Unit – 5
Project Management

Software Project Management : Estimation – LOC Based, FP Based, Make/Buy Decision, COCOMO I & II Model – Project Scheduling – Scheduling, Earned Value Analysis. Planning – Project Plan, Planning Process, RFP Risk Management – Identification, Projection – Risk Management – Risk Identification – RMMM Plan – CASE TOOLS

Software Project Management

5.1 Software Project Management-Introduction

Management is an essential activity for the computer based systems and product. The term project management involves various activities such as planning, monitoring, control of people, process and various events that occur during the development of | software.

Building the software system is a complex activity and many people get involved in - this activity for relatively long time. Hence, it is necessary to manage the project. The project management is carried out with the help of 4 P's i.e. people, product, process and project.

Hence, we will start our discussion by focusing on these elements.

5.1.1 Management Spectrum

Effective software project management focuses four P's i.e. people, product, process and project. The successful project management is done with the help of these four factors where the order of these elements is not arbitrary. Project manager has to motivate the communication between stakeholders. He should also prepare a project plan for the success of the product.

5.1.2 The People

People factor is an important issue in software industry. There is a strong need for motivated and highly skilled people for developing the software product. The Software Engineering Institute (SEI) has developed the People Management Capability Maturity Model (PM-CMM)

By using PM-CMM model software organizations become capable for undertaking complex applications which ultimately attracts or motivates the talented people.

Following are some key practice areas for software people -

- Recruitment
- Selection
- Performance management
- Training compensation
- Career development
- Organization and Work design
- Culture development.

5.1.3 The Product

Before planning the project three important tasks need to be done -

- ❖ Product objectives and scope must be established.
- ❖ Alternative solutions should be considered.
- ❖ Technical and management constraints must be identified.

The software developer and customer must communicate with each other in order to define the objectives and scope of the product. This is done as the first step in requirement gathering and analysis. The scope of the project identifies primary data, functions and behaviour of the product.

After establishing the objectives and scope of the product the alternative solutions are considered.

Finally, the constraints imposed by-delivery deadline or budgetary restrictions, personal availability can be identified.

5.1.4 The Process

The software process provides the framework from which the software development plan can be established.

There are various framework activities that needs to be carried out during the software development process. These activities can be of varying size and complexities.

Different task sets-tasks, milestones, work products and quality assurance points enable framework activities to adapt the software requirements and certain characteristics of software project.

Finally, umbrella activities such as Software Quality Assurance (SQA) and Software Configuration Management (SCM) are conducted. These umbrella activities depend upon the framework activities.

5.1.5 Project Planning Process

The first step to be taken in project management is Project planning. There are five major activities that are performed in project planning -

[1] Project estimation
[2] Project scheduling
[3] Risk analysis
[4] Quality management planning
[5] Change Management planning

Software estimation begins with a description of the scope of software product.

For the meaningful project development the scope must be bounded. The problem f for which the product is to be built is then decomposed into a set of smaller problems. 1 Each of these is estimated using historical data (metrics) and / or previous experience as I a guide. The two important issues-problem complexity and risk are considered before J final estimate is made.

There are many useful techniques for time and effort estimation. Process and project * metrics can provide historical perspective and powerful input for generation of \ quantitative estimates.

Estimation of recourses, cost and schedule for a software engineering effort requires -

- ❖ Experience,

- ❖ Access to good historical information (metrics) and
- ❖ The courage to commit to quantitative predictions when quantitative information is available.

While estimating the project, both the project planner and the customer should recognize that variability in software requirement means instability in cost and schedule. When customer changes the requirements, then estimation needs to be revisited.

5.1.6 Software Scope and Feasibility

Software scope describes four things -

- ❖ The function and features that are to be delivered to end-users.
- ❖ The data which can be input and output.
- ❖ The content of the software that is presented to user.
- ❖ Performance, constraints, reliability and interfaces that bounds the System.

There are two ways by which the scope can be defined -

[1] A scope can be defined using the narrative description of the software obtained after communication with all stakeholders.

[2] Scope can be defined as a set of use cases developed by the end users.

In the scope description, various functions are described. These functions are evaluated and refined to provide more details before the estimation of the project.

For performance consideration, processing and response time requirements are analyzed.

The constraints identify the limitations placed on the software by external hardware or any other existing system.

After identifying the scope following questions must be asked –

° Can we build the software to meet this scope?

° Is this software project feasible?

That means after identifying the scope of the project its feasibility must be ensured.

Following are the four dimensions of software feasibility. To ensure the feasibility of the software project the set of questions based on these dimension has to be answered. Those are as given below -

[1] Technology

- Is a project technically feasible?
- Is it within the state of art?
- Are the defects to be reduced to a level that satisfies the application's need?

[2] Finance

- Is it financially feasible?
- Is the development cost completed at a cost of software organization, its client, or market affordable?
- Are the defects to be reduced to a level that satisfies the application's need?

[3] Time

- Will the project's time to market beat the competition?

[4] Resource

- Does the organization have the resources needed to succeed?

- Putnam and Meyers suggests that scoping is not enough. Once scope is understood, and and feasibility have been identified the next task is estimation of the Resources required to accomplish the software development effort.

5.2 Estimation

Software project estimation is a form of problem solving. Many times the problem to be solved is too complex in software engineering. Hence for solving such problems, we decompose the given problem into a set of smaller problems.

The decomposition can be done using two approached decomposition of problem or decomposition of process. Estimation uses one or both forms of decomposition (partitioning).

5.2.1 Software Sizing

Following are certain issues based on which accuracy of software project estimate is predicated -

1. The degree to which planner has properly estimated the size of the product to be built.
2. The ability to translate the size estimate into human-effort, calendar time and money
3. The degree to which the project plan reflects the abilities of software team.
4. The stability of product requirements and the environment that supports the software engineering effort.

Sizing represents the project planner's first major challenge. In the context of project planning, size refers to a quantifiable outcome of the software project.

The sizing can be estimated using two approaches - a direct approach in which lines of code is considered and an indirect approach in which computation of function point is done.

Putnam and Myers suggested four different approaches for sizing the problem -

[1] Fuzzy logic sizing

In this approach planner must identify –

- The type of application
- Establish its magnitude on a qualitative scale and then refine the magnitude within the original range.
- Planner should also have access to historical database of the project so that estimates can be composed to actual experience.

[2] Function point sizing

Planner develops estimates of the information domain.

[3] Standard component sizing

There are various standard components used in software. These components are subsystems, modules, screens, reports, interactive, programs, batch program, files, LOC and Object-level instruction.

The project planner estimates the number of time these standard components are used. He then uses historical project data to determine the delivered size per standard component.

[4] Change sizing

This approach is used when existing software has to be modified as per the requirement of the project. The size of the software is then estimated by the number and type of reuse, addition of code, change made in the code, deletion of code.

The result of each sizing approaches must be combined statistically to create three-point estimate which is also known as expected-value estimate.

FP Based & LOC Based Estimation

5.2.2 Problem based Estimation

The problem based estimation is conducted using LOC based estimation, FP based estimation, process based estimation and use cased based estimation.

LOC and FP based data are used in two ways during software estimation -

1. These are useful to estimate the size each element of software.
2. The baseline metrics are collected from past project and LOC and FP data is used in conjunction with estimation variable to develop cost and effort values for the project. (LOC) and (FP) estimation are different estimation techniques. Yet, both have number of characteristics in common.

With a bounded statement of software scope a project planning process begins and by using the statement of scope the software problem is decomposed into the functions that can be estimated individually.

(LOC) or (FP) is then estimated for each function.

Baseline productively metrics are then applied to the appropriate estimation variable and cost or effort for the function is derived.

Function estimates are combined to obtain an overall estimate for the entire project.

Using historical data the project planner expected value by considering following variables -

1. Optimistic
2. Most likely
3. Pessimistic

For example, following formula

$$S = [S_{opt} + 4 \times S_m + S_{pess}]/6$$

considers for "most likely" estimate where S is the estimation size variable, S_{opt} represents the optimistic estimate, S_m represents the most likely estimate and S_{pess} represents the pessimistic estimate values.

5.3 LOC based Estimation

Size oriented measure is derived by considering the size of software that has been produced.

The organization builds a simple record of size measure for the software projects. It is built on past experiences of organizations.

It is a direct measure of software

Project	LOC	Effort	Cost($)	Doc (pgs.)	Errors	Defects	People
ABC	10,000	20	170	400	100	12	4
PQR	20,000	60	300	1000	129	32	6
XYZ	35,000	65	522	1290	280	87	7
: :	: :	: :	: :	: :	: :	: :	: :

A simple set of size measure that can be developed is as given below :

- Size = Kilo Lines of Code (KLOC)
- Effort = Person/month
- Productivity = KLOC/person-month
- Quality = Number of faults/KLOC - Cost = $/KLOC
- Documentation = Pages of documentation/KLOC

The size measure is based on the lines of code computation. The lines of code is defined as one line of text in a source file.

While counting the lines of code the simplest standard is :

- Don't count blank lines.
- Don't count comments.
- Count everything else.

The size oriented measure is not universally accepted method.

Advantages

1. Artifact of software development which is easily counted.
2. Many existing methods use LOC as a key input.
3. A large body of literature and data based on LOC already exists.

Disadvantages

1. This measure is dependent upon the programming language.
2. This method is well designed but shorter program may get suffered.
3. It does not accommodate non procedural languages.
4. In early stage of development it is difficult to estimate LOC.

Example of LOC based Estimation Example

Consider an ABC project with some important modules such as

1. User interface and control facilities
2. 2D graphics analysis
3. 3D graphics analysis
4. Database management
5. Computer graphics display facility
6. Peripheral control function
7. Design analysis models Estimate the project in based on LOC

Solution

For estimating the given application we consider each module as separate function and corresponding lines of code can be estimated in the following table as

Function	Estimated LOC
User Interface and Control Facilities(UICF)	2500
2D graphics analysis(2DGA)	5600
3D Geometric Analysis function(3DGA)	6450
Database Management(DBM)	3100
Computer Graphics Display Facility(CGDF)	4740
Peripheral Control Function(PCF)	2250
Design Analysis Modules (DAM)	7980
Total estimation In LOC	32620

Expected LOC for 3D Geometric analysis function based on three point estimation is -

- Optimistic estimation 4700
- Most likely estimation 6000
- Pessimistic estimation 10000

$$S = [S_{opt} + 4 * S_m + S_{pess}]/6$$

Expected value = $[4700 + (4 * 6000) + 10000]/6 = 6450$

A review of historical data indicates -

1. Average productivity is 500 LOC per month
2. Average labor cost is $6000 per month

Then cost for lines of code can be estimated as

$$cost/LOC = (6000/500) = \$12$$

By considering total estimated LOC as 32620

- Total estimated project cost = (32620*12) = $391440
- Total estimated project effort = (32620/500) = 65 Person-months

5.4 Function Oriented Metrics

The function point model is based on functionality of the delivered application.

These are generally independent of the programming language used.

This method is developed by Albrecht in 1979 for IBM.

Function points are derived using :

1. Countable measures of the software requirements domain
2. Assessments of the software complexity.

How to calculate function point?

The data for following information domain characteristics are collected :

[1] Number of user inputs

Each user input which provides distinct application data to the software is counted.

[2] Number of user outputs

Each user output that provides application data to the user is counted, e.g. screens, reports, error messages.

[3] Number of user inquiries

An on-line input that results in the generation of some immediate software response in the form of an output.

[4] Number of files

Each logical master file, i.e. a logical grouping of data that may be part of a database or a separate file.

[5] Number of external interfaces

All machine-readable interfaces that are used to transmit information to another system are counted.

The organization needs to develop criteria which determine whether a particular entry is simple, average or complex.

The weighting factors should be determined by observations or by experiments.

The count table can be computed with the help of above given table.

Now the software complexity can be computed by answering following questions. These are complexity adjustment values

1. Does the system need reliable backup and recovery ?
2. Are data communications required ?
3. Are there distributed processing functions ?
4. Is performance of the system critical ?
5. Will the system rim in an existing, heavily utilized operational environment ?
6. Does the system require on-line data entry ?

