UNIT V

5 Project Management

Syllabus

Software Project Management- Software Configuration Management - Project Scheduling- DevOps : Motivation-Cloud as a platform-Operations - Deployment Pipeline : Overall Architecture Building and Testing-Deployment - Tools - Case Study.

Contents

5.1 Software Project Management

- 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.

5.2 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.

5.2.1 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.

5.2.2 The Product

- Before planning the project three important tasks need to be done -
  1) Product objectives and scope must be established.
  2) Alternative solutions should be considered.
  3) 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.

5.2.3 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.

5.2.4 The Project

To manage the complexity of the project, we conduct planned and controlled software projects. In order to successfully manage a software project, we must understand what can go wrong and how to do it right. There are some indicators that imply poor management of the projects.

Review Question

1. *Explain the role of people, product and process in project management.*

5.3 People

People is the most important element for the success of software project. People participate in the project with different roles and responsibilities. Let us discuss these -

5.3.1 Stakeholder

Stakeholders mean persons who will be affected by the system. Following are the categories of the stakeholders -

1. **Senior Manager**
   - These are the persons who define the business issues and the decisions taken by have significant influence on the success of the project.
2. **Project Manager**
   - The project manager performs various tasks such as planning, motivating, organizing and controlling the software practitioners who are involved in software building.
3. **Practitioners**
   - The practitioners are the entities having sufficient technical skills required for engineering the product or application.
4. **Customers**
   - These stakeholders specify the requirements of the project and are interested only in outcome of the project.
5. **End-users**
   - The end-users interact with the software product once get released.
   - The team should be organized in such a way that skills and abilities of each person can be utilized at the most and this job is done by Team leader.

5.3.2 Team Leaders

Project management is an activity influenced by people. Following are the qualities that the team leader must posses -

- **Motivation**

  This ability is necessary to encourage technical people to do their work with the fullest of their ability.
- **Problem Solving**

  During project management, the project manager must diagnose the technical and organizational issues in such a manner that systematic and effective solution can be obtained. He/She must take lessons from the previous projects and must remain flexible to change the direction of the if initial problem solving methods get failed.
- **Organization**

  This is an ability to arrange the existing processes so that intial concept can be translated into the final working product.
- **Innovation**

  Innovation or ideas mean encouraging people to create and to feel creative.
- **Managerial Identity**

  The effective project manager must take the charge of complete project and must have full control over the project development process.
- **Achievement**

  The productivity of the project team can be optimized by awarding the team members for their good work and by not punishing them on the controlled risk.
- **Influence and Team Building**

  An effective project manager should be able to "read" the people in his team, he should understand their needs and must remain in controlled in stressful situations.

5.3.3 Software Team

- The **best** team structure depends on three things - i) Management style of organization ii) Number of people involved in the project and their skill levels and iii) Overall problem difficulty.

- **Mantei** described **seven factors** that should be considered for **planning** the **structure of software engineering teams** -
  - The **difficulty** of the problem to be solved.
  - The **size** of the project (may be considered in terms of LOC or in Function points).
  - The **time** that the team will stay together.
  - The required **quality and reliability** of the system to be built.
  - The **rigidity or flexibility** of the delivery date.
  - The degree of **communication** required for the project.

  Constantine suggested four **organizational paradigms** for software engineering team -
- **Closed Paradigm**

  It represents the **traditional hierarchy** of team. This team work well for producing software based on **past efforts** but it **fails** to work for **innovative ideas**.
- **Random Paradigm**

  This is a **loosely structured** and depends upon individual initiative of team members. This performs well when innovation or technological breakthrough is required but performs poorly when the orderly performance is required.
- **Open Paradigm**

  The open paradigm attempts to structure a team in such a manner that some controls are achieved using **closed paradigm** but innovation or technological break through is achieved using the **random paradigm**.
- **Synchronous Paradigm**

  It structures the team using natural compartmentalization of the problem. This team work on the piece of problem by establishing proper communication among them.

5.3.4 Agile Team

- Agile philosophy encourages -
  - Customer satisfaction.
  - Early incremental delivery of the software.
  - Small but highly motivated project team.
  - Informal methods.
  - Development simplicity.

Hence a small motivated team is called **agile team.**

- Agile philosophy stresses on **individual competency** coupled with **group collaboration** as critical success factor for the team.
- Agile teams are **self-organizing** teams. There is no **fixed Constantine team paradigm** that can be applied to the agile, instead of that the agile team can use elements of random, open, closed and synchronous paradigms.
- Many agile **models automate** the project management and technical decisions required for project accomplishment.
- Agile team is allowed to select its **own approach** for software development. The only condition is that the business requirements and organizational standards must get satisfied during software development.
- As the project proceeds the **individual competency** might be **improved** in such a way that it will be beneficial to the project. To improve the individual competency the agile team conducts the **daily meeting**, for co-ordinating and synchronizing the work. The **information** obtained during this meeting is used to adapt the useful **approach** for software development.

5.3.5 Co-ordination and Communication Issues

Following are some characteristics of modern software -

- **Scale**

  The scale of development efforts may be very large which leads to complexity and confusion in co-ordinating the team members.
- **Uncertainty**

  Continuous changes occur in the project, due to which uncertainty is common in modern software.
- **Interoperability**

  Interoperability is the key characteristic of many systems. New software must be able to communicate with existing software.

  To deal with above mentioned characteristics software engineering team must establish effective methods for co-ordinating people. This co-ordination is possible only by means of formal and informal communication among the team members.

Review Questions

1. *What are the categories of stakeholders ? What are the characteristics of effective project manager ?*
2. *List the four P's of software project management spectrum. Explain how "the people" factor contributes towards the success of software project.*
3. *What are the categories of stakeholders ? What are the characteristics of effective project manager ?*

5.4 Product

A software project manager has to examine the product in order to obtain the quantitative estimates and organizational plan. By examining the product the scope of the product can be decided.

5.4.1 Software Scope

The first step in software project management is to determine the scope of the software project. Following questions need to be answered for determining the scope of the project -

1. Context

a) How does the software built fit into the larger system and business context ?

b) What are the constraints imposed on the context of the project ?

2. Information Objectives

a) What are the visible data objects that get produced as an output of software ?

b) What are the data objects that are required for the input of the software ?

3. Function and performance

a) For transforming the input data to an output what are those required functions ?

b) What are special performance characteristics ?

The software scope must be **unambiguous and understandable** at the management and technical levels. A statement of scope must be **bounded**. **Constraints and limitations** are noted and **mitigating factors** are described.

5.4.2 Problem Decomposition

- Problem decomposition means partitioning or elaborating the problems.

- During the project scope no attempt is made to decompose the problem fully. The decomposition that is normally applied in based on two major areas -
  1. **Functionalities** that must be delivered.
  2. The **process** that will be used to deliver it.
- Normally, "**Divide and Conquer**" strategy is applied to partition the product into smaller pieces which can be managed easily as project planning begins.
- The software functions described in the scope are evaluated and refined to provide more details before the beginning of the estimation.
- As both cost and schedule are functionally oriented, some degree of decomposition is often useful.
- As the statement of scope evolves the first level of partitioning naturally occurs.

5.5 Process

- There are some framework activities that characterize the software processes. Such activities are applied to all software projects.
- The Project Manager must decide which Process Model is most appropriate for -
  1. The Customers and the People who are involved in project.
  2. The characteristics of Product.
  3. The Project Environment in which the Software works.
- When a Process Model is selected, the Software Team then defines a Preliminary **Project Plan** based on Process Framework activities. After establishing the preliminary Project Plan, Process Decomposition begins.

5.5.1 Melding the Product and the Process

- At the beginning of planning process there is the melding of Product and the Process. Each function to be engineered by the software team must go through the set of **Framework Activities** that has been defined for a Software organization.
- Assume that following are some **framework activities** -
  - Communication
  - Planning
  - Modeling
  - Construction and Release
  - Deployment
  - Customer Evaluation

- When team members work on the product functions they will apply these farmework activities to it. A matrix as shown in Fig. 5.5.1 is created.

