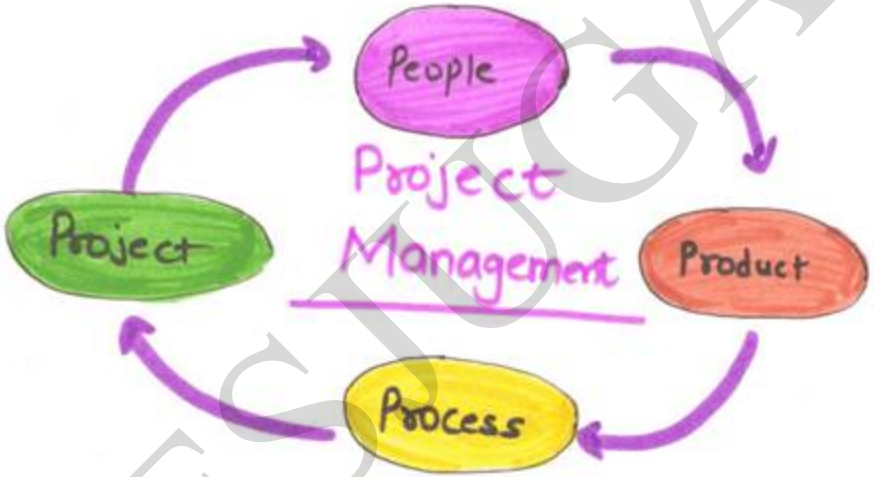
SOFTWARE ENGINEERING

Unit 2: Software Project Management Concepts



Shree Harsh Attri
(Assistant Professor)
Department: Bachelor of Computer Applications

JIMS Engineering Management Technical Campus
Greater Noida (U.P)

NOTESJUGAAD

THE MANAGEMENT SPECTRUM

Effective Software Project Management Focuses on 4 P's

1. **The People**
   - Deals with the cultivation of motivated, highly skilled people
   - Consists of the stakeholders, the team leaders, and the software team
2. **The Product**
   - Product objectives and scope should be established before a project can be planned
3. **The Process**
   - The software process provides the framework from which a comprehensive plan for software development can be established
4. **The Project**
   - Planning and controlling a software project is done for one primary reason…it is the only known way to manage complexity
   - all work required to make the product a reality

NOTESJUGAAD

THE PEOPLE

Software Engineering Institute has developed a **People Management Capability Maturity Model (PM-CMM)** to enhance the readiness of the software organizations to undertake increasingly Complex Applications by helping ***to attract, grow, motivate, deploy, and retain the talented needed*** to improve their software development capability.

The people management maturity model defines: ***recruiting, selection, performance management, training, compensation, career development, organization and work design, and team/culture development.***

NOTESJUGAAD

THE PEOPLE: THE STAKEHOLDERS

Five categories of Stakeholders

1. **Senior Managers** – define business issues that often have significant influence on the project.
2. **Project (Technical) Managers** – plan, motivate, organize, and control the practitioners who do the work.
3. **Practitioners** – deliver the technical skills that are necessary to engineer , a product or application.
4. **Customers** – specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
5. **End Users** – interact with the software once it is released for production use.

NOTESJUGAAD

THE PEOPLE: THE STAKEHOLDERS

jims Greater Noida

The Right People Skills

Qualities to look for in a **Team Leader** - MOI (Model of Leadership)

1. **Motivation** – the ability to encourage technical people to produce to their best ability.
2. **Organization** – the ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
3. **Ideas or Innovation** – the ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Team leaders should use a problem-solving management style

1. Concentrate on understanding the problem to be solved
2. Manage the flow of ideas
3. Let everyone on the team know, by words and actions, that quality counts and that it will not be compromised

NOTES.JUGAAD

THE PEOPLE: THE STAKEHOLDERS

The Team Leader must have following capabilities too:

1. **Problem Solving** – diagnose, structure a solution, apply lessons learned, remain flexible.

2. **Managerial Identity** – take charge of the project, have confidence to assume control, have assurance to allow good people to do their jobs.

3. **Achievement** – reward initiative, demonstrate that controlled risk taking will not be punished.

4. **Influence and Team Building** – be able to "read" people, understand verbal and nonverbal signals, be able to react to signals, remain under control in high-stress situations.

NOTESJUGAAD

THE PEOPLE: THE SOFTWARE TEAM

Seven project factors to consider when structuring a Software Development Team

1. The difficulty of the problem to be solved.
2. The size of the resultant program(s) in source lines of code.
3. The time that the team will stay together.
4. The degree to which the problem can be modularized.
5. The required quality and reliability of the system to be built.
6. The rigidity of the delivery date.
7. The degree of sociability (communication) required for the project.

NOTESJUGAAD

THE PEOPLE: THE SOFTWARE TEAM

Four Organizational Paradigms for Software Development Teams

1. **Closed Paradigm – traditional hierarchy** of authority; works well when *producing software similar to past efforts*; *members are less likely to be innovative.*
2. **Random Paradigm** – depends on **individual initiative** of team members; works well for projects requiring innovation or technological breakthrough; members may struggle when orderly performance is required.
3. **Open Paradigm** – hybrid of the closed and random paradigm; works well for solving ***complex problems***; *requires collaboration, communication, and consensus among members*.
4. **Synchronous Paradigm** – organizes team members based on the natural pieces of the problem; members have little communication outside of their subgroups.

NOTESJUGAAD

THE PEOPLE: THE SOFTWARE TEAM

Strengths and Weakness of Four Organizational Paradigms for Software Development Teams

1. **Closed Paradigm** –
   - Strengths: stability and predictable performance
   - Weaknesses: lack of innovation
2. **Random Paradigm** –
   - Strength: creative invention
   - Weakness: not highly stable or efficient
3. **Open Paradigm** –
   - Strength: complex problem solving (sharing of diverse opinions)
   - Weakness: slow due to debating issues
4. **Synchronous Paradigm** –
   - Strength: efficiently perform established procedures
   - Weakness: may not be highly responsive or adaptive to change.

NOTESJUGAAD

THE PEOPLE: THE SOFTWARE TEAM

Five factors that cause team toxify (i.e., a toxic team environment)

1. A hectic work atmosphere
2. High frustration that causes friction among team members
3. A fragmented or poorly coordinated software process
4. An unclear definition of roles on the software team
5. Continuous and repeated exposure to failure

How to avoid these problems?

- ✓ Give the team access to all information required to do the job
- ✓ Do not modify major goals and objectives, once they are defined, unless absolutely necessary
- ✓ Give the team as much responsibility for decision making as possible
- ✓ Let the team recommend its own process model
- ✓ Let the team establish its own mechanisms for accountability (i.e., reviews)
- ✓ Establish team-based techniques for feedback and problem solving

NOTESJUGAAD

THE PRODUCT

jims Greater Noida

Before a Project can be Planned:

- Product objectives and scope should be established
- Alternative solutions should be considered
- Technical and Management constraints should be identified like *Estimates of cost, effective assessment of risk, realistic breakdown of project tasks, or manageable project schedule.*

The scope of the Software Development must be established and bounded as follows:

1. **Context** – How does the software to be built fit into a ***larger system, product, or business context***, and what constraints are imposed as a result of the context?
2. **Information Objectives** – What ***customer-visible data objects are produced as output*** from the software? What ***data objects are required for input***?
3. **Function and Performance** – What ***functions does the software perform to transform input data into output***? Are there any special performance characteristics to be addressed?

Software project scope must be unambiguous and understandable at both the managerial and technical levels

NOTESJUGAAD

THE PROCESS

A software process provides the framework for which a comprehensive plan for software development can be established.