7. Does the on-line data entry require the input transaction to be built over multiple screens or operations ?
8. Are the master files updated on-line ?
9. Are the inputs, outputs, files or inquiries complex ?
10. Is the internal processing complex ?
11. Is the code which is designed being reusable ?
12. Are conversion and installation included in the design ?
13. Is the system designed for multiple installations in different organizations ?
14. Is the application designed to facilitate change and ease of use by the user ?

Rate each of the above factors according to the following scale :

$$\textbf{\textit{Function Points (FP)}} - \textbf{\textit{Count total}} \times (0.65 + (0.01 \times \textbf{\textit{Sum}}(F_i))$$

0	1	2	3	4	5
No influence	incidental	Moderate	Average	Significant	Essential

Once the functional point is calculated then we can compute various measures as follows

- Productivity = FP/person-month
- Quality = Number of faults/FP
- Cost = $/FP
- Documentation = Pages of documentation/FP.

Advantages

- This method is independent of programming languages.
- It is based on the data which can be obtained in early stage of project .

Disadvantages

- This method is more suitable for business systems and can be developed for that domain.
- Many aspects of this method are not validated.
- The functional point has no significant meaning. It is just a numerical value.

Example of FP based Estimation

FP focuses on information domain values rather than software functions. Thus we create a function point calculation table for ABC project.

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						*320*

For this example we assume average complexity weighting factor.

Each of the complexity weighting factor is estimated and the complexity adjustment factor is computed using the complexity factor table below. (Based on the 14 questions)

Factor	Value
Backup and recovery	4
Data communications	2
Distributed processing	0
Performance critical	4
Existing operating environment	3
Online data entry	4
Input transaction over multiple screens	5
Master files updated online	3
Information domain values complex	5
Internal processing complex	5
Code designed for reuse	4
Conversion/installation in design	3
Multiple installations	5
Application designed for change	5
Value adjustment factor	**1.17**

The estimated number of adjusted FP is derived using the following formula : -

$$FP\ ESTIMATED = (FP\ COUNT\ TOTAL * [COMPLEXITY\ ADJUSTMENT\ FACTOR]$$

$$FP\ ESTIMATED = COUNT\ TOTAL * [0.65 + (0.01 * ? \sum (Fi))]$$

Complexity adjustment factor = [0.65 + (0.01 * 52)] = 1.17
FP ESTIMATED = (381 * 1.17) = 446 (Function point count adjusted with complexity adjustment factor)

A review of historical data indicates -

1. Average productivity is 6.5 FP/Person month
2. Average labor cost is $6000 per month

Calculations for cost per function point, total estimated project cost and total effort

1. The cost per function point = (6000 / 6.5) = $923
2. Total estimated project cost = (446 * 923) = $411658
3. Total estimated effort = (446 / 6.5) = 69 Person-month.

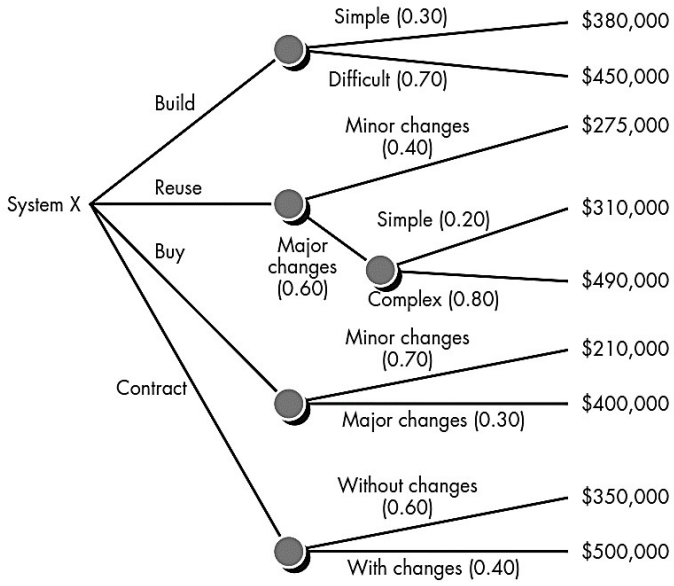
5.5 Make/Buy Decision

Software engineering managers are often faced with a make-buy decision to acquire a computer software. Normally following are the options that are used to acquire the software.

1. Purchase or buy the software.
2. Reuse existing partially built components to construct the system.
3. Build the system from scratch.
4. Contract the software development to an outside vendor.

The decision of acquisition of software is critically based on the cost. A tree structure is built to analyze the costs of software which can be acquired using any of the above given ways.

For example - Consider the make-buy decision tree for system S.



Expected cost can be computed for each branch using following formula.

$$\textbf{Expected Cost} = \sum \textbf{Path Probability of decition tree path} \times \textbf{Estimated Path cost}$$

For example for the branch $\textbf{system S} \rightarrow \textbf{Reuse}$ can be computed as:

$$\begin{aligned} Expected\ cost_{Reuse} &= 0.40(\$275\,K) + [0.60\,(0.20\,(\$310\,K) + 0.80\,(\$490\,K))] \\ &= \$110\,K + [0.60\,(\$\,62\,K + \$\,392\,K)] \\ &= \$110\,K + [0.60\,(\$\,454\,K)] \\ &= \$110\,K + \$272.4\,K \\ &= \mathbf{\$382\,K} \end{aligned}$$

Thus the expected cost at each node can be computed. It is summerised as given below

$\text{Expected cost}_{\text{reuse}} = 0.40\ (\$275K) + 0.60\ [0.20\ (\$310K) + 0.80\ (\$490K)] = \$382K$
$\text{Expected cost}_{\text{buy}} = 0.70\ (\$210K) + 0.30\ (\$400K) = \$267K$
$\text{Expected cost}_{\text{contract}} = 0.60\ (\$350K) + 0.40\ (\$500K) = \$410K$

From this we can conclude that by purchasing the software we select for lowest expected cost option. But simply cost should not be a criterion to acquire the software.

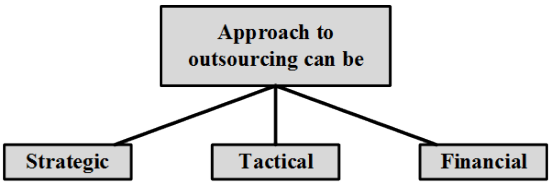
During decision making process for software acquisition following factors should also be considered.

1. Availability of reliable software.
2. Experience of developer or vendor or contractor.
3. Conformance to requirements.
4. Local politics.
5. Likelihood of changes in the software.

These are some criteria which can heavily affect the decision of make-buy of software.

5.5.1 Outsourcing

Outsourcing is a process in which software engineering activities are contracted to a third party who does the work at lowest cost with high quality.



In strategic level, the significant portion of software work can be contracted to third party.

In Tactical level, a project manager determines whether part or all of the software can be accomplished with good quality by contracting it.
In financial level, the cost is the prime factor in the decision of outsourcing.

Benefits of outsourcing

[1] Cost savings
If a software is outsourced then people and resource utilization can be reduced. And thereby the cost of the project can be saved effectively.

[2] Accelerated development
Since some part of software gets developed simultaneously by a third party, the overall development process gets accelerated.

Drawbacks of outsourcing

A software company loses some control over the software as it is developed by third person.
The trend of outsourcing will be continued in software industry in order to survive in competitive world.

5.6 COCOMO Model

COCOMO model (Constructive cost model) was proposed by Boehm.
This model estimates the total effort in terms of "person-months" of the technical project staff.

Boehm introduces three forms of cocomo. It can be applied in three classes of software project:

1. **Organic mode :** Relatively simple , small projects with a small team are handled . Such a team should have good application experience to less rigid requirements

2. **Semidetached mode:** For intermediate software projects(little complex compared to organic mode projects in terms of size). Projects may have a mix of rigid and less than rigid requirements.

3. **Embedded mode:** When the software project must be developed within a tight set of hardware and software operational constraints. Ex of complex project: Air traffic control system

Forms of cocomo model are:
1.Basic cocomo: Computes software development effort and cost as a function of programme size expressed in terms of lines of code(LOC).

The basic cocomo model takes the following form:
E=ab (KLOC)Exp(bb) persons-months
D=cb(E)Exp(db)months

Where
E- Stands for the effort applied in terms of person months
D-Development time in chronological months
KLOC-Kilo lines of code of the project

Ab,bb,cb,db are the co-efficients for the three modes are given below:

From E & D we can compute the no: of people required to accomplish the project as N=E/D

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	.38
Semi-detached	3.0	1.12	2.5	.35
Embedded	3.6	1.20	2.5	.32

Merits of Basic Cocomo model:
Basic cocomo model is good for quick, early,rough order of magnitude estimates of software project.

Limitations :

1. The accuracy of this model is limited because it does not consider certain factors for cost estimation of software. These factors are hardware constraints, personal quality and experiences, modern techniques and tools
2. The estimates of Cocomo model are within a factor of 1.3 only 29% of the time and within the factor of 2 only 60% of time.

Example:
Consider a software project using semi-detached mode with 30,000 lines of code . We will obtain estimation for this project as follows:

(1)**Effort estimation**:
E= ab (KLOC)Exp(bb)person-months
E=3.0(30)1.12 where lines of code=30000=30
KLOC E=135 person-month

(2) **Duration estimation:**
D=cb (E)Exp(db)months
=2.5(135)0.35
D=14 months

(3)**Person estimation:**
N=E/D
=135/14 N
=10 persons approx.

2. **Intermediate COCOMO:**

Computes effort as a function as a function of programme size and a lot of cost drivers that includes subjective assessment of product attributes,hardware attributes , personal attributes and project attributes.

The basic model is extended to consider a set of cost driver attributes grouped into 4 categories(Intermediate Cocomo)

(1)Product Attributes:

(a) Required software reliability
(b) Size of application software
(c) Complexity of the product

(2) Hardware Attributes:

(a) Run-time performance constraints
(b) Memory constraints
(c) Required turn around time
(d) Volatility of virtual machine

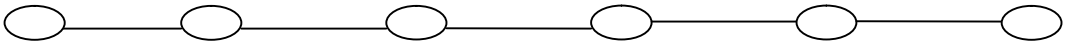
(3)Personal attributes:

(a) Analyst capability
(b) Software Engineer Capability
(c) Applications Experience
(d) Programming language experience
(e) Virtual machine Experience

(**4)Project Attributes:**

(a) Use of software tools
(b) Required development schedule
(c) Application of software engineering methods

Now these 15 attributes get a 6-point scale ranging from “very low” to “extra high”.
These ratings can be viewed as:



Very Low Low Nominal High Very high Extra high

Based on the rating effort multipliers is determined.
The product of all effort Multipliers result in “effort adjustment factor” (EAF).

The intermediate Cocomo takes the form.
E= ai(KLOC)bi*EAF

where E:Effort applied in terms of person-months
KLOC : Kilo lines of code for the project EAF : It is the effort adjustment factor

The values of ai and bi for various class of software projects are:

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Software project	a_i	b_i
Organic	3.2	1.05
Semidetached	3.0	1.12
Embedded	2.8	1.20

The duration and person estimate is same as in basic Cocomo model i.e;
D=cb (E)Exp (db) months i.e; use values of cb and db coefficients
N=E/D persons

Merits:

1.This model can be applied to almost entire software product for easy and rough cost estimation during early stage.

2.It can also be applied at the software product component level for obtaining more accurate cost estimation.

Limitations:
1.The effort multipliers are not dependent on phases.
2. A product with many components is difficult to estimate.

Example: Consider a project having 30,000 lines of code which in an embedded software with critical area hence reliability is high.The estimation can be

E=ai (KLOC)bi*(EAF)
As reliability is high EAF=1.15(product attribute)

ai=2.8
bi=1.20 for embedded software

E=2.8(30)1.20 *1.15
=191 person month

D=cb (E)db=2.5(191)0.32
=13 months approximately

N=E/D =191/13
N=15 persons approx.

DETAILED COCOMO
The Advanced COCOMO model computes effort as a function of program size and a set of cost drivers weighted according to each phase of the software lifecycle. The Advanced model applies the Intermediate model at the component level, and then a phase-based approach is used to consolidate the estimate [Fenton, 1997]. The four phases used in the detailed COCOMO model are: requirements planning and product design (RPD), detailed design (DD), code and unit test (CUT), and integration and test (IT).