Software Engineering Tasks \ Common process framework activities	Communication	Planning	Modeling	Construction and Release	Deployment	Customer Evaluation	. . .
Product functions							
Text input							
Editing and formatting							
...							
...							
Documentation							

Fig. 5.5.1 Matrix for framework activities

- On the left hand side the major product functions are described and framework activities are listed on the top. Then the software engineering work tasks would be entered in the row.
- Project Manager estimates the resource requirements for each matrix cell with Start and End Dates for each Task and Work Products to be produced (delivered) for each task.

5.5.2 Process Decomposition

- The software development team has a flexibility for choosing the software process model which is best suitable for the project. After decision of process model the software engineering tasks are decided.
- When a project manager raises a question "How do we accomplish the framework activities?" then project decomposition commences. For example - a small project might require following work tasks for performing the communication activity -
  1. A list of **clarification issues** is to be prepared.
  2. For addressing the clarification issues conduct **meetings** with customers.
  3. The developer and customer together must prepare **statement of scope**.
  4. **Review** the statement of scope.
  5. Perform **modifications** in the statement of scope if required.

- Similarly for the large projects the work task list can be prepared which may contain some additional tasks.

5.6 Project

- For a successful software project, it is necessary to understand the mistakes in the project and how to correct them. John Reel defined ten symptoms to indicate why the software projects fail -
  1. Software developers do not understand the customer's need.
  2. The scope of the project is not defined properly.
  3. Change management is done poorly.
  4. Business needs change very often.
  5. The technological changes are quite often.
  6. Users are not co-operative.
  7. Unrealistic deadlines are set.
  8. Projects do not get sponsored or Sponsorship gets lost.
  9. Lack of skilled people in the project.
  10. Project managers do not adopt the best practices for the projects.
- John Reel suggested following five points to overcome the problems in the software project -
  1. **Start on right foot** - The software developer must understand the problem, managers should set the realistic goals and customers should keep the realistic expectations from the project. The right team must be chosen to complete the project. Such a team must be given autonomy and authority. The technology needed by such team should be provided. All these factors avoid lot of problems of the software project.
  2. **Maintain momentum** - Many projects are very good initially but later on their quality decreases. Therefore project managers should provide incentives for absolute minimum income. Quality must be insisted at every stage of development.
  3. **Track progress** - Regular progress of the project must be tracked. The work products at every stages of development must be analysed.
  4. **Make smart decisions** - Simple decisions must be made by software project managers as well as team members. Whenever possible make use of the off-the shelf components. Try to identify the obvious risks and allocate more time than you think for solving complex risks.

5. **Conduct analysis** - After completion of every project try to find out what went wrong and how to correct them. Evaluate the plans and actual schedule. Get the feedback about the project from the team members and customers.

5.7 The W5HH Principle

Bary Bohem suggested an approach for addressing - project objectives, deciding milestones and schedules, roles and responsibilities, project management, technical approaches and required resources by WWWWWHH principle. The W5HH principle is nothing but the series of questions in the form of why, what, when, who, how and so on. Answers to these questions lead to definition of key project characteristics. The answers of these questions are also useful in building the resultant project plan. Following are those W5HH questions -

- **Why is the system being developed ?**

  Answer to this question assesses the validity of business reasons for the software work. It also justifies the expenditure of people, time and money.

- **What will be done ?**

  Answer to this question will help the software team member to identify the project tasks and milestones.

- **When will be done ?**

  The answer to this question will help to prepare the project schedule with identified project tasks and milestones.

- **Who is responsible for a function ?**

  By answering this question, the roles and responsibilities required to develop the system can be defined.

- **Where are they organisationally located ?**

  All the roles can not be defined within the software team itself. There are customers, users and other stakeholders holding some responsibilities.

- **How to do the job technically with proper management ?**

  Answer to this question will help to establish the technical strategy about the project.

- **How much of each resource required ?**

  This answer will be useful for deriving the project estimate or cost of the project.

Review Question

1. *Explain W5HHH principle.*

5.8 Project Estimation

AU : May-05,06,07,09,16,17,18,19,22, Dec.-07,10,14,15,16,19, Marks 16

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).

5.8.1 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.**

5.8.2 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

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

considers for "most likely" estimate where S is the estimation size variable, represents the optimistic estimate, represents the most likely estimate and represents the pessimistic estimate values.

5.8.3 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
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 5.8.1 Size measure

- 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.

5.8.4 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 $= [4700 + (4 * 6000) + 10000]/6 \rightarrow 6450$

- 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

Example 5.8.1 *Consider 7 functions with their estimated lines of code given below*

Function	LOC
Funct1	2340
Funct2	5380
Funct3	6800
Funct4	3350
Funct5	4950
Funct6	2140
Funct7	8400

Average productivity based on historical data is 620 LOC/pm and Labour rate is ₹8000 per month. Find the total estimated project cost and effort. **AU : Dec.-16, Marks 8**

Solution : We will first compute total estimation in LOC.

Function	LOC
Funct1	2340
Funct2	5380
Funct3	6800
Funct4	3350
Funct5	4950
Funct6	2140
Funct7	8400
Total estimation (LOC)	33360

- A review of historic data indicates

1) Average productivity is 620 LOC/per month

2) Average labour cost is ₹ 8000 per month

Then cost for lines of code can be estimated as

Cost/LOC = 8000/620 = ₹ 13 approx

By considering total estimated LOC as 33360

- Total estimated project cost = (33360 *13)

= ₹ 433680

- Total estimated project effort = (33360/620)

= 54 person-months

Example 5.8.2 *If Team A found 342 errors prior to release of software and Team B found 182 errors. What additional measures and metrics needed to find out if teams have removed the errors effectively ? Explain.* **AU : May-18, Marks 4**

Solution : The additional measures and metrics needed to find out if team have removed errors effectively or not are -

1. Errors per KLOC (thousand lines of code)
2. Defects per KLOC
3. $ per LOC
4. Pages of documentation per KLOC

The LOC based metric can be chosen to understand the errors.

5.8.5 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.

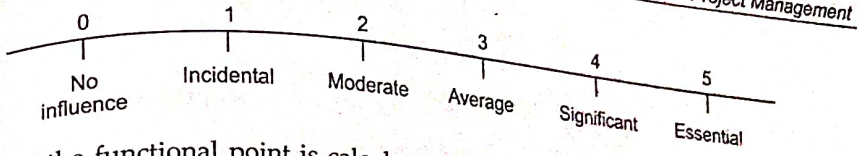
How to calculate function point ?

- 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.

Domain characteristics	Count	Weighting factor			Count
		Simple	Average	Complex	
Number of user input	X	3	4	6	
Number of user output	X	4	5	7	
Number of user inquiries	X	3	4	6	
Number of files	X	7	10	15	
Number of external interfaces	X	5	7	10	
Count Total					

- 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 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. 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 :
- **Function Points (FP)** = Count total $\times$ $(0.65 + (0.01 \times \text{Sum}(F_i))$



- 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

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

Disadvantages

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

5.8.6 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.

INFO DOMAIN VALUE	Opt.	most likely	pessimistic	esti. value	weight factor	FP
NO.OF INPUTS	25	28	32	28.1	4	112
NO. OF OUTPUTS	14	17	20	17	5	85
NO. OF INQURIES	17	23	30	23.1	5	116
NO. OF FILES	5	5	7	5.33	10	53
NO. OF EXTERNAL INTERFACES	2	2	3	2	7	15
COUNT TOTAL						381

- 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)

Sr. No.	FACTOR	VALUE (Fi).
1.	Back-up and recovery ?	4
2.	Data communication ?	2
3.	Distributed processing ?	0
4.	Performance critical ?	4
5.	Existing operational environment ?	3
6.	On-line data entry ?	4
7.	Input transactions over multiple screens ?	5
8.	Online updates ?	3
9.	Information domain values complex ?	5
10	Internal processing complex ?	5
11.	Code designed for reuse ?	4
12.	Conversion / installation in design ?	3
13.	Multiple installations ?	5
14.	Application designed for change ?	5

$\sum(F_i) \rightarrow$ 52

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* ? $\sum$ (F_i))]

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.