- **Task sets** – tasks, milestones, work products, and quality assurance points.
- **Umbrella activities** – software quality assurance, software configuration management, and measurement.

The project manager must decide which process model is most appropriate based on:

1. The customers who have requested the product and the people who will do the Work.
2. The characteristics of the product itself.
3. The project environment in which the software team works.

Once a process model is selected, a preliminary project plan is established based on the process framework activities then:

- Process decomposition begins
- The result is a complete plan reflecting the work tasks required to populate the framework activities.

NOTESJUGAAD

THE PROJECT: 5 COMMON-SENSE APPROACHES

1. **Start on the Right Foot:** working hard to understand the problem
   - ✓ Understand the problem; set realistic objectives and expectations; form a good team.
2. **Maintain Momentum:** provide incentives
   - ✓ Provide incentives to reduce turnover of people; emphasize quality in every task; have senior management stay out of the team's way.
3. **Track Progress:** track works products
   - ✓ Track the completion of work products; collect software process and project measures; assess progress against expected averages
4. **Make Smart Decisions:** keep it smile
   - ✓ Keep it simple; use existing software before writing new code; follow standard approaches; identify and avoid risks; always allocate more time than you think you need to do complex or risky tasks
5. **Conduct a Post Mortem Analysis:** lessons learned and evaluation of project
   - ✓ Track lessons learned for each project; compare planned and actual schedules; collect and analyze software project metrics; get feedback from teams members and customers; record findings in written form

NOTESJUGAAD

THE PROJECT: THE W^{5}HH PRINCIPLE

A series of questions that lead to a definition of key project characteristics and the resultant project plan.

1. **Why** is the system being developed?
   - **Assesses the validity of business reasons and justifications**
2. **What** will be done?
   - **Establishes the task set required for the project**
3. **When** will it be done?
   - **Establishes a project schedule**
4. **Who** is responsible for a function?
   - **Defines the role and responsibility of each team member**
5. **Where** are they organizationally located?
   - **Notes the organizational location of team members, customers, and other stakeholders**
6. **How** will the job be done technically and managerially?
   - **Establishes the management and technical strategy for the project**
7. **How** much of each resource is needed?
   - **Establishes estimates based on the answers to the previous questions**

NOTESJUGAAD

REFERENCE

Book:

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Website

- http://www.tutorialsspace.com/Software-Engineering/14-Project-Management-Concept-1.aspx

NOTESJUGAAD

SOFTWARE ENGINEERING

Unit 2: Software Project Planning

Shree Harsh Attri
(Assistant Professor)
Department: Bachelor of Computer Applications

JIMS Engineering Management Technical Campus
Greater Noida (U.P)

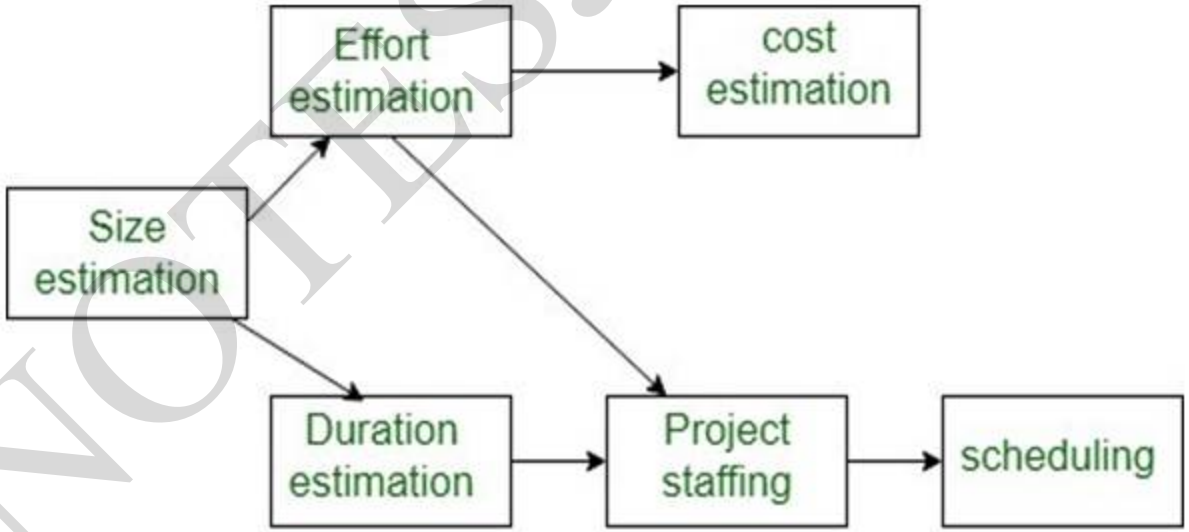
NOTESJUGAAD

Software Project Planning

Software planning begins before technical work starts, continues as the software evolves from concept to reality, and culminates only when the software is retired.

Project designing consists of subsequent essential activities:

1. **Project size:** What's going to be downside quality in terms of the trouble and time needed to develop the product?
2. **Cost:** What proportion is it reaching to value to develop the project?
3. **Duration:** However long is it reaching to want complete development?
4. **Effort:** What proportion effort would be required?



Activities during Software Project Planning

NOTESJIGAAD

SIZE ESTIMATION

Project Size Estimation is the most fundamental parameter, if it is estimated accurately then all other parameters like effort, duration, cost etc. can be determined easily.

Two *techniques* that are used to estimate project size are:

- **Lines of code or LOC**
- **Function point**

NOTESJUGAAD

SIZE ESTIMATION

Line of Code (LOC)

LOC count the total number of lines of source code in a project. The units of LOC are:

- **KLOC- Thousand lines of code**
- **NLOC- Non comment lines of code**

The size is estimated by comparing it with the existing systems of same kind. The experts use it to predict the required size of various components of software and then add them to get the total size.

Advantages:

- Universally accepted and is used in many models like COCOMO.
- Estimation is closer to developer's perspective.
- Simple to use.

Disadvantages:

- Different programming languages contains different number of lines.
- No proper industry standard exist for this technique.
- It is difficult to estimate the size using this technique in early stages of project.

NOTESJUGAAD

SIZE ESTIMATION

Line of Code (LOC)

If LOC is simply a count of the number of lines then figure shown below contains 18 LOC. When comments and blank lines are ignored, the program contains 17 LOC.

```
1.  int. sort (int x[ ], int n)
2.  {
3.  int i, j, save, im1;
4.  /*This function sorts array x in ascending order */
5.  If (n<2) return 1;
6.  for (i=2; i<=n; i++)
7.  {
8.  im1=i-1;
9.  for (j=1; j<=im; j++)
10. if (x[i] < x[j])
11. {
12. Save = x[i];
13. x[i] = x[j];
14. x[j] = save;
15. }
16. }
17. return 0;
18. }
```

"A line of code is any line of program text that is not a comment or blank line, regardless of the number of statements or fragments of statements on the line.
This specifically includes all lines containing ***program header, declaration, and executable and non-executable statements".***

NOTES BY JGAAD

FUNCTIONAL POINT ANALYSIS

- Allan J. Albrecht initially developed function Point Analysis in 1979 at IBM and it has been further modified by the International Function Point Users Group (IFPUG).
- FPA is used to make estimate of the software project, including its *testing in terms of functionality or function size of the software product*.
- The functional size of the product is measured in terms of the ***function point***, which is a standard of measurement to measure the software application.

NOTESJUGAAD

FUNCTIONAL POINT ANALYSIS

The number and type of functions supported by the software are utilized to find FPC (Function Point Count).

