

The effort adjustment factor (EAF) is calculated using 15 cost drivers. EAF is the product of fifteen cost driver (See table 17 for more details).The cost drivers are grouped into four categories:

- Product
- Computer
- Personnel

- project

Each cost driver is rated on a six-point ordinal scale ranging from low to high importance. Based on the rating, an effort multiplier is determined using Table 17 (Boehm, 1981). The product of all effort multipliers is the EAF.

Cost Driver	Description	Very Low	Low	Nominal	High	Very High	Extra High
Product							
RELY	Required software reliability	0.75	0.88	1.00	1.15	1.40	-
DATA	Database size	-	0.94	1.00	1.08	1.16	-
CPLX	Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer							
TIME	Execution time constraint	-	-	1.00	1.11	1.30	1.66
STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.56
VIRT	Virtual machine volatility	-	0.87	1.00	1.15	1.30	-
TURN	Computer turnaround time	-	0.87	1.00	1.07	1.15	-
Personnel							
ACAP	Analyst capability	1.46	1.19	1.00	0.86	0.71	-
AEXP	Applications experience	1.29	1.13	1.00	0.91	0.82	-
PCAP	Programmer capability	1.42	1.17	1.00	0.86	0.70	-
VEXP	Virtual machine experience	1.21	1.10	1.00	0.90	-	-
LEXP	Language experience	1.14	1.07	1.00	0.95	-	-

17.Intermediate Cost Driver Rating Table

Cost Driver	Description	Very Low	Low	Nominal	High	Very High	Extra High
Project							
MODP	Modern programming practices	1.24	1.10	1.00	0.91	0.82	-
TOOL	Software Tools	1.24	1.10	1.00	0.91	0.83	-
SCED	Development Schedule	1.23	1.08	1.00	1.04	1.10	-

17. Intermediate Cost Driver Rating Table

Coefficient Factor “b” and multiplicative factor “a”

*Man Month Nominal = a * KDSIB*

The above basic equation of COCOMO has the exponential factor “B” which reflects abnormalities of software project. It shows the large variance in project schedule due “X” factors. For more explanation of “X” factors see section “COCOMO 81” of this chapter.

Below is the effect of variance:

B < 1.0: In this range the project productivity increases as project size increases. This range of scale can be achieved via project specific tool but this range is difficult. Remember as the project increases more there is more scope of using reusable components thus reducing the project time.

B = 1.0: This is a linear behavior range and is often used for estimating small projects especially for prototyping projects during requirement gathering of projects.

B > 1.0: This coefficient range indicates abnormality in project. This range is a red alert saying that the project can go in to problems and will need planning. Large teams having interpersonal communications and large system integration overhead can lead to such

abnormalities. Just a short note when you foresee large teams in project the main problem is technical ego which can lead to problems.

If you notice intermediate COCOMO table (Ratings of Development Mode Table for intermediate COCOMO) max value of "b" (Embedded) is 1.20 and minimum value is "1.05" (Organic). This shows embedded project can go little here and there if not properly managed , while simple organic project has value 1.05 approximately 1.0 which shows the linear behavior. (100)1.05 = 125.89 for organic and for embedded (100) 1.20= 251. 18. The difference is of 251.18 – 125.89 = 37. 188 which show the abnormalities of embedded project and organic projects.

On the contrary coefficient factor "a" captures the multiplicative factor of the project. Change in coefficient "b" will lead to huge change in effort as compared to "a".

COCOMO II

Intermediate COCOMO provided better parameterization of the equation by introducing fifteen cost drivers. But it failed in terms to account for prototyping model and different design phases in project. Second companies where demanding more detail parameterization of coefficient "a", "b" and "c". So COCOMO II provides "cost drivers" and "scale drivers" for parameterization of "a" and "b" coefficient. Looking in to these factors a new version of COCOMO that is COCOMO II was introduced.

- COCOMO II has the following model
- Application Composition Model (For early Prototyping)
- Early Design
- Post architecture model (For subsequent portions of Life cycle)

Note: - All the above mentioned three models are used according to project needs. Example application model will be used when we are making prototype of projects. Early design model will be used in early stages of project. Post architecture model is used after the detail architecture is in place of the project.

Application Composition Model

The Application Composition model is used in prototyping to resolve potential high-risk issues such as user interfaces, software/system interaction, performance, or technology maturity. Many companies follow the prototyping life cycle model.

Object points are used in order to determine size rather than LOC. Object points are also size measuring technology. And said COCOMO takes in the size and applies the "X" factor of projects. So here we take in object points rather than LOC.

First step is from the number of screen count object points complexity for screens.

	Number and Source of Data Tables		
Number of view Contained	**Total < 4**	**Total < 8**	**Total > 8**
<3	Simple	Simple	Medium
3-7	Simple	Medium	Difficult
8+	Medium	Difficult	Difficult

18. Object Points Complexity Level for Screens

If the prototyping is having reports count those object points for reports and use table 20.

	Number and Source of Data Tables		
Number of view Contained	**Total < 4**	**Total < 8**	**Total > 8**
<3	Simple	Simple	Medium
3-7	Simple	Medium	Difficult
8+	Medium	Difficult	Difficult

19. Object Points Complexity for Reports

After we count object points for reports and screen we assign the complexity weight depending on what type of object is it. For more details see table 10.

Object Type	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component	-	-	10

20.Complexity Weight for Object Points

The weighted instances are summed to provide a single object point number. If company is using r% percent of objects from the previous projects then the formula is:

$$NOP\ (New\ Object\ Points) = (object\ points)\ x\ (100 - r)\ /\ 100$$

A productivity rate (PROD) is determined using Table 21 below (Boehm et al, 1997).

Developers experience and Capability	Very Low	Low	Nominal	High	Very High
ICASE maturity and capability	Very Low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

21 Productivity Factor

Effort can then be estimated using the following equation (Boehm et al, 1997):

$$E = NOP\ /\ PROD$$

Early Design Model

The Early Design model is used to evaluate alternative software/system architectures and concepts of operation. Example customer requirement is to integrate email with mobile device. So your technical department would write small sample code that, is this possible or not.

As said first that intermediate COCOMO did include parameterization by introducing the fifteen drivers, but did not provide any way to parameterize “a” and “b” coefficient. According to software companies experiences if projects properly managed the coefficient “a” and “b” can be managed and project time can be reduced. So some sort of customization was needed in these two coeifficient. In early design model COCOMO II provided “Scaling drivers” and “Cost drivers”. Scaling drivers affected the “b” that is exponential coefficient of basic COCOMO equation (a* KSLOC *^{b}). If you see in original COCOMO 81 the “b” coefficient value is above 1.0. That means it indicates abnormality in project. But this experience has shown that his can be controlled through proper management. So now the coefficient “b” is modified and parameterized as

$$b = 0.91 + 0.01 \times SUM(Wi)$$

Where W is the set of 5 scale factors shown in Table 22 (Boehm et al, 1997).

	Rating					
Scale Factors	VL	LO	NM	HI	VH	XH
Precedentedness (PREC)	6.20	4.96	3.72	2.48	1.24	0.00
Development Flexibility (FLEX)	5.07	4.05	3.04	2.03	1.01	0.00
Architecture / Risk Resolution (RESL)	7.07	5.65	4.24	2.83	1.41	0.00
Team Cohesion (TEAM)	5.48	4.38	3.29	2.19	1.10	0.00
Process Maturity (PMAT)	7.80	6.24	4.68	3.12	1.56	0.00

22. COCOMO II scale factors.

Precedentedness:
This scale factor captures understanding of product objectives and required technology. Means was there any previous experience in making such similar products. Will the product need any innovative ideas. Depending on the understanding capability of the project the values are taken from table 22.

Development/Flexibility:
This factor expresses the degree of conformance to software requirements and external interface standards. Does the software have to follow any specification with external interface? Does the software need to adhere to standards?.

Risk resolution:
This factor is more subjective factor it Rates the integrity and understanding of the product software architecture and the number / criticality of unresolved risk items. Software projects involve risk and if they are identified and foreseen proper estimation can be analyzed. I still remember in one of our projects we had used a third party product as heart of our project. Later when we came to know about short comings of the third party we spend more time in patching it up. So depending on risk grade this factor from table 22 given up.

TEAM Cohesion:
A big team does not mean that project get fast completed. Captures the consistency of stakeholders' objectives and the willingness of all parties to work together as a team. Big team involves integration and coordination. This factor gives the scale for synchronizing projects stake holder like user , customers , programmers , designers , project managers etc etc.

Process Maturity (PMAT):
This rating is given depending on CMM (Capability maturity model) capability of the company and the KPA (Key process areas) compliance in project. In short it Measures the maturity of the software process used to produce the product; based on the Software Capability Maturity Model So again refer the table and scale the factors.

Each scale driver has a range of rating levels: Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH), and Extra High (XH). Each rating has a specific value as shown in Table 22 given up. The project's scale factors are summed and used to determine a scale exponent.

In the same way to parameterize coefficient "a" cost drivers where provided in COCOMO II. Cost drivers capture the multiplicative factor of the basic COCOMO equation (a *

KSLOC * [b]). So the “a” coefficient is modified as follows :

$$a = 2.94 * \text{Sum of one to seven cost drivers } (EM).$$

Where EM is the effort multiplier that is sum of one to seven cost drivers. Cost drivers are again “x” factor of project. Cost drivers preciseness varies according to software phase. When I am talking about phase means not complete software life cycle phase. According to COCOMO II there are two basic phases of any software development. The first phase which we are discussing now is one when the technical design is not clear and very abstract. The other phase is next which we will discuss later (Post-architecture) where the design is clear , Use Cases are already made , Class diagrams drawn, Third party component identified with it pros and cons and ready for implementation. Below are the seven cost drivers. These cost drivers are later expanded to Post-architecture cost drivers. Table 23 clears the mapping between both the phases of COCOMOII.

“Note: - the Post-Architecture cost drivers are more than the early cost drivers as in post architecture you have better understanding that at the early stages.”

Early Cost Driver	Description	Counterpart Combined Post-Architecture Cost Driver
RCPX	Product reliability and complexity	RELY,DATA,CPLX,DOCU
RUSE	Required reuse	RUSE
PDIF	Platform difficulty	TIME,STOR,PVOL
PERS	Personnel capability	ACAP,PCAP,PCON
PREX	Personnel experience	AEXP,PEXP,LTEX
FCIL	Facilities	TOOL,SITE
SCED	Schedule	SCED

23.Early Design cost drivers and Post architecture cost drivers mapping.

This cost driver includes four sub-factors:

Product Factors

Required software reliability: As the required reliability increases, more time must be spent in the critical design and testing phases.

Database size:The size of the database is important because of the effort required to generate the test data that developers will use to exercise the program.

Software Product Complexity:Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations and user interface management operations. The complexity rating is the subjective weighted average of the selected area ratings.

Required reusability:Measures the extra efforts needed to generalize software modules developed specifically for reuse in other software programs.

Documentation match to life cycle: needs Evaluated in terms of the suitability of the project's documentation of its life cycle needs.

Platform Factors

This cost driver contains three sub-factors:

Execution time constraint: Measures the approximate percentage of the available CPU execution time that the software will use in order to achieve the system's performance objectives.

Main storage constraint: Measures the amount of constraint imposed on the software due to main memory limitations in the target computer. If memory is a problem, more time must be spent on design and coding.

Platform volatility:Volatility of the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks.

Project Factors

This cost driver also includes three sub-factors:

Use of software tools :Captures the productivity impact of tools ranging from simple edit and code tools to integrated, proactive life cycle support tools.

Required development schedule:Measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. Accelerated schedules tend to produce more effort in

the earlier phases to eliminate risks and refine the architecture, and more effort in the later phases to accomplish more testing and documentation in parallel.

Multi-site development:Multi-site development is becoming the norm rather than the exception. Determining its cost-driver rating involves the assessment and judgment-based averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia).

Personnel Factors
This cost driver encompasses six sub-factors:

Analyst capability: Analysts are personnel who work on requirements, high-level design and detailed design. The major attributes that should be considered in this rating are analysis and design ability, efficiency and thoroughness and the ability to communicate and cooperate. This rating does not consider the level of experience of the analyst with the application, platform, language or tool. Another set of coefficients measures these factors.

Application experience: This rating is dependent on the level of application experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience with this type of application.

Programmer capability:Measures the ability of the programmers who will actually perform the detailed design and write/test the physical code during the coding and unit testing phases.

Personnel continuity:Measures the project's annual personnel turnover.

Platform experience:Recognizes the importance of understanding the use of more powerful platforms, including graphical user interface, database, networking and distributed middleware capabilities.

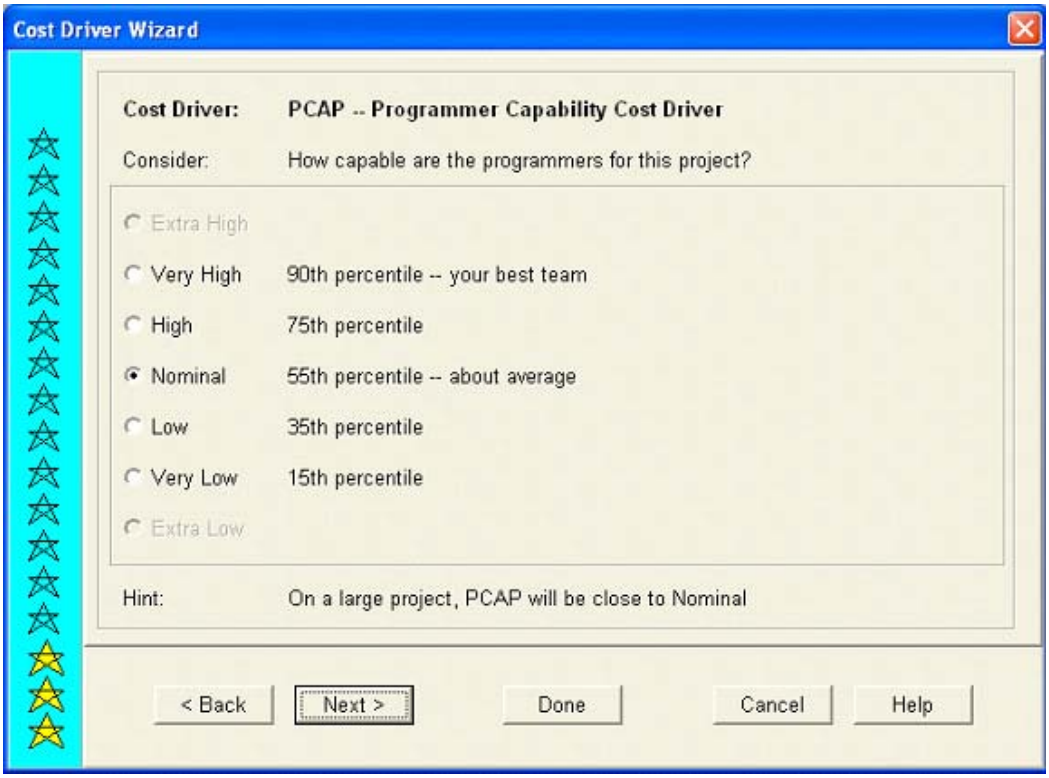
Language and tool experience:Measures the level of programming language and software tool experience of the project team developing the software system or subsystem. Software development includes the use of tools that perform requirements and design representation and analysis, configuration management, document extraction, library management, program style and formatting, consistency checking and planning and control. In addition to experience in the project's programming language, experience on the project's supporting tool set also affects the development effort.

The explanation of all the above points is now been to put actual value in table 24. Boehm et al, 1997

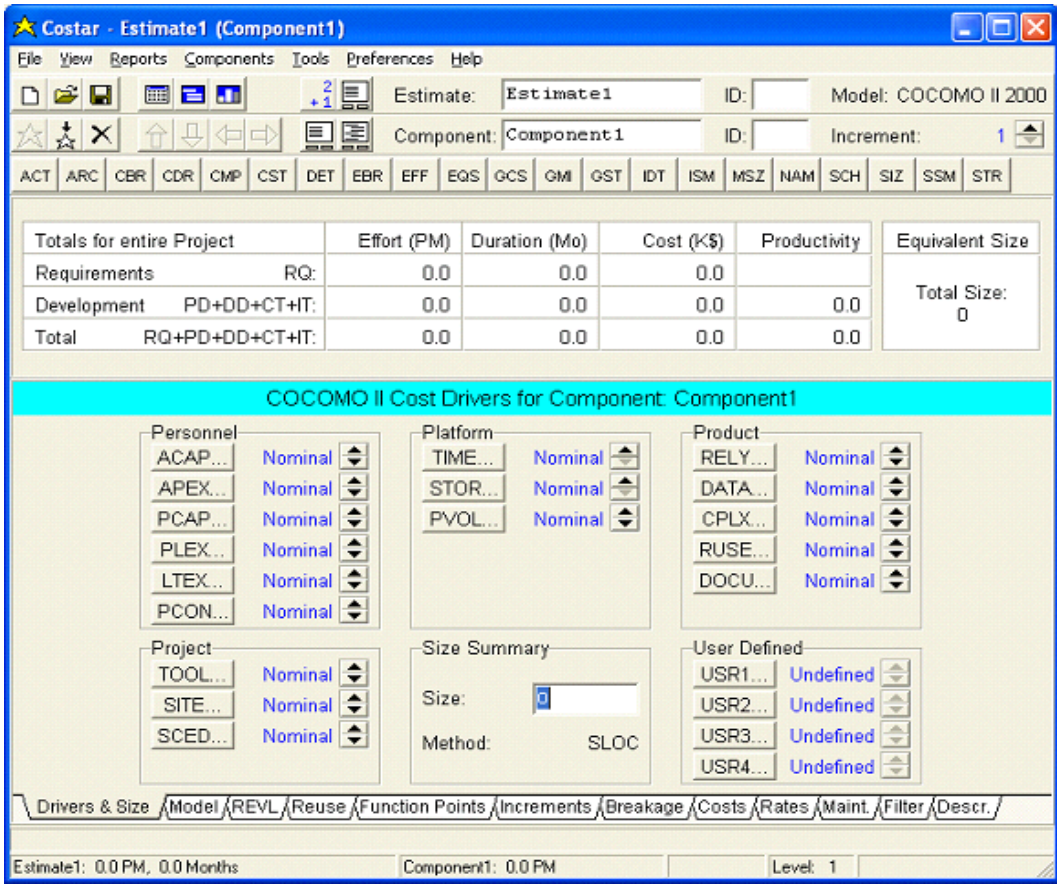
Attributes	VL	LO	NM	HI	VH	XH
Required Reliability (RELY)	0.82	0.92	1.00	1.10	1.26	-
Database Size (DATA)	-	0.90	1.00	1.14	1.28	-
Product Complexity (CPLX)	0.73	0.87	1.00	1.17	1.34	1.74
Required Reusability (RUSE)	-	0.95	1.00	1.07	1.15	1.24
Documentation Required (DOCU)	0.81	0.91	1.00	1.11	1.23	-
Execution Time Constraints (TIME)	-	-	1.00	1.11	1.29	1.63
Main Storage Constraint (STOR)	-	-	1.00	1.05	1.17	1.46
Platform Volatility (PVOL)	-	0.87	1.00	1.15	1.30	-
Analyst Capability (ACAP)	1.42	1.19	1.00	0.85	0.71	-
Applications Experience (APEX)	1.22	1.10	1.00	0.88	0.81	-
Programmer Capability (PCAP)	1.34	1.15	1.00	0.88	0.76	-
Personnel Continuity (PCON)	1.29	1.12	1.00	0.90	0.81	-
Platform Experience (PLEX)	1.19	1.09	1.00	0.91	0.85	-
Language and Tools Experience (LTEX)	1.20	1.09	1.00	0.91	0.84	-
Use of Software Tools (TOOL)	1.17	1.09	1.00	0.90	0.78	-
Multiple Site Development (SITE)	1.22	1.09	1.00	0.93	0.86	0.80
Required Development Schedule (SCED)	1.43	1.14	1.00	1.00	1.00	-

24.Full list of Post-Architecture cost drivers

"Note : Table 23 shows the mapping between the both the phases. Cost driver like RCPX in early phase is later expanded in to RELY, DATA, CPLX, DOCU. So RCPX = RELY+DATA+CPLX+DOCU. If we say RCPX is nominal then RELY = 1, DATA =1 , CPLX =1 and DOCU =1 as that is the nominal rating of the cost drivers For more details see Table 24. So if I have not shown the table of how the early design cost drivers will be calculated, just see the mapping and give rating accordingly by taking average"



25.Sample of CO-Star software showing customization of Cost Driver.



26.Sample screen of COSTAR showing full view of software.

Costar evaluation version is provided with CD feel free to install it and use it, it's in "COCOMO" folder of the CD.