Cost Driver	Rating	RPD	DD	CUT	IT
ACAP	Very Low	1.80	1.35	1.35	1.50
	Low	0.85	0.85	0.85	1.20
	Nominal	1.00	1.00	1.00	1.00
	High	0.75	0.90	0.90	0.85
	Very High	0.55	0.75	0.75	0.70

Analyst capability effort multiplier for detailed COCOMO Estimates for each module is combined into subsystems and eventually an overall project estimate. Using the detailed cost drivers, an estimate is determined for each phase of the lifecycle.

5.7 COCOMO II Model

COCOMO II is applied for modern software development practices addressed for the projects in 1990's and 2000's.

The sub-models of COCOMO II model are -

[1] Application composition model

For estimating the efforts required for the prototyping projects and the projects in which the existing software components are used application-composition model is introduced.

The estimation in this model is based on the number of application points. The application points are similar to the object points.

This estimation is based on the level of difficulty of object points.

Boehm has suggested the object point productivity in the following manner.

Developers experience and capability	Very low	low	Nominal	High	Very high
CASE maturity	Very low	Low	Nominal	High	Very high
Productivity (NOP/Month)	4	7	13	25	50

Effort computation in application-composition model can be done as follows -

$$PM = (NAP^{(1-\%reuse/100)})/PROD$$

Wher

PM means effort required in terms of person-months.

NAP means number of application points required.

% reuse indicates the amount of reused components in the project. These reusable components can be screens, reports or the modules used in previous projects.

PROD is the object point productivity. These values are given in the above table.

[2] An early design model

This model is used in the early stage of the project development. That is after gathering the user requirements and before the project development actually starts, this model is used. Hence approximate cost estimation can be made in this model.

The estimation can be made based on the functional points.

In early stage of development different ways of implementing user requirements can be estimated.

The effort estimation (in terms of person month) in this model can be made using the following formula

$$Effort = A \times size^B \times M$$

Boehm has proposed the value of coefficient A = 2.94.

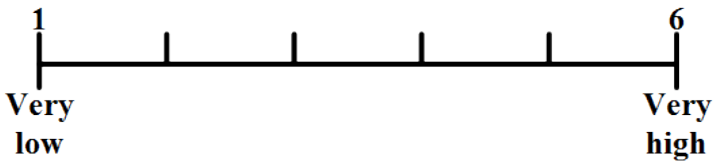
Size should be in terms of Kilo source lines of code i.e. KSLOC. The lines of code can be computed with the help of function point.

The value of B is varying from 1.1 to 1.24 and depends upon the project.

M is based on the characteristics such as

- Product reliability and complexity (RCPX)
- Reuse required (RUSE)
- Platform difficulty (PDIF)
- Personnel capability (PERS)
- Personnel experience (PREX)
- Schedule (SCED)
- Support facilities (FCIL)

These characteristics values can be computed on the following scale -



Hence the effort estimation can be given as

$$PM = 2.94 \times size^B \times M$$
$$M = RUSE \times PDIF \times PERS \times PREX \times SCED \times FCIL$$

[3] A reuse model

This model considers the systems that have significant amount of code which is reused from the earlier software systems. The estimation made in reuse model is nothing but the efforts required to integrated the reused models into the new systems.

There are two types of reusable codes : black box code and white box code. The black box code is a kind of code which is simply integrated with the new system without modifying it. The white box code is a kind of code that has to be modified to some extent before integrating it with the new system, and then only it can work correctly.

There is third category of code which is used in reuse model and it is the code which can be generated automatically. In this form of reuse the standard templates are integrated in the generator. To these generators, the system model is given as input from which some additional information about the system is taken and the code can be generated using the templates.

The efforts required for automatically the generated code is

$$PM = (ASLOC \times AT/100)/ATPROD$$

where

AT is percentage of automatically generated code.

ATPROD is the productivity of engineers in estimating such code

Sometimes in the reuse model some white box code is used along with the newly- developed code. The size estimate of newly developed code is equivalent to the reused code. Following formula is used to calculate the effort in such a case -

$$ESLOC = ASLOC \, x \, (1 - AT/10Q) \, X \, AAM$$

where

ESLOC means equivalent number of lines of new source code.

ASLOC means the source lines of code in the component that has to be adapted.

AAM is adaptation Adjustment multiplier. This factor is used to take into account the efforts required to reuse the code.

[4] Post architecture model

This model is a detailed model used to compute the efforts. The basic formula used in this model is

$$Effort - Ax SizeB \, x \, M$$

In this model efforts should be estimated more accurately. In the above formula A is the amount of code. This code size estimate is made with the help of three components -

1. The estimate about new lines of code that is added in the program.
2. Equivalent number of source lines of code (ESLOC) used in reuse model.
3. Due to changes in requirements the lines of code get modified. The estimate of amount of code being modified.

The exponent term B has three possible values that are related to the levels of project complexity. The values of B are continuous rather than discrete. It depends upon the five scale factors. These scale factors vary from very low to extra high (i.e. from 5 to 0).

These factors are

Scale factor for component B	Description
Precedentedness	This factor is for previous experience of organization. Very low means no previous experience and high means the organization knows the application domain.
Development flexibility	Flexibility in development process. Very low means the j typical process is used. Extra high means client is responsible for defining the process goals.
Architecture/risk resolution	Amount of risk that is allowed to carry out. Very low: means little risk analysis is permitted and extra high means f high risk analysis is made. $
Team cohesion	Represents the working environment of the team. Very low j cohesion means poor communication or interaction between j the team members and extra high means there is no j communication problem and team can work in a good j spirit. 1
Process maturity	This factor affects the process maturity of the organization. This value can be computed using Capacity Maturity Model 1 (CMM) questionnaire, for computing the estimates CMM j maturity level can be subtracted from 5.

Add up all these rating and then whatever value you get, divide it by 100. Then add the resultant value to 1.01 to get the exponent value.

This model makes use of 17 cost attributes instead of seven. These attributes are used to adjust initial estimate.

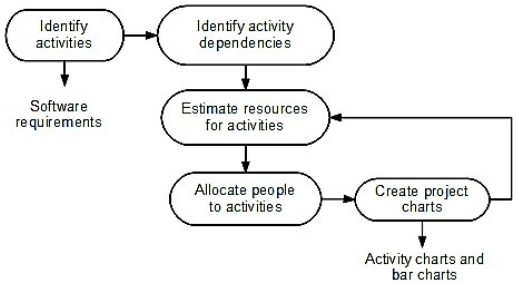
Cost attribute	Type of attribute	Purpose
RELY	Product	System reliability that is required.
CPLX	Product	Complexity of system modules 5'
DATA	Product	Size of the data used from database
DOCU	Product	Some amount of documentation used
RUSE	Product	Percentage of reusable components
TIME	Computer	Amount of time required for execution
PVOL	Computer	Volatility of development platform.)
STOR	Computer	Memory Constraint
ACAP	Personnel	Project analyst's capability to analyses the project.
PCAP	Personnel	Programmer capability
PCON	Personnel	Personnel continuity.
PEXP	Personnel	Programmer's experience in project domain.
LTEX	Personnel	Experience of languages and tools that are used.
AEXP	Personal	Analyst's experience in project domain.
TOOL	Project	Use of software tools
SCED	Project	Project schedule compression.
SITE	Project	Quality of inter-site and multi-site working.

Scheduling and Tracking

5.8 Scheduling and Tracking

While scheduling the project, the manager has to estimate the time and resource of the project. All the activities in the project must be arranged in coherent sequence. The schedules must be continually updated because some uncertain problems may occur during the project life cycle. For new projects initial estimates can be made optimistically.

During the project scheduling the total work is separated into various small activities. And time required for each activity must be determined by the project manager. For efficient performance some activities are conducted in parallel.



The project manager should be aware of the fact that: Every stage of the project may not be problem-free. Some of the typical problems in project development stage are:

- People may leave or remain absent.
- Hardware may get failed.
- Software resource may not be available.

To accomplish the project within given schedule the required resources must be available when needed. Various resources required for the project are

- Human effort
- Sufficient disk space on server
- Specialized hardware
- Software technology
- Travel allowance required by the project staff.

Project schedules are represented as the set of chart in which the work-breakdown structure and dependencies within various activities is represented.

5.8.1 Relationship between People and Effort

People work on the software project doing various activities such as requirements gathering, design, analysis, coding and testing.

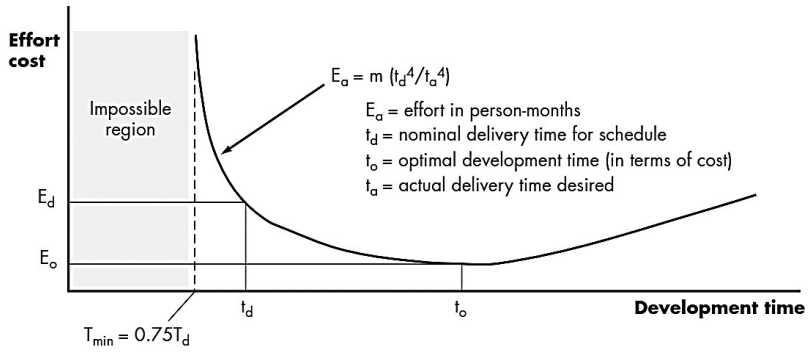
There is a common myth among the software managers that by adding more people in the project, the deadline can be achieved. But this is not true - as by adding more people in the project, first we need to train them for the tools and technologies that are getting used in the project. And only those people can teach the new people who are already working. Thus during teaching or training the time will be simply wasted and there won't be the progress in the project.

Once the effort required to develop a software has been determined, it is necessary to determine the people or staff requirement for the project.

Putnam first studied the on how much staffing is required for the projects. He extended the work of Norden who had earlier investigated the staffing pattern

The Putnam-Norden-Rayleigh Curve(PNR Curve) represents the relationship between effort applied and delivery time for the software project.

Following equation shows the relationship of project effort as a function of project delivery time.



The curve rises sharply left to t_d indicating that project delivery time can not be compressed beyond 0.75 t_d.

Beyond this the failure region is shown and failure risk becomes high.

Software equation: The software equation can be derived from the PNR curve.

It represents the nonlinear relationship between time to complete the project and human effort applied to the project. It can be denoted as

$$L = P \times E^{1/3} t^{4/3}$$

that is

$$E = L^3 / P^3 t^4$$

That also implies that by extending the end date six months, we can reduce the number of people

5.8.2 Task Sets

Definition of task set: The task set is a collection of software engineering work tasks, milestones, and work products that must be accomplished to complete particular project.

Every process model consists of various tasks sets. Using these tasks sets the software team define, develop or ultimately support computer software.

There is no single task that is appropriate for all the projects but for developing large, complex projects the set of tasks are required. Hence every effective software process should define a collection of task sets depending upon the type of the project.

Using tasks sets the high quality software can be developed and any unnecessary work can be avoided during software development.

The number of tasks sets will vary depending upon the type of the project. Various types of projects are enlisted below -

[1] Concept Development project

These are the projects in which new business ideas or the applications based on new technologies are to be developed.

[2] New application development project

These projects are developed for satisfying a specific customer need.

[3] Application upgradation project

These are kind of projects in which existing software application needs a major change. This change can be for performance improvement, or modifications within the modules and interfaces.

[4] Application maintenance project

These are kind of projects that correct, adapt or extend the existing software applications.

[5] Reengineering projects

These are the projects in which the legacy systems are rebuilt partly or completely.

Various factors that influence the tasks sets are -

1. Size of project
2. Project development staff
3. Number of user of that project
4. Application longetivity
5. Complexity of application
6. Performance constraints
7. Use of technologies

Task set example: Consider the concept development type of the project. Various tasks sets in this type of project are -

1. **Defining scope:** This task is for defining the scope, goal or objective of the project.
2. **Planning**: It includes the estimate for schedule, cost and people for completing the desired concept.
3. **Evaluation of technology risks**: It evaluates the risk associated with the technology used in the project.
4. **Concept implementation**: It includes the concept representation in the same manner as expected by the end user.