The steps in function point analysis are:

1. **Count the number of functions of each proposed type:** Find the number of functions belonging to the types: *External Inputs, External Outputs, External Inquires, Internal Files, External Interface Files.*
2. **Compute the Unadjusted Function Points(UFP):** Categorise each of the five function types as ***simple, average or complex*** based on their complexity. Multiply count of each function type with its weighting factor and find the weighted sum.

NOTESJUGAAD

FUNCTIONAL POINT ANALYSIS

3. **Find Total Degree of Influence(TDI):** Use the ***14 general characteristics*** of a system to find the degree of influence of each of them. The sum of all 14 degrees of influences will give the TDI. The range of TDI is 0 to 70. The 14 general characteristics are: *Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, On-Line Data Entry, End-user Efficiency, Online Update, Complex Processing Reusability, Installation Ease, Operational Ease, Multiple Sites and Facilitate Change*. Each of above characteristics is evaluated on a scale of 0-5.
4. **Compute Value Adjustment Factor(VAF):** Use the following formula to calculate VAF, also known as ***Complexity Adjustment Factor*** **(CAF)**

$$\text{VAF or CAF} = (\text{TDI} * 0.01) + 0.65$$

5. **Find the Function Point Count:** Use the following formula to calculate FPC

$$\text{FPC} = \text{UFP} * \text{VAF}$$

NOTESJUGAAD

FUNCTIONAL POINT ANALYSIS

Count the Number of Functions

The principle of Albrecht's function point analysis (FPA) is that a system is decomposed into functional units.

- Inputs : information entering the system
- Outputs : information leaving the system
- Enquiries : requests for instant access to information
- Internal logical files : information held within the system
- External interface files : information held by other system that is used by the system being analyzed.

NOTESJUGAAD

FUNCTIONAL POINT ANALYSIS

The five functional units are divided in two categories:

1. **Data Function Types:**
   - **Internal Logical Files (ILF):** A user identifiable group of logical related data or control information maintained within the system for example: ***Databases and Directories***
   - **External Interface files (EIF):** A user identifiable group of logically related data or control information referenced by the system, but maintained within another system. This means that EIF counted for one system, may be an ILF in another system for example: ***Shared databases and shared routines.***

NOTESJUGAAD

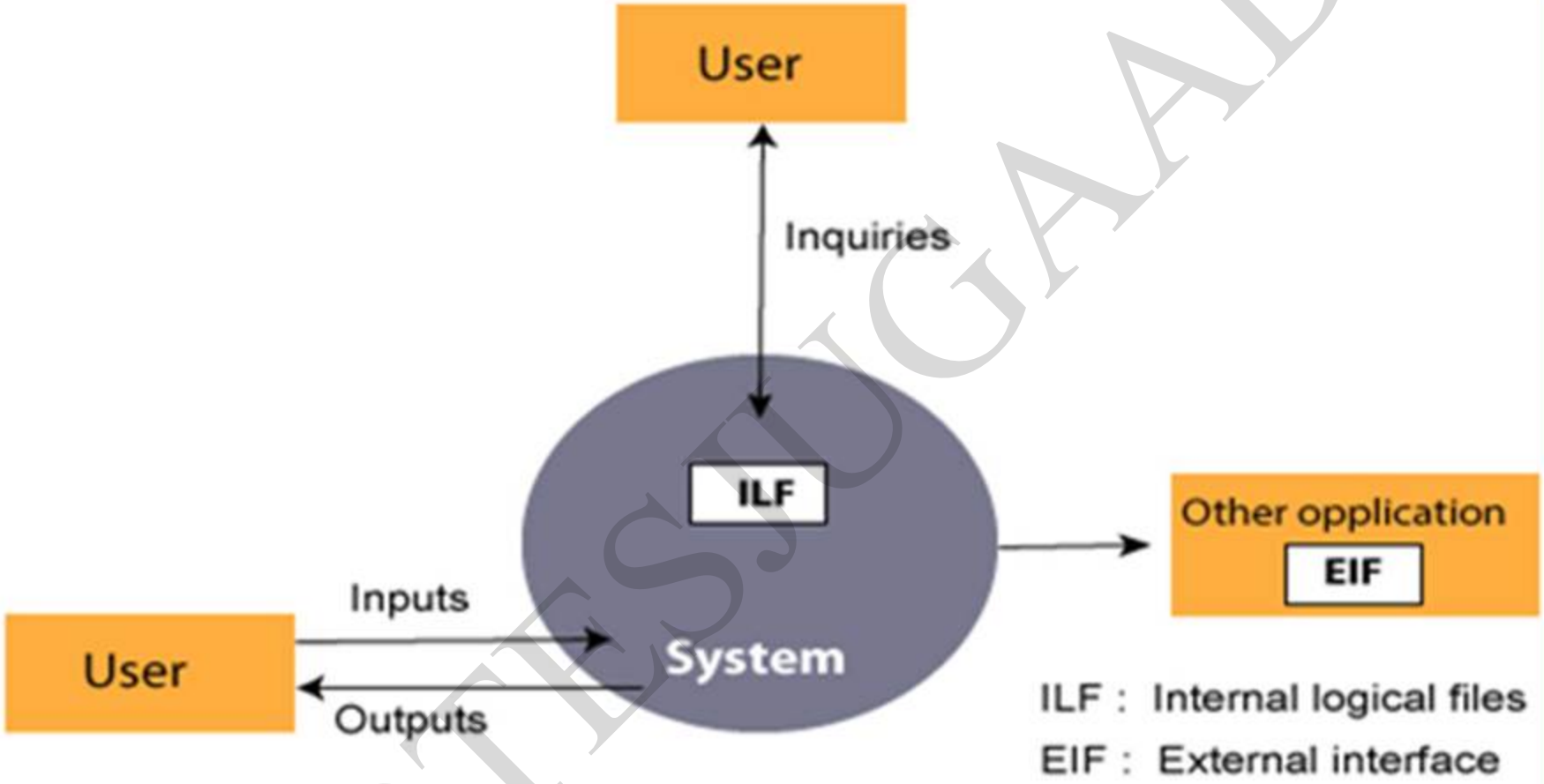
FUNCTIONAL POINT ANALYSIS

The five functional units are divided in two categories:

2. **Transactional function types**
   - **External Input (EI):** An EI processes data or control information that comes from outside the system. The EI is an elementary process, which is the smallest unit of activity that is meaningful to the end user in the business for example: ***Input Screens and Tables***
   - **External Output (EO):** An EO is an elementary process that generate data or control information to be sent outside the system for example: ***Output Screens and Reports***
   - **External Inquiry (EQ):** An EQ is an elementary process that is made up to an input-output combination that results in data retrieval for example: ***Prompts and Interrupts***

NOTESJUGAAD

FUNCTIONAL POINT ANALYSIS



FPAs Functional Units System

श्री हर्ष अत्रि (सहायक आचार्य)

NOTESJUGAAD

FUNCTIONAL POINT ANALYSIS

Special Features of FPA

- Function point approach is independent of the language, tools, or methodologies used for implementation; i.e. they do not take into consideration programming languages, data base management systems, processing hardware or any other data base technology.
- Function points can be estimated from requirement specification or design specification, thus making it possible to estimate development efforts in early phases of development.
- Function points are directly linked to the statement of requirements; any change of requirements can easily be followed by a re-estimate.
- Function points are based on the system user's external view of the system, non-technical users of the software system have a better understanding of what function points are measuring.

NOTESJUGAAD

FUNCTIONAL POINT ANALYSIS

Compute the Unadjusted Function Points(UFP)

Data Functional Types and ***Transactional Function Types*** are assigned some weights to measure the Complexity.