Example 5.8.3 *Study of requirement specification for ABC project has produced following results : Need for 7 inputs, 10 outputs, 6 inquiries, 17 files and 4 external interfaces. Input and external interface function point attributes are of average complexity and all other function points attributes are of low complexity. Determine adjusted function points assuming complexity adjustment value is 32.*

Solution : Given that :

7 inputs

10 Outputs

6 inquiries

17 files

4 external interfaces

Average complexity for inputs and external interfaces. Low complexity for remaining parameters.

Adjusted function point value $\sum(F_i) = 32$.

Let us calculate count total value.

Measurement parameters	Count	Weighting factor				
		×	Simple	Average	Complex	
Number of user inputs	7	×		4		28
Number of user outputs	10	×	4			40
Number of user inquiries	6	×	3			18
Number of files	17	×	7			119
Number of external interfaces	4	×		7		28
Count total						233

Function point = Count total × $[0.65 + 0.01 \times \Sigma(F_i)]$

= $233 \times [0.65 + 0.01 \times 32]$

= $233 \times [0.65 + 0.32]$

= 233×0.97

FP = 226.01

Hence adjusted function point is **226.01**

Example 5.8.4 *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 unadjusted and adjusted function point counts ?* **AU : May-16, Marks 4**

Solution : The unadjusted function points are calculated using predefined weights for each function type which is as given below.

Functional Units	Weighting Factors		
	Low	Average	High
External Input	3	4	6
External Output	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface File	5	7	10

The Unadjusted Function Point (UFP)

$$= \Sigma \text{ Functional units} \times \text{Weighting factor.}$$

$$= ((10 \times 3) + (8 \times 7) + (13 \times 7) + (17 \times 10) + (11 \times 4))$$

$$= (30 + 56 + 91 + 170 + 44)$$

$$UFP = 391$$

Adjusted Function Point (FP)

$$= UFP \times \text{Complexity adjustment factor}$$

$$= 391 \times 1.10$$

$$FP = 430.10$$

Example 5.8.5 *Consider the following function point components and their complexity. If the total degree of influence is 52, find the estimated function points.*

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

AU : Dec.-16, Marks 8

Solution : We will first compute **count total**

Function type	Estimated count	Weight	FP count
ELF	2	7	2 × 7 = 14
ILF	4	10	4 × 10 = 40
EQ	22	4	22 × 4 = 88
EO	16	5	16 × 5 = 80
EI	24	4	24 × 4 = 96
	Count Total (VFP)		318

We will compute complexity

$$\text{Adjustment factor (CAF)} = \text{Degree of influence} \times 0.01 + 0.65$$

$$= 52 \times 0.01 + 0.65$$

$$CAF = 1.17$$

$$FP = UFP \times CAF$$

$$= 318 \times 1.17$$

$$FP = 372$$

Example 5.8.6 *An application has 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 adjustment factor of 1.10. What is the unadjusted and adjusted function point-count ?* **AU : May-17, Marks 5**

Solution : The unadjusted function points are calculated using predefined weights for each function type which is as given below

Functional Units	Weighting Factors		
	Low	Average	High
External Input	3	4	6
External Output	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

The Unadjusted Function Point (UFP)

$$= \sum \text{Funtional units} \times \text{Weighting Factor}$$

$$= (10 \times 3) + (12 \times 7) + (20 \times 7) + (15 \times 10) + (12 \times 4)$$

$$= (30 + 84 + 140 + 150 + 48)$$

$$\text{UFP} = 452$$

Adjusted Function Point (FP)

$$= \text{UFP} \times \text{Complexity adjustment factor}$$

$$= 452 \times 1.10$$

$$\textbf{FP} = \textbf{497.2}$$

Example 5.8.7 *Compute the function point FP for a payroll program that reads a file of employees and file of information for the current month and prints cheques for all the employees. The program is capable of handling an interactive command to print an individually requested cheque immediately.* **AU : May-09, Marks 16**

Solution : From given data -

1. Reading a file of employee and file of information for current month = 2 input operations.
2. Print cheques of employee = 1 output
3. There are two files : File of employee and file of information for current month.
4. Interactive command means = User enquiries.

We assume complexity adjustment factor and weighting factors are average.

	Count		Average weighting factor	
Number of input	2	×	4	8
Number of user output	1	×	5	5
Number of user inquiries	1	×	4	4
Number of files	2	×	10	20
Count total			= 37	

$$\text{Function point} = \text{Count total} \times \left[0.65 + \left[0.01 \times \sum_{i=1}^{14} F_i\right]\right]$$

$$= 37 \times (0.65 + (0.01 \times (14 \times 3)) = 37 \times (0.65 + 0.42)$$

$$= 37 \times (1.07)$$

$$\textbf{Function point} = \textbf{39.59}$$

Example 5.8.8 *Compute function point value for a project with following information domain characteristics :*

No. of external inputs - 30

No. of external outputs - 52

No. of external inquiries - 22

No. of log files - 12

No. of external interface files - 2

Assume complexity adjustment values for above are average (4, 5, 4, 10, 7 respectively). **AU : Dec.-14. Marks 6**

Solution : Funcion Point (FP) = Count total $\times \left[0.65 + \left[0.01 \times \sum_{i=1}^{14} F_i\right]\right]$

Now we will compute count total

$$\text{Count total} = \sum (\text{Information domain characteristic} \times \text{Complexity adjustment values})$$

$$= [(30 \times 4) + (52 \times 5) + (22 \times 4) + (12 \times 10) + (2 \times 7)]$$

$$= 120 + 260 + 88 + 120 + 14$$

$$\textbf{Count total} = \textbf{602}$$

$$FP = 602 \times \left[0.65 + \left[0.01 \times \sum_{i-1}^{14} F_i\right]\right]$$

$$= 602 \times [0.65 + (0.01 \times (14 \times 3))]$$

$$= 602 \times (0.65 + 0.42)$$

$$= 602 \times 1.07$$

$$= 644.14$$

∴ **Function point = 644.14**

Review Questions

1. *How is functional point analysis methodology is applied in estimation of software size ? Explain. Why FPA methodology is better than LOC methodology ?* **AU : May-17, Marks 8**
2. *Discuss about the metrics for small organizations.* **AU : Dec.-15, Marks 6**
3. *Explain the function point approach to establish the size of a project.* **AU : Dec.-10, 14, Marks 10**
4. *Discuss the process of function point analysis. Explain function point analysis with sample cases for components of different complexity.* **AU : May-18, Marks 13**
5. *List the features of LOC and FP based estimation models. Compare the two models and list the advantages of one over other.* **AU : May-19, Marks 13**
6. *Outline the steps in function point analysis with an example.* **AU : Dec.-19, Marks 15**
7. *Brief out the importance of LOC and FP cost estimation.* **AU : May-22, Marks 7**

5.9 Make Buy Decision

AU : May-16, Marks 8

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.

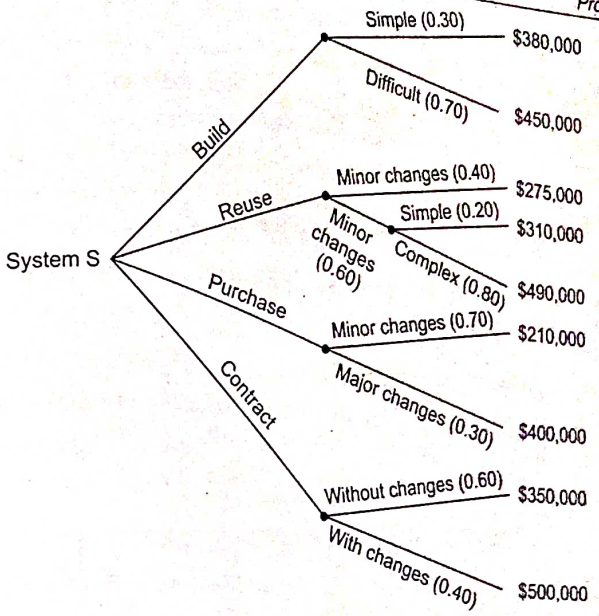


Fig. 5.9.1

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

$$\text{Expected cost} = \begin{matrix}\Sigma \text{ Path Probability} \\ \text{of decision tree path}\end{matrix} \times \begin{matrix}\text{Estimated path} \\ \text{cost}\end{matrix}$$

For example for the branch **system S → Reuse** can be computed as :

$$\text{Expected cost}_{\text{Reuse}} = 0.40(\$275\text{ K}) + [0.60\ (0.20\ (\$310\text{ K}) + 0.80\ (\$490\text{ K}))]$$

$$= \$110\text{ K} + [0.60\ (\$\ 62\text{ K} + \$\ 392\text{ K})]$$

$$= \$110\text{ K} + [0.60\ (\$\ 454\text{ K})]$$

$$= \$110\text{ K} + \$272.4\text{ K}$$

$$= \$382\text{ K}$$

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

Node	Expected cost
Build	$429 K
Reuse	$382 K
Purchase	$267 K
Contract	$410 K

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.