The Early Design model equation is same as the basic equation defined in upper basic COCOMO equation but the "a" and "b" coefficient equations are modified:

$$E = a\,KSLOC^{b}$$

Where modified “a” and “b” coefficients are as follows (brief explanation of both coefficients is given in this section above):

*a= 2.94 * Sum of one to seven cost drivers (EM) .*

b = 0.91 + 0.01 x SUM (Wi)

Time to develop is calendar time in months.

*Time to develop = (3.67) *(E)* ^{F}

*F = 0.28 + 0.2 * (b-0.91)*

Note the Size is in KSLOC (Source lines of code which is after achieved by applying SEI check list for lines of code).

An unadjusted function point count (UFC) is used for sizing. This value is converted to LOC using tables such as those published by Jones, excerpted in Table 23 below (Jones, 1996).

“This technique is also termed as backfiring. That is converting from one measure to other measure using practical data. In this scenario we are converting from Unadjusted function points to LOC. Practical experiences has shown that very weird result come out. So beware of using backfiring technologies.”

Language	Level	Min	Mode	Max
Machine Language	0.10	-	640	-
Assembly	1.00	237	320	416
C	2.50	60	128	170
RPG II	5.50	40	58	85
C++	6.00	40	55	140
Visual C++	9.50	-	34	-
PowerBuilder	20.00	-	16	-
Excel	57.00	-	5.5	-

27.Programming language levels and ranges of source code statements per function point

The Post-Architecture Model

The Post-Architecture model is used during the actual development and maintenance of a product. Function points or LOC can be used for sizing, with modifiers for reuse and software breakage. Boehm advocates the set of guidelines proposed by The Software Engineering Institute in counting lines of code (Park, 1992). The Post-Architecture model includes a set of 17 cost drivers (A shown in table 24 above) and a set of 5 factors determining the projects scaling component. The difference between early design and Post-architecture is the coefficient “a”. In early design it was sum of seven cost factors and in post-architecture its sum of seventeen cost drivers. Other things in equations that is Calculating Time for development, Calculation of “b” coefficient etc remain same.

$$E = a * KSLOC^{b}$$

$$a = 2.94 * \text{Sum of one to seventeen cost drivers } (EM)$$

$$b = 0.91 + 0.01 \times SUM(Wi)$$

$$\text{Time to develop} = (3.67) * (E)^{F}$$

$$F = 0.28 + 0.2 * (b - 0.91)$$

The EAF is calculated using the 17 cost drivers shown in Table 24 above (Boehm et al, 1997).

Chapter 3
Function Points

Introduction to Function Points

> *"This document contains material which has been extracted from the IFPUG Counting Practices Manual. It is reproduced in this document with the permission of IFPUG."*

Function Point Analysis was developed first by Allan J. Albrecht in the mid 1970s. It was an attempt to overcome difficulties associated with lines of code as a measure of software size, and to assist in developing a mechanism to predict effort associated with software development. The method was first published in 1979, then later in 1983. In 1984 Albrecht refined the method and since 1986, when the International Function Point User Group (IFPUG) was set up, several versions of the Function Point Counting Practices Manual have been coming out.

> *"The best way to understand any complicated system is breaking the system in to smaller subsystem and try to understand those smaller sub-systems . In Function Point you break complicated huge system into smaller systems and estimate those smaller pieces, then total up all the subsystem estimate to come up with final estimate."*

Basics of Function Points

Following are some terms used in FPA [Function Point analysis].

Application Boundary

The first step in FPA is defining boundary. There are two types of major boundaries:

- Internal Application Boundary
- External Application Boundary

I will state features of external application boundary, so that internal application boundary would be self explained.

External Application Boundary can be identified using following litmus test:

- Does it have or will have any other interface to maintain its data, which is not developed by you. Example: Your Company is developing an "Accounts Application" and at the end of accounting year, you have to report to tax department. Tax department has his own website where companies can connect and report there Tax transaction. Tax department application has other maintenance and reporting screens been developed by tax software department. These maintenance screens are used internally by the Tax department. So Tax online interface has other interface to maintain its data which is not your scope, thus we can identify Tax website reporting as External Application.
- Does your program have to go through a third party API or layer? In order your application interacts with Tax Department Application probably your code have to interact through Tax Department API.
- The best litmus test is to ask yourself do you have full access over the system. If you have full rights or command to change then its internal application boundary or else external application boundary.

Elementary Process

As said in introduction FPA is breaking huge systems in to smaller pieces and analyzing them. Software application is combination of set of elementary processes.

> *EP is smallest unit of activity that is meaningful to the user.EP must be self contained and leave the application in a consistent state.*

When elementary processes come together they form a software application.

> *Note:-Elementary process is not necessarily completely independent or can exist by itself. So we can define elementary process as small units of self contained functionality from user perspective.*

Dynamic and static elementary process

There are two types of elementary process

- Dynamic Elementary process
- Static Elementary process

Dynamic elementary process moves data from internal application boundary to external application boundary or vice-versa.

Examples of dynamic elementary process:

- Input data screen where user inputs data in to application.Data moves from the input screen inside application.
- Transaction exported in export files in XML or any other standard.
- Display reports which can come from external application boundary and internal application boundary.

Static elementary process maintains data of application either inside application boundary or in external application boundary.

Examples of static elementary process:

- In a customer maintenance screen maintaining customer data is static elementary process.

Elements of Function Points

Following are the elements of FPA.

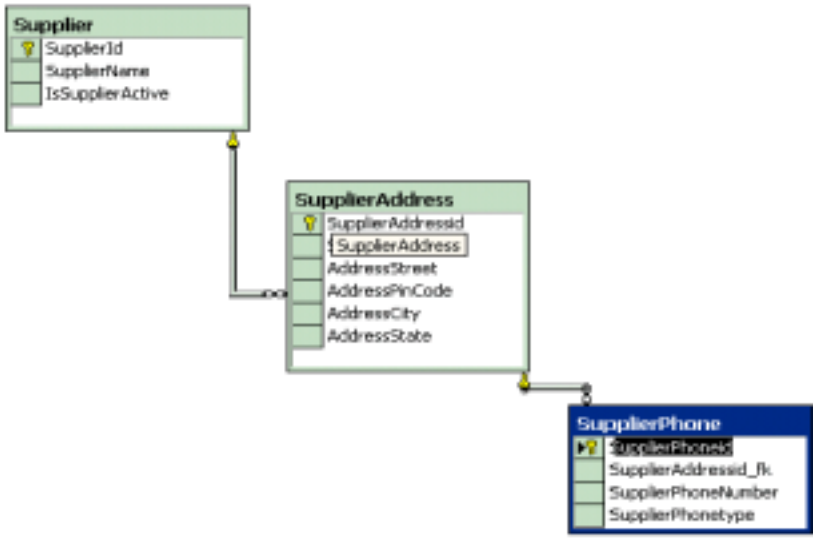
Internal Logical Files (ILF)

Following are points to be noted for ILF

- ILF are logically related data from user point of view.
- They reside in Internal Application boundary and are maintained through elementary process of application.
- ILF can have maintenance screen or probably not.

Caution: - Do not make a mistake of mapping one to one relationship between ILF and technical database design, then FPA can go very misleading. The main difference between ILF and technical database is ILF is logical view and database is physical structure (Technical Design). Example Supplier database design will have tables like Supplier, Supplier Address, SupplierPhonenumbers, but from ILF point of view its only Supplier. As logically they are all Supplier details.

28.Figure



External Interface File (EIF)

- They are logically related data from user point of view.
- EIF reside in external application boundary.

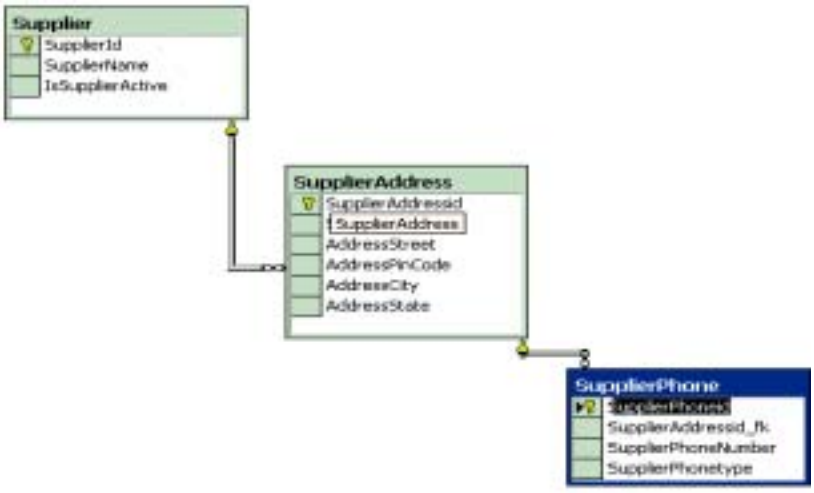
- EIF is used only for reference purpose and are not maintained by internal application.
- EIF is maintained by external application.

Record Element Type (RET)

Following are points to be noted for RET

- RET are sub-group element data of ILF or EIF.
- If there is no sub-group of ILF then count the ILF itself as one RET.
- A group of RET's within ILF are logically related. Most probably with a parent child relationship. Example: - Supplier had multiple addresses and every address can have multiple phone numbers (See detail image below which shows database diagrams).So Supplier, SupplierAddress and Supplier phone numbers are RET's.

29.Figure



Please note the whole database is one supplier ILF as all belong to one logical section.RET quantifies the relationship complexity of ILF and EIF.

DET (Data element types)

Following are the points to be noted for DET counting

- Each DET should be User recognizable. Example in the above given figure we have kept auto increment field (Supplierid) for primary key.Supplierid field from user point of view never exists at all , its only from software designing aspect , so does not qualifies for DET.
- DET should be non-recursive field in ILF. DET should not repeat in the same ILF again, it should be counted only once.
- Count foreign keys as one DET. “Supplierid” does not qualifies as DET but its relationship in “supplieraddress” table is counted as DET. So “Supplierid_fk” in supplieraddress table is counted as DET.Same folds true for “Supplieraddressid_fk”.

File Type Reference (FTR)

Following are points to be noted for FTR

- FTR is files or data referenced by a transaction.
- FTR should be ILF or EIF.So count each ILF or EIF read during process.
- If the EP is maintaining an ILF then count that as FTR.So by default you will always have one FTR in any EP.

External Input (EI)

Following are points to be noted for EI

- It’s a dynamic elementary process [For definition see “Dynamic and Static Elementary Process” Section] in which data is received from external application boundary.
  Example: - User Interaction Screens, when data comes from User Interface to Internal Application.
- EI may maintain ILF of the application, but it’s not compulsory rule. Example: - A calculator application does not maintain any data, but still the screen of calculator will be counted as EI.

- Most of time User Screens will be EI, again no hard and fast rule.Example: - An import batch process running from command line does not have screen, but still should be counted as EI as it helps passing data from External Application Boundary to Internal Application Boundary.

External Inquiry (EQ)

Following are points to be noted for EQ

- It's a dynamic elementary process in which result data is retrieved from one or more ILF or EIF.
- In this EP some input request has to enter the application boundary.
- Output results exits the application boundary.
- EQ does not contain any derived data. Derived data means any complex calculated data. Derived data is not just mere retrieval but are combined with additional formulae to generate results. Derived data is not part of ILF or EIF, they are generated on fly.
- EQ does not update any ILF or EIF.
- EQ activity should be meaningful from user perspective.
- EP is self contained and leaves the business in consistent state.
- DET and processing logic is different from other EQ's.
- Simple reports form good base as EQ.

Note:- No hard and fast rules that only simple reports are EQ's.Simple view functionality can also be counted as EQ.

External Output (EO)

Following are points to be noted for EO

- It's a dynamic elementary process in which derived data crosses from Internal Application Boundary to External Application Boundary.
- EO can update an ILF or EIF.

- Process should be the smallest unit of activity that is meaningful to end user in business.
- EP is self contained and leaves the business in a consistent state.
- DET is different from other EO's.So this ensures to us that we do not count EO's twice.
- They have derived data or formulae calculated data.

Major difference between EO and EQ is that data passes across application boundary. Example: - Exporting Accounts transaction to some external file format like XML or some other format. Which later the external accounting software can import. Second important difference is in EQ its non-derived data and EO has derived data.

General System Characteristic Section (GSC)

This section is the most important section. All the above discussed sections are counting sections. They relate only to application. But there are other things also to be considered while making software, like are you going to make it an N-Tier application, what's the performance level the user is expecting etc these other factors are called GSC. These are external factors which affect the software a lot and also the cost of it. When you submit a function point to a client, he normally will skip everything and come to GSC first. GSC gives us something called as VAF (Value Added Factor).

There are 14 points considered to come out with VAF (Value Added factor) and its associated rating table.

1) Data Communications

How many communication facilities are there to aid in the transfer or exchange of information with the application or system?

Rating	Description
0	Application is pure batch processing or a standalone PC.
1	Application is batch but has remote data entry or remote Printing.
2	Application is batch but has remote data entry and remote Printing
3	Application includes online data collection or TP (Teleprocessing) front end to a batch process or query system.
4	Application is more than a front-end, but supports only one Type of TP communications protocol.
5	Application is more than a front-end, and supports more than One type of TP communications protocol.

30.Table

2) Distributed data processing

How are distributed data and processing functions handled?

Rating	Description
0	Application does not aid the transfer of data or processing Function between components of the system.
1	Application prepares data for end user processing on another component of the system such as PC spreadsheets and PC DBMS.
2	Data is prepared for transfer, then is transferred and processed on another component of the system (not for end-user Processing).
3	Distributed processing and data transfer are online and in One direction only.
4	Distributed processing and data transfer are online and in Both directions.
5	Processing functions are dynamically performed on the most Appropriate component of the system.

31. Table

3) Performance

Did the user require response time or throughput?

Rating	Description
0	No specialperformance requirements were stated by the User.
1	Performance and design requirements were stated and Reviewed but no specialactions were required.
2	Response time or throughput is criticalduring peak hours.No specialdesign for CPU utilization was required.Processing deadline is for the next business day.
3	Response time or throughput is criticalduring all businesshours. No specialdesign for CPU utilization was required.Processing deadline requirements with interfacing systems Are constraining.
4	In addition, stated user performance requirements are stringent enough to require performance analysis tasks in the Design phase.
5	In addition, performance analysis tools were used in thedesign, development, and/or implementation phases to meetThe stated user performance requirements.

32.Table

4) Heavily used configuration

How heavily used is the current hardware platform where the application will be executed?

Rating	Description
0	No explicit or implicit operational restrictions are included.
1	Operational restrictions do exist, but are less restrictive than a typical application. No special effort is needed to meet the Restrictions.
2	Some security or timing considerations are included.
3	Specific processor requirements for a specific piece of the Application is included.
4	Stated operation restrictions require special constraints on the application in the central processor or a dedicated Processor.
5	In addition, there are special constraints on the application in The distributed components of the system.

33.Table

5) Transaction rate

How frequently are transactions executed; daily, weekly, monthly, etc.?

Rating	Description
0	No peak transaction period is anticipated.
1	Peak transaction period (e.g., monthly, quarterly, seasonally, Annually) is anticipated.
2	Weekly peak transaction period is anticipated.
3	Daily peak transaction period is anticipated.
4	High transaction rate(s) stated by the user in the application requirements or service level agreements are high enough to Require performance analysis tasks in the design phase.
5	High transaction rate(s) stated by the user in the application requirements or service level agreements are high enough to require performance analysis tasks and, in addition, require the use of performance analysis tools in the design, Development, and/or installation phases.

35. Table

6) On-Line data entry

What percentage of the information is entered On-Line?

Rating	Description
0	All transactions are processed in batch mode.
1	1% to 7% of transactions are interactive data entry.
2	8% to 15% of transactions are interactive data entry.
3	16% to 23% of transactions are interactive data entry.
4	24% to 30% of transactions are interactive data entry.
5	More than 30% of transactions are interactive data entry.

34.Figure

7) End-user efficiency

Was the application designed for end-user efficiency? There are seven end-user efficiency factors which govern how this point is rated.

Sr no	End-user Efficiency Factor
1	Navigational aids (for example, function keys, jumps, dynamically generated menus)
2	Menus
3	Online help and documents
4	Automated cursor movement
5	Scrolling
6	Remote printing (via online transactions)
7	Preassigned function keys
8	Batch jobs submitted from online transactions
9	Cursor selection of screen data
10	Heavy use of reverse video, highlighting, colors underlining, and other indicators
11	Hard copy user documentation of online transactions
12	Mouse interface
13	Pop-up windows.
14	As few screens as possible to accomplish a business function
15	Bilingual support (supports two languages; count as four items)
16	Multilingual support (supports more than two languages; count as six items).

36.Table

Rating	Description
0	None of the above.
1	One to three of the above.
2	Four to five of the above.
3	Six or more of the above, but there are no specific user Requirements related to efficiency.
4	Six or more of the above, and stated requirements for end-user efficiency are strong enough to require design tasks forhuman factors to be included (for example, minimize keyStrokes, maximize defaults, use of templates).
5	Six or more of the above, and stated requirements for end-user efficiency are strong enough to require use of specialtools and processes to demonstrate that the objectives have Been achieved.

37.Table

8) On-Line update

How many ILF's are updated by On-Line transaction?

Rating	Description
0	None of the above.
1	Online update of one to three controlfiles is included.Volume of updating is low and recovery is easy.
2	Online update of four or more controlfiles is included.Volume of updating is low and recovery easy.
3	Online update of major internallogicalfiles is included.
4	In addition, protection against data lost is essentialand has Been specially designed and programmed in the system.
5	In addition, high volumes bring cost considerations into the Recovery process. Highly automated recovery procedures With minimum operator intervention are included.

38.Table

9) Complex processing

Does the application have extensive logical or mathematical processing?

Sr no	Complex Processing Factor
1	Sensitive control (for example, special audit processing) and/or application specific security Processing
2	Extensive logical processing
3	Extensive mathematical processing
4	Much exception processing resulting in incomplete transactions that must be processed again, for example, incomplete ATM transactions caused by TP interruption, missing data values, or failed edits
5	Complex processing to handle multiple input/output possibilities, for example, multimedia, or device independence

39.Table

Rating	Description
0	None of the above.
1	Any one of the above.
2	Any two of the above.
3	Any three of the above.
4	Any four of the above.
5	All five of the above

40.Table

10) Reusability

Was the application developed to meet one or many user's needs?

Rating	Description
0	No reusable code.
1	Reusable code is used within the application.
2	Less than 10% of the application considered more than one user's needs.
3	Ten percent (10%) or more of the application considered more than one user's needs.
4	The application was specifically packaged and/or documented to ease re-use, and the application is customized by the user at source code level.
5	The application was specifically packaged and/or documented to ease re-use, and the application is customized for use by means of user parameter maintenance.

41.Table

11) Installation ease

How difficult is conversion and installation?

Rating	Description
0	No special considerations were stated by the user, and no special setup is required for installation.
1	No special considerations were stated by the user but special setup is required for installation.
2	Conversion and installation requirements were stated by the user, and conversion and installation guides were provided and tested. The impact of conversion on the project is not considered to be important.
3	Conversion and installation requirements were stated by the user, and conversion and installation guides were provided And tested. The impact of conversion on the project is Considered to be important.
4	In addition to 2 above, automated conversion and installation Tools were provided and tested.
5	In addition to 3 above, automated conversion and installation Tools were provided and tested.

42.Table

12) Operational ease

How effective and/or automated are start-up, back up, and recovery procedures?

Rating	Description
0	No special operational considerations other than the normal Back-up procedures were stated by the user.
1-4	One, some, or all of the following items apply to the Application. Select all that apply. Each item has a point Value of one, except as noted otherwise.
	Effective start-up, back-up, and recovery processes were Provided, but operator intervention is required.
	Effective start-up, back-up, and recovery processes were provided, but no operator intervention is required (count as Two items).
	The application minimizes the need for tape mounts.
	The application minimizes the need for paper handling.
5	The application is designed for unattended operation. Unattended operation means no operator intervention is required to operate the system other than to start up or shut Down the application. Automatic error recovery is a feature Of the application.

43.Table

13) Multiple sites

Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?

Rating	Description
0	User requirements do not require considering the needs of More than one user/installation site.
1	Needs of multiple sites were considered in the design, and the application is designed to operate only under identical Hardware and software environments.
2	Needs of multiple sites were considered in the design, and the application is designed to operate only under similar Hardware and/or software environments.
3	Needs of multiple sites were considered in the design, and the application is designed to operate under different Hardware and/or software environments.
4	Documentation and support plan are provided and tested to support the application at multiple sites and the application is as described by 1 or 2.
5	Documentation and support plan are provided and tested to support the application at multiple sites and the application is as described by 3.