Counting function points

Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
External logical files (ILF)	7	10	15
External Interface files (EIF)	5	7	10

NOTES JUGAAD

Calculation of Unadjusted Function Points(UFP)

Functional Units	Count	Complexity		Complexity Totals	Functional Unit Totals
External Inputs (EIs)		Low x 3	=		
		Average x 4	=		
		High x 6	=		
External Outputs (EOs)		Low x 4	=		
		Average x 5	=		
		High x 7	=		
External Inquiries (EQs)		Low x 3	=		
		Average x 4	=		
		High x 6	=		
External logical Files (ILFs)		Low x 7	=		
		Average x 10	=		
		High x 15	=		
External Interface Files (EIFs)		Low x 5	=		
		Average x 7	=		
		High x 10	=		
Total Unadjusted Function Point Count					

NOTESJUGAAD

Calculation of Unadjusted Function Points(UFP)

The weighting factors are identified for all functional units and multiplied with the functional units accordingly.

The procedure for the calculation of UFP in mathematical form is given below:

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

Where i indicate the row and j indicates the column of Table 1

W_{ij} : It is the entry of the ith row and j^{th} column of the table 1

Zij : It is the count of the number of functional units of Type *i* that have been classified as having the complexity corresponding to column *j*.

NOTES UGAAD

COMPUTING FUNCTION POINT

Organizations that use function point methods develop a criterion for determining whether a particular entry is Low, Average or High. Nonetheless, the determination of complexity is somewhat subjective.

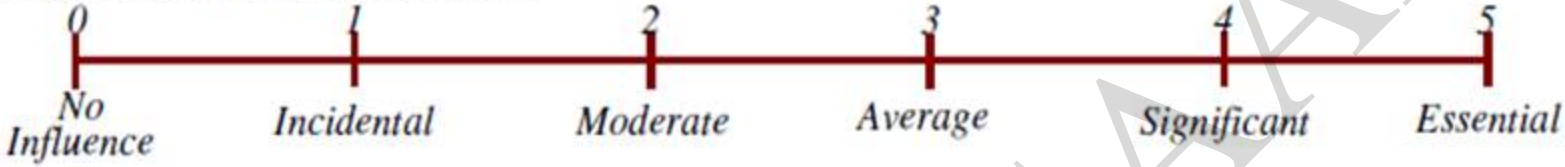
$$FP = UFP * CAF$$

Where CAF is complexity adjustment factor and is equal to $[0.65 + 0.01 \times \Sigma F_i]$. The F_i (i=1 to 14) are the degree of influence and are based on responses to questions

NOTESJUGAAD

COMPUTING FUNCTION POINT

Rate each factor on a scale of 0 to 5.



Number of factors considered (F_i)

1. Does the system require reliable backup and recovery ?
2. Is data communication required ?
3. Are there distributed processing functions ?
4. Is performance critical ?
5. Will the system run 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. Is the inputs, outputs, files, or inquiries complex ?
10. Is the internal processing complex ?
11. Is the code designed to be 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 ?

NOTESJUGAAD

COMPUTING FUNCTION POINT

Functions points may compute the following important metrics:

Productivity	=	FP / persons-months
Quality	=	Defects / FP
Cost	=	Rupees / FP
Documentation	=	Pages of documentation per FP

These metrics are controversial and are not universally acceptable. There are standards issued by the International Functions Point User Group (IFPUG, covering the Albrecht method) and the United Kingdom Function Point User Group (UFPGU, covering the MK11 method). An ISO standard for function point method is also being developed.

NOTESJUGAAD

EXAMPLE 1:

Consider a project with the following functional units:

Number of user inputs	= 50
Number of user outputs	= 40
Number of user enquiries	= 35
Number of user files	= 06
Number of external interfaces	= 04

Assume all complexity adjustment factors and weighting factors are average. Compute the function points for the project.

NOTESJUGAAD

EXAMPLE 1:

Solution

We know

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

UFP = 50 x 4 + 40 x 5 + 35 x 4 + 6 x 10 + 4 x 7

= 200 + 200 + 140 + 60 + 28 = 628

CAF = $(0.65 + 0.01\ \Sigma F_i)$

= (0.65 + 0.01 (14 x 3)) = 0.65 + 0.42 = 1.07

FP = UFP x CAF

= 628 x 1.07 = 672

NOTESJUGAAD

EXAMPLE 2:

An application has the following:

10 low external inputs, 12 high external outputs, 20 low internal logical files, 15 high external interface files, 12 average external inquiries, and a value of complexity adjustment factor of 1.10.

What are the unadjusted and adjusted function point counts ?

NOTESJUGAAD

EXAMPLE 2:

Solution

Unadjusted function point counts may be calculated using as:

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

= 10 x 3 + 12 x 7 + 20 x 7 + 15 + 10 + 12 x 4

= 30 + 84 +140 + 150 + 48

= 452

FP = UFP x CAF

= 452 x 1.10 = 497.2.

NOTESJUGAAD

EXAMPLE 3:

Consider a project with the following parameters.

(*i*) External Inputs:

(a) 10 with low complexity

(b) 15 with average complexity

(c) 17 with high complexity

(*ii*) External Outputs:

(a) 6 with low complexity

(b) 13 with high complexity

(*iii*) External Inquiries:

(a) 3 with low complexity

(b) 4 with average complexity

(c) 2 high complexity

NOTESJUGAAD

EXAMPLE 3:

(*iv*) Internal logical files:

(a) 2 with average complexity

(b) 1 with high complexity

(v) External Interface files:

(a) 9 with low complexity

In addition to above, system requires

i. Significant data communication

ii. Performance is very critical

iii. Designed code may be moderately reusable

iv. System is not designed for multiple installation in different organizations.

Other complexity adjustment factors are treated as average. Compute the function points for the project.

NOTESJUGAAD

EXAMPLE 3:

Solution: Unadjusted function points may be counted

Functional Units	Count	Complexity		Complexity Totals	Functional Unit Totals
External Inputs (EIs)	10	Low x 3	=	30	
	15	Average x 4	=	60	
	17	High x 6	=	102	192
External Outputs (EOs)	6	Low x 4	=	24	
	0	Average x 5	=	0	
	13	High x 7	=	91	115
External Inquiries (EQs)	3	Low x 3	=	9	
	4	Average x 4	=	16	
	2	High x 6	=	12	37
External logical Files (ILFs)	0	Low x 7	=	0	
	2	Average x 10	=	20	
	1	High x 15	=	15	35
External Interface Files (EIFs)	9	Low x 5	=	45	
	0	Average x 7	=	0	
	0	High x 10	=	0	45
Total Unadjusted Function Point Count					424

NOTESJUGAAD

EXAMPLE 3:

$$\sum_{i=1}^{14} F_i = 3+4+3+5+3+3+3+3+3+3+2+3+0+3=41$$

$$\begin{aligned} CAF &= (0.65 + 0.01 \text{ x } \Sigma F_i) \\ &= (0.65 + 0.01 \text{ x } 41) \\ &= 1.06 \\ FP &= UFP \text{ x } CAF \\ &= 424 \text{ x } 1.06 \\ &= 449.44 \end{aligned}$$

Hence $\boxed{FP = 449}$

NOTESJUGAAD

COST ESTIMATION

Software Cost Estimation is the process of predicting the effort required to develop a software system.