5.9.1 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.

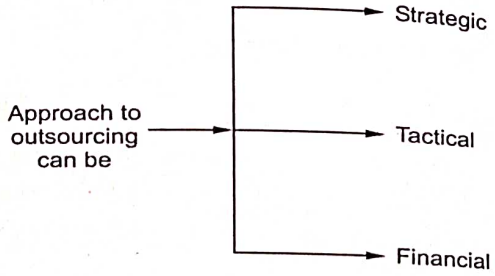


Fig. 5.9.2 Approaches for outsourcing

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

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.

Review Question

1. Write short note - Make/Buy decision.

5.10 COCOMO I Model

AU : May-16. Marks 8

AU : May-07,08,14,15,17,22. Dec.-05,13,22. Marks 16

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 an estimate of the number of man-months it will take to develop a software product. COCOMO predicts the efforts and schedule of a software product based on size of the software. COCOMO stands for "**CO**nstructive **CO**st **MO**del".

COCOMO has three different models that reflect the complexity -

- Basic model
- Intermediate model
- Detailed model.

Similarly there are three classes of software projects.

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

2) Semi-detached projects : In this class an intermediate projects in which teams with mixed experience level are handled. Such projects may have mix of rigid and less than rigid requirements.

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

Let us understand each model in detail.

1) Basic model : The basic COCOMO model estimates the software development effort using only **Lines of Code**. Various equations in this model are -

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

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

$$P = E/D$$

Where E is the **effort** applied in person-months.

D is the development time in chronological months.

KLOC means kilo line of code for the project.

P is total number of persons required to accomplish the project.

The coefficients a_b, b_b, c_b, d_b for three modes are as given below.

Software projects	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table 5.10.1

Merits of basic COCOMO model

Basic COCOMO model is good for quick, early, rough order of magnitude estimates of software project.

Limitations of basic model

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 experience, modern tachniques 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 -

i) Effort estimation

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

i.e. $E = 3.0\ (30)^{1.12}$ where lines of code = 30000 = 30 KLOC

$$E = 135 \text{ person-month}$$

ii) Duration estimation

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

$$= 2.5(135)^{0.35}$$

$$D = 14 \text{ months}$$

iii) Persons estimation

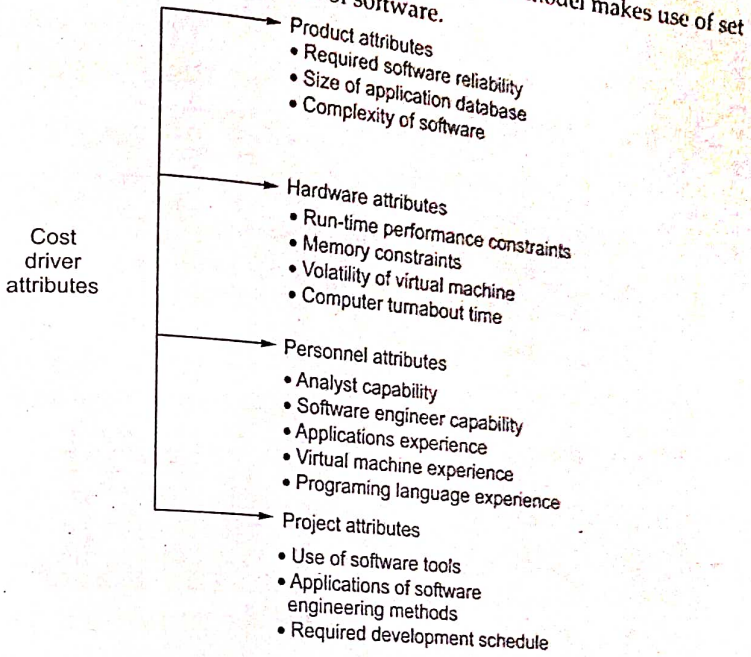
$$P = E/D$$

$$= 135/14$$

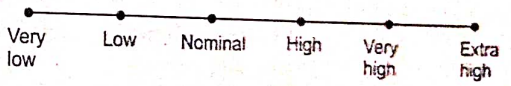
$$P = 10 \text{ persons approximately}$$

2) Intermediate model

This is an extension of Basic COCOMO model. This estimation model makes use of set of "Cost driver attributes" to compute the cost of software.



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



The effort multipliers for each cost driver attribute is as given in following table. The product of all effort multipliers result in "Effort Adjustment Factor" (EAF).

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 software	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 virtual machine		0.87	1.00	1.15	1.30	
Computer turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Use of software tools	1.24	1.10	1.00	0.91	0.82	
Applications of software engineering methods	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Table 5.10.2

The formula for effort calculation can be -

$$E = a_i (KLOC)^{b_i} \cdot EAF \text{ person-months}$$

The values for a_i and b_i for various class of software projects are -

Software project	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Table 5.10.3

The duration and person estimate is same as in basic COCOMO model. i.e.

$D = c_b(E)^{d_b}$ months — i.e. use values of c_b and d_b coefficients that are in Table 5.10.1

$P = E/D$ persons

Merits of intermediate model

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 of intermediate model

1. The estimation is within 20 % of actual 68 % of the time.
2. The effort multipliers are not dependent on phases.
3. A product with many components is difficult to estimate.

Example

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

$$E = a_i (KLOC)^{b_i} \cdot EAF$$

As reliability is high, EAF = 1.15 (product attribute)

$a_i = 2.8$, $b_i = 1.20$ } for embedded software

$$\therefore \quad E = 2.8(30)^{1.20} * 1.15$$
$$= 191 \text{ person-month}$$
$$D = c_b(E)^{d_b} = 2.5(191)^{0.32}$$
$$= 13 \text{ months approximately}$$
$$P = E/D$$
$$= 191/13$$
$$P = 15 \text{ persons approximately}$$

3) Detailed COCOMO model

The detailed model uses the same equations for estimation as the Intermediate Model. But detailed model can estimate the effort (E), duration (D) and persons (P) of each of development phases, subsystems, modules.

The experimentation with different development strategies is allowed in this model.

Four phases used in detailed COCOMO model are -

1. Requirements Planning and Product Design (RPD)
2. Detailed Design (DD)
3. Code and Unit Test (CUT)
4. Integrate and Test (IT)

The effort multipliers for detailed COCOMO are

Phases	Very low	Low	Nominal	High	Very high
RPD	1.80	0.85	1.00	0.75	0.55
DD	1.35	0.85	1.00	0.90	0.75
CUT	1.35	0.85	1.00	0.90	0.75
IT	1.50	1.20	1.00	0.85	0.70

Using these detailed cost drivers, an estimate is determined for each phase of the lifecycle.

Review Questions

1. *Describe in detail COCOMO model for software cost estimation. Illustrate considering a suitable example.* **AU : May-17, Marks 13**
2. *Discuss about the COCOMO models (Basic, intermediate and detailed) for cost estimation.* **AU : May-14, 15, Marks 16**
3. *Explain how effort and cost estimation are determined using COCOMO model ?* **AU : Dec.-20, Marks 13**
4. *Discuss about the basic COCOMO model with advantages and limitations.* **AU : Dec.-22, Marks 8**
5. *Explain COCOMO model for software estimation.* **AU : May-22, Marks 7**

5.11 COCOMO II Model

AU : Dec.-07,10,11,14,15,16,19, May-07,16,17,18, Marks 16

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 -

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

where

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 :

$$\text{Effort} = A \times \text{size}^B \times M$$

where

Boehm has proposed the value of coefficient **A = 2.94.**

Size should be in terms of Kilo source lines of code i.e. KSLOC.