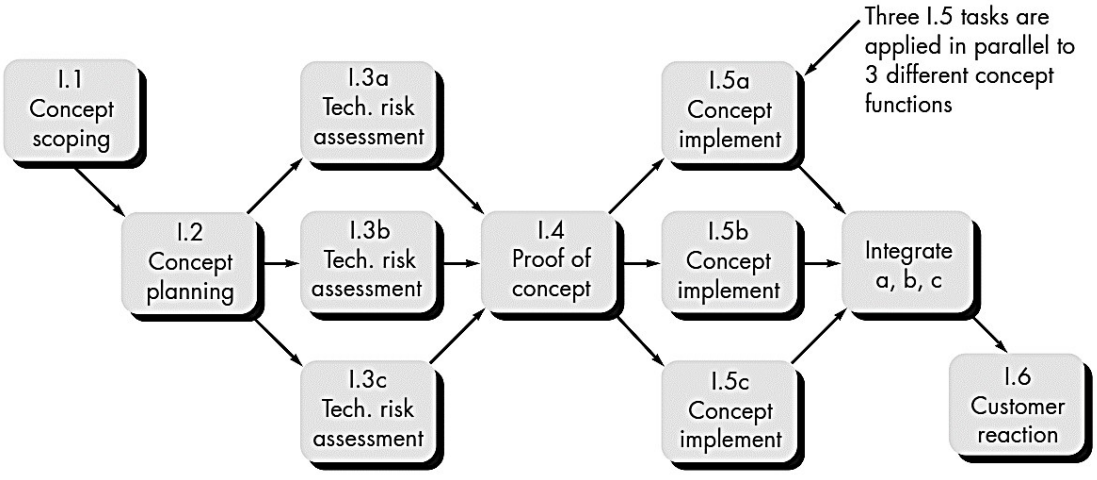
5.8.3 Task Network

The task is a small unit of work.

The task network or an activity network is a graphical representation, with:

- Nodes corresponding to activities.
- Tasks or activities are linked if there is a dependency between them.
- The task network for the product development is as shown in the below figure

The task network definition helps project manager to understand the project work breakdown structure.



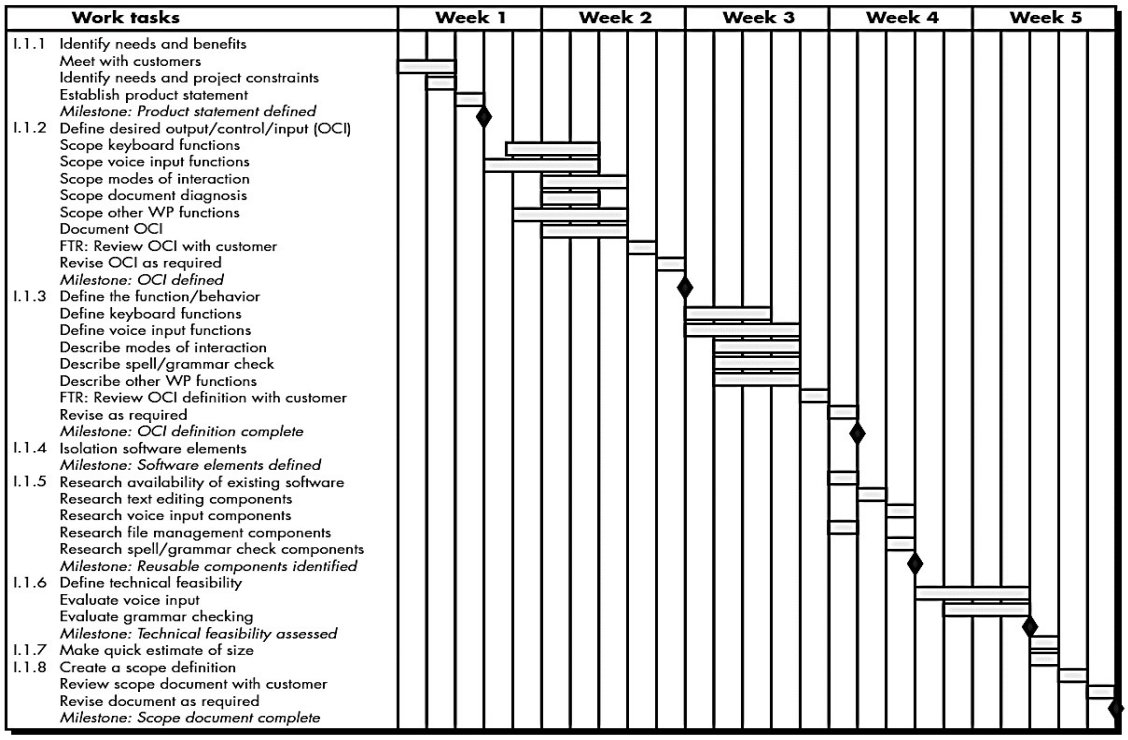
The project manager should be aware of interdependencies among various tasks. It should be aware of all those tasks which lie on the critical path.

5.8.4 Time Line Chart

In software project scheduling the timeline chart is created. The purpose of timeline chart is to emphasize the scope of individual task. Hence set of tasks are given as input to the time line chart.

- The time line chart is also called as Gant chart.
- The time line chart can be developed for entire project or it can be developed for individual functions.
- In time line chart
  1. All the tasks are listed at the leftmost column.
  2. The horizontal bars indicate the time required by the corresponding task.
  3. When multiple horizontal bars occur at the same time on the calendar, then that means concurrency can be applied for performing the tasks.
  4. The diamonds indicate the milestones.

In most of the projects, after generation of time line chart the project tables are prepared. In project tables all the tasks are listed along with actual start and end dates and related information.



Work tasks	Planned start	Actual start	Planned complete	Actual complete	Assigned person	Effort allocated	Notes
I.1.1 Identify needs and benefits							Scoping will require more effort/time
Meet with customers	wk1, d1	wk1, d1	wk1, d2	wk1, d2	BLS	2 p-d	
Identify needs and project constraints	wk1, d2	wk1, d2	wk1, d2	wk1, d2	JPP	1 p-d	
Establish product statement	wk1, d3	wk1, d3	wk1, d3	wk1, d3	BLS/JPP	1 p-d	
Milestone: Product statement defined	wk1, d3	wk1, d3	wk1, d3	wk1, d3			
I.1.2 Define desired output/control/input (OCI)							
Scope keyboard functions	wk1, d4	wk1, d4	wk2, d2		BLS	1.5 p-d	
Scope voice input functions	wk1, d3	wk1, d3	wk2, d2		JPP	2 p-d	
Scope modes of interaction	wk2, d1		wk2, d3		MLL	1 p-d	
Scope document diagnostics	wk2, d1		wk2, d2		BLS	1.5 p-d	
Scope other WP functions	wk1, d4	wk1, d4	wk2, d3		JPP	2 p-d	
Document OCI	wk2, d1		wk2, d3		MLL	3 p-d	
FTR: Review OCI with customer	wk2, d3		wk2, d3		all	3 p-d	
Revise OCI as required	wk2, d4		wk2, d4		all	3 p-d	
Milestone: OCI defined	wk2, d5		wk2, d5				
I.1.3 Define the function/behavior							

5.8.5 Tracking Schedule

Project schedule is the most important factor for software project manager. It is the duty of project manager to decide the project schedule and track the schedule.

Tracking the schedule means determine the tasks and milestones in the project as it proceeds.

Following are the various ways by which tracking of the project schedule can be done –

1. Conduct periodic meetings. In this meeting various problems related to the project get discussed. The progress of the project is reported to the project manager.
2. Evaluate results of all the project reviews.
3. Compare 'actual start date' and 'scheduled start date' of each of the project task.
4. Determine if the milestones of the project is achieved on scheduled date.
5. Meet informally the software practioners. This will help the project manager to solve many problems. This meeting will also be helpful for assessing the project progress.
6. Assess the progress of the project quantitatively.

Thus for tracking the schedule of the project the project manager should be an experienced person. In fact project manager is the only responsible person who is controlling the software project. When some problems occur in the project then addition resources may be demanded, skilled and experienced staff may be employed or project schedule can be redefined.
For handling the severe deadlines, project manager uses a technique of time boxing. In this technique each it is under stood that the complete product cannot be delivered on given time. Part by part i.e. in the series of increments the product can be delivered to the customer.
The project manager uses time box technique means he is associating each task with a box. That means each task is put in a "time box" and within that time frame each task must be completed. When the current task reaches to boundary of its time box, then the next task must be started (even if current task is remaining incomplete).

Some researchers had argued upon - leaving the task incomplete when current task reaches to the boundary but for this argument the counterpart is that even if the task is remaining incomplete it reaches to almost completion stage and remaining part of it can be completed in the next successive increment.

5.9 Earned Value Analysis

Earned Value Analysis (EVA) is technique of performing quantitative analysis of the software project.

Earned value system provides a common value scale for every task of software project.

The EVA acts as a measure for software project progress.

With the help of quantitative analysis made in EVA, we can know how much percentage of the project is completed.

The earned value analysis can be made using following steps.

1) The Budgeted Cost of Work Scheduled (BCWS) is an estimated cost for the work that has been scheduled. This value is obtained for every individual task in the software project. In this activity the work of each software engineering task is planned. The $BCWS_i$ is the effort planned for work task i. Hence at every point in the progress of project the $BCWS_i$ are calculated and the total cost is the summation of all the $BCWS_i$.

2) At the completion of the project the BCWS values for all work tasks are summed to derive the budget of the project. The calculation of budgeted actual cost is

$$BAC = \sum(BCWS_i) \text{ for all tasks } i$$

3) Then budgeted cost of work performed (BCWP) is computed. The value of BCWP is the sum of all the BCWS values of all the corresponding tasks that have actually been completed by a point in time on the project schedule.

The difference between BCWS and BCWP is that BCWS represents values for the project activities that are planned and BCWP represents the values of the project activities that are Completed.

Various types of computations in EVA are given as follows

1) $SPI = BCWP/BCWS$

Where SPI is the software performance index. It represents the project efficiency. If the value of SPI is 1.0 then it represents that execution of project is very efficient.

2) $SV = BCWP - BCWS$

Where SV indicates the scheduled variance.

3) $Project\ scheduled\ for\ completion = BCWS/BAC$

Where project scheduled for completion indicates the percentage of work which should be completed by time t.

$$Percent\ complete = BCWP/BAC$$

Percent complete represents the percent of project which is actually completed by time t.

4) $ACWP = \sum Efforts\ expended\ on\ work\ task\ that\ have\ been\ completed\ by\ time\ t.$

Where ACWP refers to .Actual Cost Work Performance. This value helps in computing the cost factor of the project.

$$CPI - BCWP/ACWP$$

Where CPI indicates the cost performance index. This value represents whether the performance of project is within the defined budget or not. The value 1.0 indicates that the project is within the defined budget

Thus EVA helps in identifying the project performance, cost of performance and project scheduling difficulties. This ultimately helps the project manager to take the appropriate corrective actions.

5.9.1 Error Tracking

While developing the software project many work products such as SRS, design document, source code are being created. Along with these work products many errors may get generated. Project manager has to identify all these errors to bring quality software.

Error tracking is a process of assessing the status of the software project.

The software team performs the formal technical reviews to test the software developed. In this review various errors are identified and corrected. Any errors that remain uncovered and are found in later tasks are called defects.

The defect removal efficiency can be defined as

$$DRE = E/(E + D)$$

where

DRE is the defect removal efficiency,

E is the error

D is defect.

The DRE represents the effectiveness of quality assurance activities. The DRE also helps the project manager to assess the progress of software project as it gets developed through its scheduled work task.

During error tracking activity, following metrics are computed

1. Errors per requirements specification page : denoted by E_{req}
2. Errors per component - design level : denoted by E_{design}
3. Errors per component - code level : denoted by E_{code}
4. DRE - requirement analysis
5. DRE - architectural design
6. DRE - component level design
7. DRE - coding

The project manager calculates current values for E_{req}, E_{design} and E_{code}.
These values are then compared with past projects. If the current result differs more than 20 % from the average, then there may be cause for concern and investigation needs to be made in this regard.