Cost Estimation

A number of estimation techniques have been developed and are having following attributes in common :

- Project scope must be established in advance
- Software metrics are used as a basis from which estimates are made
- The project is broken into small pieces which are estimated individually

To achieve reliable cost and schedule estimates, a number of options arise:

- Delay estimation until late in project
- Use simple decomposition techniques to generate project cost and schedule estimates
- Develop empirical models for estimation
- Acquire one or more automated estimation tools

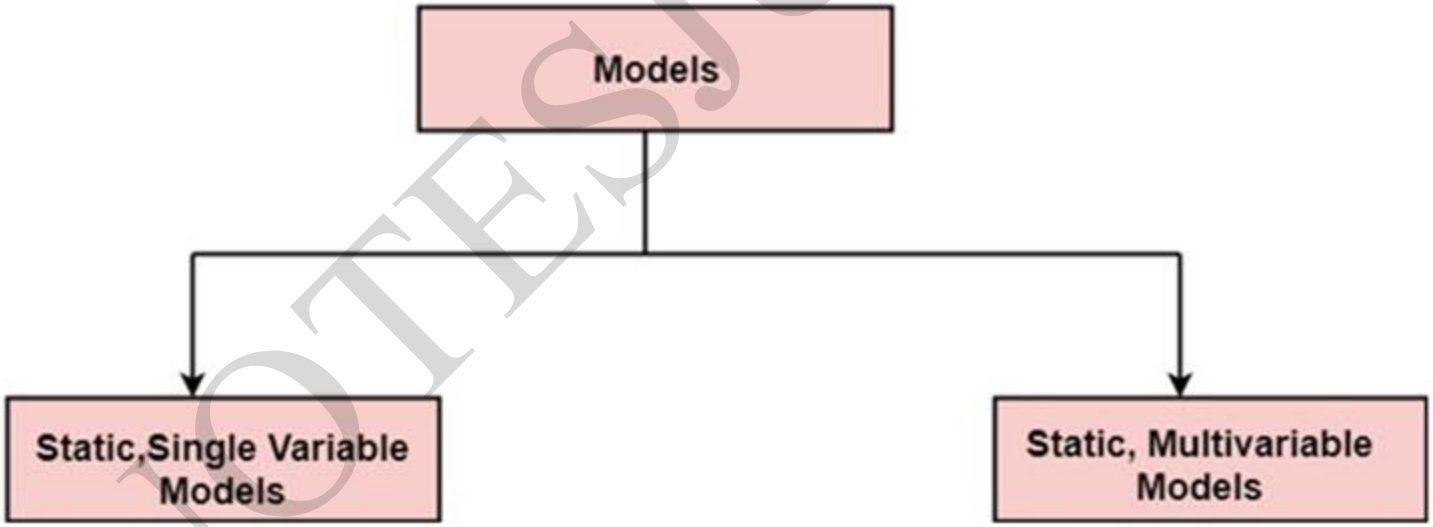
NOTESJUGAAD

COST ESTIMATION

Uses of Cost Estimation

1. During the planning stage, one needs to choose how many engineers are required for the project and to develop a schedule.
2. In monitoring the project's progress, one needs to access whether the project is progressing according to the procedure and takes corrective action, if necessary.

Cost Estimation Models



NOTESJUGAAD

COST ESTIMATION

Static, Single Variable Models

Methods using this model use an equation to estimate the desired values such as cost, time, effort, etc. They all depend on the same variable used as predictor (say, size). An example of the most common equations is :

$$C = a L^b \quad \text{(i)}$$

C is the cost, L is the size and a,b are constants

$$E = 1.4 L^{0.93}$$

$$DOC = 30.4 L^{0.90}$$

$$D = 4.6 L^{0.26}$$

Effort (E in Person-months), documentation (DOC, in number of pages) and duration (D, in months) are calculated from the number of lines of code (L, in thousands of lines) used as a predictor.

The ***Software Engineering Laboratory*** established a model called ***SEL model***, for estimating its software production.

NOTESJUGAAD

COST ESTIMATION

Static, Multivariable Models

These models are often based on equation (i), they actually depend on several variables representing various aspects of the software development environment, for example method used, user participation, customer oriented changes, memory constraints, etc.

WALSTON and **FELIX** develop the models at IBM provide the following equation gives a relationship between ***lines of source code and effort***:

$$E = 5.2\,L^{0.91}$$

In the same manner ***duration of development*** is given by

$$D = 4.1\,L^{0.36}$$

NOTESJUGAAD

EXAMPLE 4:

Compare the Walston-Felix model with the SEL model on a software development expected to involve 8 person-years of effort.

(a)Calculate the number of lines of source code that can be produced.

(b)Calculate the duration of the development.

(c)Calculate the productivity in LOC/PY

(d)Calculate the average manning

NOTESJUGAAD

EXAMPLE 4:

Solution

The amount of manpower involved = 8 PY = 96 person-months

(a) Number of lines of source code can be obtained by reversing equation to give:

$$L = (E/a)^{1/b}$$

Then

$$L(SEL) = (96/1.4)^{1/0.93} = 94264 \text{ LOC}$$

$$L(W\text{-}F) = (96/5.2)^{1/0.91} = 24632 \text{ LOC.}$$

NOTESJUGAAD

EXAMPLE 4:

(b) Duration in months can be calculated by means of equation

$$D(SEL) = 4.6\,(L)^{0.26}$$

$$= 4.6\,(94.264)^{0.26} = 15 \text{ months}$$

$$D(W\text{-}F) = 4.1\,L^{0.36}$$

$$= 4.1(24.632)^{0.36} = 13 \text{ months}$$

(c) Productivity is the lines of code produced per person/month (year)

$$P(SEL) = \frac{94264}{8} = 11783\, LOC / Person - Years$$

$$P(W - F) = \frac{24632}{8} = 3079\, LOC / Person - Years$$

NOTESJUGAAD

EXAMPLE 4:

(d) Average manning is the average number of persons required per month in the project.

$$M(SEL) = \frac{96P - M}{15M} = 6.4 Persons$$

$$M(W - F) = \frac{96P - M}{13M} = 7.4 Persons$$

NOTESJUGAAD

REFERENCE

Book:

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Website

- http://www.tutorialsspace.com/Software-Engineering/14-Project-Management-Concept-1.aspx
- https://www.javatpoint.com/software-engineering-functional-point-fp-analysis
- http://www.professionalqa.com/functional-point-analysis
- https://www.geeksforgeeks.org/software-engineering-project-size-estimation-techniques/
- https://www.javatpoint.com/software-cost-estimation

NOTESJUGAAD

SOFTWARE ENGINEERING

Unit 2: Constructive Cost Model (COCOMO)

Shree Harsh Attri
(Assistant Professor)
Department: Bachelor of Computer Applications

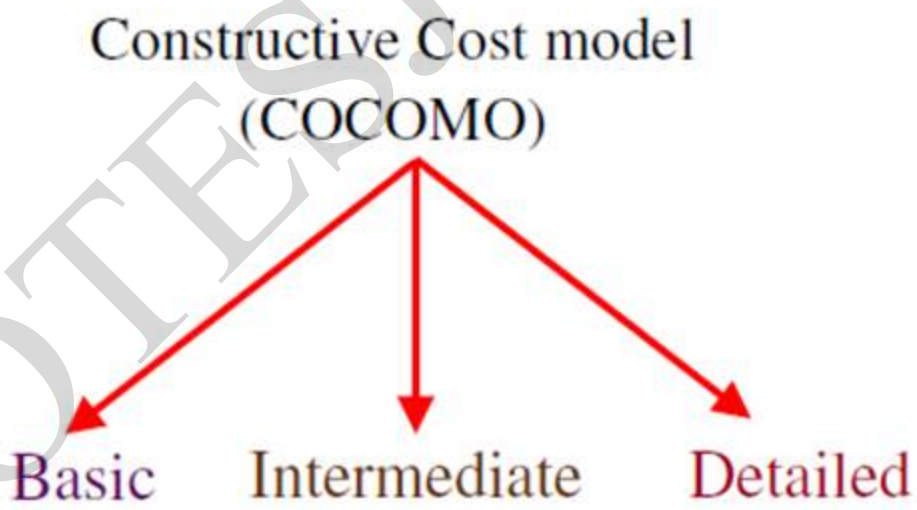
JIMS Engineering Management Technical Campus
Greater Noida (U.P)

NOTESJUGAAD

COCOMO MODEL

- COCOMO is one of the most widely used software estimation models in the world.
- This model is developed in 1981 by Barry Boehm to give estimation of number of man-months it will take to develop a software product.
- COCOMO predicts the efforts and schedule of software product based on size of software.