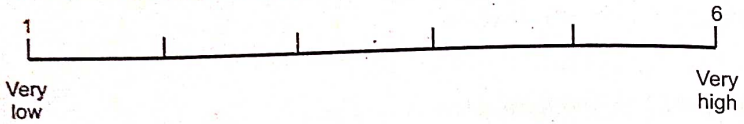
The lines of code can be computed with the help of function point.

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

1. Product reliability and complexity (RCPX)
2. Reuse required (RUSE)
3. Platform difficulty (PDIF)
4. Personnel capability (PERS)
5. Personnel experience (PREX)
6. Schedule (SCED)
7. 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 -
- $ESLOC = ASLOC \times (1-AT/100) \times AAM$

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

$$Effort = A \times Size^{B} \times M$$

- 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 organisation. Very low means no previous experience and high means the organisation knows the application domain.
Development flexibility	Flexibility in development process. Very low means the 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 high risk analysis is made.
Team cohesion	Represents the working environment of the team. Very low cohesion means poor communication or interaction between the team members and extra high means there is no communication problem and team can work in a good spirit.
Process maturity	This factor affects the process maturity of the organisation. This value can be computed using Capacity Maturity Model (CMM) questionnaire, for computing the estimates CMM 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
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 analyse 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.

Example 5.11.1 *Describe in detail COCOMO model for software cost estimation. Use it to estimate the effort required to build software for a simple ATM that produces 12 screens, 10 reports and has 80 software components. Assume average complexity and average developer maturity. Use application composition model with object points.* **AU : Dec.-16. Marks 16**

Solution : The number of screens, reports and components are weighted according the table as given below

Object type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
Software Components			10

As there is average complexity and average developer maturity, we will compute object point as

Object point = $12 * 2 + 10 * 5 + 80 * 10$

Object point = 874

Example 5.11.2 *Using COCOMO, estimate time required for the following :*

1) A semi-detached model of software project of 2000 lines.

2) An embedded model of software of 30,000 lines.

3) An organic model of software of one lakh lines.

4) An organic model of software of 10 lakh lines. **AU : May-07. Marks 8**

Solution : To estimate time using basic model of COCOMO following formula can be used.

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

where E is the effort in person-month.

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

where D is development time in chronological months.

$$P = E/D$$

where P is total number of persons involved in the project. The constants are -

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

1) Given that, System = Semidetached

Lines of code = 2000 lines = 2 KLOC

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

$E = 3.0(2)^{1.15}$

$E = 6.65$ person-month

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

$D = 4.8$ months

$\therefore$ $P = E/D$

$P = 1.3 \approx 1$ person

Thus 1 person can handle this project within approximately **5 months.**

2) Given that, System = Embedded

Lines of code = 30,000 lines = 30 KLOC

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

$= 3.6(30)^{1.20}$

$E \approx 213$ person - month

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

$= 2.5(213)^{0.32}$

$\therefore$ $D \approx 14$ months

$P = E/D$

$\approx 213/14$

≈ 15 persons.

That means 15 persons can complete this project within approximately **14 months.**

3) Given that,system = Organic

Lines of code = 1 lakh = 100 KLOC

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

$\therefore$ $= 2.4\ (100)^{1.05}$

$E \approx 302$ person-month

$D = c_b(E)^{d_b} = 2.5(302)^{0.38}$

≈ 21 months

$P = E/D$

$\therefore$ $\approx 302/21$

≈ 14 persons.

That means this project can be completed within 21 months by 14 persons, approximately,

4) Given that, System = Organic

Lines of code = 10 lakh = 1000 KLOC

$\therefore$ $E = a_b(KLOC)^{b_b} = 2.4(1000)^{1.05}$

≈ 3390 person - month

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

$\approx 2.5\ (3390)^{0.38}$

≈ 55 months

$\therefore$ $P = E/D$

$\approx 3390/55$

≈ 61 persons

This project can be completed within 55 months by 61 people approximately.

Example 5.11.3 *Calculate the effort and duration using the above details for basic COCOMO model.*

Given,

Number of user inputs = 15

Number of user outputs = 3

Number of external interfaces = 11

1 function point = 20 LOC (as fourth generation language is used).

Values of constant used in basic COCOMO model. a = 2.4, b = 1.05, c = 2.5, d = 0.38.

AU : Dec.-11, Marks 8

Solution : We will first calculate FP from given data. Assume all complexity adjustment factors and weighting factors as average.

$\therefore$ $UFP = 15 \times 4 + 3 \times 5 + 11 \times 7 = 152$

$$CAF = 0.65 + 0.01 \times \left(\sum F_i\right) = 0.65 + 0.01 \times (14 \times 3) = 1.07$$

$$FP = UFP \times CAF = 152 \times 1.07 = 163$$

$$1FP = 20 \text{ LOC}$$

$$\therefore \quad 163 = 3260 \text{ LOC} = 3.26 \text{ KLOC.}$$

The basic COCOMO equation take the form :

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

$$D = C_b(KLOC)^{d_b}$$

For organic mode

$$E = 2.4\,(3.26)^{1.05} \qquad E = 8.28 \text{ PM}$$

$$D = 2.5\,(3.26)^{0.38} \qquad D = 3.9 \text{ M}$$

Review Questions

1. *Discuss about COCOMO II model for software estimation.* **AU : Dec.-15, Marks 10**
2. *Write short notes - COCOMO II.* **AU : May-16, Marks 8**
3. *Describe in detail COCOMO model for software cost estimation. Illustrate considering suitable example.* **AU : May-17, Marks 16, May-18, Marks 9**
4. *Elaborate the cost estimation COCOMO II cost estimation model.* **AU : Dec.-19, Marks 13**

5.12 Software Configuration Management

Definition : Software configuration management is a set of activities carried out for identifying, organizing and controlling changes throughout the lifecycle of computer software.

- During the development of software change must be managed and controlled in order to improve quality and reduce error.
- Hence Software Configuration Management is a quality assurance activity that is applied throughout the software process.
- The **origins of changes** that are requested for software are -
  1. New business or market positions cause the changes in the requirements. Due to which the changes need to occur.
  2. New stakeholders may require some changes in the existing requirements.

3. Due to business growth or project extension, it is essential to makes changes in the project.
4. Sometimes due to schedule or budge constraints, changes are must in project.

- The software configuration management is concerned with managing evolving software systems.
- Software configuration management is a set of tracking and control activities that begin when a software development project begins and terminates when the software is taken out of operation.

5.12.1 SCM Basics

5.12.1.1 Goals in Change Management

- During software development process, various roles and tasks are involved.
- The goal of project manager is to ensure that the project is getting completed within given schedule and budget. Hence project managers continuously track the progress of the project.
- The configuration managers ensures that the changes in the projects is appropriately taking place and the testing is conducted thoroughly after the modification in the project.
- The goal of software engineer is to work on the assigned module to produce efficient code and error free code.
- Finally the customer uses the product, analyze it and may request for the change or find out the bugs.

5.12.1.2 Elements of Configuration Management System

- Susan Dart identified four important elements for software configuration management system. These elements are -

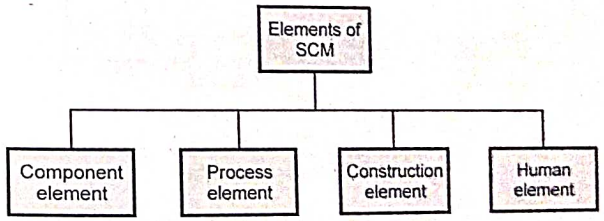


Fig. 5.12.1

1. **Component Elements :** It consists of collection of tools that are used for file management system. For example - databases.

2. **Process Elements :** It consists of actions and tasks used during change management and use of software.
3. **Construction Elements :** It is a collection of tools that automate the construction of software.
4. **Human Elements :** It consists of set of tools that are used by software team to implement software configuration management.