These error tracking metrics can also be used for better target review and testing resources.

5.10 Project Plan

A project plan must be prepared in advance from the available information. The project planning is an iterative process and it gets completed only on completion of the project. This process is iterative because new information gets available at each phase of project development. Hence the plan needs to be modified on regular basis for accommodating new requirements of the project.

Following is a structure of project plan -

Project plan

1.1. Introduction

The goals, objectives and constraints of the project must be described in this section.

1.2. Project organization

The organization of development team should be described. The number of people involved along with their roles must be described in detail.

1.3. Risk analysis

All possible risks should be identified. Not only this but, the possible risk reduction strategies must be decided.

1.4. Hardware and software requirements

This section specifies the required hardware and software.

1.5. Work breakdown

Various project activities are grouped together to define the project work breakdown. Deciding the project milestone and deliverables is an important task in defining the work breakdown.

1.6. Project schedule

The tentative schedule of project activities must be determined.

1.7. Report generation

The structure of the project report, when it should be generated must be decided.

5.11 Risk Management

Definition of risk: The risk denotes the uncertainty that may occur in the choices due to past actions and risk is something which causes heavy losses.

Definition of risk management: Risk management refers to the process of making decisions based on an evaluation of the factors that threats to the business.

Various activities that are carried out for risk management are –

[1] Risk identification
[2] Risk projection
[3] Risk refinement
[4] Risk mitigation, monitoring and management.

5.11.1 Software Risks

There are two characteristics of the risks

1. The risk may or may not happen. It shows the uncertainty of the risks.
2. When risks occur, unwanted consequences or losses will occur.

5.11.2 Different types of risk

1. Project risk

Project risks arise in the software development process then they basically affect budget, schedule, staffing, resources, and requirements. When project risks become severe then the total cost of project gets increased.

2. **Technical risk**

These risks affect quality and timeliness of the project. If technical risks become reality then potential design implementation, interface, verification and maintenance problems gets created. Technical risks occur when problem becomes harder to solve.

3. **Business risk**

When feasibility of software product is in suspect then business risks occur. Business risks can be further categorized as

1. **Market risk**

   When a quality software product is built but if there is no customer for this product then it is called market risk (i.e. no market for the product).

2. **Strategic risk**

   When a product is built and if it is not following the company's business policies then such a product brings strategic risks.

3. **Sales risk**

   When a product is built but how to sell is not clear then such a situation brings sales risk.

4. **Management risk**

   When senior management or the responsible staff leaves the organization then management risk occurs.

5. **Budget risk**

   Losing the overall budget of the project is called budget risk.

Another categorization of risk proposed by Charette is –

1. Predictable Risk
2. Unpredictable Risk

Known risks are those risk that are identified after evaluating the project plan. These risks can also be identified from other sources such as environment in which the product gets developed, unrealistic deadlines, poor requirement specification and software scope. There are two types of known risks - predictable and unpredictable risks.

Predictable risks

Predictable risks are those risks that can be identified in advance based on past project experience.

For example: Experienced and skilled staff leaving in between or improper communication with customer resulting in poor requirement specification.

Unpredictable risks

Unpredictable risks are those risks that cannot be guessed earlier.

For example certain changes in Government policies may affect the business project.

5.11.3 Reactive and Proactive Risk

Reactive and proactive risk strategies are the approaches used for managing the risks.

Reactive risk strategy

- Reactive risk management is a risk management strategy in which when project gets into trouble then only corrective action is taken. But when such risks cannot be managed and new risks come up one after the other, the software team flies into action in an attempt to correct problems rapidly. These activities are called "firefighting" activities.
- Resources are utilized to manage such risks. And if still the risks do not get managed then project is in danger.
- In this strategy no preventive care is taken about the risks. They are handled only on their occurrences.
- This is an older approach of risk management.

Proactive risk strategy

- Proactive risk management strategy begins before the technical activity by considering the probable risk.
- In this strategy potential risks are identified first then their probability and impact is analyzed. Such risks are then specified according to their priorities (i.e. high priority risks should be managed first!). Finally the software team prepares a plan for managing these risks.
- The objective of this strategy is to avoid the risks *(prevention is better than cure!!!)*. But it is not possible to avoid all the risks, hence team prepares the risk management plan in such a manner that risk controlling can done efficiently.
- This is an intelligent strategy for risk management and now a day it is used by most of the IT industries.

5.11.4 Risk Identification

Risk identification can be defined as the efforts taken to specify threats to the project plan. Risks identification can be done by identifying the known and predictable risks.

The risk identification is based on two approaches

1) **Generic risk identification**

   It includes potential threat identification to software project.

2) **Product-specific risk identification**

   It includes product specific threat identification by understanding people, technology and working environment in which the product gets built.

Normally the risk identification is done by the project manager who follows following steps -

Step 1: Preparation of risk item check list

The risk items can be identified using following known and predictable components

1) **Product size** - The risk items based on overall size of the software product is identified.
2) **Business impact** - Risk items related to the marketplace or management can be predicted.
3) **Customer characteristics** - Risks associated with customer-developer communication can be identified.
4) **Process definition** - Risks that get raised with the definition of software process. This category exposes important risks items because whichever is the process definition made, is then followed by the whole team.
5) **Development environment** - The risks associated with the technology and tool being used for developing the product.
6) **Staff size and experience** - Once the technology and tool related risks items are identified it is essential to identify the risk associated with sufficient highly experienced and skilled staff who will do the development.
7) **Technology to be built** - complexity of the system should be understood and related risk items needs to be identified.

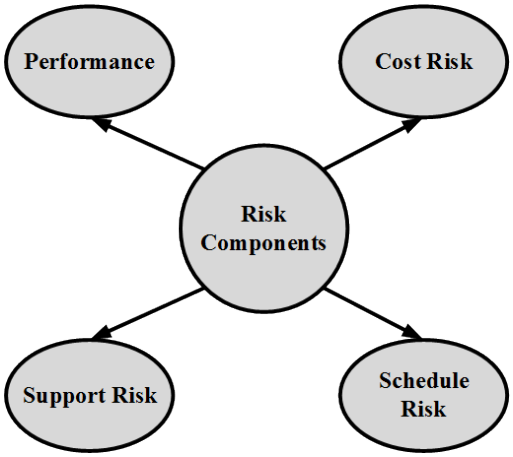
After preparing a risk item checklist a questionnaire is prepared. These set of questions should be answered and based on these answers the impact or seriousness of particular risk item can be judged.

Step 2: Creating risk components and drivers list.

The set of risk components and drivers list is prepared along with their probability of occurrence. Then their impact on the project can be analyzed.

Risk Components and Drivers

U.S. Air force has written a guideline for risk identification which is based on identification of risk component and risks drivers. It has suggested following types of risk components -



[1] Performance risk

It is the degree of uncertainty that the product will satisfy the requirements

[2] Cost risk

It is the degree of uncertainty that the project will maintain the budget.

[3] Support risk

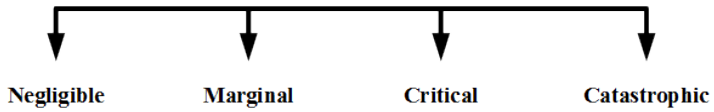
It is the degree of uncertainty that the software project being developed will be easy to correct, modify or adapt.

[4] Schedule risk

It is the degree of uncertainty that the software project will maintain the schedule and the project will be delivered in time.

Associated with these components are the risk drivers that are used to analyses the impact of risk. These four risk drivers are listed below

For the risk impact assessment a table is built in which impact of each risk driver on each software component can be specified.



How to Asses Overall Project Risk ?

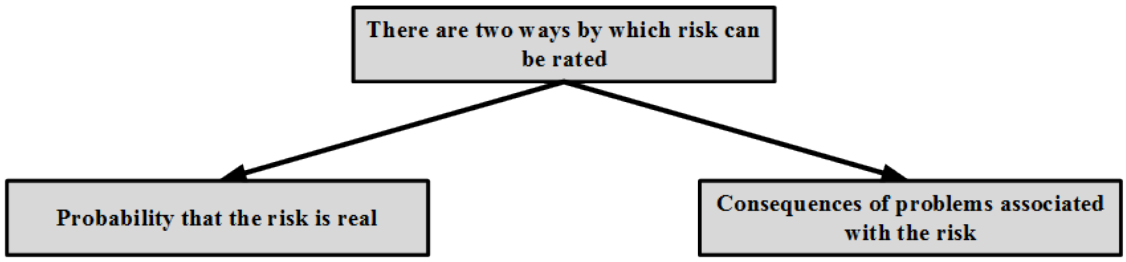
The best approach is to prepare a set of questions that can be answered by project managers in order to assess the overall project risks. These questions can be

1. Will the project get proper support by the customer manager?
2. Are the end-users committed to the software that has been produced?
3. Is there a clear understanding of requirements?
4. Is there an active involvement of the customer in requirement definition?
5. Is that the expectations set for the product are realistic?
6. Is project scope stable?
7. Are there team members with required skills?
8. Are project requirements stable?
9. Does the technology used for the software is known to the developers?
10. Is the size of team sufficient to develop the required product?
11. Is that all the customers know the importance of the product/ requirements of the system to be built?

Thus the number of negative answers to these questions represents the severity of the impact of the risk on overall project.

5.11.5 Risk Projection

The risk projection is also called risk estimation.



The project planner, technical staff, project manager performs following steps to perform following steps for risk projection -

- ❖ Establish a scale that indicates the probability of risk being real.
- ❖ Enlist the consequences of the risk.
- ❖ Estimate the impact of the risk on the project and product.
- ❖ Maintain the overall accuracy of the risk projection in order to have clear understanding of the software that is to be built.

These steps help to prioritize the risks. Once the risks are prioritized then it becomes easy to allocate the resources for handling them.

Building Risk Table

Building the risk table is the simplest and most commonly used technique adopted by project managers in order to project the risks. The sample risk table is as given below

Risk Table

	Category	Probability	Impact	RMMM
Is the skilled staff available	Staff	50 %	Catastrophic	
Is that the team size sufficient	Staff	62 %	Critical	
Have the staff received sufficient training	Staff	25 %	Marginal	
Will technology meet the expectations?	Technology	30 %	Critical	
Is the software management tool	Environment	40 %	Negligible	
How much amount of j reused software is	Project size	60 %	Marginal	
Will customer change the requirement?	Customer	20 %	Critical	

While building the risk table

The project team first of all enlists all probable risks with the help of risk item checklist.

Each risk is then categorized. As we know various categories of risk can be

1. Project size
2. Technology
3. Customer
4. Staff
5. Business
6. Developing environment.

Probability of occurrence of each risk is then estimated by each team member individually.

Then impact of each risk is assessed. While calculating the impact of each risk, each using the cost drivers each component of risk (performance, cost, support, and schedule) is assessed and it then averaged to quote the overall impact of particular risk.

After building this table it is then sorted by probability and impact. The high probability and high impact risks will be at the top of the table. And low probability and low impact risk will be at the bottom of the table. This arrangement of the table is called first-order prioritization.

Then the project manager goes through this first-order prioritized risk table and draws a horizontal line at some point in the table. This line is called cut off line. The risks table above the cut off line is now considered for further risk analysis,

The risk table below the cut off line is again sorted and a second-order prioritization is applied on this table.

The risk table above the cut-off line is having the risks with high probability and high impact and such risks should occupy the significant amount of management time.

All the risks that lie above the cut off line should be managed. Using Risk mitigation, monitoring and management plan the last column of the risk table is filled up.

Assessing Risk impact

While assessing the risks impact three factors are considered

- Nature of risk
- Scope of the risk
- Timing at which risk occurs

Nature of risk denotes the type or kind of risk. For example if software requirement is poorly understood, the software processes gets poorly designed and ultimately it will create a problem in unit testing. Scope of the risk means severity of the risk. And timing of risk means determining at which phase of software development life cycle the risk will occur and how long it will persist.