44.Table

14) Facilitate change

Was the application specifically designed, developed, and supported to facilitate change?.

The following characteristics can apply for the application:

Sr no	Facilitate factors
0	None of above
1	Flexible query and report facility is provided that can handle simple requests; for example,and/or logic applied to only one internal logical file (count as one item).
2	Flexible query and report facility is provided that can handle requests of average complexity,for example, and/or logic applied to more than one internal logical file (count as two items).
3	Flexible query and report facility is provided that can handle complex requests, for example,and/or logic combinations on one or more internal logical files (count as three items).
4	Business control data is kept in tables that are maintained by the user with online interactive Processes, but changes take effect only on the next business day.
5	Business control data is kept in tables that are maintained by the user with online interactive Processes and the changes take effect immediately (count as two items).

45.Table

Rating	Description
0	None of the above.
1	Any one of the above.
2	Any two of the above.
3	Any three of the above.
4	Any four of the above.
5	All five of the above.

All the above GSC are rated from 0-5.Then VAF is calculated from the equation below

VAF = 0.65 + ((sum of all GSC factor)/100).

Note: - GSC has not been accepted in software industry widely. Many software companies use Unadjusted Function point rather than adjusted. ISO has also removed GSC from its books and only kept unadjusted function points as the base for measurement. Read GSC acceptance in software industry

Rating Tables for All elements of Function Points

Below shown are look up tables which will be referred during counting.

EI Rating Table			
	Data Elements		
FTR	1 to 4	5 to 15	Greater than 15
Less than 2	3	3	4
Equal to 2	3	4	6
Greater than 2	4	6	6

47.Table

This table says that in any EI (External Input), if your DET count (Data Element) and FTR (File Type Reference) exceed these limits, then this should be the FP (Function Point). Example, if your DET (data element) exceeds >15 and FTR (File Type Reference) is greater than 2, then the Function Point count is 6. The rest down tables also show the same things. These tables will be there before us when we are doing function point count. The best is put these values in Excel with formulae so that you have to only put quantity in the appropriate section and you get the final value.

EO Rating Table			
	Data Elements		
FTR	1 to 5	6 to 19	Greater than 19
Less than 2	4	4	5
2 or 3	4	5	7
Greater than 2	5	7	7

48.Table

EQ Rating Table			
	Data Elements		
FTR	1 to 5	6 to 19	Greater than 19
Less than 2	3	3	4
2 or 3	3	4	6
Greater than 2	4	6	6

49.Table

ILF Rating Table			
	Data Elements		
RET	1 to 19	20 to 50	51 or more
1 RET	7	7	10
2 to 5	7	10	15
Greater than 6	10	15	15
EIF Rating Table			
RET	1 to 19	20 to 50	51 or more
1 RET	5	5	7
2 to 5	5	7	10
Greater than 6	7	10	10

50.Table

Steps to Count Function Points

This section will discuss the practical way of counting the FP and coming out with a Man/Days on a project.

- Counting the ILF, EIF, EI, EQ, RET, DET, FTR (this is basically all sections discussed above): This whole FP count will be called as "unadjusted function point".
- Then put rating values 0 to 5 to all 14 GSC. Adding total of all 14 GSC to come out with total VAF. Formula for VAF = 0.65 + (sum of all GSC factor/100).
- Finally, make the calculation of adjusted function point. Formula: Total function point = VAF * Unadjusted function point.
- Make estimation how many function points you will do per day. This is also called as "Performance factor".
- On basis of performance factor, you can calculate Man/Days

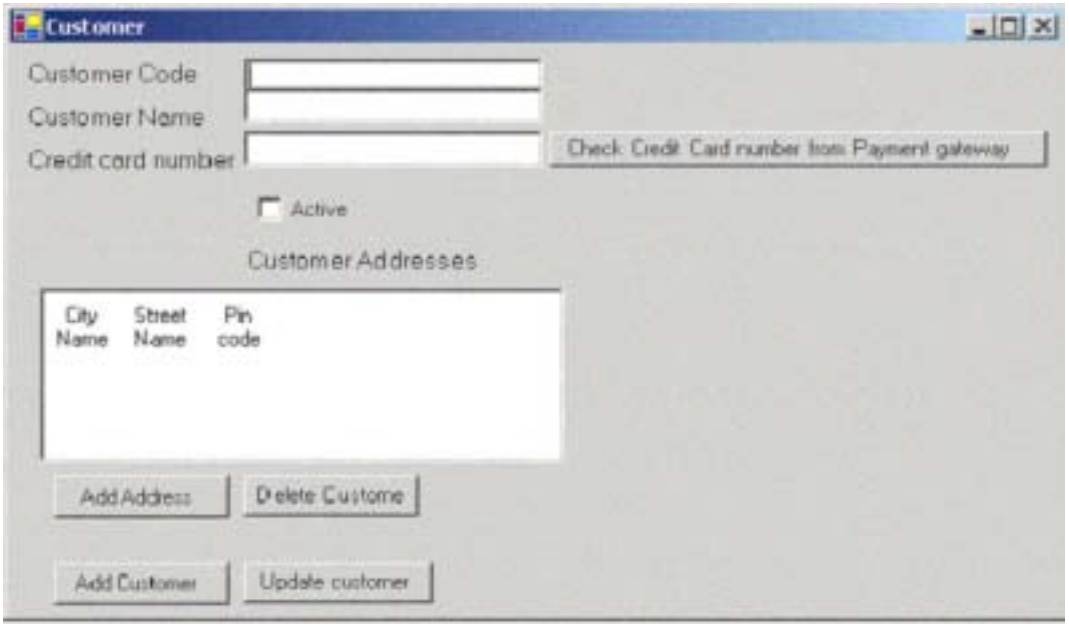
Let's try to implement these details in a sample customer project.

Sample Customer Project

We will be evaluating the customer GUI. So i will just scope what the customer GUI is all about.

Following is the scope of the customer screen:-

- Customer screen will be as shown below.
- After putting the customer code and Customer name. They will be verified credit card check.
- Credit Card check is a external system.
- Every Customer can have multiple addresses.
- Customer will have add, update functionality



51.Figure

There is one ILF in the above screen:

- The customer ILF.

There is one EIF in the above form.

- Credit Card System

Following the ILF counting rules

- ILF are logically related data from user point of view.Customer and Customer addresses belong logically to customer category.
- ILF reside in Internal Application boundary and are maintained through elementary process of application. Customer resides in inside application boundary as we have full access over it.

So hence goes the counting below for ILF

ILF Customer		
Description	**Number of DET**	**Number of RET**
There are total 9 DETs, all add and update buttons, even the credit check button, the address list box, check box active, all text boxes. There is only one RET, the customer addresses.	9	1
So according to the above ILF ranking table	Total function	7

52.Table

EIF lie outside the application boundary.

EIF Credit card Information		
Description	**Number of DET**	**Number of RET**
The credit card information referenced is EIF. Note this file is only referenced for credit card check. There's only one textbox credit card number and hence one DET is put in the side column. and RET 0. Looking at the above rating table the totalFP is 5.	1	1
So according to the above ranking table	Totalfunction	5

53.Table

Following EIF rules define in the previous sections:

- It's a dynamic elementary process [For definition see "Dynamic and Static Elementary Process" Section] in which data is received from external application boundary.Customer detail is received from external boundary that is customer input screen.
- EI may maintain ILF of the application, but it's not compulsory rule. In this sample project Customer ILF is maintained.
- So there are two EI one for Add and one from update.

Its is two because processing logic for add and update is very different.

EIAdd Customer		
Description	**Number of DET**	**Number of FTR**
There are total 9 DETs, all add and update buttons, even the credit check button, the address list box, check box active, all text boxes. There are 3 FTRs, one is the address and the second is the credit card information and third is customer him self	9	3
So according to the above ranking table	Total function	6

54.Table

EIUpdate Customer		
Description	**Number of DET**	**Number of RET**
There are total 9 DETs, all add and update buttons, even the credit check button, the address list box, check box active, all text boxes. There are 3 FTRs, one is the address and the second is the credit card information and third is customer him self.	9	3
So according to the above ranking table	Total function	6

55.Table

While counting EI I have seen many people multiplying it by 3.That means we are going to do all CRUD functionality (ADD, UPDATE, and DELETE).This is not fair as it just shows laziness of the Cost estimation team. Here the customer screen has add and update. I can say the 2 * 6 that's = 12 FP for this EI customer. But later when some refers to your FP sheet he will be completely lost.

Following are rules to recognize EO

- Data should cross application boundary and it should involve complex logic.

Credit card check process can be complex as the credit card API complexity is still not known. Data that is credit card information crosses from credit card system to Customer system

EO check credit card		
Description	Number of DET	Number of RET
One DET Credit Card number and one RET credit card itself. Note if there are no RET we count default as one. Look for RET counting rules defined in previous section.	1	1
So according to the above ranking table	Total function	4

57.Table

Following are rules to recognize EQ:

- It's a dynamic elementary process in which result data is retrieved from one or more ILF or EIF. For editing the customer we will need to retrieve the customer details.
- In this EP some input request has to enter the application boundary. The customer code is inputted from the same screen.
- Output results exits the application boundary. The customer details is displayed while the customer is editing the customer data.

- EQ does not contain any derived data. The above customer data which is displayed does not contain any complex calculations.

EQ Display Customer Edit Information		
Description	Number of DET	Number of FTR
There are 5 DETs to be retrieved Customer Code,Customer Name,Credit Card number,Active,Customer Address.Only customer details and customer address will be referenced.	5	2
So according to the above ranking table	Total function	3

58.Table

So now, let's add the total function point got from above tables:

Section Name	Function Point Counted
ILF Customer	7
EO Credit Card check system	4
EIF credit card information	5
EI Customer (Add and update)	12
EQ display customer edit information	3
Total Unadjusted Function Points	31

59.Table

So unadjusted function point comes to 31.Please note i have said this as Unadjusted function as we have not accounted other variance factor of project (Programmers leaving job, Language we will use, what architecture etc etc).

In order to make it adjusted function point, we have to calculate and tabulate the GSC and come out with the VAF.

GSC	Value(0-5)
Data communications	1
Distributed data processing	1
Performance	4
Heavily used configuration	0
Transaction rate	1
On-Line data entry	0
End-user efficiency	4
On-Line update	0
Complex processing	0
Reusability	3
Installation ease	4
Operationalease	4
Multiple sites	0
Facilitate change	0
Total	22

60.TableSo using formulae:

VAF = 0.65 + ((sum of all GSC factor)/100). = 0.65 + (22/100) = 0.87.

This factor affects the whole FP like anything, be very particular with this factor. So now, calculating the

Adjusted FP = VAF * Total unadjusted

FP = 0.87 * 31 = 26.97 = rounded to 27 FP.

Now we know that the complete FP for the customer GUI is 27 FP. Now calculating the efficiency factor, we say that we will complete 3 FP per day that is 9 working days. So, the whole customer GUI is of 9 working days (Note do not consider Saturday and Sundays in this). I know upper manager people will say make it 7 FP per day and over load the programmer. That's why programmer works at night.

Considering SDLC (System Development Life Cycle)

This section is introduced to give an insight of different development software cycle. Quotations are heavily affected by which development cycle you follow.SDLC is overall process of developing information systems through multi-step process systems from investigation of initial requirements through analysis, design, implementation and maintenance. The days are gone when one COBOL programmer used to analyze, test and implement software systems. Systems have become complex, huge team members are involved, architects, analyst, programmers, testers, users etc. To manage this number of SDLC models have been created.

Following are popular models which are listed:-

- Waterfall Model.
- Spiral Model.
- Build and Fix model.
- Rapid prototyping Model.
- Incremental Model.

This section we will go in to fair depth of different SDLC models. As the quotation depends heavily on which type of SDLC you will follow. Example if client is not looking at quality and interested in only product rather than quality, the cost of the project will be less. But at heavy sacrifice of quality.

Water Fall Model

This is the oldest model. It has sequence of stages; output of one stage becomes input of other.

Following are stages in Waterfall model:-

- System Requirement: - This initial stage of the project where end user requirements are gathered and documented.
- System Design: - In this stage detail requirements, screen layout, business rules, process diagram, pseudo code and other documentations are prepared. This is first step in technical phase.
- Implementation: - Depending on the design document actual code is written here.
- Integration and Testing: - All pieces are brought together and tested. Bugs are removed in this phase.
- Acceptance, Installation and Deployment: - This is final stage where software is put in production and runs actual business.
- Maintenance: - This is least glamorous phase which runs forever. There is detail introduction about how to put maintenance quotation later in this book. Changes, correction, addition etc are done in this phase.

Waterfall is suited for low risk in areas of User Interface and performance requirements, but high risk in budget and schedule predictability and control. Waterfall assumes that all requirements can be specified in advance. But unfortunately requirement grows and changes through various stages, so it needs feedback from one stage to other.

Spiral Model

Spiral Model removes the drawback of waterfall model, by providing emphasis to go back and reiterate earlier stages a number of times as project progresses. On broader level it's a series of short waterfall cycles, each producing an early prototype representing a part of entire project. It also helps demonstrate a Proof of Concept at early software life cycle.

Build and Fix Model

This is the most way freelancers work Write some code and keep modifying it until the customer is happy. This approach can be quite dangerous and risky.

Rapid Prototyping Model

This model is also called as Rapid Application Development. The initial emphasis is on creating prototype that looks and acts like the desired product. Prototype can be created by using tools which is different from those used for final product. Once the prototype is approved, its discarded and real software development is started from scratch. The problem with this model is that sometimes the prototype moves ahead to become the final live product which can be bad from design point of view. It's a effective model but can have higher costing than other models as you require programmers during the initial phase of the software cycle.

Incremental Model

In this model we divide products in to builds, where section of product are created and tested separately. Here errors are found in requirement phase itself, user feedback is taken for each stage and code is tested after it's written.

The main intention of introducing this section is because quotations are heavily affected by which software life cycle you follow. Because deliverables change according to SLDC model the project manager chooses for the project. Example for waterfall model we will have Requirement documents, Design documents, Source code and testing plans. But for prototyping models in addition to the documents above we will also need to deliver the rough prototype. For build and fix model we will not deliver any of the documents and the only document delivered will be source code. So according to SDLC model deliverables change and hence the quotation. This book will mainly concentrate on waterfall model and spiral model deliverables. We will divide the estimation across requirement, design, implementation (coding) and testing .In what way the estimation has to divide across all deliverables is all up to the project manager and his plans.

Phase	Percentage distribution effort
Requirements	10 % of total effort
Design Phase	20 % of total effort
Coding	60 % of total effort
Testing	10 % of total effort

61.Table

The above sample is total 100 % distribution of effort across various phases. But as said it up to the project manager to change according to scenarios.

Ok now from the above function point estimation the estimation is 7 days let's try to divide it across all phases.

Phase	Percentage distribution effort	Distribution of man/days across phases
Requirements	10 % of total effort	0.9 days
Design Phase	20 % of total effort	1.8 days
Coding	60 % of total effort	5.4 days
Testing	10 % of total effort	0.9 days
Total		9 days

62.Table

The above table shows the division of project man/days across project.Now let's put down the final quotation. Just a small comment about test cases.

Total number of Test Cases = (Function Point) raised to power of 1.2.This is as suggested from caper Jones.

*Number of Acceptance Test Cases = 1.2 * Function Points*

20-25 % of total effort can be allocated to testing phase. Test cases are non-deterministic. That means if test passes it takes “X” amount of time and if it does not then to amend it take “Y” amount of time etc etc.

Final Quotation

One programmer will sit on the project with around 1000 $ salary / Month. So his 9 days salary comes to 290 dollars approx. The upper quotation format is in its simplest format. Every company has his quotation format accordingly. So no hard and fast rule of quotation template. But still if interested http://www.microsoft.com/mac/resources/templates.aspx?pid=templates has good collection of decent templates.

XYZ SOFTWARE COMPANY				
To: TNC Limited, Western road 17, California. Quotation number: 90 Date: 1/1/2004 Customer ID :- 20090DATAENTRY				
Quantity	Description	Discount	Taxable	Total
1	Customer Project	0%	0%	290 dollars
Quotation Valid for 100 days Goods delivery date with in 25 days of half payment Quotation Prepared by :- XYZ estimation department Approved by :- SPEG department XYZ				

CustomerSampleFP.xls is provided with the CD which has all estimation details which you can refer for practical approach. If you have downloaded e-book then you will need to download cddata.zip for all files.

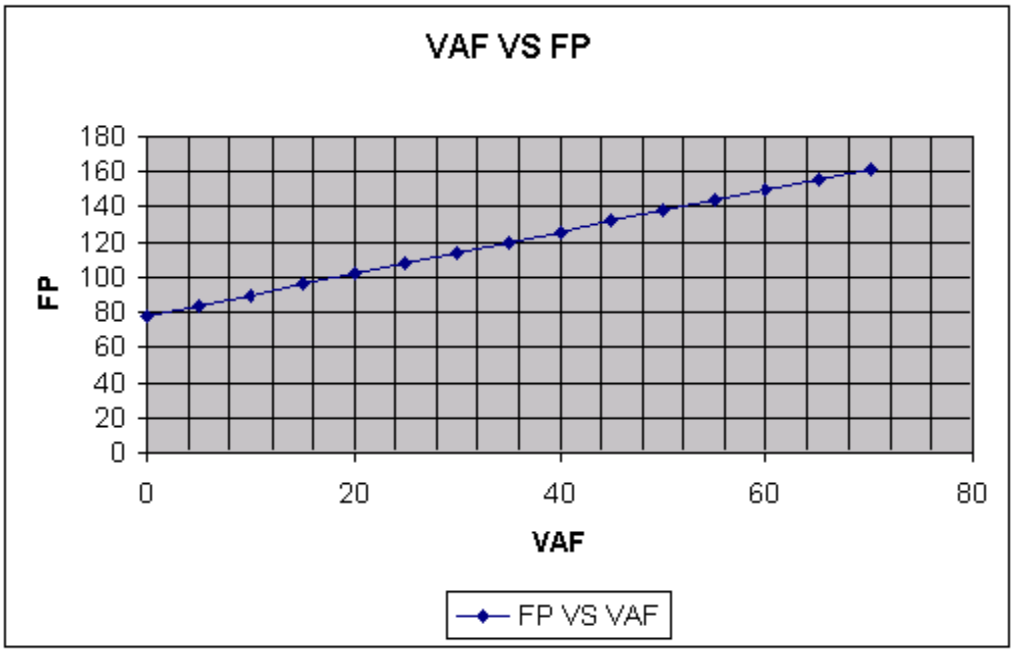
GSC Acceptance in Software industry

GSC factors have been always a controversial topic. Most of the software companies do not use GSC, rather than they base line UAFP or construct there own table depending on company project history. ISO has also adopted function point as unit of measurement, but they also use UAFP rather than AFP. Let's do a small experiment to view relationship between FP, AFP, GSC and VAF.In this experiment we will assume UAFP = 120 and then lot graph with GSC increment of five. So the formulae is VAF = 0.65 + (GS/100).

Here's the table with every five incremental values in formulae and plot.

FP	GSC
78	0
84	5
90	10
96	15
102	20
108	25
114	30
120	35
126	40
132	45
138	50
144	55
150	60
156	65
162	70

63.Table



64.Figure

The following are the observation from the table and plot:-

- Graph is linear. It also captures that nature of complexity is linear.
- If the GSC value is zero then VAF is 0.65. So the graph starts from UAFP*0.65.GSC = 35 AFP = UAFP.So the VAF = 1.
- When GSC < 35 then AFP < UAFP. That means complexity decreases.
- When GSC > 35 then AFP > UAFP. That means complexity increases.

Readers must be wondering why 0.65? There are fourteen GSC factor from zero to five. So the maximum value of VAF = 0.65 + (70/100) = 1.35. In order that VAF does not have any affect i.e. UAFP = FP VAF should be one. VAF will be one when GSC is 35 i.e. half of 70. So in order to complete value “1” value “0.65” is taken. Note value is 0.35 when GSC is 35 to complete the one factor “0.65” is required.

But following is the main problem related to GSC. GSC is applied throughout FP even when some GSC does not apply to whole function points. Here's the example to demonstrate GSC problem.

Let's take 11th GSC factor "installation ease". The project is of 100 UAFP and there is no consideration of installation previously by client so the 11th factor is zero.

GSC with installation ease with ZERO	
GSC	Value(0-5)
Data communications	1
Distributed data processing	1
Performance	4
Heavily used configuration	0
Transaction rate	1
On-Line data entry	0
End-user efficiency	4
On-Line update	0
Complex processing	0
Reusability	3
Installation ease	0
Operationalease	4
Multiple sites	0
Facilitate change	0
Total	18

65.Table

VAF = 0.65 + (18/100) = 0.83. So the FP = 100 * 0.83 = 83 Function Points. But later the client demanded for full blown installation for the project with auto updating when new version is released. So we change out GSC table with installation ease to 5.