5.12.1.3 Baselines

- The IEEE (IEEE Std. No. 610.12-1990) defines a baseline as :
  - A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
- A baseline is a milestone in the development of software that is marked by the delivery of one or more software configuration items and the approval of them is obtained through formal technical review.
- The baseline is the shared project database. It is an SCM task to maintain the integrity of the set of artifacts.

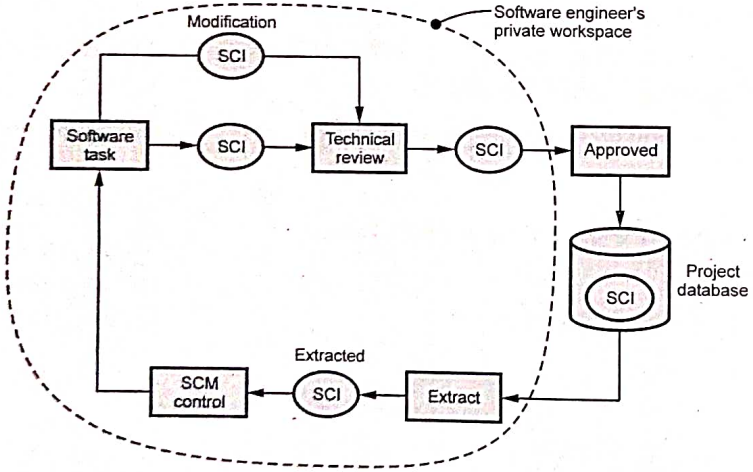


Fig. 5.12.2 Software baseline

- The elements of a design model have been first documented and reviewed. From this design model errors are identified and corrected. Once all parts of the model have been reviewed, corrected and then approved, the design model becomes a baseline.

- Fig. 5.12.2 represents the common software baseline.
  - **Step 1 :** The Software engineering task produce one or more SCIs.
  - **Step 2 :** These SCIs are then reviewed and approved.
  - **Step 3 :** These are then placed in project database or project library.
  - **Step 4 :** When software engineer wants to make changes to baselined SCI, it is copied to software engineers private workspace.
  - **Step 5 :** The extracted SCI is modified only if SCM controls are followed.

5.12.1.4 Software Configuration Items

- Software configuration item is basically the information that is created as a part of software engineering process.
- Examples of Software Configuration Items are
  - Computer programs
    - Source programs
    - Executable programs
  - Documents describing the programs
    - Technical manual
    - Users manual
  - Data
    - Program components or functions
    - External data
    - File structure
- For each type of item, there may be a large number of different individual items produced. For instance there may be many documents for a software specification such as project plan, quality plan, test plan, design documents, programs, test reports, review reports.
- These SCI or items will be produced during the project, stored, retrieved, changed, stored again and so on.
- Each configuration item must have a unique name and a description or specification which distinguishes it from other items of the same type.

5.12.2 SCM Repository

- Software Configuration Items (SCI) are maintained in project repository or project library.

- The software repository is basically a database that acts as a center for both accumulation and storage for software engineering information.
- The software engineer interacts with repository using tools that are integrated within it.

5.12.2.1 Role of Project Repository

- Software repository is a collection of information accessed by software engineers to make appropriate changes in it.
- This repository is handled using all the modern database management functions.
- It must maintain important properties such as data integrity, sharing and integration.
- The repository must maintain uniform structure and format for software engineering work products.
- To achieve these capabilities, the repository is defined in terms of **meta-model**. The meta model determines -

  i) How information is stored in repository ?

  ii) How data can be accessed by using the tool ?

  iii) How the security and integrity of the data is maintained ?

  iv) How the existing model can be extended to accommodate new requirements ?

5.12.2.2 Features

The Software Configuration Management can be performed by interacting with repository. The repository must have toolset that provides support for following features -

1. **Versioning :** As project progresses various versions of individual work products may get created. The repository must be able to maintain all these versions and permit the developer to go back to previous versions during testing and debugging.
2. **Dependency Tracking and Change Management :** The data elements stored in the repository are related to each other. The repository must have an ability to keep track of these relationship and to maintain data integrity.
3. **Requirements Tracing :** If the constructed components, its designed is tracked, then particular requirement can be traced out. The repository must have this ability to trace the requirement from constructed components.

4. **Configuration Management :** The repository must be able to keep track of series of configurations representing project milestones and production releases.
5. **Audit Trails :** The audit trail.

5.12.3 SCM Process

The primary objectives of Software Configuration Management process (SCM) are -

1. **Configuration Identification :** Identify the items that define the software configuration.
2. **Change Control :** Manage changes to one or more items.
3. **Version Control :** Facilitate to create different versions of the application.
4. **Configuration Authentication :** To ensure that the quality of the software is maintained as the configuration evolves over the time.

The SCM process must be developed in such a way that the software team must answer the following set of questions -

1. How does the software team identify the software configuration items ?
2. How does the software team control the changes in the software before and after delivering it to the customer ?
3. How does the software team manage the versions of the programs in the software package ?
4. How does team get ensured that the changes are made properly ?
5. Who is responsible for approving the changes in the software ?

The answers to these questions lead the definition of five tasks of SCM and those are - Identification, change control, version control and configuration audit and status reporting.

5.12.3.1 Identification of Objects in Software Configuration

The software configuration items must be separately named and identified as object.

- These objects must be arranged using **object oriented approach.**
- There are two categories of objects - **basic objects** and **aggregate objects.**
- The **basic object** is unit of information created during requirements analysis, design, coding or testing. For example basic object can be part of **source code.**
- Aggregate object is a collection of basic objects and other aggregate objects. For example SRS or data model can be aggregate object.

- Each object can be uniquely identified because it has got -
  1. **Name :** The name of the object is nothing but the collection of characters (string) or some text. It is unique.
  2. **Description :** For describing the object, the object description can be given. This description contains document, program or some other description such as project identifier or version information.
  3. **List of resources :** The resources are the entities that are used for accessing, referencing and processing of objects. The data types and functionalities can serve as a resource.
  4. **Realization or identification :** It is pointer to object.
- The configuration object identification can also consider relationships that exist between the named objects.
- If a change is made to one configuration object it is possible to determine which other configuration objects in the repository are affected by the change.
- Basically object evolve throughout the software process. During the process of object identification the evolution of objects along with its process must be identified.
- Major modifications in the object must be noted.

5.12.3.2 Change Control

- Changes in any software projects are vital. Sometimes, introducing small changes in the system may lead to big problems in product.
- Similarly, introducing some changes may enhance the capabilities of the system.
- According to **James Bach** too little changes may create some problems and too big changes may create another problems.
- For a large software engineering project, uncontrolled change creates lot of chaos. For managing such changes, human procedures or automated tools can be used.
- The change control process is shown by following Fig. 5.12.3.
  (See Fig. 5.12.3 on next page)
  - **Step 1 :** First of all there arises a need for the change.
  - **Step 2 :** The change request is then submitted by the user.
  - **Step 3 :** Developers evaluate this request to assess technical merits, potential side effects and overall impact on system functions and cost of the project.