COCOMO has three different models that reflect complexity



NOTESJUGAAD

COCOMO MODEL

There are **three types** of Software Projects

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

2. **Semi-Detached:** In this class, intermediate project in which **team with mixed experience level are handled**. Such project may have mix of rigid and less than rigid requirements.

3. **Embedded:** In this class, **project with tight hardware, software and operational constraints** are handled.

NOTESJUGAAD

COCOMO MODEL

Comparison of Three COCOMO Modes

Mode	Project size	Nature of Project	Innovation	Deadline of the project	Development Environment
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & In house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer Interfaces required

NOTESJUGAAD

COCOMO MODEL

Software Cost Estimation should be done through three stages:

1. Basic Model
2. Intermediate Model
3. Detailed Model

NOTESJUGAAD

jims Greater Noida

COCOMO MODEL

1. Basic Model

The basic COCOMO model estimate the software development effort using only **Lines of code**.

Various equations in this model are:

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (E)^{d_b}$$

where,
E is the **Effort** applied in person-months
D is the **Development time** in chronological months.
KLOC is the estimated number of **delivered lines of code** for the project.

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO coefficients

NOTESJUGAAD

jims Greater Noida

COCOMO MODEL: BASIC MODEL

When effort and development time are known, the average staff size to complete the project may be calculated as:

$$\text{Average staff size } (SS) = \frac{E}{D} \, Persons$$

When project size is known, the productivity level may be calculated as:

$$\text{Productivity } (P) = \frac{KLOC}{E} \, KLOC / PM$$

NOTESJUGAAD

jims Greater Noida

COCOMO MODEL: BASIC MODEL

Example 1:

Suppose that a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three modes i.e., organic, semidetached and embedded.

(i) Organic mode

$$E = 2.4(400)^{1.05} = 1295.31 \text{ PM}$$

$$D = 2.5(1295.31)^{0.38} = 38.07 \text{ PM}$$

(ii) Semidetached mode

$$E = 3.0(400)^{1.12} = 2462.79 \text{ PM}$$

$$D = 2.5(2462.79)^{0.35} = 38.45 \text{ PM}$$

(iii) Embedded mode

$$E = 3.6(400)^{1.20} = 4772.81 \text{ PM}$$

$$D = 2.5(4772.8)^{0.32} = 38 \text{ PM}$$

NOTESJUGAAD

jims Greater Noida

COCOMO MODEL: BASIC MODEL

Example 2:

A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the effort, development time, average staff size and productivity of the project.

Solution

The semi-detached mode is the most appropriate mode; keeping in view the size, schedule and experience of the development team.

Hence $E = 3.0(200)^{1.12} = 1133.12$ PM

$D = 2.5(1133.12)^{0.35} = 29.3$ PM

Average staff size $(SS) = \frac{E}{D} Persons = \frac{1133.12}{29.3} = 38.67 Persons$

Productivity $= \frac{KLOC}{E} = \frac{200}{1133.12} = 0.1765 KLOC / PM$

$P = 176 LOC / PM$

NOTESJUGAAD

jims Greater Noida

COCOMO MODEL:

2. Intermediate Model

The basic COCOMO Model considers that the effort is only a function of the number of lines of code and some constants calculated according to the various software systems.

The **Intermediate** COCOMO model recognizes these facts and refines the initial estimates obtained through the basic COCOMO model by using a set of **15 cost drivers** based on various attributes of software engineering.

NOTESJUGAAD

COCOMO MODEL: INTERMEDIATE MODEL

Cost Drivers are grouped into 4 categories:

Product Attributes

- Required Software Reliability (RELY)
- Database Size (DATA)
- Product Complexity (CPLX)

Computer Attributes

- Execution Time Constraint (TIME)
- Main Storage Constraint (STOR)
- Virtual Machine Volatility (VIRT)
- Computer Turnaround Time (TURN)

Personnel Attributes

- Analyst Capability (ACAP)
- Application Experience (AEXP)
- Programmer Capability (PCAP)
- Virtual Machine Experience (VEXP)
- Programming Language Experience (LEXP)

Project Attributes

- Modern Programming Practices (MODP)
- Use of Software Tools (TOOL)
- Required Development Schedule (SCED)

jims Greater Noida

NOTES JUGAAD

COCOMO MODEL: INTERMEDIATE MODEL

Multipliers of different cost drivers

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Product Attributes						
RELY	0.75	0.88	1.00	1.15	1.40	--
DATA	--	0.94	1.00	1.08	1.16	--
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
TIME	--	--	1.00	1.11	1.30	1.66
STOR	--	--	1.00	1.06	1.21	1.56
VIRT	--	0.87	1.00	1.15	1.30	--
TURN	--	0.87	1.00	1.07	1.15	--

Multiplier values for effort calculations

NOTESJUGAAD

COCOMO MODEL: INTERMEDIATE MODEL

Cost Drivers	RATINGS					
	Very low	Low	Nominal	High	Very high	Extra high
Personnel Attributes						
ACAP	1.46	1.19	1.00	0.86	0.71	--
AEXP	1.29	1.13	1.00	0.91	0.82	--
PCAP	1.42	1.17	1.00	0.86	0.70	--
VEXP	1.21	1.10	1.00	0.90	--	--
LEXP	1.14	1.07	1.00	0.95	--	--
Project Attributes						
MODP	1.24	1.10	1.00	0.91	0.82	--
TOOL	1.24	1.10	1.00	0.91	0.83	--
SCED	1.23	1.08	1.00	1.04	1.10	--

Multiplier values for effort calculations

NOTESJUGAAD

COCOMO MODEL: INTERMEDIATE MODEL

Intermediate COCOMO equations

$E = a_i (KLOC)^{b_i} * EAF$
$D = c_i (E)^{d_i}$
$SS = E/D$ persons
$P = KLOC/E$

EAF = Effort Adjustment factor
E = effort
D = Deployment time
SS = staff size
P = productivity
a_i, b_i, c_i, d_i = Coefficients

Project	a_i	b_i	c_i	d_i
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Coefficients for intermediate COCOMO

NOTESJUCAAD

COCOMO MODEL: INTERMEDIATE MODEL

Example 3:

Case 1

A new project with estimated 400 KLOC embedded system has to be developed. Project manager has a choice of hiring from two pools of developers: Very highly capable with very little experience in the programming language being used

Case 2

Developers of low quality but a lot of experience with the programming language. What is the impact of hiring all developers from one or the other pool ?

NOTESJUGAAD

COCOMO MODEL: INTERMEDIATE MODEL

Solution

This is the case of embedded mode and model is intermediate COCOMO.

Hence $$E = a_i (KLOC)^{d_i}$$

$$= 2.8\ (400)^{1.20} = 3712 \text{ PM}$$

Case I: Developers are very highly capable with very little experience in the programming being used.