U.S. Air Force has suggested following steps in order to determine the impact of risk -

1. The probability of all the components of risk (performance, cost, support and schedule) is calculated and averaged.
2. Using risk drivers (catastrophic, critical, marginal, negligible) the impact of risk on each components is determined.
3. Build the risk table and analyses the high impact, high probability risks.

Risk exposure

The risk exposure can be calculated by following formula

$$Risk\ Exposure = Probability\ of\ occurrence\ of\ risk \times Cost$$

For example: Consider a software project with 77 percent of risk probability in which 15 components were developed from the scratch. Each component have on an average 500 LOC and each LOC have an average cost of $10. Then the risk exposure can be calculated as ,

$$First\ of\ all\ we\ will\ compute\ cost$$
$$= Number\ of\ components \times LOC \times cost\ of\ each\ LOC$$
$$= 15 \times 500 \times 10 = \$75000$$

Then

$$Risk\ Exposure = Probability\ of\ occurrence\ of\ risk \times Cost$$
$$= 77 / 100 \times 75000 = \$57750$$

Thus risk exposure for each risk from risk table is calculated. The total risk exposure of all risks helps in determining the final cost of the project.

5.11.6 Risk Refinement

Risk refinement is a process of specifying the risk in more detail. The risk refinement can be represented using CTC format suggested by D.P.Gluch.

The CTC stands for condition-transition-consequence. The condition is first stated and then based on this condition sub conditions can be derived. Then determine the effects of these sub conditions in order to refine the risk. This refinement helps in exposing the underlying risks. This approach makes it easier for the project manager to analyze the risk in greater detail.

Risk Mitigation, Monitoring & Management

5.12 RMMM

RMMM stands for risk mitigation, monitoring and management. There are three issues in strategy for handling the risk is

[1] Risk avoidance
[2] Risk monitoring
[3] Risk management.

Risk mitigation

Risk mitigation means preventing the risks to occur (risk avoidance). Following are the steps to be taken for mitigating the risks.

1. Communicate with the concerned staff to find of probable risk.
2. Find out and eliminate all those causes that can create risk before the project starts.
3. Develop a policy in an organization which will help to continue the project even though some staff leaves the organization.
4. Everybody in the project team should be acquainted with the current development activity.
5. Maintain the corresponding documents in timely manner. This documentation should be strictly as per the standards set by the organization.
6. Conduct timely reviews in order to speed up the work.
7. For conducting every critical activity during software development, provide the additional staff if required.

Risk monitoring

In risk monitoring process following things must be monitored by the project manager,

1. The approach or the behavior of the team members as pressure of project varies.
2. The degree in which the team performs with the spirit of "team-work".
3. The type of co-operation among the team members.
4. The types of problems that are occurring.
5. Availability of jobs within and outside the organization.

The project manager should monitor certain mitigation steps. For example.

If the current development activity is monitored continuously then everybody in the team will get acquainted with current development activity.

The objective of risk monitoring is

1. To check whether the predicted risks really occur or not.
2. To ensure the steps defined to avoid the risk are applied properly or not.
3. To gather the information which can be useful for analyzing the risk.

Risk management

Project manager performs this task when risk becomes a reality. If project manager is successful in applying the project mitigation effectively then it becomes very much easy to manage the risks.

For example, consider a scenario that many people are leaving the organization then if sufficient additional staff is available, if current development activity is known to everybody in the team, if latest and systematic documentation is available then any "new comer' can easily understand current development activity. This will ultimately help in continuing the work without any interval.

5.13 RMMM Plan

The RMMM plan is a document in which all the risk analysis activities are described. Sometimes project manager includes this document as a part of overall project plan. Sometimes specific RMMM plan is not created, however each risk can be described individually using risk information sheet. Typical template for RMMM plan or Risk information sheet can be,

Risk information sheet			
Risk ID: P02-4-32	Date: 5/9/09	Prob: 80%	Impact: high
Description: Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.			
Refinement/context: Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards. Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components. Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.			
Mitigation/monitoring: 1. Contact third party to determine conformance with design standards. 2. Press for interface standards completion; consider component structure when deciding on interface protocol. 3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.			
Management/contingency plan/trigger: *RE* computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly. Trigger: Mitigation steps unproductive as of 7/1/09.			
Current status: 5/12/09: Mitigation steps initiated.			
Originator: D. Gagne		Assigned: B. Laster	

The risk information sheet can be maintained by database systems. After documenting the risks using either RMMM plan or Risk information sheet the risk mitigation, monitoring and analysis activities are stopped.

5.14 CASE Tools

What is CASE ?

The Computer Aided Software Engineering (CASE) tools automate the project- nagement activities, manage all the work products.

Importance of CASE Tools

- The CASE tools assist to perform various software life cycle activities such as analysis, design, coding and testing.
- Software engineers and **project managers** make use of CASE tools.
- The use of CASE tools in the software development process **reduces** the significant amount of **efforts.**

- CASE is used in conjunction with the process model that is chosen.
- CASE tools help software engineer to produce the **high quality software efficiently.**
- Use of CASE tool **automates** particular **task of software development activity.** But it is always useful to use a set of CASE tools instead of* using a single CASE tool. If different CASE tools are not integrated then the data generated by one tool will be an input to other tools. This may also require a format conversions, as tools developed by different vendors may use different formatting. There are chances, that many tools do not allow exporting data and maintain data in proprietary formats.

5.14.1 Building Blocks of CASE

The CASE tools may exist as a single tools or a collection of multiple tools. These _tools must communicate with various elements such as hardware, database, people, network, operating system and so on. This communication creates an **integrated environment.**

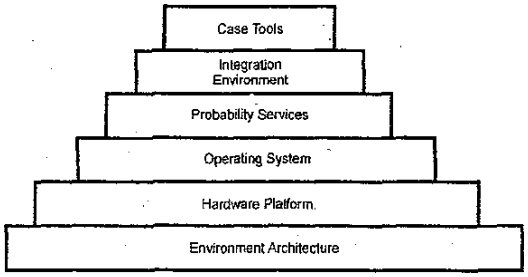


Fig. 5.10.1 Building block for CASE Toois

Fig. 5.10.1 represents the building block for CASE. The bottom most layer **consists of C environment architecture** and **hardware platform.** The environment architecture **consists** of collection of system software and human work pattern "that is applied during the software engineering process.

- A set of probability services connects the CASE tools with the integration framework.
- The **integration framework** is a collection of specialized programs which allows
- the CASE tools to communicate with the database and to create same look and (feel for the end-user. Using probability services CASE tools can communicate with the cross platform elements.
- At the top of this building block a collection of CASE tools exist. CASE tools , basically assist the software engineer in developing a complex Component.
- CASE tools can exist in variety of manner. A single CASE tool can be used, or a , collection of CASE tools may exists which acts as some package. The CASE tools may serve as a bridge between other tools.

5.14.2 Taxonomy

- To create an effective CASE environment, various categories of tools can be developed. ,
- CASE tools can be classified by
  1. by function or use (
  2. by user type (e.g. manager, tester), or
  3. by stage in software engineering process (e.g. requirements, test)
- The taxonomy of CASE tools is as given below.

Business process engineering tools
This tool is used to model the business information flow. It represents business data objects, their relationships and how data objects flow between different business areas / within a company.

Process modeling and management tools
It models software processes. First the processes need to be understood then only it could be modeled. This tool represents the key elements of the processes. Hence it is possible to carry out work tasks efficiently.

Project planning tools
These tools help to plan and schedule projects. Examples are PERT arid CPM. The objective of this tool is finding parallelism and eliminating bottlenecks in the projects.

Risk analysis tools
It helps in identifying potential risks. These tools are useful for building the risk table and thereby providing detailed guidance in identification and analysis of risks. Using his tool one can categorize the risks as catastrophic, critical, marginal, or negligible. A cost is associated with each risk which can be calculated at each stage of development.

Project management tools
These track the progress of the project. These tools are extension to the project planning tools and the use of these tools is to update plans and schedules.

Requirements tracing tools
The objective of these tools is to provide a systematic approach to isolate customer requirements and then to trace these requirements in each stage of development.

Metrics and management tools
These tools assist to capture specific metrics that provide an overall measure of quality. These tools are intended to focus on process and product characteristics. For ample "defects per function point", "LOC/person-month".

Documentation tools
Most of the software development organizations spend lot of time in developing the documents. For this reason the documentation tools provide a way to develop documents efficiently. For example - word processors that give templates for the organization process documents.

System software tools
These tools provide services to network system software, object management and distributed component support. For example - e-mail, bulletin boards, and www access.

Quality assurance tools
These are actually metrics tools that audit source code to insure compliance with language standards.

Database management tool
It provides consistent interfaces for the project for all data, in particular the [figuration objects are primary repository elements.

Software configuration management tools
It assists with identification, version control, change control, auditing, and status accounting.

Analysis and design tools

It creates models of the system. Some create formal models. Others construct data flow models. These models contain representation of data, function and behavior. Such tools helps in architectural, component level and interface design.

PRO/SIM tools
These are prototyping and simulation tools. They can help predict real time system response and allow mockups of such systems to be fashioned.

Interface design and development tools
These tools are used in developing user interface. It includes various components such as menu, icons, buttons, scrolling mechanisms etc. For example - JAVA, Visual Studio

Prototyping tools
These tools support to define screen layout rapidly for interactive applications.

Programming tools
The programming tool category include the programs that support most of the conventional programming languages. For example - compilers, debuggers, editors, database query languages.

Web development toots
These tools help in developing the web based applications. The various components of these tools are text, graphics, form, scripts, and applets.

Integration and testing tools
These tools include various category of tools such as data acquisition tools, static measurement, dynamic measurement, simulation, cross functional tools.

Static analysis tools
The static testing tools are used for deriving the test cases. There are three types of static testing tools.

a. Code based testing tools - These tools take source code as input and generate test cases.
b. Specialized testing language - Using this language the detailed test specification can be written for each test case.
c. Requirement-based testing tools - These tools help in designing the test cases as per user requirements.

Dynamic analysis tools
These interact with an executing program, checking path coverage, and testing assertions. Intrusive tools insert code in the tested program. Non-intrusive tools use a separate hardware processor.

Test management tools
These tools manage and co-ordinate regression testing, perform comparisons of put, and act as test drivers.

Client/server testing tools
The client server tools are used in client server environment to exercise the GUI and work communication requirements for client and server.

Reengineering tools
These tools perform a set of maintenance activities. These tools perform various functions such as

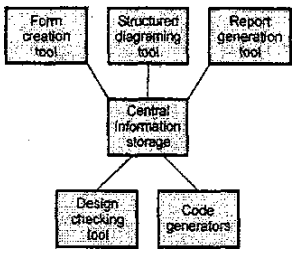
- Reverse engineering to specification tools.
- Code restructuring.
- On-line system re-engineering.

5.14.3 Workbenches

- CASE workbench is a set of tools which supports a particular phase in the software process.
- These tools work together to provide the complete support to the software development activity.

In the workbench common services are provided which are used by all the tools, te kind of data integration is also supported

- For example **-Analysis and Design Workbench** (Refer Fig. 5.10.2)
    - It support the generation of system models during design and analysis activities.
    - It provides graphical editors plus shared repository.
    - It may include code generators to create source code from design information



Advantages

- It is available at low cost.
- There is productivity improvement due to support of workbenches.
- It results in standardized documentation for software system.

Drawbacks

- These systems are closed environments with tight integration with tools.
- The import and export facilities for various types of data is limited.
- It is difficult to adapt method for specific organizational need.

Environments

- CASE tools create a pool of software engineering information. The integrated CASE environment allows a transfer of information into and out of this pool. For such transfer there is a need for some architectural components. These components are -
    - Database for storing the information
    - Object management system. Because using the objects information can be transferred in the information pool.
    - Control mechanism.
    - User interface
- The Fig. 5.14.3 shows the simple model for integrated CASE environment. This environment consists of various levels.