GSC with installation ease “5”

GSC with installation ease with FIVE	
GSC	Value(0-5)
Data communications	1
Distributed data processing	1
Performance	4
Heavily used configuration	0
Transaction rate	1
On-Line data entry	0
End-user efficiency	4
On-Line update	0
Complex processing	0
Reusability	3
Installation ease	5
Operational ease	4
Multiple sites	0
Facilitate change	0
Total	23

Figure 66

So VAF = 0.65 + (23/100) = 0.88 so the FP = 100 * 0.88 = 88. The difference is of only 5 FP which from no way a proper effort estimate. To make an autoupdation for a software versioning can no way be done in 5 function points , just think downloading new version , deleting the old version , updating any database structure changes etc etc. So that's the reason GSC is not accepted in software industry. Best ways is baseline your UAFP and make your estimation on base of UAFP.

Enhancement Function Points

Major software project fail not because of programmer's or project managers but due to moody and changing customers. In one of our huge projects we had good programmers, very enthusiastic. The project started of well but customer called ten times in a day to change something or other. Believe me programmers get pissed if the customer is changing his plans every fortnight. Well from this book point of view we have to evaluate this changes which can be addition or deletion of requirements. Function point group has come out with a methodology called as "Enhancement Function Points".

Down is the formulae

*Formulae of EFP (Enhanced Function Points) = (ADD + CHGA) * VAFA + (DELFP) * VAFB*

ADD: - This is new function points added. This value is achieved by counting all new EP (Elementary process) given in change request.

CHGA: - Function points which are affected due to CR. This value is achieved by counting all DET, FTR, ILF, EI, EO and EQ which are affected. Do not count elements which are not affected.

VAFA: - This is VAF factor which is because of CR. Example previously the application was desktop and now is changed to web so the GSC factor is affected.

DELFP: - When CR is for removing some functionality this value is counted. It's rare that customer removes functionalities (at least in India), but if they ever estimator has to take note of it by counting the deleted elementary process.

VAFB: - Again removal affects Value added factor.

Once we are through with calculating enhanced function points, it time to count total function points of the application.

$$Total\ Function\ points = [UFPB + ADD + CHGA] – [CHGB – DELFP]$$

UFPB: - Function points previously counted before enhancement.

ADD: - Newly added functionality which leads to new function points after enhancements.

CHGA: - Changed function points counted after enhancements.

CHGB: - Changed function points before enhancements.

DELFP: - Deleted function points.

Enhancement function points is not covered from practical angle in this book and is left to the user's as exercise. CR (Change request) is covered in more detail in Change request chapter.

Chapter4
Use Case Points

Introduction to Use Case Modeling

Working in Ericsson in the late 1960s Ivar Jacobson devised Use-Case Documents. Thanks to Ivar Jacobson to come out with such a wonderful way of communication by using Use Case Documents. Later Use Case Documents became subset of UML. In 1994, Alistair Cockburn constructed the 'Actors and Goals conceptual model' while writing use case guides for the IBM Consulting Group. It provided guidance as how to structure and write use cases. Use Case is document which describes "WHAT" our system will do at high-level and from user perspective. Use Case does not capture "HOW" the system will do. Use Cases can also be termed as smallest unit of delivery. They capture interest of the stake holders of the system. Use Case Documents are user's vision in more understandable format. . It's the document which can stand not only for programmer, architecture but also for the stake holders. Its document which stands between the Customer and Programmers/Architecture/Business analyst/Etc.It also serves as handover when any new programmer comes in the project. Use Case document also serve as valuable input to the design of software. In short it serves in the whole life cycle of software development.

Ok let's make one good sweet definition of Use Case which comes from Ivar Jacobson's "Object-oriented Software Engineering (OOSE)"

> *"Use Case is a sequence of transactions in a system, whose task is to yield a measurable value to an individual actor of the system"*

Parts of Use Case

There is no standard in writing Use Cases. Example a simple narrative statement about a system can also be a Use Case. A detail description can also be a Use case.

Example a simple statement something like this:-

"User enters his credit card information. Credit Card information is validated with payment gateway. After credit card information is valid user is validated inside the system."

The above requirement gathered is narrative and at very high level of abstraction, but it's a Use Case. Below is a detail version of it.

1. Use enters Credit Card information.
2 Credit Card length is checked. If length is appropriate the information is sent to payment gateway or else appropriate error messages are displayed.
3. Payment gateway sends back details saying that does the credit card have enough balance to complete the transaction.
4. If amount in his account is not enough "Insufficient amount error message" is displayed.
5. If amount is sufficient to complete the transaction appropriate amount is deducted from his account.

Use Case can be any format example:

- Simple narrative.
- Tabular column
- Maintained in software.

Use Case structure and depth can vary from organization to organization and from individual to individual.

Let's try to understand various terminologies in Use Case modeling.

Actor

Actor is "A ROLE" that someone or something in the environment can play in relation to the business. Example: - Accountant is role in accounting system.

Primary Actor

Actor who initiates the transaction.

Supporting Actor

In order to achieve goals primary actor uses other actor they are called as Supporting Actor.

Transaction

An atomic set of activities that are performed either fully or not at all.

Measurable Value

The performance of the task (or transaction) has some visible, quantifiable impact on primary actor who initiated the task or transaction.

Scenarios

It's a sequence of steps that can have success and failure scenarios.

Use Case

Use Case is a sequence of transactions in a system, whose task is to yield a measurable value to an individual actor of the system.

Extension

Use Case can have linkages one use case can use others. Example Use Case "Validate User" will be used by "Order Product" to complete a transaction.

Introduction to Use Case Points

Karner identified that this document can also be used to measure and estimate effort. This section will make a walk through of karners work and give one sample example. So let's start with the definition.

Use Case Point is software sizing and measurement based on Use Case Document." Use Case Point" is based on work by gustav karner in 1993.It was written as a diploma thesis at university of linkoping This work is modification of work by Allen Albrecht on function points.

> *As said before Use Case says "WHAT" a system can do rather than "HOW" it will do it.*

We will be using Use Case documents for estimation so the "WHAT" of the Use Case document has to very precisely defined. Requirements and scope of the project becomes clearer as the project proceeds. Scope is clearer at the design phase rather than requirement phase. But quotations are prepared at very early phase so it's difficult to revert back once

the quotation is given to client. In short Use Case has to be in detail so that estimation has good accuracy. While requirements are gathered the writer of the Use Case can write in one of the following modes:-

- Summary Goal Level.
- User Goal Level
- Sub-Function Level.

Summary Goal level are the most top in hierarchy.

Summary level Use Cases run over hours, days or years.

They are long running Use Cases which is collection of Use goal level Use Cases. They provide a table of content for the User Goal level Use Cases.

User-Goal level Use Cases are the greatest interest from user perspective. It's a elementary process. As compared to Summary Goal level they are not long running.

User Goal levels are completed in one sitting (1-20 minutes).

They can have Sub-Function Use Cases below them.Sub-Function level Use Cases are smallest unit of Use Case Types. Mostly required to carry out User-Goal level Use Cases. They are mostly introduced to improve clarity or to be used by other Use Cases. Example: - Save File, Search Supplier etc.

Let's try to see a sample and take that sample to all three levels of goal describes up. Here's a simple summary-goal level Use Case for opening a new bank account. This book has taken simple template later more detail templates will be explored.

Use Case Name	Create Bank Account.
Actor	Customer
Goal	Summary GoalLevel
Steps of Use Case	Customer Fill in personaldetailform . PersonalDetailform is submitted to the scrutiny department for genuineness of the details. After scrutiny if the details are not proper the customer is intimated to fill details again. If details are proper the bank account detail is sent to customer by post.

66.Table

“Create Bank Account “Use Case is a summary level goal level as this Use Case is not completed in one sitting. The whole process will take some time (that’s minimum of one day).Estimation using summary level Use Case can give very misleading results. So we will not use summary level Use Case for estimation.

Now let’s take the same summary level goal level to a user goal level. So “Create Bank Account” will further be broken down in to Elementary process type of Use Case. User goal level Use Case is the right level of Use Case on which estimation can be done. Note: - A User goal level Use Case can be easily mapped to an elementary process of Function Points.

Use Case “Create Bank Account” is further divided in to:-

- Use Case “Customer Applies for New Account”.
- Use Case “Scrutiny of Account Details”.
- Use Case “Intimate Customer about Bank Details”.

Use Case Name	Customer Applies for New Bank Account
Actor	Customer
Goal	User-Goal Level
Steps of Use Case	Customer Fills in personal detail information like First Name, Last Name, Middle Name, Type of Bank Account (Current Account, Saving Account etc)
	After personal information of customer is filled, he has to put in address information. Customer can fill multiple addresses. Address information has the following details :Address1, Address2, Pin code, City, Country
	After Address information is filled Customer fills in the Phone information. Customer will be provided with option to fill in multiple phone numbers. Phone number has the following information :Phone Number, Phone Type (Mobile, Landline).
	After all information is filled up details are pushed to the scrutiny pool.
	Customer information is also sent to the printer so that customer has one hardcopy of the application form.

67.Table

This Use Case does not involve different scenarios and other details of the Use Case. Now the above Use Case “Customer Applies for New Bank Account “ can be completed in one sitting (Max this process will take 30 minutes on Computer Screen).So it’s a User Goal level. If you look the Data details in user goal level gives more detail picture and estimation will be more refined. The other two Use Cases “Scrutiny of new bank details” and “Intimate Customer about status” is left to users as home work. As the whole intention was to show what a user goal level use case is. So at least let’s promise our self that before starting estimation using Use Case points our Use Case should at least be user goal level Use Case.

Let's try introducing some sub-function level Use Cases for the above project. Application form of the bank also needs bank account type to be filled in. So there will be a search screen for bank account type. So we introduce one more Use Case "Select Bank Account Type".

Use Case Name	Select Bank Account Type
Actor	Customer
Goal	Sub-Function Type
Steps of Use Case	User put in account type to search.
	User is displayed with bank account types the bank has.
	User can select and view the bank account type details.
	When user double click he is returned back to "Customer Applies for New Bank Account"

68.Table

The above described is a sub-function level type Use Case. Note the last step returns back to "Customer Applies for New Bank Account" Use Case. Sub-Function Use Cases are mainly helper Use Cases, which are either extended or used. Having Sub-Function level Use Cases during estimation is very good. But sub-function level Use Cases mainly becomes clear only during design phase and not during requirement. But yes if you have it nothing like it. But at least have user goal level during requirement phase.

Steps of Use Case Point Estimation

The first four steps are steps for normal identification of Use Case and after that are steps for Use Case Points.

- Identify Actors.
- Identify Goals for the Actors.
- Define transaction and steps for the actor to achieve the goal.
- Define alternate scenarios if any present for the Use Case

- Determine the UAW (Unadjusted Actor weight): The first step is to classify all the actors in to following classification. Table 2.0 will give you clear idea of how to classify the actors. Second column is the litmus test for making decision which type of actor falls in which category. The last column provides the factor of complexity

Classification	Litmus Test to Recognize Classifications	Value/Factor
Simple actors	Simple actors are those which communicate to System through API	1
Average actors	Average actors are recognized if they following properties : 1)Actors who are interacting the system through some protocol(HTTP,FTP, or probably some user defined protocol) 2)Actor which are data store(Files, RDBMS)	2
Complex	Complex actor is interacting normally through GUI.	3

69.Table

- Determine number of UUCW (Unadjusted Use case Weight): The second step is to count Use Cases and assign weights depending on number of scenarios and number of transactions.

Use case Type	Litmus test to decide the Classification	Value/Factor
Simple	Greater than or equal to 3 transactions	5
Average	Between 4 to 7 transactions	10
Complex	Greater than 7 transactions	15

70.Table

- Determine Total UUCP (Unadjusted Use Case Point) : Total UUCP = Total UAW + Total UUCW
- Computing technical and environmental factor: Final step is to take in to account the technical complexity. All technical factors will be assigned a value from 0 to 5 depending on complexity.

	Technical factor	Weight	Description
T1	Distributed System	2	Is the system having distributed architecture or centralized architecture?
T2	Response time	1	Does the client need the system to fast? Is time response one of the important criteria?
T4	Complex Internal Processing	1	Is the Business process very complex? Like complicated accounts closing, Inventory tracking, heavy tax calculation etc.
T5	Reusable Code	1	Do we intend to keep the reusability high? So will increase the design complexity.
T6	Installation Ease	0.5	Is client looking for installation ease? By default we get many installers which create package. But if the client is looking for some custom installation probably depending on module wise .One of our client has requirement that when the client wants to install he can choose which modules he can install. If the requirement is such that when there is a new version there should be auto installation. These factors will count when assigning value to this factor
T7	Easy use	0.5	Is user friendly at the top priority?

	Technical factor	Weight	Description
T8	Portable	2	Is the customer looking for also cross platform implementation?
T9	Easy to change	1	Is the customer looking for high customization in the future?. So that also increases the Architecture design complexity and hence this factor.
T10	Concurrent	1	Is the customer looking at large numbers of users working with locking support? This will increase the architecture complexity and hence this value.
T11	Security objectives	1	Is the Customer looking at having heavy security like SSL or have to write custom code logic for encryption.
T12	Direct access to third parties	1	Does the project depend in using third party controls? So for understanding the third-party controls and studying its pros and cons considerable effort will be required. So this factor should be rated accordingly.
T13	User training facilities	1	Will the software from user perspective be so complex that separate training has to be provided? So this factor will vary accordingly

71.Table

- Equation for Tfactor = sum(T1....T13)
- TCF (Technical Complexity Factor): TCF = 0.6 + (0.01 * Tfactor).
- EF(Environmental Factor): There are other factors like trained staff, motivation of programmers etc which has quiet a decent impact on the cost estimate

	EnvironmentalFactor	Weight	Description
E1	Familiarity with project	1.5	Are all the people working in the project familiar with domain and technical details of the project? So probably you will spend your most time in explaining them all know-how's.
E2	Application experience	0.5	How much is the application experience?
E3	Objects-oriented Experience	1	As use-case documents are inputs to Object oriented design. Its important that people on the project should have basic knowledge of OOP's concept.
E4	Lead analyst capability	0.5	How the analyst who is leading the project?. Does he have enough knowledge of the domain?

	EnvironmentalFactor	Weight	Description
E5	Motivation	1	Are the programmers motivated for working on the project? As instability in project will always lead to people leaving halfway there source code. And the hand over becomes really tough. This Factor you can put according to how software industry is going on? Example if the software market is very good put this at maximum value. As good the market more the jobs and more the programmers will jump.
E6	Stable requirements	2	Is the client clear of what he wants?. I have seen clients expectations are the most important factor in stability of requirements. If the client is of highly changing nature put this value to maximum.
E7	Part-Time Staff	-1	Are there part-time staffs in project like consultants etc?
E8	Difficult programming language	-1	How the language complexity Assembly,Vb6,c++,c etc

72.Table

- Efactor = SUM (e1...e8).
- Calculating Environmental Factor = EF = 1.4 + (-0.03 * Efactor)
- AUCP (Adjusted Use Case Points): Finally calculating the Adjusted Use case points.AUCP = UUCP * TCF * EF
- Multiplying by Man/Hours Factor: AUCP * Person/Hours/AUCP.

Karner [13] proposed a factor of 20 staff hours per use case point for a project estimate.

While sharks states that field experience has shown that effort can range from 15 to 30 hours per use case point.

Schneider and winters proposed number of staff hours per use case point depends on the environmental factors. The number of factors in E1 through E6 that are below 3 are counted and added to the number of factors in E7 through E8 that are above 3. If the total is 2 or less, the general idea is to use twenty staff hours per UCP; if the total is 3 or 4, use twenty-eight staff hours per UCP.If the number exceeds 5, it is usually recommended that changes should be made to the project so the number can be adjusted because, in this case, the risk is unacceptably high. Another possibility is to increase the number of staff hours to thirty-six per use case point.

Guide Lines for Technical Factors

This section will define guidelines how to rate from 0 – 5 the ratings for technical factors. Please note there is no official release as such for these ratings. So following is the methodology adopted by this book. From points 1 to 9 it's brought from Function Point rating table and the later tables are calibrated on my experience of using Use Case Points.

1 .Distributed System

Rating	Description
0	Application does not aid the transfer of data or processing Function between components of the system.
1	Application prepares data for end user processing on another component of the system such as PC spreadsheets and PC DBMS.
2	Data is prepared for transfer, then is transferred and processed on another component of the system (not for end-user Processing).
3	Distributed processing and data transfer are online and in One direction only.
4	Distributed processing and data transfer are online and in Both directions.
5	Processing functions are dynamically performed on the most Appropriate component of the system.

73.Table

2. Performance Objectives

Rating	Description
0	No specialperformance requirements were stated by the User.
1	Performance and design requirements were stated and Reviewed but no specialactions were required.
2	Response time or throughput is criticalduring peak hours.No specialdesign for CPU utilization was required.Processing deadline is for the next business day.
3	Response time or throughput is criticalduring all businesshours. No specialdesign for CPU utilization was required.Processing deadline requirements with interfacing systems Are constraining.
4	In addition, stated user performance requirements are stringent enough to require performance analysis tasks in the Design phase.
5	In addition, performance analysis tools were used in thedesign, development, and/or implementation phases to meetThe stated user performance requirements.

74.Table

3. End User Efficiency-

Was the application designed for end-user efficiency? There are seven end-user efficiency factors which govern how this point is rated.

Sr no	End-user Efficiency Factor
1	Navigational aids (for example, function keys, jumps, dynamically generated menus)
2	Menus
3	Online help and documents
4	Automated cursor movement
5	Scrolling
6	Remote printing (via online transactions)
7	Preassigned function keys
8	Batch jobs submitted from online transactions
9	Cursor selection of screen data
10	Heavy use of reverse video, highlighting, colors underlining, and other indicators
11	Hard copy user documentation of online transactions
12	Mouse interface
13	Pop-up windows.
14	As few screens as possible to accomplish a business function
15	Bilingual support (supports two languages; count as four items)
16	Multilingual support (supports more than two languages; count as six items).

75.Table

Rating	Description
0	None of the above.
1	One to three of the above.
2	Four to five of the above.
3	Six or more of the above, but there are no specific user Requirements related to efficiency.
4	Six or more of the above, and stated requirements for end-user efficiency are strong enough to require design tasks for human factors to be included (for example, minimize keyStrokes, maximize defaults, use of templates).
5	Six or more of the above, and stated requirements for end-user efficiency are strong enough to require use of special tools and processes to demonstrate that the objectives have Been achieved.

76.Table

4. Complex Processing

Does the application have extensive logical or mathematical processing?

Sr no	Complex Processing Factor
1	Sensitive control(for example, specialaudit processing)and/or application specific security Processing
2	Extensive logicalprocessing
3	Extensive mathematicalprocessing
4	Much exception processing resulting in incomplete transactions that must be processed again, for example, incomplete ATM transactions caused by TP interruption, missing data values, or failed edits
5	Complex processing to handle multiple input/output possibilities, for example, multimedia, or device independence

77.Table

Rating	Description
0	None of the above.
1	Any one of the above.
2	Any two of the above.
3	Any three of the above.
4	Any four of the above.
5	Allfive of the above

78.Table

5. Reusable Code

Rating	Description
0	No reusable code.
1	Reusable code is used within the application.
2	Less than 10% of the application considered more than one user's needs.
3	Ten percent (10%) or more of the application considered more than one user's needs.
4	The application was specifically packaged and/or documented to ease re-use, and the application is customized by the user at source code level.
5	The application was specifically packaged and/or documented to ease re-use, and the application is customized for use by means of user parameter maintenance.

79.Table

6. Easy to Install

Rating	Description
0	No special considerations were stated by the user, and no special setup is required for installation.
1	No special considerations were stated by the user but special setup is required for installation.
2	Conversion and installation requirements were stated by the user, and conversion and installation guides were provided and tested. The impact of conversion on the project is not considered to be important.
3	Conversion and installation requirements were stated by the user, and conversion and installation guides were provided And tested. The impact of conversion on the project is Considered to be important.
4	In addition to 2 above, automated conversion and installation Tools were provided and tested.
5	In addition to 3 above, automated conversion and installation Tools were provided and tested.

80.Table

7. Easy to Use

Rating	Description
0	No special operational considerations other than the normal Back-up procedures were stated by the user.
1-4	One, some, or all of the following items apply to the Application. Select all that apply. Each item has a point Value of one, except as noted otherwise.
	Effective start-up, back-up, and recovery processes were Provided, but operator intervention is required.
	Effective start-up, back-up, and recovery processes were provided, but no operator intervention is required (count as Two items).
	The application minimizes the need for tape mounts.
	The application minimizes the need for paper handling.
5	The application is designed for unattended operation. Unattended operation means no operator intervention is required to operate the system other than to start up or shut Down the application. Automatic error recovery is a feature Of the application.