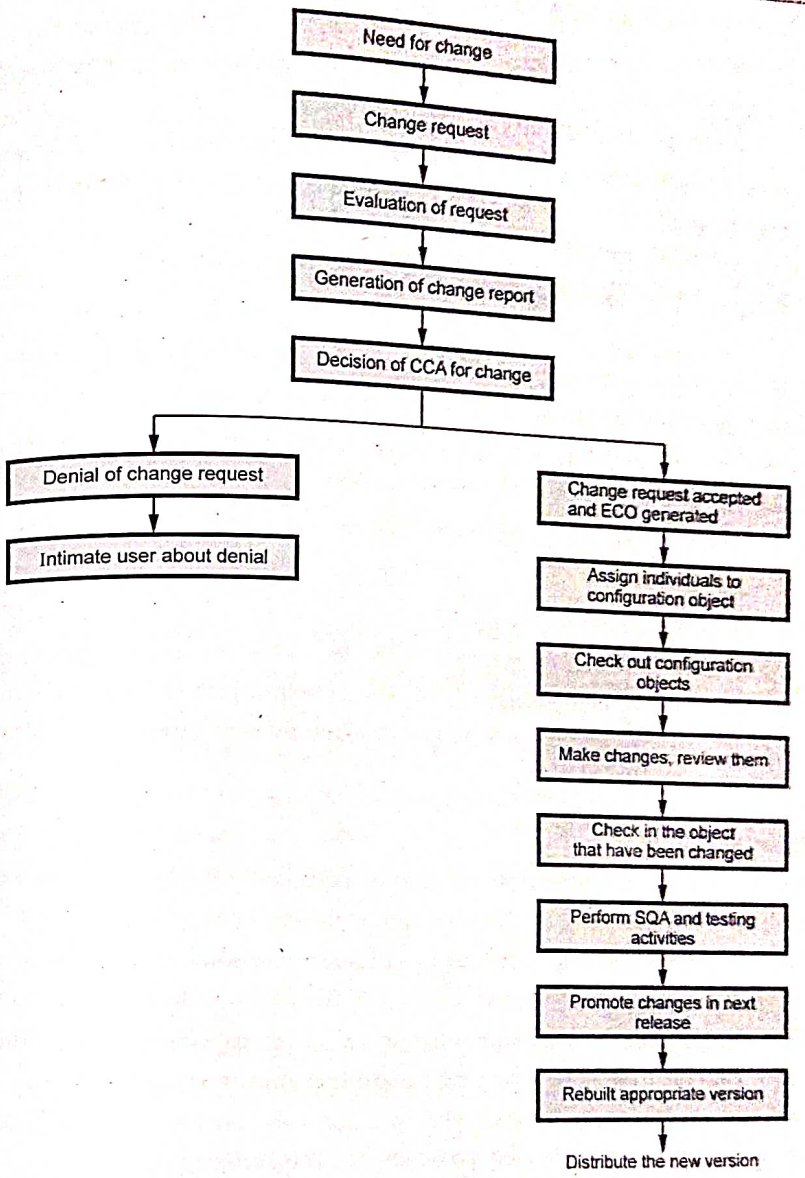


Fig. 5.12.3 Change control process

- **Step 4 :** A change report is then generated and presented to the Change Control Authority (CCA).
- **Step 5 :** The change control authority is a person or a group of people who makes a final decision on status or priority of the change.
- **Step 6 :** An Engineering Change Order (ECO) is generated when the change gets approved. In ECO the change is described, the restrictions and criteria for review and audit are mentioned.
- **Step 7 :** The object that needs to be changed is checked out of the project database.
- **Step 8 :** The changes are then made on the corresponding object and appropriate SQA activities are then applied.
- **Step 9 :** The changed object is then checked in to the database and appropriate version control is made to create new version.

The checked in and checked out mechanisms require two important elements -

- Access control
- Synchronization control

The **access control** mechanism gives the authority to the software engineer to access and modify the specific configuring object. The **synchronization control** mechanism allows to make parallel changes or the changes made by two different people without overwriting each other's work.

5.12.3.3 Version Control

- Version is an instance of a system which is functionally distinct in some way from other system instances.
- Version control works to help manage different versions of configuration items during the development process.
- The configuration management allows a user to specify the alternative configurations of the software system by selecting appropriate version.
- Certain attributes are associated with each software version. These attributes are useful in identifying the version. For example : The attribute can be 'date', 'creator', 'customer', 'status'.
- In practice the version needs an associated name for easy reference.

- Different versions of a system can be shown by an evolution graph as shown in Fig. 5.12.4.
- Each version of software system is a collection of software configuration items.

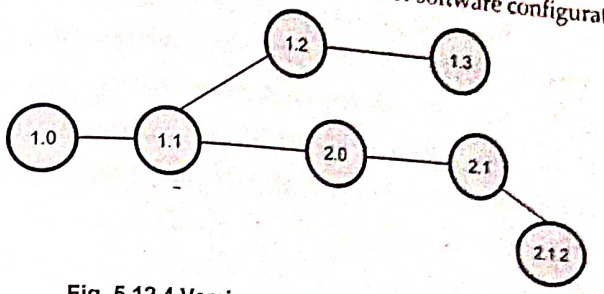


Fig. 5.12.4 Version numbering in evolution graph

5.12.3.4 Configuration Audit

- In order to ensure that the change has been properly implemented or not two activities are carried out.
  1. Formal Technical Review (FTR)
  2. Software Configuration Audit
- In Formal Technical Review, the correctness of configuration object is identified and corrected. It is conducted by technical reviewer.
- The software configuration audit assess the configuration object for the characteristics that are not reviewed in formal technical review. It is conducted by software quality assurance group.
- Following are some primary questions that are asked during configuration audit -
  1. Whether FTR is conducted to assess the technical correctness ?
  2. Whether or not the change specified by ECO has been made ?
  3. If additional changes need to be made or not ?
  4. Whether the software engineering standards are properly followed ?
  5. Do the attributes of configuration objects reflect the change ?
  6. Whether all the SCI are updated properly ?
  7. Whether the SCM process (object identification, change and version control, configuration audit and status reporting) are properly followed ?
- The above questions can be asked as a part of formal technical review.

5.12.3.5 Status Reporting

- The status reporting focuses on communication of changes to all people in an organization that involve with changes.
- During status reporting following type of questions were asked.
  1. **What happened ? :** What are the changes that are required ?
  2. **Who did it ? :** Who will be handling these changes ?
  3. **When did it happen ? :** The time at which these changes are arised.
  4. **What else will be affected ? :** The objects or part of the software that might be reflected due to these changes.

Review Questions

1. *What is configuration management repository ? Discuss role and features of SCM repository.*
2. *Which are the layers of SCM process ? Explain each in detail.*

5.13 Project Scheduling

AU : May-05,15,16,17,22, Dec.-06,07,15,16, Marks 16

- 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**.

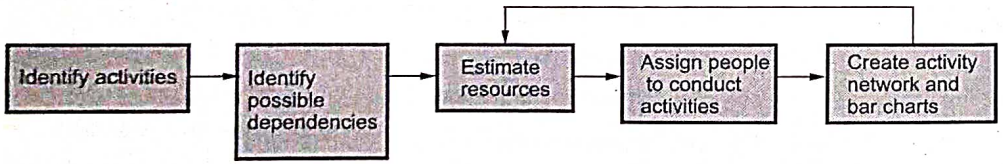


Fig. 5.13.1 Project scheduling process

- 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 :

i) People may leave or remain absent.

ii) Hardware may get failed.

iii) 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 -

i) Human effort.

ii) Sufficient disk space on server.

iii) Specialized hardware.

iv) Software technology.

v) 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.

5.13.1 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.

$$E_a = m\left(t_d^4 / t_d^4\right)$$

Where

E_a denotes the effort in person months

t_d denotes the nominal delivery time for the schedule

t_a denotes the actual delivery time desired.

- Let t_o will be the delivery time at which least effort is expended.
- As we move to left of t_o and accelerate delivery the curve rises non linearly.
- 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 non linear 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

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

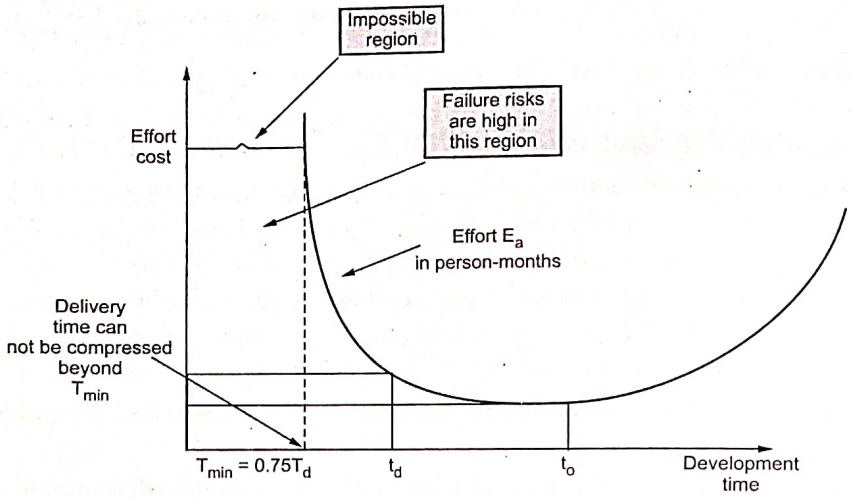


Fig. 5.13.2 Effort and delivery time

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

5.13.2 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 -
  - **Defining scope :** This task is for defining the scope, goal or objective of the project.
  - **Planning :** It includes the estimate for schedule, cost and people for completing the desired concept.
  - **Evaluation of technology risks :** It evaluates the risk associated with the

technology used in the project.