EAF = 0.82 x 1.14 = 0.9348

E = 3712 x .9348 = 3470 PM

D = $2.5\ (3470)^{0.32}$ = 33.9 M

Avg. Staff Size = E/D = 3712/33.9 = 109 persons

Productivity = KLOC/E = 400/3712 = 0.1077 KLOC/person-month

NOTESJUGAAD

COCOMO MODEL: INTERMEDIATE MODEL

Case II: Developers are of low quality but lot of experience with the programming language being used.

EAF = 1.29 x 0.95 = 1.22

E = 3712 x 1.22 = 4528 PM

D = $2.5\ (4528)^{0.32}$ = 36.9 M

Avg. Staff Size = E/D = 4528/36.9 = 122 persons

Productivity = KLOC/E = 400/4528 = 0.0884 KLOC/person-month

Case II requires more effort and time. Hence, low quality developers with lot of programming language experience could not match with the performance of very highly capable developers with very litter experience.

NOTESJUGAAD

COCOMO MODEL

3. Detailed COCOMO Model

Detailed COCOMO incorporates all characteristics of the ***Intermediate*** version with an ***assessment of the cost driver's impact*** on each step of the Software Engineering Process.
The detailed model uses different effort multipliers for ***each cost driver attribute***. In detailed COCOMO, the whole software is divided into different modules and then COCOMO is applied in different modules to estimate effort and then sum the effort.

The Phases of Detailed COCOMO are:

1. Planning and Requirements
2. System Design and Detailed Design
3. Module Code and Test
4. Integration and Test

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

NOTES.IUCAAD

COCOMO MODEL: DETAILED MODEL

Development Phase

Plan / Requirements

EFFORT	:	6% to 8%
DEVELOPMENT TIME	:	10% to 40%

% depend on mode & size

Design

Effort	:	16% to 18%
Time	:	19% to 38%

Programming

Effort	:	48% to 68%
Time	:	24% to 64%

Integration & Test

Effort	:	16% to 34%
Time	:	18% to 34%

NOTESJUGAAD

COCOMO MODEL: DETAILED MODEL

Effort and schedule fractions occurring in each phase of the lifecycle

Lifecycle Phase Values of μ_p **Effort**

Mode & Code Size	Plan & Requirements	System Design	Detailed Design	Module Code & Test	Integration & Test
Organic Small S≈2	0.06	0.16	0.26	0.42	0.16
Organic medium S≈32	0.06	0.16	0.24	0.38	0.22
Semidetached medium S≈32	0.07	0.17	0.25	0.33	0.25
Semidetached large S≈128	0.07	0.17	0.24	0.31	0.28
Embedded large S≈128	0.08	0.18	0.25	0.26	0.31
Embedded extra large S≈320	0.08	0.18	0.24	0.24	0.34

NOTESJUGAAD

COCOMO MODEL: DETAILED MODEL

Effort and schedule fractions occurring in each phase of the lifecycle

Lifecycle Phase Values of τ_p **Schedule**

Mode & Code Size	Plan & Requirements	System Design	Detailed Design	Module Code & Test	Integration & Test
Organic Small S≈2	0.10	0.19	0.24	0.39	0.18
Organic medium S≈32	0.12	0.19	0.21	0.34	0.26
Semidetached medium S≈32	0.20	0.26	0.21	0.27	0.26
Semidetached large S≈128	0.22	0.27	0.19	0.25	0.29
Embedded large S≈128	0.36	0.36	0.18	0.18	0.28
Embedded extra large S≈320	0.40	0.38	0.16	0.16	0.30

NOTESJUGAAD

COCOMO MODEL: DETAILED MODEL

The Calculation

Multiply all 15 Cost Drivers to get **Effort Adjustment Factor(EAF)**

E(Effort) = a_b**(KLOC)**^{b_b} * **EAF**(in Person-Month)

D(Development Time) = c_b**(E)**^{d_b} (in month)

E_p **(Total Effort)** = μ_p * **E** (in Person-Month)

D_p **(Total Development Time)** = τ_p * **D** (in month)

NOTESJUGAAD

jims Greater Noida

COCOMO MODEL: DETAILED MODEL

Example 4:

Consider a project to develop a full screen editor

The major components identified and their sizes are:

(i)	Screen Edit	– 4KLOC
(ii)	Command Lang Interpreter	– 2KLOC
(iii)	File Input and Output	– 1KLOC
(iv)	Cursor movement	– 2KLOC
(v)	Screen Movement	– 3KLOC

Assume the ***Required software reliability is high, product complexity is high, analyst capability is high & programming language experience is low***.

Use COCOMO model to determine

1. **Overall Cost and Schedule Estimates**
2. **Cost & Schedule Estimates for different Phases**

NOTESJUGAAD

COCOMO MODEL: DETAILED MODEL

Solution

Size of modules : 4 + 2 + 1 + 2 + 3 = 12 KLOC **[Organic]**

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.75	0.88	1.00	**1.15**	1.40	--
CPLX	0.70	0.85	1.00	**1.15**	1.30	1.65
ACAP	1.46	1.19	1.00	**0.86**	0.71	
LEXP	1.14	**1.07**	1.00	0.95	--	--

EAF = 1.15 * 1.15 * 0.86 * 1.07 = 1.2169

(a) The initial effort estimate for the project is obtained from the following equation

$$E = a_i (KLOC)^{bi} \times EAF$$

$$= 3.2(12)^{1.05} \times 1.2169 = 52.91 \text{ PM}$$

Development time $D = C_i(E)^{di}$

$$= 2.5(52.91)^{0.38} = 11.29 \text{ M}$$

NOTESJUGAAD

jims Greater Noida

COCOMO MODEL: DETAILED MODEL

(b) Cost and Schedule estimate is calculated as:

$$E_p = \mu_p E$$

$$D_p = \tau_p D$$

Phase value of μ_p (Effort) and τ_p (Schedule / Development Time Duration)

	Plan & Requirement	System Design	Detail Design	Module Code & Test	Integration & Test
Organic Small μ_p	0.06	0.16	0.26	0.42	0.16
Organic Small τ_p	0.10	0.19	0.24	0.39	0.18

NOTESJUGAAD

COCOMO MODEL: DETAILED MODEL

Phase wise Effort & Development Time Duration

	Effort (E)	Duration (D)	Ep (in person-months)	Dp (in months)
Plan & Requirement	52.9	11.29	0.06*52.9 = 3.17	0.10*11.29=1.12
System Design	52.9	11.29	0.16*52.9=8.46	0.19*11.29=2.14
Detail Design	52.9	11.29	0.26*52.9=13.74	0.24*11.29=2.70
Module Code & Test	52.9	11.29	0.42*52.9=22.21	0.39*11.29=4.40
Integration & Test	52.9	11.29	0.16*52.9=8.46	0.18*11.29=2.03

NOTESJUGAAD

REFERENCE

Book:

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Website

- http://www.tutorialsspace.com/Software-Engineering/14-Project-Management-Concept-1.aspx
- https://www.javatpoint.com/software-engineering-functional-point-fp-analysis
- http://www.professionalqa.com/functional-point-analysis
- https://www.geeksforgeeks.org/software-engineering-project-size-estimation-techniques/
- https://www.javatpoint.com/software-cost-estimation

NOTESJUGAAD

SOFTWARE ENGINEERING

Unit 2: Risk Management

Shree Harsh Attri
(Assistant Professor)
Department: Bachelor of Computer Applications

JIMS Engineering Management Technical Campus
Greater Noida (U.P)

NOTESJUGAAD

RISK MANAGEMENT

What is Risk ?