81.Table

8. Portable

Rating	Description
0	Application should cater to only operating system.
1	Application should cater to only one type of family of operating system. Means it will cater to windows OS family example windows 2000, winnt, windows 9x etc. Application should not cater to multiple families of OS.
2	Application should cater to at least two different family of OS example Windows and Linux.
3	Application should cater to three different family of operating system.
4	Application should cater to four different family of operating system.
5	Application should cater to five different family of operating system.

82.Table

9. Easy to Change

Was the application specifically designed, developed, and supported to easy to change?. The following characteristics can apply for the application:

Sr no	Facilitate factors
0	None of above
1	Flexible query and report facility is provided that can handle simple requests; for example,and/or logic applied to only one internal logical file (count as one item).
2	Flexible query and report facility is provided that can handle requests of average complexity,for example, and/or logic applied to more than one internal logical file (count as two items).
3	Flexible query and report facility is provided that can handle complex requests, for example,and/or logic combinations on one or more internal logical files (count as three items).
4	Business control data is kept in tables that are maintained by the user with online interactive Processes, but changes take effect only on the next business day.
5	Business control data is kept in tables that are maintained by the user with online interactive Processes and the changes take effect immediately (count as two items).

83.Table

Rating	Description
0	None of the above.
1	Any one of the above.
2	Any two of the above.
3	Any three of the above.
4	Any four of the above.
5	All five of the above.

84.Table

10. Concurrent

Rating	Description
0	No Concurrency required
1	Concurrency required in 10% of project data.
2	Concurrency required in 30% of the project data.
3	Concurrency required in 50% of the project data.
4	Concurrency required in 70% of the project data.
5	Concurrency required in 100% of the project data.

85.Table

11. Special Security Features

Rating	Description
0	Security aspect is not important.
1	Simple Third party installation will take care of security.
2	For implementing security third party API's needed to be incorporated in coding. 100% of security is incorporated by using third party API's in through coding. But the API's are well known in market and easy to understand.
3	API's are not well known and needs a larger understanding curve to understand.
4	Application is combination of third party API's and custom security
5	Full security of application is incorporated by custom coding.

86.Table

12. Providing Direct Access to Third Parties

Rating	Description
0	No access needed to third party product.
1	Only 5% of functionality needed to be accessed by third party software's.
2	Only 10% of functionality needed to be accessed by third party software's.
3	Only 20% of functionality needed to be accessed by third party software's.
4	Only 50% of functionality needed to be accessed by third party software's.
5	100% of functionality needed to be accessed by third party software's.

87.Table

13. Special User Training Required

Rating	Description
0	No user training required.
1	Simple instructions are required to make user understand the system.
2	Help files are supplied to user which will be referred by user during using the software.
3	With help files user has to be provided with expert guidance in initial stage.
4	Special training needed to be provided.
5	Special training is required and certification has to be acquired in order that user is eligible to use the product.

88.Table

Guide Lines for Environmental Factors

Guide lines for environmental factors. There are no guidelines as such provided by any governing body these are all based on my personal experience and should be used at there own risk.

1. Familiarity with project

Rating	Description
0	None.
1	20% percent familiarity with project.
2	40% percent familiarity with project.
3	60% percent familiarity with project.
4	80% percent familiarity with project.
5	100% familiarity with project.

89.Table

2. Application experience

Rating	Description
0	None.
1	20% percent application experience.
2	40% percent application experience.
3	60% percent application experience.
4	80% percent application experience.
5	Expert.

90.Table

3. Objects-oriented Experience

Rating	Description
0	No prior experience in Object Oriented Concepts.
1	Only theoretical experience.
2	2 years experience in Object Oriented Concepts.
3	4 years experience in Object Oriented Concepts.
4	10 years experience in Object Oriented Concepts.
5	15 years experience in Object Oriented Concepts full expert.

91.Table

4. Lead analyst capability

Rating	Description
0	No lead analyst in the project.
1	3 years experience lead analyst in project.
2	5 years experience lead analyst in project
3	8 years experience lead analyst in project.
4	10 years experience lead analyst in project.
5	Expert with 15 years of experience as lead analyst.

92.Table

5. Motivation

Rating	Description
0	No motivation.
1	Low motivation. Team works only when directed. No special initiative from team members.
2	5% of team are high motivated and have self initiative. 90% team members work only as directed; no special initiative exists from these team members.
3	20% of team are high motivated and have self initiative. 80% team members work only as directed; no special initiative exists from these team members.
4	50% of team are high motivated and have self initiative. 50% team members work only as directed; no special initiative exists from these team members.
5	High motivation and self initiation in all members.

93.Table

6. Stable requirements

Rating	Description
0	No stability. Every meeting with customer changes around 80% deviation from original requirements.
1	Requirements changing around 60% of the original requirements.
2	Requirements changing around 40% of the original requirements.
3	Requirements changing around 20% of the original requirements.
4	Requirements changing around 5% of the original requirements.
5	Stable.

94. Table

7. Part time staff

Rating	Description
0	No part time staff.
1	10% of members are part-time staff.
2	30% of members are part-time staff.
3	50% of members are part-time staff.
4	80% of members are part-time staff.
5	100% of members are part-time staff.

95.Table

8. Difficult programming language

Rating	Description
0	Easy within one week the language can be picked up.
1	At least two week is needed to pick up the language.
2	At least one month is needed to pick up the language.
3	Special training needed for the language.
4	Special training needed for the language and need help during the project.
5	Difficult needs only experience people.

96.Table

Use Case Structure Matters

The structure of use-case matters a lot. According to (Bente Anda,Hege Dreiem,Dag I.K Sjoberg and Magne jorgensen) the following aspects of structure has an impact :

- Use Case scenarios should be non-recursive and processing logic should be different from other Use Cases.
- Role/Actor should be non-recursive.
- Minimum Use Goal level Use Case should be present.
- If two Use Cases have 60 % scenario in common try to accommodate then in one Use Case using alternate scenario. Example CRUD (Create, Read, Update and Delete) should be fitted in one Use Case using alternate scenario.

- If Use Case is exceeding more than 25 transactions split them in to two Use Cases.
- Every transaction in Use Case should add business value from User Point of view.
- Function level Use Case should be used to capture algorithmic complexity
- If two actors have 80 percent in common put them in generalization relationship and count them only once. This will increase the accurate of the estimate.
- Karner recommends that included and extending use case should not be counted. But according to (Bente Anda,Hege Dreiem,Dag I.K Sjoberg and Magne Jorgensen) have a different opinion in there practical experience. In some use cases they found include and extended use cases as essential functionalities and reducing them will reduce steps and hence the estimation.

Sample Data Entry Project

Let's start with a sample fiction project.Heres the scope of the project.

TNC company till now was using manual way of maintaining its customer database and there credit card information. Data entry operator manually validates credit card information from external payment gateway. They maintain Customer Code, Customer Name, Customer Address, Customer phone and validated Customer Credit card information in Customer registry. Customer Code is unique for a customer So TNC manually check for the validations and enters in the customer registry.TNC wants the data entry project to be automated.

TNC is looking for the following automation:

- Customer Code assigned should be checked for uniqueness automatically.
- Customer Code should not exceed 8 length.
- Credit card validation should be automatic for the current System.TNC has already given the API documentation of how to interact with the third party payment system.
- Credit card length should not exceed more than 10 length.

- Data entry operator should be able to add/update/Delete customer information.
- The database will be in the TNC head office and only data entry operators will be allowed to use the Data entry Software.
- Software should work on Windows platform. At this moment TNC has Windows 2000 client installed in all computers.

Writing use case for Sample Data Entry Project

I have used Alistair Cockburn's template for the "Use Case point" example. Use Case Template varies from person to person, project to project and organization to organization. I found Alistair's template to be complete so just took it. But there's no hard and fast rule that you have to follow this template. What will matter is what steps you write in the Use Case.

Use Case Transactions

It's an atomic set of activities that are either performed entirely or not all. What is a use case transaction and what's not just see if the transaction is adding any business value or else do not include it as a transaction.

Example the user switches on the computer, user clicks on add button or any GUI is not valid business transaction step. But example the Customer Code is validated for credit card information is a valid business transaction. Use Case points are heavily affected by the way the Actors and its transactions are identified. So Use Case Document should be written by predefined guidelines, uniformly in a project. Just take a meeting with whole project team before starting writing Use Case. As the depth of the Use Case Document will affect estimation by 40%.

Sample Customer Use Case

Applying Use Case Points

Let Start Applying Use Case Points to the upper given document.

Use Case #	DATAENTRYPROJECTCUST-1009
Action	Add New Customer
Main success scenario (or basic Flow)	1)Data entry operator receives customer information.
	2)Data entry operator enters following information Customer Code :- a)Customer Name b)Customer Address c)Customer Phone
	3)Customer Code is checked if it exists in Customer Table.
	a)If the Customer Code is existing then "Duplicate Customer Code"error is raised.
	b)If the Customer Code is more than 8 length then "Customer code length limit crossed"error is raised.
	4)After step 3 is passed ok.Data entry operator enters Credit Card information.
	a)If the credit card length is more than 10 length then "Credit card length limit crossed"error is raised.
	5)Credit card information is send to the externalpayment gateway.Appropriate APIs of the externalpayment gateway willbe used for validity.
	6)ExternalPayment Gateway returns "OK" if credit card is validated or else willreturn "NOT VALID" flag.
	7)Data entry operator then adds the customer in database.

Use Case #	DATAENTRYPROJECTCUST-1009
Action	Add New Customer
Main success scenario (or basic Flow)	1)Data entry operator receives customer information.
	2)Data entry operator enters following information Customer Code :- a)Customer Name b)Customer Address c)Customer Phone
	3)Customer Code is checked if it exists in Customer Table.
	a)If the Customer Code is existing then "Duplicate Customer Code"error is raised.
	b)If the Customer Code is more than 8 length then "Customer code length limit crossed"error is raised.
	4)After step 3 is passed ok.Data entry operator enters Credit Card information.
	a)If the credit card length is more than 10 length then "Credit card length limit crossed"error is raised.
	5)Credit card information is send to the externalpayment gateway.Appropriate APIs of the externalpayment gateway willbe used for validity.
	6)ExternalPayment Gateway returns "OK" if credit card is validated or else willreturn "NOT VALID" flag.
	7)Data entry operator then adds the customer in database.

Use Case continued in next page.....

Use Case #	DATAENTRYPROJECTCUST-1009
Alternate scenario (Extensions)	Update Existing Customer
	1)Data Entry operator enter Customer Code to retrieve the customer which has to be updated.
	2)Data Entry operator make appropriate changes to the customer information. Allsteps and business validation from to 6 of Add new Customer is repeated.
	3)Data Entry operator update the customer information.
Alternate scenario (Extensions)	Delete Existing Customer
	1)Data Entry Operator enters Customer Code to retrieve the Customer which has to be deleted.
	2)Data Entry Operator Deletes the Customer. Data Entry Operator is alerted "Are you sure you want to delete the Customer?"
	3)If the Data entry operator clicks "Yes". Then the customer is deleted from the database.
	4)If the Data entry operator click 'NO "no action is taken.

Use Case #	DATAENTRYPROJECTCUST-1009
Success Guarantee (Post conditions)	1)Customer is added to Customer Database.
	2)Customer is updated to Customer Database.
	3)Customer is deleted from Customer Database
Special Requirements (including Business rules)	If Credit Card Payment Gateway API changes the interaction of the data entry customer module will have to changed accordingly.
Technology and Data Variations List	
Frequency of occurrence	
Notes and Open Issues	

Determining Unadjusted Use Actor Weights (UAW):

In this project we have identified only one actor “Data Entry Operator”. The upper Actor (Data entry operator) is complex as data entry operator will be interacting through GUI.So UAW=3 as per table Table: 2.0.

- Determine number of UUCW (Unadjusted Use case Weight): There are 12 transactions [Adding also the alternative flows] in Table 6.0 Use Case. So the above Use Case is complex according to Table: 3.0.So referring Table: 3.0 UUCW=15.
- Now calculating the total UUCP = 15 + 3 = 18.
- Determining the technical Factor

	Technical factor	Weight	Value	Weighted Value= Weight * Value	Description
T1	Distributed System	2	1	2	Is the system having distributed architecture or centralized architecture?
T2	Response time	1	4	4	Does the client need the system to fast? Is time response one of the important criteria?
T3	End user efficiency	1	3	3	How's the ends users efficiency?
T4	Complex Internal Processing	1	2	2	Is the Business process very complex? Like complicated accounts closing, Inventory tracking, heavy tax calculation etc.
T5	Reusable Code	1	1	1	Do we intend to keep the reusability high? So will increase the design complexity.

	Technical factor	Weight	Value	Weighted Value= Weight * Value	Description
T6	Installation Ease	0.5	0	0	Is client looking for installation ease? By default we get many installers which create package. But if the client is looking for some custom installation probably depending on module wise .One of our client has requirement that when the client wants to install he can choose which modules he can install. If the requirement is such that when there is a new version there should be auto installation. These factors will count when assigning value to this factor
T7	Easy use	0.5	4	2	Is user friendly at the top priority?
T8	Portable	2	1	2	Is the customer looking for also cross platform implementation?

	Technicalfactor	Weight	Value	Weighted Value= Weight * Value	Description
T9	Easy to change	1	0	0	Is the customer looking for high customization in the future?. So that also increases the Architecture design complexity and hence this factor.
T10	Concurrent	1	0	0	Is the customer looking at large numbers of users working with locking support? This will increase the architecture complexity and hence this value.
T11	Security objectives	1	0	0	Is the Customer looking at having heavy security like SSL or have to write custom code logic for encryption.

	Technical factor	Weight	Value	Weighted Value= Weight * Value	Description
T12	Direct access to third parties	1	3	3	Does the project depend in using third party controls? So for understanding the third-party controls and studying its pros and cons considerable effort will be required. So this factor should be rated accordingly.
T13	User training facilities	1	0	0	Will the software from user perspective be so complex that separate training has to be provided? So this factor will vary accordingly
	Total			19	

98.Table

- Depending on the table calculating the Technical Factor: TCF = 0.6 + (0.01 * Tfactor) = 0.6 + (0.01 * 19) = 0.79
- Calculating environmental factor

	Environmental Factor	Weight	Value	Weighted Value= Weight * Value	Description
E1	Familiarity with project	1.5	5	7.5	Are all the people working in the project familiar with domain and technical details of the project? So probably you will spend your most time in explaining them all know-how's.
E2	Application experience	0.5	5	2.5	How much is the application experience?
E3	Objects-oriented Experience	1	5	5	As use-case documents are inputs to Object oriented design. It's important that people on the project should have basic knowledge of OOP's concept.
E4	Lead analyst capability	0.5	5	2.5	How the analyst who is leading the project?. Does he have enough knowledge of the domain?

	Environmental Factor	Weight	Value	Weighted Value= Weight * Value	Description
E5	Motivation	1	1	1	Are the programmers motivated for working on the project? As instability in project will always lead to people leaving half way there source code. And the hand over becomes really tough. This Factor you can put according to how software industry is going on? Example if the software market is very good put this at maximum value. As good the market more the jobs and more the programmers will jump.
E6	Stable requirements	2	4	8	Is the client clear of what he wants? I have seen clients expectations are the most important factor in stability of requirements. If the client is of highly changing nature put this value to maximum.
E7	Part-Time Staff	-1	0	0	Are there part-time staffs in project like consultants etc?
E8	Difficult programming language	-1	3	-3	How the language complexity Assembly,Vb6,c++,c etc
	Total			23.5	

99.Table

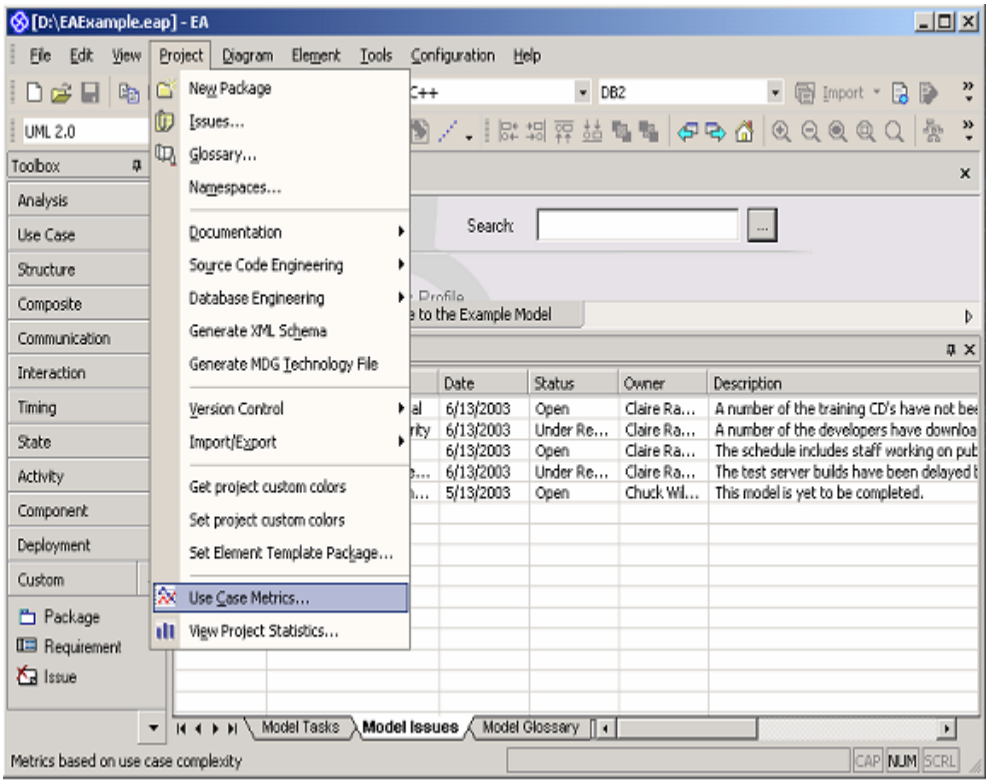
- According to [Kirsten Ribu Master of Science Thesis] Environmental factor play very important role in the estimation. A slight variation will increase the use case point by very very drastic amount. Even small adjustments of an environmental factor, for instance by half a point, can make a great difference to the estimate. Difference 3 to 2.5 increased the estimate by 4580 hours, from 10831 to 15411 hours, or 42.3 percent. This means that if the values for the environmental factors are not set correctly, there may be disastrous results -- Sources [Kirsten Ribu Master of Science Thesis] do see links below.
- Using formulae for calculating

  EF = 1.4 + (-0.03 * Efactor) = 1.4 + (-0.03 * 23.5) = 0.695
- Calculating AUCP = UUCP * TCF * EF = 18 X 0.79 X 0.695 = 9.88

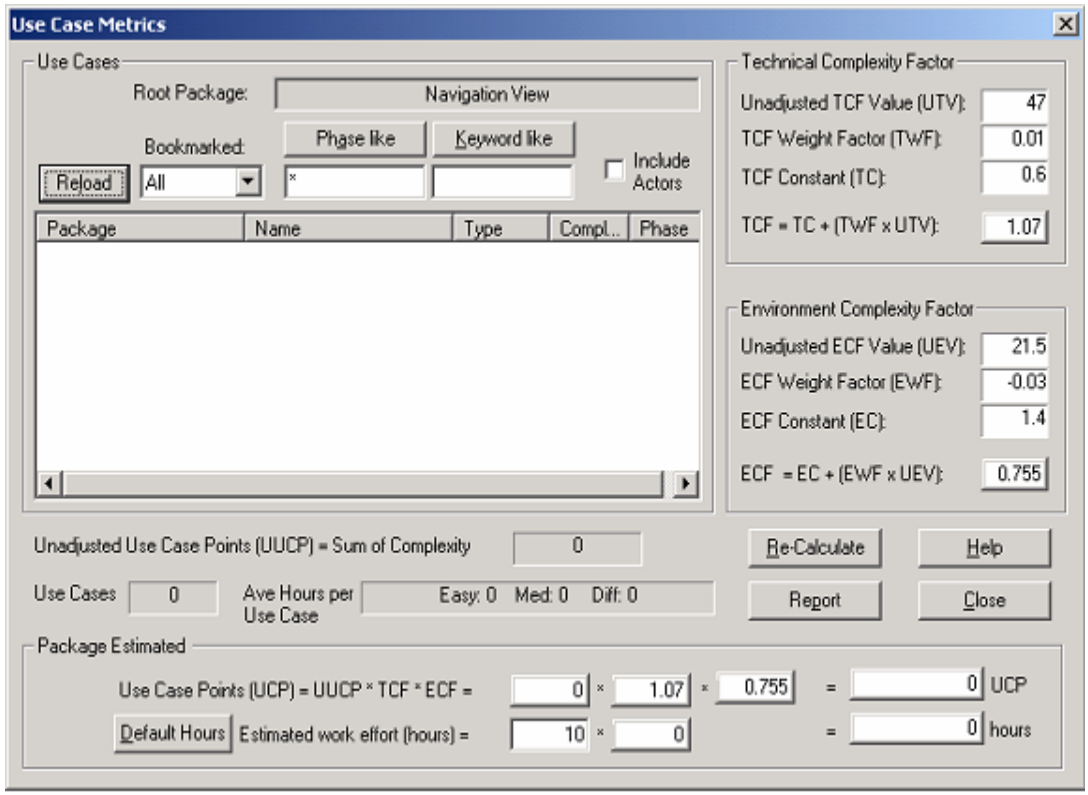
  approx = 10 Use Case Points. I have put the approximation as its only creates 3 to 4 hours of difference.
- Calculating According to Karner i.e. 20 staff hours per use case points = 10 X 20 = 200 hours for the total project.
- If programmer works for 8 hours for a day then 340/8 = 25 days.