<table>
<tr><td colspan="10">User Interface Layer
(Interlace tool kit. Presentation Protocol)</td></tr>
<tr><td colspan="10">Tool Management Layer</td></tr>
<tr><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td><td>CASE Tools</td></tr>
<tr><td colspan="10">Object Management Layer
(Integration services, configuration management services)</td></tr>
<tr><td colspan="10">Shared Repository Layer
(CASE database, Access control functions)</td></tr>
</table>

Fig. 5.14.3 Model for integrated CASE environment

- The **user interface layer** consists of **interface tool kit** and **presentation protocols.** The **interface tool kit** consists of collection of software required for interface management and display objects. The **presentation protocol** decides a common look and feel of the presentation interface.

Then comes a **tools layer.** It consists of set of Tools Management Services (IMS).

- The next layer is **Object management Layer (OML).** It performs the configuration management. The services of this layer allows the identification of all the configuration objects. So that the case tool can be plugged into the integrated CASE environment.
- The bottom most layers is shared **repository layer.** It consists of **CASE database** and **access control functions.** These access control functions help the **object** management layer to access the CASE Database.

Components of CASE

Various components of CASE tools are -

1. **Central Repository:**
   - The central repository contains the common, integrated and consistent information
   - The central repository acts like a data dictionary
   - It contains product specification, requirement documents, project management information and reports.
2. **Upper Case-Tools**
   - Uppercase tool focus on the planning, analysis phase and sometimes the design phase of the software development lifecycle.
3. **Lower Case Tools :**
   - Lower CASE Tool Computer-Aided Software Engineering (CASE) software tool that directly supports the implementation (programming) and integration tasks.
   - Lower CASE tools support database schema generation, program generation, implementation, testing, and configuration management
4. **Integrated CASE tools :**
   - These are type of tools that integrate both upper and lower CASE, for example making it possible to design a form and build the database to support it at the same time.
   - An automated system development environment that provides numerous tools to create diagrams, forms and reports.
   - It also offers analysis, reporting and code generation facilities and seamlessly shares and integrates data across and between tools.

Part – A

Estimation.

1. What does empirical estimation model means?

The empirical estimation model uses empirically (experimentally) derived formulae to predict effort as a function of lines of code or function point. In empirical estimation model, the empirical values of LOC of FP are put. The empirical model supports a limited number of software project. Typical estimation model is derived using regression analysis on data collected from past software projects.

2. Define basic equation for the effort estimation models. **[ND 2012]**

The basic equation for the effort estimation models are - $E = A + B \times (eV)c$

Where E = Effort in person-month

A, B and C = are empirically derived constants.

eV = Estimation variable either LOC or FP based method.

3. What do you mean by estimation risk?

Estimation risk is measured by the degree of uncertainty in the quantitative estimates for cost, schedule and resources.

FP Based & LOC Based Estimation

4. What is called the function point (FP)?

A function point is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points measure software size. The cost (in dollars or hours) of a single unit is calculated from past projects.

5. How is productivity and cost related to function points? **[ND 2016]**

Once the functional point is calculated then we can compute various measures as follows

- ❖ Productivity = FP/person-month
- ❖ Quality = Number of faults/FP
- ❖ Cost = $/FP
- ❖ Documentation = Pages of documentation/FP.

6. List out the different approaches to size of the software.

The two approaches used for estimating the size of the software are

- LOC i.e. computing the lines of code.
- FP i.e. computing function point of the program.

7. Differentiate between size oriented and function oriented metrics.

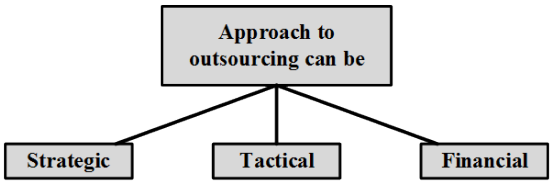
Size oriented metrics	Function oriented metrics
Size oriented software metrics is by considering the size of the software that has been produced.	Function oriented metrics use a ; measure of functionality delivered by the software

For a size oriented metric the software organization maintains simple records in tabular form. The typical table entries are: Project name, LOC, Effort, Pages of documents, errors, defects, total number of people working on project.	Most widely used function oriented metric is the Function Point (FP) computation of the function point is based on characteristics of software's information domain and complexity.

Make/Buy Decision

8. Define outsourcing.

Outsourcing is a process in which software engineering activities are contracted to a third party who does the work at lowest cost with high quality.



9. What are the advantages of Outsourcing?

[1] Cost savings

If a software is outsourced then people and resource utilization can be reduced. And thereby the cost of the project can be saved effectively.

[2] Accelerated development

Since some part of software gets developed simultaneously by a third party, the overall development process gets accelerated.

10. List out the disadvantages of outsourcing.

[1] A software company loses some control over the software as it is developed by third person.

[2] The trend of outsourcing will be continued in software industry in order to survive in competitive world.

COCOMO II

11. Mention difference between organic mode and embedded mode in COCOMO model [May/June 2014]

Organic mode	Embedded mode
The small size projects are handled using organic mode.	The large size projects are handled using embedded mode.
The experienced developers develop such projects.	The embedded mode projects are handled by little previous experienced people,

12. List out the advantages of COCOMO II Model.

The following are the major aspects of COCOMO II model that favour its usage,

- Five exponential scale factors
- Adaptation Adjustment Multiplier
- Three sizing options
- Three levels of cost model granularity

- Three development processes
- New cost drivers
- Bayesian calibration

13. What are the drawbacks of COCOMO II Model?

1. It is hard to accurately estimate KDSI early on in the project, when most effort estimates are required.
2. KDSI, actually, is not a size measure it is a length measure.
3. Extremely vulnerable to misclassification of the development mode
4. Success depends largely on tuning the model to the needs of the organization, using historical data which is not always available

Project Planning and Risk Management

14. What is project planning?

Project planning is an activity in which the project plan is prepared. This plan is mainly concerned with schedule and budget.

15. What are software planning activities?

The project planning activities can be carried out in following steps.

1. Identify the constraints in the project.
2. Make initial assessment of the project.
3. Define project milestones and deadlines.
4. While developing the project.
    a. Prepare project schedule.
    b. Initiate project activities as per the project schedule.
    c. Review progress of the project.
    d. Update project schedule if required.
    e. Revise project constraints and deliverables.
    f. If any problem arises then perform technical review.
5. Repeat the step 4 until the project is ready.

16. What is risk assessment?

Risk assessment is an activity of assessing likelihood of the risk and impact of that risk. It is given to be a triplet as $[R_i, I_i, X_i]$ where is R_i, risk, I_i, is likelihood of risk and X_i is the impact of the risk.

17. Write down the algorithm used in Risk assessment.

[1] For the given project define the risk reference level.
[2] Establish the relationship between the triplet $[R_i, I_i, X_i]$ and risk reference level.
[3] Predict the set of reference points that define a region of termination bounded by area of certainty.
[4] Predict how compound combinations of risks will affect a reference level.

18. Highlight the activities in project planning. **[MJ 2015]**

The following are the activities involved in project planning

1. Identify project objectives
2. Identify project infrastructure
3. Analyze project characteristics
4. Identify products and activities
5. Estimate effort for activity
6. Identify activity risks

7. Allocate resources
8. Review/ publicize plan
9. Execute plan
10. Lower level planning

19. What is risk? Give an example of risk. Types of Risks? [ND 2015]

The software risk can be defined as a problem that can cause some loss or can threaten the success of the project.

Example: Experienced and skilled staff leaving the organization in between.

Types of risks

1.Project Risks

2.Technical Risks

3. Business risks

RMMM

20. Define risk management [ND 2016]

Risk management is the identification, assessment, and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events or to maximize the realization of opportunities. Risk management's objective is to assure uncertainty does not deflect the endeavor from the business goals

21. What are risk management activities?

Following are risk management activities -

[1] Risk identification - All possible risks are anticipated and a list of potential risks is prepared.

.

[2] Risk analysis - Under this activity the after effects of risks are obtained and risks are prioritized accordingly.

[3] Risk planning - Risk avoidance or risk minimization plan is prepared.

[4] Risk monitoring - Identified risks are mitigated.

22. What are predictable risks? Give example for such risks.

Predictable risks are those risks that can be identified in advance based on past project experience.

For example: Experienced and skilled staff leaving the organization in between.

Scheduling and Tracking

23. What information does a software project plan provide?

Software project plan provides following information -

The quality plan describes the quality procedures and standards that will be used in a project.

The validation plan describes the approach, resources and schedule required for system validation.

The configuration management procedures and structures used are also described by the project plan.

The maintenance requirements of the system, maintenance cost and effort requirement information is described by the software project plan.

24. What is scheduling?

Scheduling is an activity that distributes estimated effort across the planned project duration. These efforts are related to software engineering tasks.

25. What is error tracking?
Error tracking is a process of finding out and correcting the errors that may occur during the software development process at various stages such as software design, coding or documenting.

26. What is the use of performing Earned Value Analysis? [ND 2018]
Earned Value Analysis (EVA) is an industry standard method of measuring a project's progress at any given point in time, forecasting its completion date and final cost, and analyzing variances in the schedule and budget as the project proceeds

27. State the importance of scheduling activity in Project management. [MJ 2015]
Software project scheduling is an activity that distributes estimated effort across the duration of project cycle by allocating effort to each specific task that is associated with all process.
Software project scheduling guides the following

- Compartmentalization of the project
- Correct allocation of time,
- Correct effort validation,
- Defining responsibility
- Defining outcomes

Process and Project Metrics

28. What is the standardization for the software metrics?
The standardization of software metrics indicates the attributes of effective software metrics. The software metrics should follow following standards.

Simple and computable - The software metrics should be easy to learn and derive.

Empirically and intuitively persuasive - The software metrics should satisfy the intuitive notions about the product attribute.

Consistent and objective - The software metrics should give the unambiguous results.

Consistent in its use of units and dimensions - The units and dimensions used in the software metrics should be consistent.

Programming language independent - The software metrics should be based on analysis model, design model and structure of the program. But it should be independent of syntax or semantics of the program.

Effective mechanism for high quality feedback - The software metric should provide information to software engineer whether the software product being developed is of quality or not.

29. What are project indicators and how do they help a project manager?
Project Indicators mean combination of metrics that provides insight into the software process, project or product. An indicator is a tool that helps you and your organization know how far your project is from achieving your goals and whether you are headed in the right direction.

30. Distinguish between direct and indirect measures of metrics.
Direct metrics - Direct metrics is directly measurable attribute. For example lines of code, execution speed, size of memory, some defects.

Indirect metrics - Indirect metrics are the aspects that are not immediately measurable. For example functionality, reliability, maintainability.

31. List down few process and product metrics. **[MJ 2016]**

Product metrics are

1. Size metric - This metric is used for measuring the size of the software.
    a. LOC based meritc - This metric makes use of lines of code to measure the software.
    b. FP based metric - This metric makes use of functional points to measure the software.
2. Complexity metric - A software module can be described by a control flow graph.
    a. Cyclomatic complexity
    b. McCabe complexity metric.
3. Quality metric -
    a. Defects
    b. Reliability metric
    c. Maintainability metric
4. Process metrics are based on following factors
    a. Application of methods and tools.
    b. Use of standards for the system.
    c. Effectiveness of management of system.
    d. Performance of development of system.

32. Define software measure. **[MJ 2015]**

Software measure is a numeric value for a attribute of a software product or process. There are two types of software measures-direct measure and indirect measure. The direct measure refers to immediately measurable attributes. For example lines of code. The indirect measure refers to the aspects that are not immediately quantifiable or measurable. For example functionality of the program.

33. Name the metrics for specifying non-functional requirements.

Reliability, response time, storage capacity, usability are the metrics for functional requirements.

34. Define software quality. **[MJ 2014]**

The software quality can be defined as the degree to which a system component or process meets specific requirements.

35. Write note on Risk Information Sheet. **[ND 2017]**

A risk information sheet is a means of capturing information about a risk. Risk information sheets are used to document new risks as they are identified. They are also used to modify information as risks are managed. It is a form that can be submitted to the appropriate person or included in a database with other project risks. In the absence of a database, this becomes a primary means of documenting and retaining information about a risk.