Tomorrow's problems are today's risks.

*"**Risk** is a problem that may cause some loss or threaten the success of the project, but which has not happened yet".*

Risk Management is the process of identifying, addressing and eliminating these problems before they can damage the project.

NOTESJUGAAD

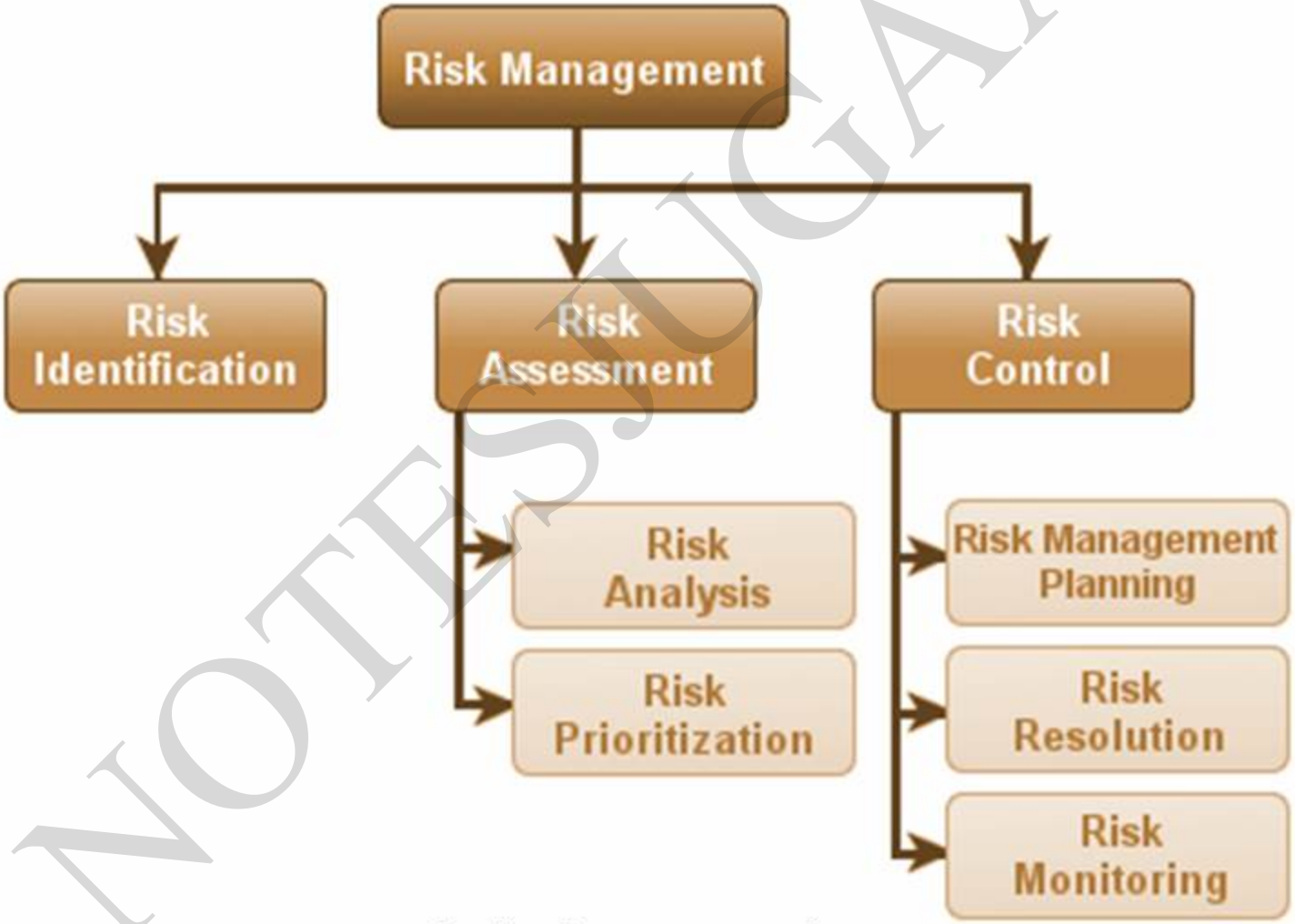
RISK MANAGEMENT

Following are the categories of the Risk

1. **Project Risk:**
   a) If the project risk is real then it is probable that the project schedule will slip and the cost of the project will increase.
   b) It identifies the potential schedule, resource, stakeholders and the requirements problems and their impact on a software project.
2. **Technical Risk:**
   a) If the technical risk is real then the implementation becomes impossible.
   b) It identifies potential design, interface, verification and maintenance of the problem.
3. **Business Risk:**
   a) If the business risk is real then it harms the project or product.
   b) This type of risk expresses the risks of building a superb product that nobody needs, losing monetary fund or personal commitments etc.

NOTESJUGAAD

Risk Management Activities



NOTESJUGAAD

Risk Identification

A project may consists of large variety of risk thus the project organizer needs to anticipate the risk in the project as early as possible so that the impact of risk can be reduced by making effective risk management planning.

To identify the significant risk there are different types of risks which can affect a software project:

a) **Technology risks:** Risks that assume from the software or hardware technologies that are used to develop the system.

b) **People risks:** Risks that are connected with the person in the development team.

c) **Organizational risks:** Risks that assume from the organizational environment where the software is being developed.

d) **Tools risks:** Risks that assume from the software tools and other support software used to create the system.

e) **Requirement risks:** Risks that assume from the changes to the customer requirement and the process of managing the requirements change.

f) **Estimation risks:** Risks that assume from the management estimates of the resources required to build the system

NOTESJUGAAD

jims Greater Noida

Risk Assessment

Risk Analysis and Risk Prioritization

- This stage usually involves the analysis and prioritization of the risks, i.e. possible outcomes of each identified risk is being assessed based on which risks are categorized and accordingly, prioritized.
- Based on the degree of impact, possessed by each risk, they are being assigned severity levels, namely 'High', 'Medium' and 'Low'. And based on their severity, they are prioritize i.e. High risks are considered as top priority whereas the low risk is regarded for the bottom most priority.

During the risk analysis process, there is a need to consider every identified risk and make a perception of the **probability** and seriousness of that risk.

So risk analysis is analyzed as follows:

a) The probability of the risk might be determined as very low (0-10%), low (10-25%), moderate (25-50%), high (50-75%) or very high (+75%).

b) The effect of the risk might be determined as catastrophic (threaten the survival of the plan), serious (would cause significant delays), tolerable (delays are within allowed contingency), or insignificant.

NOTESJUGAAD

Risk Control

During this stage, risks are managed, controlled and moderated, based on their priority so as to achieve the desired results.

It is generally divided into three activities as follows:

- **Risk Management Planning:** It involves a proper and effective plan to deal with the each identified risk.
- **Risk Resolution:** It refers to the execution of the plans, outlined during the risk management planning stage so as to either remove or fix identified risks.
- **Risk Monitoring:** It involves, regular monitoring and tracking, the development progress, in the direction, of resolving risk issues, which may include revaluation of the risks, their likelihood to occur, etc., and taking and implementing necessary & appropriate actions, wherever necessary.

NOTESJUGAAD

REFERENCE

Book:

Software Engineering (3rd ed.), By K.K Aggarwal & Yogesh Singh, Copyright © New Age International Publishers, 2007

Website

- https://www.professionalqa.com/risk-management-activity
- https://www.javatpoint.com/software-engineering-risk-management
- https://www.geeksforgeeks.org/software-engineering-risk-management/
- https://www.tutorialride.com/software-engineering/risk-management-in-software-engineering.htm

NOTESJUGAAD