Calculating according to Schneider and winters from e1 to e6 there are only 3 properties that are below 3.and from e7 to e8 there are none value above 3.So the total is 3 so we use 28 staff hours.10 X 28 = 280 hours. If programmer works for 8 hours then 35 days. If this step is not understood look at the steps defined in theory of use case points.

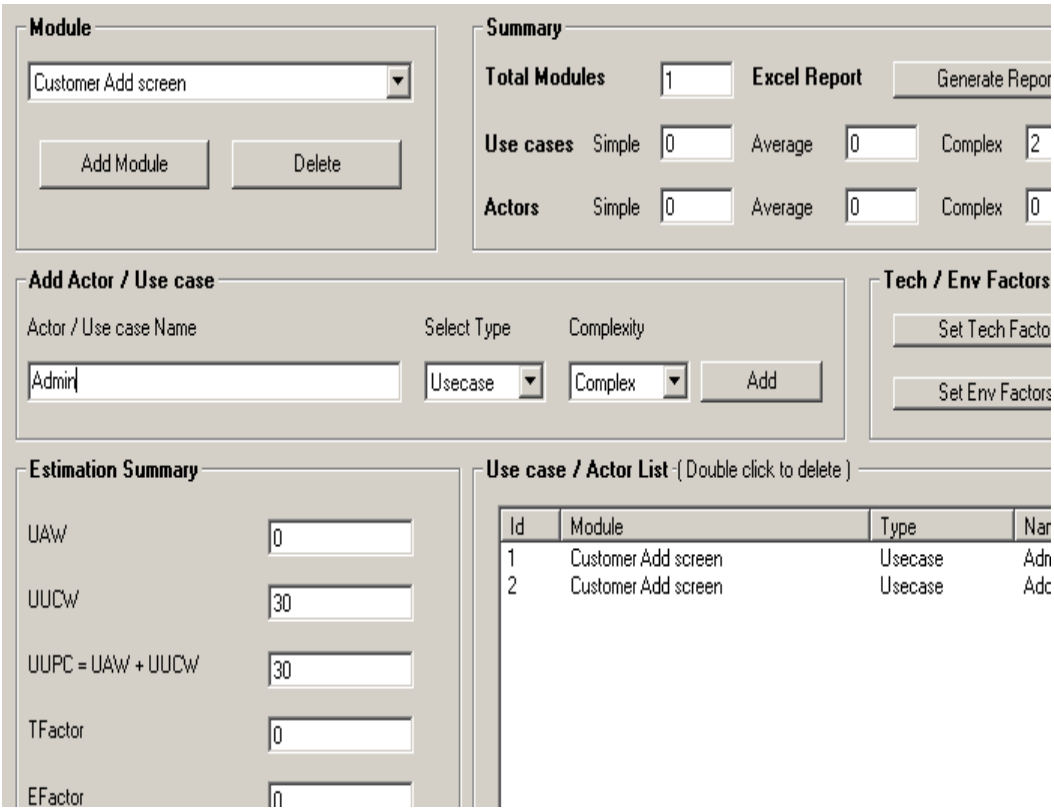
If we apply sixth sense we will find karner approach is coming to round about figure. It really depends what you want to follow karner or Schneider approach. Best is that after 2 or 3 projects what's ever is coming accurate from history take that approach. Best approach is to use excel and incorporate formulae's properly. With CD "Enterprise Architect" developed by Sparx Systems is given. It's a full blown UML tool but they have a very decent section for calculating "Use Case Metrics". The best thing with using EA is that if you are logging your project details in EA it will use the existing Use Cases in project and estimate.



100.Process to access Use Case Points section in EA



101. EA Use Case Point screen



EzEstimate sample screen one more use case point software

Final Quotation

XYZ SOFTWARE COMPANY				
To: TNC Limited, Western road 17, California. Quotation number: 90 Date: 1/1/2004 Customer ID: - 20090DATAENTRY				
Quantity	Description	Discount	Taxable	Total
1	Data Entry Module	0%	0%	840 $
Quotation Valid for 100 days Goods delivery date with in 25 days of half payment Quotation Prepared by: - XYZ estimation department Approved by :- SPEG department XYZ				

In this quotation I have taken karners value that's 25 days. One programmer will sit on the project with around 1000 $ salary. So his 25 days salary comes to 840 dollars approx. The upper quotation format is in its simplest format. Every company has his quotation format accordingly. So no hard and fast rule of quotation template. But

Still if interested http://www.microsoft.com/mac/resources/templates.aspx?pid=templates has good collection of decent templates.

In this chapter we are not showing phase wise distribution as shown in Function points it's given as exercise to readers.

Please note many people have asked this question so thought to put it in the book. Can actor rating (1 – simple, 2 – medium and 3 – complex) and Use Case rating (5 – simple, 10 – medium and 15 – complex) be customized and changed? So let's try to get what will be consequences of customizing my views are the following:-

- If the rating tables are customized two companies can not compare estimation for the same project. The basic purpose of consistent software measure will be defeated.
- UCP to FP conversion can go very inconsistent. So my suggestion is not change the ratings rather play around with productivity factor (Hours / use case) in order you are feeling the ratings are not matching.

Advantages and disadvantages of Use Case Points

Advantages of Use Case Points

- #### Automation:

Use Case document if structured properly for a company (Uniformly) we can use automation tools. In case of FP this is difficult.

Disadvantages of Use Case Points

- #### Can not be used during initial phase:

Estimations are normally done at earlier stage of projects. When we say earlier means during the first and second meet with the client. Use case documents are mostly prepared after the project sign off. So during earlier stage this document will not be available. Preparing Use Case document during first and second meet with client means wasting your resources in case you do not get the project. For initial phase of project you can use “Function points”. For function points no formal document is needed. Refer Function Points Chapter in this book.

- #### No Standard Use Case Document:

The document structure of use is not standard still. It varies not only from company to company but also from project to project. So the estimation has significant variance according to structure of Use Case documents. Even the writing matter to a large extent. And also how one does identifies use-case

and the transaction associated with it. Free textual descriptions may lead ambiguous specification [AP98].

Maintenance estimation problems:

Function point [Refer Function Points in this book] failed for maintenance projects. Use Case was derived from function points so same holds true for Use Case Points. There is a round way to prepare quotation amount for maintenance projects which is described later in this chapter.
But still its not recommended solution.

Actor identification need technical details:

In order that the actor is classified we need to know technical details like which protocol the actor will use. So estimation can be done by technical guys. Use Case is technical dependent.

Chapter 5

WBS and SMC

Why WBS and SMC

Function Points, Use Case Points and other scientific estimation ways are useful in most situation for preparing estimation and quotation. But many times you will find yourself in situation where function points, Use Case Points, LOC can not be applied.Definelty you can not give a reason to upper management saying no we can not give estimation. You have to give quotation amount with risk and assumptions.Theres where WBS with SMC approach will come to rescue. Please note this approach does not have international body or recognition, but is approach which I use when all methodologies fail. This approach is improved way of freelancer and expert judgment approach [See chapter 1 for details].So do not shy way to use this approach when you find yourself in panic.

Introduction to WBS

"Work Break Down" Structure is a result oriented family tree that captures all the work of project in a organized way. WBS was initially developed by US defense establishment and it is described in military standard (MIL-STD) 881B (25 Mar 93) as follows:

> *"A Work Break Down Structure is a product oriented family tree composed of hardware, software, services, data facilities------ [IT] displays and defines the products to be developed and/or produced and relates the elements of work to be accomplished to each other and to the end products".*

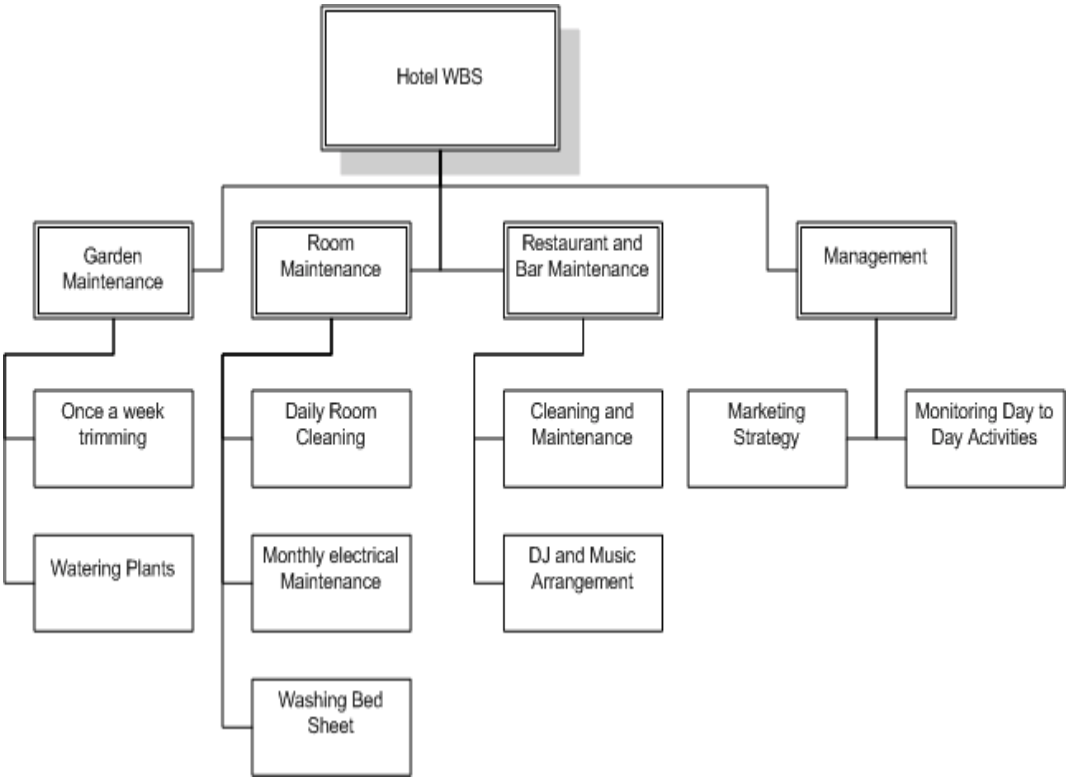
The Basic fundamental is large and complex projects are organized and broken down in smaller units. A 1000000 $ project is simply small $500 projects.

The main use of WBS is at beginning of a project for defining project scope, estimating cost and also planning. WBS breaks thousand of tasks in to smaller manageable chunks. Psychologist say human brains can comprehend 7-9 items simultaneously. A project with 1000 tasks will be difficult to understand.

So from Quotation point of view we will break project in to smaller task, group them logically and assign cost to smaller task. Later add up all smaller task cost and come out with total project cost.

Best Way to Design WBS

Everybody has designed WBS in day to day life knowingly or unknowingly. Below is simple WBS of hotel day to day process.



102.Figure

The hotel management who has prepared this list does not even know if this is a WBS structure. For him its simple department and activities “TODO” list.

The basic approach Hotel management has taken to make this WBS is as follows:

- In primary level or top level he categorized department.

- For every department work activity was logically classified.

This type of WBS is called as “Activity oriented structure “.Now management can estimate cost for every activity and later total up to come out with total running cost of Hotel.

WBS was initially defined as product oriented family tree. But to comprehend huge complexity process-oriented WBS structure where introduced.WBS structure can be built on Noun and Verbs. If the project is too activity based or process based example a simple call centre project has

- Agent Make Outbound Call
- Agent Received Inbound Call
- Agent Hangs up the phone

For this kind of project process oriented approach is best.

If project is noun based or end item approach WBS is appropriate example:

- Hardware Systems
- Server Space
- Technical Manuals
- Accounts Modules
- Invoice Module

But in real software projects the WBS will be both product and process oriented.

SMC model

Once the project is broken down in smaller components, its time to assign effort to it. There are two ways either to assign man-days or to assign rating is the task Simple, Medium or Complex. Now man-days or rating value is completely a judgment or sixth sense depending on past experience. This is the biggest disadvantage of WBS approach. The approach of the book will be following

- To apply SMC model in first iteration of estimation.
- Then using sixth sense and past experience tidy up the man-days.

Again SMC model will have table for Man-days

Rating Table

Rating	Man/Days
Simple	7
Medium	15
Complex	30

103.Table

This table is an assumption that in general according to rating this much man-day will be needed by a single person. Also note that rating table is also taking in to consideration Analysis, Design and testing phase which is also important part of software cycle. Please note if wanted according to project requirements this table can be tailored. I know many readers would not just agree to my assumption table. In software project to say that any simple task will require 7 days or a complex task will require 30 days can make estimation completely wrong. If wanted estimator can customize the rating table something like this:-

Rating	Man/Days
Very Simple	2
Simple	4
Semi-medium	6
Medium	8
Semi-Complex	15
Complex	20
Highly Complicated	30

104.Table

But for sample Tapri project we will use the Simple WBS table rating.

Tapri Online Shopping Mall

Tapri is largest retail shop all over world. They cater to various types of products right from household products to office products. Due heavy volume demand from customer they are looking at online selling of these products.Tapri has its main head office in India , but has sub head offices scattered all over the world. In every country again the retail shop are in every state and every region.

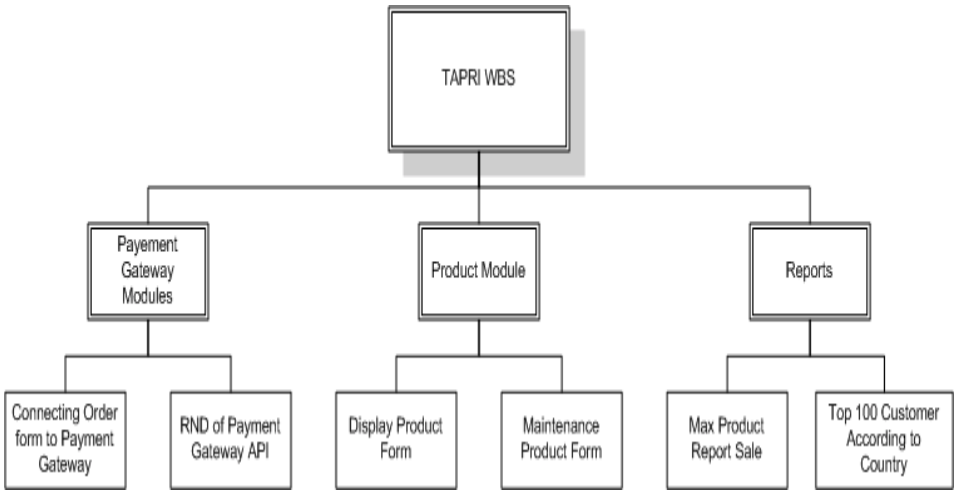
Tapri already has a static website, but has following limitation:

- New products can not be added or change by simple users. It needs a technical guy who knows HTML and FTP.
- If customer wants to order a product, there is no payment gateway at this moment. Customer see's the product and then sends email to the respective head office region for delivery of the product.

Tapri is looking for the following features in website:

- Tapri office people should be able to upload new products and update there prices.
- Tapri marketing people should be able to see the following reports
- Which is the maximum selling product?
- Top 100 customers according to country for giving them selling points.
- Tapri is not looking forward to change neither the look and feel of the current website nor any matter of the current website to be changed. Only the products page of the current website has to be made dynamic and integrated with the payment gateway.

Tapri WBS break up



105.Figure

Table of all root task and there Man-days

Root Task	Rating	Man/Days
Maintenance Product form	Medium	15
Display product Form	Simple	7
RND (Know how of the API) of payment gateway	Simple	7
Connecting Order form to Payment Gateway	Simple	7
Max Product Report Sale	Simple	7
Top 100 Customer According to Country	Simple	7
		50

106.Table

The above simple rating table is used and these results are evaluated. In this chapter we are not showing phase wise distribution as shown in Function points it's given as exercise to readers.

Chapter 6

Project Scenarios and Quotation

Getting Practical

After reading different methodologies readers must have got to a saturation point. Even I understand its time to be practical and talk figures, what's the best methodology and how to implement it in organization. If your organization is already using some methodologies its fine, you have people who have belief in software measures. But if you are in a company which has just started and you are the guy who has been given the task to implement process its tough believe me.

Following are the broader level guidelines to incorporate software measure:-

Know your productivity factor

If you are thinking by just making PowerPoint presentation non-believers of your company will be convinced forget it. The best way to go to convince them is to apply software measure methodology to past project and come out with figures and then compare them with the actual results of the project. For example you have thought to implement Use Case Points methodology in your company. So take Use Cases of the past project and start counting. After that take those results and compare with the actual project figures. Example your one of the past project is 50 UCP , but the project took one month that's approximately 30 days that is approx 1.6 UCP / DAY that becomes your productivity factor. Do the similar analysis for other past project and then make a average which is the productivity factor of the company.

Please note: - The productivity factor should be averaged with the same category project. Example if you have a project department for Microsoft, SAP, JAVA etc do not mix there productivity factors up. IN short productivity factor will be different for Microsoft, SAP, and JAVA etc. Example developing accounting project and developing a B2B site using commerce server has different productivity factor, which can not be categorized in one or averaged.

Power Point Presentation

Once you have the productivity factor from past projects and the analysis, give a nice presentation to the management.

Define your mechanism

Once your management has given you green flag, define the mechanism by which you will implement software measure. Are you going to use any software for it, or simple Excel, probably manually etc etc?. This book will use Excel to implement software measure. With this book CD is provided with Excel templates (TemplareFPandUCP.xls) so that you can kick start software measure in your company. You can also modify excel according to your company requirements.

Choose right people and Training

Once you have defined the mechanism it time to give training about the template and the fundamentals of the software measure. If there is any certification for the software measures try to pursue people to acquire the certificate. Example Function Points has certification which is conducted by www.ifpug.org. Do not ever think to appoint resource on full time basis as estimator. Believe me estimation and counting becomes boring in long term. Rather let this be shared and consecutive work load. Project manager with technical background are the right persons to assign this task. I had tried to give estimation to programmers but the effect was not impressive. Do not every give it to marketing persons, who can make estimation in terms of getting the project rather than calculating estimation.

Note: If the estimator tries to estimate from the point of view of getting project or to give less quotation it will end no where. Try to understand estimation is not minimized by not counting some elements. But rather proper project plan and automation tools. Always keep the gap between estimation and negotiation. Never mix them up. Marketing person tend to be on more negotiation mood rather than proper estimation.

Streamline your process

Once you have the template and the resource allocated for estimation. Stream line how a sales enquiry flows in your company and how the project managers will co-ordinate with the marketing department to incorporate estimation methodology.

Template Explanation (UCP and FP)

In the CD provided with this book excel templates are provided so that you can practically implement software measure in your company. This section will make a walk through the excel template (TemplateFCandUP.xls) in CD. If you have downloaded free E-book from website then you need to download ZIP file for getting these templates.

In order to open the template you will need MS excel 2002 minimum.

TemplateFPandUCP.xls has following tabs or sections:-

- Brief Description of project

  This section has free text of project scope.

- Total Estimation

  This section has complete summary of estimation template. TemplateFpandUCP.xls has provision for two types of software measure Function Point and Use Case Points (As this Project only deals with preparing quotation and estimation at initial stage, so this book is focusing on this two methodology).

“Total estimation” is divided in to three main sections

“Function Point Analysis” section gives analysis of Function Point data. None of the section is allowed to be filled, leaving the project name. They use other section and give analysis of Function Point Data.