36. List out the principles of Project Scheduling **[ND 2017]**

Basic principles guide software project scheduling:

- Compartmentalization
- Interdependency
- Time allocation
- Effort allocation
- Effort validation
- Defined responsibilities
- Defined outcomes
- Defined milestones

37. What are the different types of productivity estimation measures? [MJ 2017]

There are various methods by which software productivity is measured, but whichever method is employed the goal should be uniform: to give software managers and professionals a set of useful, tangible data points for sizing, estimating, managing, and controlling software projects with rigor and precision (Jones 1). Some of the more common methods of measuring software productivity are Function Point Analysis, Constructive Cost Modeling, and Cyclomatic Complexity.

38. List two customer related and technology related risks. [MJ 2017]

Customer related risks

- Have you worked with he customer in the past?
- Does the customer have a solid idea of what is required?
- Has the customer taken the time to write this down?
- Will the customer agree to spend time in formal requirements gathering meetings to identify project scope?

Technology risks

- Is the technology to be built new to your company?
- Do the customer requirements demand the creation of new algorithms, input or output technology?
- Does the software interface with new or unproven hardware?

39. List two advnatges of using COCOMO Model. [MJ 2019]

Advantages of COCOMO estimating model are:

- COCOMO is factual and easy to interpret. One can clearly understand how it works.
- Accounts for various factors that affect cost of the project.
- Works on historical data and hence is more predictable and accurate.
- The drivers are very helpful to understand the impact on the different factors that affect the project costs.

40. Compare Project risk vs Business Risk. [MJ 2019]

- **Business risks** are largely about the decisions related to products and services offered in the market. When a company decides to manufacture and sell a specific product, there is always a risk with regards to the product not working as well as the company had expected. Sometimes marketing campaigns fail to sell a product. Other examples of business risks are changes in raw material costs, changes in shipping charges, new technological developments and so on.
- To put it very simply, business risks are typically more general as compared to project risks. Also, they would have an impact on nearly all aspects of the business.
- **Project risks** are different from business risks in the sense that they refer to an uncertain condition/event that may affect one or more project objectives.
- When estimating project risks, several people may end up listing all possible risks that could have an impact on the project. However, it is vital to do a careful assessment and see clearly whether you're listing specific project risks or common business risks. In the latter case, the list would be unmanageably lengthy.
- In other words, a business risk may be misinterpreted as a project risk. For example, key team members exiting the project/organization or going on a long sick leave should ideally be categorized as a business risk. It is not in the immediate risk management realm of the project and would be better handled by the HR department as part of the overall business process. In other words, this type of risk needs to be assessed under general business risks.

41. List CASE tools for the following phases od SDLC : Design ,Testing: [MJ 2019]

Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

42. What is budgeted cost of work scheduled? [ND 2019]

Budgeted Cost of Work Scheduled (BCWS) is the sum of the budgets for all work scheduled to be accomplished with a given time period. It also includes the cost of previous work completed and can address a specific period of performance or a date in time.

43. Write any two differences between 'Known Risks and predictable risks. [ND 2019]

Known Risk :

1) It can be uncovered after careful evaluation project plan, business and technical environment in which the project is being developed, other reliable information resources.

2) E.g. unrealistic delivery date, lack of software poor development environment.

Predictable Risk:

1) Predictable risks are extrapolated from past project experience.

2) E.g. staff turnover, poor communication with the customer, dilution of staff effort as ongoing maintenance requests are serviced.

PART B Questions Bank

Q.No	Questions	Univ. QP
1	Explain the various aspects of Project Management in detail	
2	What is Software Sizing? Explain why is it important for software estimation process [May/June 2013 – 8M]	[MJ 2013]
3	Explain in detail how Lines of Code can be used for Estimation with a suitable example [May/June 2014 – 16M]	[MJ 2016]
4	Explain the process of estimation using COCOMO II model in detail	[ND 2017/2015] [MJ 2016]
5	Why project planning is important? Explain the structure of a well-defined project plan	
6	Explain what are classified as risk and explain various types of risks.	
7	Discuss in detail how are risks identified and what are the various components and drivers used in identification of risks [OR] State the need for Risk Management and explain the activities under Risk Management. [May/June 2015 – 16M]	[MJ 2016] [MJ 2015] [ND 2017/2015]

8	Write a short note on RMMM	
9	Explain why scheduling is considered to be an important activity of Software Engineering process [OR] Write short note on Project Scheduling [May/June 2015 – 8M]	[MJ 2015]
10	Briefly discuss the role of Task Sets and Task Networks in effective implementation and tracking of software projects. [OR] Write a short note on Timeline Charts and Task Networks. [May/June 2015 – 8M]	[MJ 2015]
11	Explain why schedules are to be tracked in software development and why time line charts are used [Nov/Dec 2012 – 16M]	[ND 2012]
12	Discuss why Earned Value Analysis and Error Tracking are to be done and how they help in improving quality. (or) Explain Earned Value Analysis in Detail	[ND 2016] [MJ 2017]
13	Write a short note on various Qualitative metrics used in Process and Project Quality assessment.	
14	Explain in detail the various Metrics to measure the quality of the software that is being developed.	[ND 2015]
15	Explain with an example how function point based estimation is independent of the programming language being used. [Nov/Dec 2014 – 16M]	[ND 2014]
16	Identify the situation when outsourcing is done in a software project and what decision does an software engineer has to make.	
17	Define the term Risk Projection. Explain how it can be useful in assessing the impact on software development activities	
18	Explain the process of Cost estimation using COCOMO model in detail	[MJ 2017] [ND 2016]
19	Write Short Notes on Make./Buy Decision	[MJ 2016]
20	Explain Function Point concept in detail	[ND 2016]
21	Elaborate the cost estimation COCOMO II cost estimation model.	[ND 2019]
22	Present a detailed note on risk management	[ND 2019]
23	Outline the steps in function point analysis with an example.	[ND 2019]
24	List the features of LOC and FP Based estimation models. Compare the two models and list the advantages of one over other.	[MJ 2019]
25	Define Risk. List the types of risk and give example for each. List and explain the phases in risk management.	[MJ 2019]

University Questions

Part A

1. Highlight the activities in Project planning. **A/M 2015**

2. State the importance of scheduling activity in Project Management. **A/M 2015**
3. Define Risk and list its types. **N/D 2015**
4. Mr. Koushan is the project manager on a project to build a new cricket stadium in Mumbai, india after six months of work, the project is 27% complete. At the start of the project koushan estimated that it would cost $ 50,0000.000. What is the Earned value? **N/D 2015**
5. List a few process and project metrics. **M/J 2016**
6. Will exhaustive testing guarantee that the program is 100% correct? **M/J 2016**
7. What is risk management? **N/D 2016**
8. How is productivity and cost related to function points? **N/D 2016**
9. What are different types of productivity estimation measures? **A/M 2017**
10. List two customers related and technology related risks. **A/M 2017**
11. List out the principles of project scheduling. **N/D 2017**
12. Write a note on Risk Information Sheet (RIS). **N/D 2017**
13. What is EVA?

Part B

A/M 2015

1. State the need for Risk management and explain the activities under Risk Management. (16)
2. Write short notes on the following
   a. Project Scheduling (8)
   b. Project Timeline chart and Task network. (8)

N/D 2015

3. Explain in detail about the risk management in a software development life cycle. (16)
4. Discuss about COCOMO II model for software estimation. (10)
5. Discuss about the metrics for small organizations (6)

M/J 2016

6. Write short notes on the following :
   i) Make/Buy decision (8)
   ii) COCOMO II (8)
7. An application has the following 10 low external inputs, 8 high external outputs, 13 low internal logical files, 17 high external interface files, 11 average external inquires and complexity adjustment factor of 1.10. What are the unadjusted and adjusted function point counts? (4)
8. Discuss Putnam resources allocation model. Drive the time and effort equations. (12)

N/D 2016

9. Suppose you have a budgeted cost of a project as Rs. 9,00,000. The project is to be completed in 9 months. After a month, you have completed 10 percent of the project at a total expense of Rs. 1,00,000. The planned completion should have been 15 percent. You need to determine whether the project is on-time and on-budget? Use earned value analysis approach and interpret. (8)
10. Consider the following Function point components and their complexity. If the total degree of influence is 52, find the estimated function points. (8)

Function type	Estimated count	Complexity
ELF	2	7
ILF	4	10
EQ	22	4
EO	16	5
EI	24	4

11. Describe in detail COCOMO model for software cost estimation. Use it to estimate the effort required to build software for a simple ATM that produce 12 screens, 10 reports and has 80 software components. Assume average complexity and average developer maturity. Use application composition model with object points. (16)

A/M2017

12. Describe in detail COCOMO model for software cost estimation. Illustrate considering a suitable example. (13)

13. Given the following project plan of tables table1 and table2:

Table 1

ID	Task	Immediate Predecessor	Expected Duratic (days)	Budget($)
A	Meet with Client		5	500
B	Write SW	A	20	10000
C	Debug SW	B	5	1500
D	Prepare draft manu	B	5	1000
E	Meet with clients	D	5	1000
F	Test SW	C,E	20	2000
G	Make modification	F	10	8000
H	Finalize manual	G	10	5000
I	Advertise	C,E	20	8000

(*) all dependencies are assumed to be FS – Finsh to Start
And the following progress status:

Table 2

ID	Task	Status	Actual start(days)	Actual duration(days	Actual costs($)
A	Meet with client	100%			1500
B	Write SW	100%	+5days		9000
C	Debug SW	100%	+15days	+10 days	2500
D	Prepare draft manual	100%	As per other delays	+5 days	1000
E	Meet with clients	100%	As per other delays		1000
F	Test SW	100%	As per other delays		750
G	Make modifications	0%	As per other delays		0
H	Finalize manual	0%	As per other delays		0
I	Advertise	10%	+5 on top of other delays		1000

Perform an analysis of the project status at week13, using EVA. Use the CPI and SPI to determine project efficiency. Explain the process involved.

N/D 2017

14. Explain in detail about the risk management in a software development life cycle. (13)
15. Discuss about COCOMO II model for software estimation. (8)
16. Explain about the factors that cause difficulty in testing software. (5)

A/M2018

17. Describe in detail COCOMO model for software cost estimation. (9)
18. If Team A found 342 errors prior to release of software and Team B found 182 errors. What additional measures and metrics are needed to find out if the teams have removed the errors effectively ? Explain. (4)
19. Discuss the process of function point analysis. Explain function point analysis with sample case for components of different complexity. (13)

PART C

A/M2017

1. Consider the online book stores. It accept individual/bulk orders, process payments, triggers delivery of the books. Some of the major features of the system include:
    a. Order Books
    b. User friendly online shopping cart function
    c. Create, view, modify, and delete book to be sold
    d. To store inventory and sales information in database
    e. To provide an efficiency inventory system
    f. Register for book payment options
    g. Request book delivery

h. Add a wish list
i. Place request for book not available
j. To be able to print invoice to members and print a set of summary reports
k. Internet access.

Analysis the system using the context diagram and level 1 DFD for the system. Explain the components of DFD.

N/D 2017

2. List out the various umbrella activities which support software development process and discuss about their necessity in maintaining the quality in both software process and product that is being developed for railway reservation system. (15)
3. Model a data flow diagram for a :library Management System" State the functional requirements you are considering. (15)

A/M2018

4. What is the purpose of DFD? What are the components of DFD? Construct DFD for the following system:

   An on-line shopping system for XYZ provides many services and benefits to its members and staffs. Currently, XYZ staffs manually handle the purchasing information with the use of basic office software, such as Microsoft Office Word and Excel. It may results in having mistakes easily and the process is very inconvenient. XYZ needs an online shopping system at their Intranet based on the requirement. XYZ needs an online shopping system at their Intranet based on the requirement of users. XYZ online shopping system has five key features:

   i. To provide the user friendly online shopping cart function to members to replace hardcopoy ordering form:
   ii. To store inventory and sales information in database to resuce the human mistakes, increase accuracy and enhance the flexibility of information processing;
   iii. To provide an efficient inventory system which can help the xya staffs to gain enough information to update the inventory;
   iv. To be able to print invoice to members and print a set of summary reports for XYZ's internal usage;
   v. To design the system that is easy to maintain and upgrade.
5. Consider the problem of determining the number of different words in an input file. Carry out structured design by performing transform and transaction analysis construct the structured chart.