No	Field Name	Explanation
1	Project Name	Put the name of the project whose counting has to be done. This is the only field in this section which is not read-only.
2	ILF	This is also read-only and does sum of all ILF from the ILF sheet.
3	EIF	This is also read-only field and does sum of all EIF from the EIF sheet.
4	EO	This is also read-only field and does sum of all EO from the EO sheet.
5	EQ	This is not input section and does sum of all EQ from the EQ sheet.
6	EI	This is also read-only field and does sum of all EI from the EI sheet.
7	Total Unadjusted Function Points	This is sum of ILF + EIF + EO + EQ + EI. This is section also is read-only.
8	Total GSC	This is also read-only field and takes data from GSC sheet.
9	Total Adjusted Function Points	This is also read-only field. Total unadjusted Function points * GSC. i.e. Column 7 * Column 8
10	Productivity Factor	This is also read-only field and it is the productivity factor of your company.
11	Total Man days	This is also read-only field and gives you man days depending on productivity factor column.

107.Table

- “Use Case Point Analysis” section gives analysis of Use Case Point data.

No	Field Name	Explanation
1	TotalUAW	This is read-only field and is sum of all actor weights from the “UAW” sheet.
2	TotalUUCW	This is read-only field and is sum of all Use Case weights from the “UUCW” sheet.
3	TotalUUCP	This is read-only field and is addition of UAW + UUCW
4	Adjusted UCP	This is read-only field. Its uses the “Technical Factor” and “EnvironmentalFactor” values. In short it applied the following formulae
UUCP * TCF * EF	5	Hours
Multiply it by 20 for the hours.	6	Days (By Karner Way)
That’s according to karner divide by eight. i.e Adjusted UCP / 8.		

108.Table

- “Comparison between Unadjusted Function Points and Unadjusted Use Case Points” section gives analysis of difference between both software measure methodologies. This section is specially introduced for companies who are looking to migrate from “Function Points” to “Use Case Points” and some to compare estimation with your business partner.

No	Field Name	Explanation
1	Total Difference between Unadjusted Values (UUCP-UAFP)	Read only field which uses UUCP and UAFP from above two sections. Its mathematical difference between UUCP-UAFP.
2	Factor Difference UAFP/UUCP	Read only field which uses UUCP and UAFP from above two sections. It gives multiplicative factor by dividing UAFP/UUCP.
3	Number of elements in FP	This is just a caption
4	EI	Total number of EI in this project. Note we have EI section in the first section "Function Point analysis" that's the sum and this is total number of EI in this project.
5	EO	Total EO 's in this project.
6	EQ	Total number of EQ 's in this project.
7	ILF	Total Number of ILF 's in this project.
8	EIF	Total Number of EIF 's in this project.
9	Total Elements in FP	This is total elements that's sum of Total number of EI + EO + EQ + ILF + EIF.
10	Number of Use Case in UCP	Total number of Use Case in this project.
11	Number of actors	Total number of actors in this project.
12	Total Elements in UCP	Sum of total number of Use Case and actors.
13	Total Difference in Elements (FP-UCP)	This is read-only column and is the difference between total numbers of elements in FP subtracted from Total number of elements in UCP.

109.Table

No	Field Name	Explanation
1	Functionality	Type in description of the ILF elementary process.
2	DET	Type in DET's in this elementary process.
3	RET	Type in RET's in this elementary process.
4	Value	This is read-only and does the calculation using DET and RET from Function point table.
5	DET's Considered	Short note of the DET's considered in this elementary process.
6	RET's considered	Short note of the RET's considered in this elementary process.

110.Table

- EIF section

No	Field Name	Explanation
1	Functionality	Type in description of the EIF elementary process.
2	DET	Type in DET's in this elementary process.
3	RET	Type in RET's in this elementary process.
4	Value	This is read-only and does the calculation using DET and RET from Function point table.
5	DET's Considered	Short note of the DET's considered in this elementary process.
6	RET's considered	Short note of the RET's considered in this elementary process.

111.Table

- EO section

No	Field Name	Explanation
1	Functionality	Type in description of the EO elementary process.
2	DET	Type in DET's in this elementary process.
3	FTR	Type in FTR's in this elementary process.
4	Value	This is read-only and does the calculation using DET and FTR from Function point table.
5	DET's Considered	Short note of the DET's considered in this elementary process.
6	FTR's considered	Short note of the FTR's considered in this elementary process.

112.Table

- EQ section

No	Field Name	Explanation
1	Functionality	Type in description of the EQ elementary process.
2	DET	Type in DET's in this elementary process.
3	FTR	Type in FTR's in this elementary process.
4	Value	This is read-only and does the calculation using DET and FTR from Function point table.
5	DET's Considered	Short note of the DET's considered in this elementary process.
6	FTR's considered	Short note of the FTR's considered in this elementary process.

113.Table

- EI section

No	Field Name	Explanation
1	Functionality	Type in description of the EI elementary process.
2	DET	Type in DET's in this elementary process.
3	FTR	Type in FTR's in this elementary process.
4	Value	This is read-only and does the calculation using DET and FTR from Function point table.
5	DET's Considered	Short note of the DET's considered in this elementary process.
6	FTR's considered	Short note of the FTR's considered in this elementary process.

114.Table

- GSC section

No	Field Name	Explanation
1	GSC attribute	This is read-only and is caption. In tool tip you will see the rating guidelines.
2	Definitions	This is read-only and is short description of what The GSC attribute is.
3	Value	This where you will input the rating of the GSC. All GSC is from 1-5.
4	GSC	This is read-only and applies the formulae :-0.65 + SUM (of all GSC attribute)/100.

115.Table

- Counting rule for FP

This is rules and regulation for counting FP. These are guidelines provided by www.ifpug.org.

- UAW

No	Field Name	Explanation
1	Actor Name	Enter actor name here.
2	Weight	Weight 1, 2 or 3 according to Use Case Actor rating table.
3	Description so that third person can understand	Short note of what the actor is all about.

116.Table

- UUCW

No	Field Name	Explanation
1	Use Case Name	Enter Use Case Name.
2	Number of transaction	Enter number of transaction in the Use Case included with alternate scenario.
3	Weight	This is read-only field. This field uses the Look up section and Number of transaction to calculate values.
4	Description so that third person can understand	I use this to put the transaction description here.

117.Table

- Technical Factor

No	Field Name	Explanation
1	TechnicalFactor	Read-only field, a short description of the technicalfactor.
2	Weight	This is read-only field which specifies the weight for the technicalfactor.
3	Value	Enter the rating for the technicalfactor value.
4	Weighted Value	Weight * value = weighted value. This is read-only field.
5	Description so that third person can understand	Short note of why the technicalfactor scaled in that way.

118.Table

- Environmental Factor

No	Field Name	Explanation
1	EnvironmentalFactor	Read-only field, a short description of the environmentalfactor.
2	Weight	This is read-only field which specifies the weight for the environmentalfactor.
3	Value	Enter the rating for the environmentalfactor value.
4	Weighted Value	Weight * value = weighted value. This is read-only field.
5	Description so that third person can understand	Short note of why the Environmentalfactor scaled in that way.

119.Table

- Lookup

  This is look up table for Use Case values. This section is made only for formulae purpose and should not be changed.

- Counting Rules for UCP

  There are two sections first section specifies Guidelines to write a Use Case. The second section specifies the mapping between Function points and Use Case points so that both software measures can be compared. These guidelines are all on my experience basis and have no governing body as such. So for your company purpose you can make these guidelines more mature.

Backup Project

Fast Track Company is an Accounting firm established some one year back. They have around twenty employees in there accounting section. Accounting section is maintaining huge data in excel and word. The data maintained is of prime importance to Fast Track. All employees store there backup information on file server located inside the office premises. Later in evening the network administrator backs up the file on tape. Fast Track wants the back up procedure to be automated.

Following parameters govern the way backup process is processed:-

- Start time when the backup process will start.
- Folders that has to be backed up.
- Folder where the back up will be done.
- Path where the log file will be stored.

In case of back up fails a report is generated and stored in log file.

Let's start first with writing a Use Case

Assumptions: - Trust me having a good assumption list helps out.

- Back ups are not incremental they just override existing back up's.
- Log file has only details when the back up failed and which file the back up failed.
- Requirements windows 2000 server with .NET framework installed.
- Programming language is VB.NET

Let's write the Use Case.

- Identifying the Actor: - There are two actors Network Administrator and Timer. If you are raising your eyebrows about how timer is an actor cool down. Let's revisit our definition of actors in Use Case. Actor has roles. Timer is playing a role of initiating back up process on specific time.
- Identifying Roles of Actor :-

Actor	Role
Timer	Start Timer.
Network Administrator	1) Define Backup Policy. 2) Start Backup

So we have identified two actors and three Use Cases. Let's start writing them in detail.

Use Case #	UC1001
Use Case Name	'Define Back Up Policy"
Description	This use case is used by network administrator for specifying back up parameters.
Scope and Level	
Level	User-GoalLevel
Primary and secondary actors	Network Administrator is the primary actor
Stakeholders and Interests	
Trigger	Network administrator clicks on menu " Define Backup Policy"
Preconditions	Should have admin levelrights
Assumptions	
Failed End Condition	
Action	
Main success scenario (or basic Flow)	1.Network Administrator opens define Backup Policy. Back up policy screen starts with default values for start time equal to "12:00 PM " if there is not policy defined previously. 2.Network administrator can specify the following details a)Start time if he wants to change. b)Folders which is to be backed up. This is multiple entries data. c)Backup folder where the backup will be done. d)Log files location where the log details will be stored. 3)Administrator can then start the " Back up Task" . This initiates the 'Start Timer" Use Case. 4)Administrator also has option to start the back up at that moment. In that case " Start Backup" Use Case is triggered.

Use Case #	**UC1002**
Use Case Name	"Define Back Up Policy"
Description	This Use Case defines how the timer task will work in backend.
Scope and Level	
Level	User-Goal Level
Primary and secondary actors	Primary Actor :- Timer
Secondary Actor: - Network Administrator.	
Trigger	Network administrator clicks on menu " Define Backup Policy"
Preconditions	Should have admin level rights
Assumptions	
Failed End Condition	
Action	
Main success scenario (or basic Flow)	1. Timer checks the current time with time specified by the Network Administrator. 2. If the time is Equal to or greater than the time specified by the network administrator it start the " Start Backup process" . 3. If the time is less than time specified by network administrator it goes to step 1 again of this use case,

Use Case #	UC1003
Use Case Name	"Start Backup"
Description	This Use Case will define how the actual Back Up Process works.
Scope and Level	
Level	Sub-Function Level
Primary and secondary actors	Primary Actor :- Timer, Network Administrator
Stakeholders and Interests	
Trigger	
Preconditions	
Assumptions	
Failed End Condition	
Action	
Main success scenario (or basic Flow)	1) Program picks the folder from folders specified by the network administrator to be backed up. 2) Program copies all files to the folder where the backup will be saved. 3) Program checks if any folders are remaining to be backed up. If any folder are remaining to be backed up it repeats step2 of this Use Case 4) If there are no folders remaining it returns the control to Use Case invoking it. That means either " Start Timer" or " Define Backup Policy" .
Alternate scenario (Extensions)	1. If there is failure in backup the program writes the following details to log file Filename and Date and time.

120. Table

Applying Use Case points

For Use Case Points refer TemplateFpandUCP.xls which has all details of the Use Case Point estimation of BackupProject. But here's the explanation

1. Calculating Unadjusted Actor Weights: - There are two actors Network Administrator and Timer.

Actor Name	Weight	Description so that third person can understand
Network Administrator	3	Network administrator has GUI which will be used for defining backup policy
Timer	1	It will only use simple API for timer functionality. NO database interaction or GUI present

121. Table

2. Calculating Unadjusted Use Case Weight : -

Use Case Name	Number of Transaction	Weight	Description so that third person can understand
Network Administrator	4	10	Has 4 transaction look at the Use Case
Start Timer	3	5	Refer Use Case
Start Backup	5	10	Refer Use Case

3.Technical Factor

	Technical factor	Weight	Value	Weighted Value= Weight * Value	Description
	Distributed System	2	1	2	Application copies files to main back up system
	Response time	1	0	0	No special requirement as its batch process which runs at night speed is of not prime importance
	End user efficiency	1	0	0	Simple navigation for Network administrator.
	Complex internal processing	1	2	2	There is extensive logical processing in terms of browsing through folders and files and copying them across
	Reusable code	1	0	0	This is first time the company is making this application so no reusable code is available and second there is no scope for reusability in application itself as its too small

123.Table

	Technical factor	Weight	Value	Weighted Value= Weight * Value	Description
	Installation ease	0.5	0	0	Network administrator are technicalguys who can do themselves
	Easy use	0.5	4	2	As used by network administrator no specialuser freindliness is needed.As network administrator are themself quiet technicalguys.
	Portable	2	0	0	Only should work on windows 2000 server. See assumptions for details
	Easy to change	1	0	0	No specialrequirement as such for now.
	Concurrent	1	0	0	No requirement as such
	Security objectives	1	0	0	Its in house no requirement as such
	Direct access to third parties	1	0	0	No requirement as such
	User training facilities	1	0	0	Network administrator can understand the application by themselves
	TotalTCF			0.66	

123.Table

4. Environmental Factor

	Environmental Factor	Weight	Value	Weighted Value= Weight * Value	Description so that third person can understand
	Familiarity with project	1.5	5	7.5	everybody know the project very well as the requirements are very clear
	Application experience	0.5	2	1	20% percent application experience people are available
	Object-oriented programming experience	1	2	2	Have 2 years of OOPs experience
	Lead analyst capability	0.5	2	1	Have around 2 years experience lead analyst leading the project
	Application experience	0.5	2	1	20% percent application experience people are available
	Object-oriented programming experience	1	2	2	Have 2 years of OOPs experience
	Lead analyst capability	0.5	2	1	Have around 2 years experience lead analyst leading the project
	Motivation	1	0	0	Project being small team has less motivation
	Stable requirements	2	5	10	Requirements do no changes
	Part-time Staff	-1	0	0	No part time staff
	Difficult programming language	-1	1	-1	Programming language is VB.NET
	Total			0.785	

124.Table

5. Total Unadjusted Use Case = UAW + UUCW = 29

6. Adjusted Use Case Point = 29 * 0.66 * 0.785 = 15 approx.

7. Total Man days = 37 man days approx.

Applying Function Points

1. Counting ILF

Functionality	DET	RET	Value
Backup Policy	7	2	7

Following are the DET's considered in ILF

DET 's considers	RET 's considered
Start time, Folders to be backed up, Backup folder name , Log file path , Add back folders , Start Backup , Start timer	Folders to be backed up, Backup policy

126. Table

2. Counting EIF

There are no EIF in this project

3. Counting EO

There is no EO in this project

4. Counting EQ

Functionality	DET	RET	Value
Display folders to be backed up	2	1	3
Load Backup policy Details for Edit	7	1	3

127. Table

DET's considers	RET's considered
Folder Name	Folders
Start time, Folders to be backed up, Backup folder name, Log file path, Add back folders, Start Backup, Start timer	Backup policy

128. Table

5. Counting EI

Functionality	DET	FTR	Value
Add Backup Policy	7	2	4
Update Backup Policy	7	2	4
Check timer and start back up	8	4	6

129. Table

DET's considers	FTR's considered
Start time, Folders to be backed up, Backup folder name, Log file path, Add back folders, Start Backup, Start timer	Backup, Folders to be backed up
Start time, Folders to be backed up, Backup folder name, Log file path, Add back folders, Start Backup, Start timer	Backup, Fold\ ers to be backed up
Time of the timer, Start time, Folders to be backed up, Backup folder name, Log file path, Add back folders, Start Backup, Start timer	Current time, backup policy, backup folders and log file

130. Table

6. GSC

GSC Attribute	Definitions	Value
Data communications:	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?	1
Distributed data processing	How are distributed data and processing functions handled?	5
Performance	Did the user require response time or throughput?	5
Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?	1
Transaction rate	How frequently are transactions executed; daily, weekly, monthly, etc.?	5
On-Line data entry	What percentage of the information is entered On-Line?	5
End-user efficiency	Was the application designed for end-user efficiency?	1
On-Line update	How many ILF's are updated by On-Line transaction?	5
Complex processing	Does the application have extensive logical or mathematical processing?	5
Reusability	Was the application developed to meet one or many user's needs?	5
Installation ease	How difficult is conversion and installation?	5
Operational ease	How effective and/or automated are start-up, back up, and recovery procedures?	5
Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?	5
Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?	5
GSC		1.23

131. Table

7. Total Unadjusted Adjusted Function Points = 27

8. Total Adjusted Function Points = 27 * 1.23 = 33.21

9. Productivity factor = 0.85 FP per day

10. Total Man days from function points = 33.21 * 0.85 = 39 man / days approx.

There is one excel sheets provided with CD “TemplateFpandUCP.xls” which will give you more clarity about how the counting is done.

From here onwards this book will describe the project but the estimation will be in excel file which is supplied in CD.

Researcher Training Institute Project

Researcher Training institute established in 1994, provides education training in various sector. These sector categories can be

- Computer
- Tele-communications
- Personality Improvement etc

Until date researcher training institute have been maintaining data manually. But due to volume of work and market expectation researcher institute now wants the get automised.Following are the data they are maintaining manually

- When a trainee comes for enquiry counselor enters the enquiry in enquiry sheet. This enquiry sheet is filled by the counselor during initial interview with client. Following is the data collected during enquiry :
  - Personal details (Name , First Name , Last Name , Date of Birth)
  - Address Details
  - Educational Details
  - Courses Enquiry Done
- When the enquiry is completed and the trainee is ready to take a course, course initiation form is filled. The course initiation form has following details
  - Course Name

- Trainer Name
- Course Start Date
- Course End Date

- After course is completed trainer takes a test. This test is oral and marks are entered manually. Trainer then prints a certificate for the trainee and the course is closed.

Counselor and trainer should have facility to add new categories (Computer, telecommunications etc) and new courses in the categories.

First try to estimate yourself and then compare the results from the excel. For answer of estimation see "ReseacherTrainingFPAndUCP.xls" provided in CD.

Converting Use Case Points to Function Points

Use Case Points have started becoming very popular in most software shops now. The popularity of Use Case Points is because Use Case documents are integral part of UML. Almost all software shops have adapted to UML which makes UCP easier to adopt. On other hand function points is popular because of its maturity and the user perceptiveness nature approach to estimation. Many companies are migrating to UCP because of its simplicity and availability of Use Case Documents. The question arises how does a company migrate? Companies who have estimated hundred of projects using function points and already developed a good baseline of productivity factor, how do they apply FP productivity factor to UCP, and what's the factor that can convert Function points to Use Case points.

> *Please note the main problem is not of learning or getting acquainted to UCP counting procedure, but the base lined productivity factor which is collected across years.*

Let's try to analyze how we can map elements of Use Case points to Function Points. Function Points have the following elementary process EO, EI, EQ, ILF and EIF.Use Case Points has the following Actors / Role and Use Case. In order the Function points and Use Case Points should have consistent conversion factor, this mapping should be proper.

So let's define our guidelines of mapping and comparison between Function points and Use Case Points so that we can conclude our difference factor:-

- Comparison can only be done on unadjusted values. That means comparison can be done only between Unadjusted Use Case points and Unadjusted Function points. Because after multiplying there respective non-functional factors (i.e. In case of Function Points it's GSC and in case of Use Case Points its TF and EF) the comparison will be baseless. So our comparison will be only on unadjusted values of UCP and FP for accuracy.
- Every elementary process should be mapped to Use Case or at least a group of elementary process should belong to Use Case. Every EI, EO and EQ should be mapped to Use Case.
- Every ILF and its associated EI should be mapped to Use Case.
- Every EIF and its associated EI should be mapped to Use Case.
- Group of EP's can map in one Use Case.
- Its possible EP and Use Cases will not have one to one mapping.
- Actors have no equivalent mapping in Function Points.
- Follow Use Case writing guidelines as mentioned in "Use Case Structure Matters" Section so that the use case writing is uniform and hence the conversion.

Conversion Factor for Function Points to Use Case Points

After observing around 50 plus projects the following are things observed:-

- Unadjusted Function Points is higher than Unadjusted Use Case Points.
- Unadjusted Function Points approx equal 0.9 * Unadjusted Use Case Points.

$$UAFP = (Approx)\ 0.9\ UUCP$$

These results where obtained when the "Use Case Structure Matters" (See the use case chapter section) guidelines where followed. So any deviations from those guidelines this conversion can go weird.

- Multiplication factor (i.e. 0.9) increases with increase of difference in following equation.

Number of (ILF + EIF + EO + EQ + EI) – Number of (Actor + Use Case)

So more the difference the more is the multiplication factor. Strictly speaking the difference should be in between 0.9 to 1.3 but anything more varying than this then some elements is not mapping or there is some serious error in counting.

Chapter 8

Change Request, Maintenance and Quotation

Introduction

"When there's a change there's an opportunity ..... Not always"

In software projects meeting client's expectation is an achievement, but meeting changing expectation with out financial loss is a biggest achievement. How many times do you remember client said something during requirement phase, changed during design phase and again something new popped during implementation phase? Software companies put there quotation amount depending on scope gathered during requirement phase. But scope changes during design phase and continues till implementation and deployment phase.

Here's a beautiful example of how customer can change his requirement

Day1 :-

Mr. Customer : I want a customer screen which should have following things

- Customer Code
- Customer Name
- Customer Address
- Customer Pin code
- Customer Phone number

Day 2 :-

Mr. Customer: The customer screen should also have facility to add multiple addresses.

Day 3 :-

Mr. Customer: The customer screen should also have facility to add multiple phones according to each address.

On day 1 what looked like a simple customer screen has now been heading towards customer ERP.For software companies to give these changes free of cost will lead only to loss for the project. But again saying no to the customer for changes will not be healthy for software project. This section will deal with how CR affects various phases of software cycle and hence the quotation.

Identifying Billable Change Request

First let's define change request before even deciding what's a billable change request?

> *"Change request is a change or enhancement from the current project specification ".*

Now's is the next question which type of CR you should bill.CR means change request hence forth I will be using this acronym. Changes like changing color, caption etc do they come under billable change request. So in this section will try to differentiate between billable and non-billable change request. Believe me no client understands billable CR.During initial stage of the project whatever quotation amount is given; they try only to stick to that. You try to tell the customer, sir there are extra changes, considerable effort will be required, and they will just not accept things. That's where software companies have to educate the clients, that software is not free. It cost's to pay programmers salary , infrastructure etc.Many customer just say oh its one extra field , one extra report , one extra master form etc etc.

All CR can not be billable. Project Manager should be able to differentiate between billable and non-billable CR.

There are two types of basic CR from client:

- Adding new modules in the existing project.
- Amendments in the existing project.

All new modules are 90 percent times billable. Until the client is expecting everything for free. To identify billable CR in amendments given by client is little difficult. Amendments given by customer can range from adding or removing extra field to changing whole business logic. The basic thumb rule can be if the CR by client is changing business logic it falls under billable CR category. Please note that does not mean you do not record unbillable CR, when number of unbillable CR does above certain amount, they amount to one billable CR.

Revising when a CR is billable

- When CR is for a new module.
- When CR amendments change business logic.
- When unbillable CR reached certain amount. Normally I take ten unbillable CR as one billable CR.

Source of CR

In previous section we have seen the source of CR only from end customer who will use your product. But there is no hard and fast rule that changes will emerge only from customer.

Following are sources from which CR can originate:

1 .Stake holders (Example – Customer)

2. Project team member (Project manager , project leader , programmers etc)

3. Project testing department.

4. Project reviewers

Any CR raised other than stake holder is mostly non-billable. If project team is changing something technically may be for speed purpose, for better design can not be billable. In short any changes initiated other than client is non-billable.

Put the classification diagram here

If you look at the classification diagram internal CR is initiated because of external CR.If internal CR is independent of external CR then it can be non-billable. As many times in project technical changes are made even when there is no external CR raised.

Impact Analysis

From project team point of view it only affects the source code. But source code is not the only document that is changed. To be honest if CR is big then it can change almost all document if the software life cycle. Here are some lists below.

- SRS (System requirement Specification) Document:- SRS document is prepared during requirement phase.Definetly this document will change as there is CR.
- SDD (System Design Document) :- SDD documents are prepared after requirement phase.SDD documents have technical know how of the

projects.SDD can have database Design , UML documents (Class diagrams , object diagrams , use cases etc).SDD documents are most affected due to CR SDD document change leads to source code document change.

- UTP (Unit Test Plan) Document: - UTP documents have unit test plan of the project. It is not necessary that CR will change this document, but can in most of the cases.UTP is testing plan for small units of project. Example you have "country master "maintenance module. You have made a class for its maintenance. You can test this class in isolation using tools like NUNIT.As said UTP is testing small units of project in isolation. Small units of project can be class, method, function etc.CR can affect UTP.

- SITP (System integrated Test plan) Document : As said in previous section UTP is testing smaller unit in isolation.SITP is testing all UTP's in integration example in UTP you have tested Invoice business class but later invoice is integrated with accounting and printing modules and tested , that type of testing is called as SITP.CR can affect SITP documents.

- IA: IA documents are screen shots sent to client about look and feel of user interface during initial stage of the project. There's no hard and fast rule that CR will change IA but it can in many cases.

- PMP (Project Management Plan): PMP document is the bible of project. It has the whole plan, resources allocated, dead line et.Depending on complexity this document can change heavily.

- Estimation and Quotation documents: The quotation amount will change depending on whether it's billable CR or not.

- Source code: This is the only one document which surely changes with any CR i.e. internal CR or external CR.It's very rare that company will maintain so many documents. If company is following strict process of documentation then these all documents are impacted in case company maintains it. If company is not following any process than source code is the only one document which is affected.

133.Figure

Change Request Form

Below is given a sample change request form. The below CR form is just a sample and there is no hard fast rule to follow it in this way. Many software companies maintain there CR through software. The first two sections are companies name and project name. Software companies normally fill in CR forms for bunch of CR rather than filling one CR by one. It better to maintain CR in excel as you can filter it by billable and source. Then there is approved by section which says that has client agreed to treat this as a valid CR. Please note Billable CR has no meaning if it's not approved by customer end. Then comes the CR No which is just a unique number assigned to CR so that it can be tracked. A short description followed by is it Billable CR.Then source who made the CR followed by effort estimation in Man/Days.

Change Request Form			
Company Name :- Xyz Software Company			
Project Name :- Accounting Project			
Approved by:IT department Head			
CR No	Description of CR	Billable	Source
1	Changed Color of Accounting Screen	No	Customer
2	Accounts should close according to Financial Year	Yes	Customer
3	Add check box extra field in voucher form print yes or no	No	Customer

134.Table

Impact Analysis Form

Impact Analysis Form								
		Impacted Section						
CR No	SDD	SRS	SITP	UAT	PMP	EST	SC	No of Man hours
1	NO	NO	NO	NO	NO	NO	YES	0.1
2	YES	YES	YES	YES	YES	YES	YES	4
3	YES	YES	YES	NO	YES	NO	YES	0.1

135.Table

Impact analysis is done once customer has agreed on the change request form. Impact analysis form is indirectly estimation or the quotation for the CR raised. Note backward tracking is done by CR No Column.CR No is used from change request form. Also note for non-billable CR No 1 and 2 we have put the value as 0.1.If you see CR No 1 “Changed Color of accounting screen” is definitely a CR which will not be charged but a lot of this kind of CR will become one billable CR.

That’s 10 Non-billable CR * 0.1 man/hours = 1 man/hour. Let the client know changes can not continue indefinetly.Impact analysis form also has which sections have been updated SDD,SRS,SITP,UAT etc , just to show to the client various places the impact is done and to guide during estimation.SC means source code. Incase if you are just wondering.

Estimating CR

As said before there are two types of CR

- Adding New Modules
- Changes in existing system

For new modules FP, UCP or WBS can be used. If you have skipped FP, UCP and WBS do read them once so that this chapter goes like a simple breeze. Main challenge is in estimating changes in CR.Its know fact that effort required to make changes in existing system requires more effort than creating new systems. Above all unstructured and undocumented code has more learning curve.CR does not have fixed pattern as such.CR changes from customer to customer and also what type of request it is.

Below is some typical example of CR

- Change of textbox size, Color, GUI design changes etc.
- Changes in business logic.
- Changing stored procedure, database design etc.

In short there is no fixed pattern of CR.Due to this estimating using current metric methods is difficult. This book recommends WBS for amendment type of CR and for any new functionality you can use FP, UCP or WBS.

Rating Table for CR

Rating	Hours
Non-billable CR	0.1
Simple CR	1
Medium CR	2
Complex CR	3

136.Table

Our normal rating table for WBS had only simple, medium and complex. We have added one extra level Non-billable CR.WBS rating table can be further modified to fit according to project. Below is simple modified WBS CR rating table.

Rating	Hours
Non-billable CR	0.1
Semi-Simple CR	1
Simple CR	2
Semi-Medium CR	3
Medium CR	4
Semi-Complex CR	5
Complex CR	6

137.Table

Impact analysis sheet gives the most detail view of CR.We will try to apply this WBS rating table to impact analysis sheet.Heres how the impact analysis sheet will look like when WBS rating table is applied.

Impact Analysis Form									
Impacted Section									
CR No	SDD	SRS	SITP	UAT	PMP	EST	SC	WBS Rating	Man/Hours
1	NO	NO	NO	NO	NO	NO	YES	NBCR	0.1
2	YES	YES	YES	YES	YES	YES	YES	CCR	3
3	YES	YES	YES	NO	YES	NO	YES	NBCR	0.1
									3.2

138.Table

Note effort is in man/hours this is because CR is the smallest unit of effort. Also note the extra column added WBS rating column.

Let's revise the steps to estimate CR

- Fill in the CR form get its approval by customer.
- Prepare impact analysis from the CR form.
- Construct your WBS rating table.
- Apply your WBS rating table to the impact analysis estimation sheet.

Software Maintenance

Recently in our building we appointed electrical maintenance engineering. The main work that he will be doing is as follows:-

- Rectify electrical faults if they occur.
- To prevent any faults to occur by checking current electrical systems on daily basis.

Ok that was maintenance from electrical business model point of view. Now let's see what does maintenance stands in software industry. Let's try to understand definition of maintenance first

"Software maintenance is to rectify any bugs that occur in production system and to ensure future bugs do not occur by proper house-keeping"

Maintenance are boring job from programmer point of view, hence productive level is very low as motivation is very low. Maximum resignations occur on maintenance types of jobs.Whateever definition is defined in upper section is theoretical. In software industry practically maintenance also involves making new modules and handling CR. Normal trend in software industry is to complete the project , when product goes under production appoint one or two software engineers for maintenance. Software Company is stable and mature when it has large maintenance projects rather than new projects. Remember it's during maintenance you get CR and also new projects. Most software companies use sixth-sense to prepare quotation. The management of the software company just asks a question how much full time engineers will be required. If lets say two engineers then quotation is

Quotation = Two engineers salary + profit to be earned by company.

But this approach can be quiet misleading as there is large possibility that the guy who is doing maintenance can be overloaded or with out work. Software metrics had different approach for maintenance .They look at how many engineer can handle how many software metric points. That means is it one FTE (Full Time Engineers) can handle 100 Function Points ,Two FTE (Full Time Engineers) can handle twenty Use Case Points (UCP) , one FTE can handle 20 WBS.If the quotation is prepared on basis of FTE / Software Metric Points , it gives a evaluation of how good the software is. That is if suppose you have ten FTE / 100 FP, that means there is some serious problem with the software design, probably has to be made from scratch or a change in architecture. But this approach has some serious drawbacks. Let's say the project is of 100 UCP (Use Case Points) and that project has ten masters and one transaction maintenance screen. Bugs normally occurs in transaction screen rather than master screens. Master screens are used once in a month. If you go according to metric rules FTE/FP and you appoint two FTE for the work. I am sure one FTE will be seen in canteen whole day. From my perspective it depends on how customized software and how many bugs are there. If the software before going to production has already all house keeping screens in place, company will probably have no maintenance. So many times customer expects from the FTE to handle not only software bugs but also look at other problems like networking issues, software installation, internet connection etc. These all work is not part of the software project at all.

Consider the following sales project. It has following master tables

- Country Master
- City Master
- Region Master
- Customer Master

It has two transaction tables

- Sales
- Invoices

All the tables need to have a maintenance screen. Total UCP for masters is 100 and transaction is 50 UCP.Total project is of 150 UCP (Please note above estimation are just assumptions).After the project is completed customer decided to give maintenance to the same company. Company decided to appoint one FTE.But if you closely look at the

project programmer would only be maintaining 50 UCP.Now think the other scenario also customer also expects that FTE should look after :

- Hardware problem.
- Internet Connection
- Train users

In these situations FTE will be quiet overloaded. The next section will cover a better way of quoting maintenance project rather than proportional way FTE/Software points.

Task per Day (TPD) Approach

As we seen in previous section definition of maintenance in software industry is different than in other industries. Software maintenance can involve:

- Enhancements
- Changes
- New modules

So the approach of FTE/ Software points can be very misleading. In previous section we had talked about electrical maintenance. Let's say he is also given extra responsibility to supervise other works like gardening work, security work etc.Definetly he will be quiet overloaded, in this situation you will either appoint extra guy for supervising or increase his salary for the overload. Same holds true for software maintenance. If a FTE is getting appointed at client side, just chalk down what how many tasks he will perform in day and put quotation according to that. Down shown is a simple task sheet or daily house-keeping the maintenance engineer has to do.

Task per Day to be performed	Number of Hours
Check all services are working	1
Look at error log files and take appropriate actions	2
Backing database daily	0.30
Check for internet connections	0.15
Antivirus updates	0.30
Attend any problems users are facing on software	2

139.Task Per Day Table

Looking at TPD to be performed and number of hours required you easily forecast number of FTE to be required. Basically there are two types of maintenance task:

- Fixed maintenance work example backing database , antivirus updates , log file supervision etc
- On demand work example bugs in software, new enhancements in project if any etc.

Fixed tasks are easy to predict as you have know how of what is to be done before hand. But on demand task is difficult to predict and can only be done with understanding with client.

Chapter 8

Points to Remember During Prepare Quotation

The below points are not related to any quotation procedure as such. But if kept in mind will help you to come out with confident quotation

1) Warranty periods:

Warranty periods are normally given to customer if they want any amendments or touch up. Let's say if the project is of 1 year and after 1 year still they want to make some minor changes. Depending on companies policies you can give warranty period of 1 month. Warranty periods are provided to attract customers. Now it's up to the companies policies would they like to charge for this warranty period?

2) Creative documents:

Creative documents like flash movies, images, look and feel of website take a lot time. Estimator overlooks creative aspect of the project which can overshoot if your client is very keen about the look and feel aspects. Specially if the estimator is programmer himself he is least bother about the creative aspect. In web projects creative aspect can consume huge effort. There is no estimation technology as such for creative documents, best go the ADHOC way and append the figure to the final estimate.

3) User Acceptance:

User Acceptance is phase when user accepts your product and goes live. User signs off saying yes the delivery are according to specification. This is also the closing phase and probably when you will get the final delivery amount. But User Acceptance phase is not simple as it looks. Many times client due there daily schedule do not give time to user acceptance and it gets delayed for several reason. From your side some programmers are unnecessarily engaged in user acceptance phase.

There are two ways you can estimate User Acceptance:-

- Take User Acceptance as 5% of the total cost of project.

- Ask the user how much time he will take for user acceptance.

User Acceptance can pretty complicated if there there is third party in between. Example your final client is "A" but you have got this project from company "B".So your company coordinates with "B".So here you will have Twice user acceptance phase. So estimate accordingly.

4) Change of psychology:

Estimator should not be biased. If you are employee of the company do not add extra charge or subtract extra charges. These all things will be handled at the negotiation tables between the software company director and the customer. A estimator job is to show the real cost of the software to the company. In short estimator should not be bothered about negotiation phase and will we get this project or not?. Leave that work to the companies' director. And if you are the director of the company think that thing after estimation is over.

5)Sixth Sense Approach:

Any of the software measurement ways (Use case, Function points, LOC etc) are evolving and under practice. After looking the figure try to give Sixth sense based on your past experience. Some time estimation will be fair if went the ADHOC way.

6)Modular Estimation:

In huge projects with 100 of modules its better to estimate modular wise. Example if a client is asking for a customer module, Supplier module and Accounts module. Estimate them differently so that on negotiation table with client you can show him the complete break up. Second this approach is very useful for phase wise estimation. Client can decide either to not take the module (Due to financial concerns) or move it to phases.

7)Information Creep and Grey Areas:

Estimation are normally done at the initial phase itself probably with one or two meets with client we have to give estimation. So but natural many of the areas there can be creep. The best way for such situation is to think the maximum possibility and estimate. Example if any customer says that he needs chat module and no clarification is made till

what the depth it is, estimate to maximum possibility of how can chat application are made. Later during negotiation table show client the estimation basis. So according to the client financial budget changes will be made.

8)Other Costing:

Any of the Software estimation methodology do not give cost for non-software factors.

- If the software is using any third-party components example crystal reports etc estimate them in a ADHOC way.
- Example if in the project company is also providing web hosting, domain name, hard ware etc put them separately.
- Any training involved estimates them separately.
- Any traveling charges if it's an On-Shore project.

9)Assumptions:

As estimation is done at the initial stage there can be lot of creep and gray areas. Due to gray areas estimation has to be supported by proper assumptions.

10)Review from Third Party:

Before sending the costing to the client review it internally from third person who is not involved in the project. Iterations: Iterate it as many as times possible and at various phases. Example use function point to iterate during scoping phase that's initial phase. And Use case Point during the System requirement phase. This gives a good idea before implementing that is the costing proper

11)Two teams Estimation:

During estimation have two teams which will do the estimation. So that cross verification can be done on the error percent of the estimation.

12)Adding buffer:

The most tempted thing a estimator can do is add buffer. Buffers are normally added if the estimator is the programmer himself so that he does not have to sit late night. Example

for programming integration with payment gateway maximum you will require 3 days (which includes understanding the API and coding a sample to connect with payment gateway). Estimator adds a buffer for his comfort 1 week. As said in point Change of psychology think unbiased way. Leave it to the upper director and management level to add buffer

Acronym and Definitions

LOC(Lines of code):-Measurement methodology which uses Lines of code to determine the size of software.

MOM(Minutes of Meeting):-A document which describes what was discussed in the meeting.

RDBMS(Relational Database Management System):-A database management system in which data can be viewed and manipulated in tabular form. Data can be sorted in any order and tables of information are easily related or joined to each other.

OOP(Object Oriented Programming):-A programming technology in which program components are put together from reusable building blocks known as objects

WBS(Work Break Down Structure):-The list of tasks and subtasks defined for a project. This list is done in a hierarchical fashion, grouping sets of related tasks under a common parent task.

SEI(Software Engineering Institute.):-A federally funded research and development center that is under contract to Carnegie Mellon University and is devoted to the advancement of software engineering and the quality of software support systems. The SEI carries out its mission through two principal areas of work: software Engineering Management Practices, and Software Engineering Technical Practices.

CR (Change Request):-A formally submitted artifact that is used to track all stakeholder requests (including new features, enhancement requests, defects, changed requirements, etc.) along with related status information throughout the project lifecycle. All change history will be maintained with the Change Request, including all state changes along with dates and reasons for the change. This information will be available for any repeat reviews and for final closing.

FP(Function Points):-A sizing methodology for software projects based on functions of the software.

SLOC(Source Lines of Code):-(Source Line of Code) One line in a computer program. In many languages, each SLOC ends with a semicolon. SLOC is used in COCOMO and PROBE as the basis for estimating software-development time.

COTS(Commercial off-the-shelf software):-Commercial off-the-shelf, commercially available products that can be purchased and integrated with little or no customization, thus facilitating customer infrastructure expansion and reducing costs.

FPA(Function Point Analysis)

SMC(Simple , Medium and Complex)

FTP(File Transfer Protocol):-File Transfer Protocol - a mechanism for transferring files from one computer to another, often across a network or via a modem.

HTML(Hyper Text Markup Language):-The document format language used on the World Wide Web. Web browsers read HTML and display the page.

DLOC(Delivered Lines of Code):- Actual number of lines in a source document.

FTE(Full time Engineers) :- Resources working full time in a work.

EP(Elementary process)

UCP(Use Case Points):- Use Case points method is a software sizing and estimation based on Use case document.

UAW(Unadjusted actor weights):- A numeric sum of value of actors after giving the classification and before multiplying the technical complexity factor of the system. (When you go through steps of how to calculate UAW this will be more clear)

UUCW(Unadjusted Use case Weight):- A numeric sum of value of Use cases after classifying and before multiplying the technical complexity factor of the system. (When you go through steps of how to calculate UUCW this will be more clear)

UUCP(Unadjusted Use Case Points):- Sum of UAW and UUCW

API(Application Programming Interface):-Application programs used for accessing services provided by some lower-level module (such as operating system)

GUI(Graphical User Interface):-A computer terminal interface, such as Windows, that is based on graphics instead of text.

Use Case Transactions:-Its an atomic set of activities that are either performed entirely or not all.

Tfactor(Technical factor):-Total of all technical factor. See for more details in steps in estimation. See table 4.0 for more details

TCF(Technical Complexity Factor):-Factor which defines the complexity of the project. For more details see steps for UCP estimation.TCF is calculated from Tfactor.

EF(Environment Factor):-This is multiplying factor.

AUCP(Adjusted Use Case Points):-This the value obtained after multiplying with Efactor and Tfactor.

EA(Enterprise Architect):-Software program shipped with CD for Use Case Points.