- **Concept implementation :** It includes the concept representation in the same manner as expected by the end user.

Example 5.13.1 *How to compute a task set selector for the project ? Explain with suitable illustration.* **AU : IT, Dec.-07, Marks 8**

Solution : Before computing the task set selector value let us list out the 'adaption criteria' used for software process in software project.

1. Size of project.
2. Total number of skilled users.
3. Complexity and difficult level of mission.
4. How long the application will exist ?
5. Requirements stability.
6. Customer-developer communication.
7. Project staff.
8. Maturity of technology used.
9. Performance constraints.
10. Re-engineering factor.
11. Embedded and non embedded characteristics.

Now following are the steps that must be followed for computing task set selector.

Step 1 : Prepare a table in which all the above mentioned adaption criteria must be entered. Assign appropriate grade to each adaption criteria. These grades range from 1 to 5.

Step 2 : Then assign corresponding weighting factor to each adaption criteria. This weighting factor ranges from 0.8 to 1.2.

Step 3 : Then there are some entry point multiplier based on -

i) Concept development ii) New development

iii) Enhanced application iv) Maintenance project

v) Re-engineering project

These values can be 0 or 1 indicating relevance of adaption criteria.

Step 4 : Computer product value for each production criteria using -

Grade × Weighting factor × Entry point multiplier

Step 5 : Take average of all 'product' values. This will indicate task set selector.

For example

Adaption criteria	Grade	Weighting factor	Entry point multiplier					Product
			Conc. dev.	New dev.	Enha.	Maint.	Re-engg.	
Size of project	3	1.1		1				3.3
Total number of skilled users	3	1.1		1				3.3
Complexity of mission	4	1.2		1				4.8
How long application will exist	3	0.9		1				2.7
Requirement stability	2	1.1		1				2.2
Customer-developer comm.	2	1.0		1				2.0
Project staff	3	1.2		1				3.6
Maturity of technology	2	1.0		0				2.0
Performance constraint	3	0.8		1				2.4
Reengineering factor	0	1.1		0				0
Embedded and non-embedded characteristics	2	1.1		1				2.2
								2.6

Task set selector value

5.13.3 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 Fig. 5.13.3.

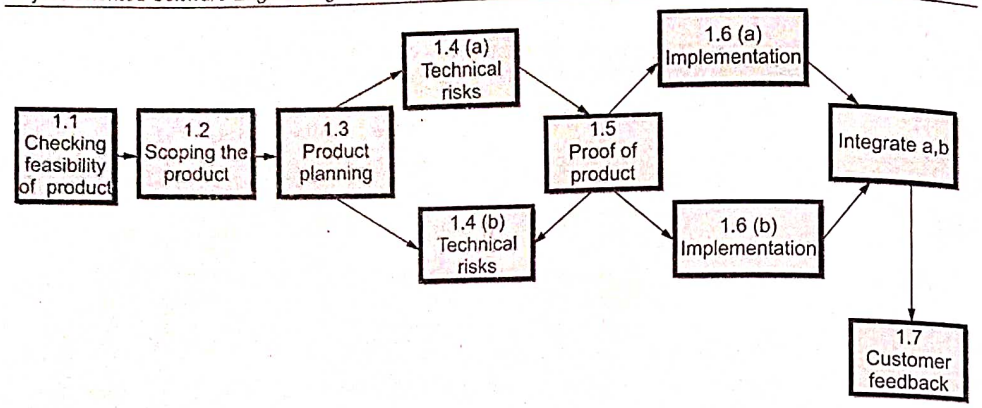


Fig. 5.13.3 Task network

- 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.

5.13.4 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.

Example

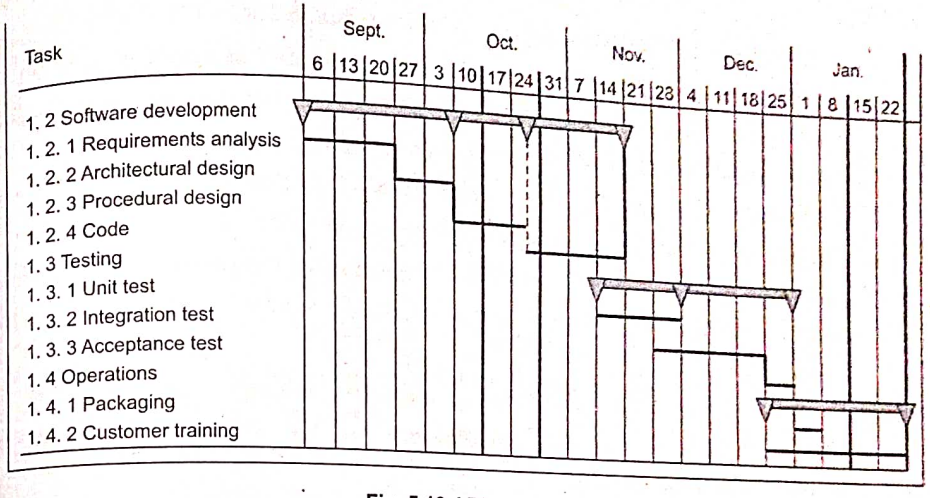


Fig. 5.13.4 Time line chart

Project table

Tasks	Planned start	Actual start	Planned end	Actual end	Effort assignment
Requirement analysis	6th Sept'05	6th Sept'05	20th Sept'05	22nd Sept'05	Jayashree, Padma, Lucky
Architectural design	27th Sept'05	27th Sept'05	3rd Oct'05	7th Oct'05	Trupti, Varsha
Procedural design	10th Oct'05	12th Oct'05	24th Oct'05	25th Oct'05	Varsha, Sachin, Devendra
⋮	⋮	⋮	⋮	⋮	⋮
Customer training	1st Jan'06	4th Jan'06	22nd Jan'06	25th Jan'06	Smita, Yogita

Table 5.13.1 Project table

5.13.5 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 ways by which tracking of schedule can be done -

1. **Periodic meeting -** The regular periodic meetings should be conducted. In these meeting each team member should report the progress of the project. Also various problems encountered in project should be openly discussed.
2. **Evaluation of reviews -** The results of all the reviews during software engineering process should be evaluated.
3. **Milestone accomplishment -** To track the schedule of the project it is necessary to determine whether formal project milestones have been accomplished on the scheduled date.
4. **Meeting with practitioners -** Conduct informal meetings with practioners to obtain subjective assessment of progress to date. Also the problems arising in project should be discussed.
5. **Earned value analysis -** The project can be assessed quantitatively using earned value analysis.

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 understood 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 boundry 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.

5.13.6 Earned Value Analysis

- The Earned Value Analysis (EVA) takes into consideration the project context for planned and actual expenditure.

- This analysis is made to find out project scope, schedule and resource characteristics.
- The EVA acts as a measure for software project progress.
- Various measures are determined during EVA. These measures are -

1) **Planned Value (PV) :** It denotes the planned cost of the work. The planned value is developed by first determining all of the work, that must be accomplished for successful project result.
2) **Actual Cost (AC) :** It represents the actual amount that the business has to expend on the project.

$$AC = \sum \text{Efforts expended on work task have been completed by time t}$$

3) **Earned Value (EV) :** It is project manager's estimate of the amount of originally budgeted work completed.
4) **Budget At Completion (BAC) :** It represents total budget for the project.
5) **Schedule variance (SV) :** It indicates the status of the schedule. It represents whether work is ahead or behind the plan. If SV is negative the project is behind schedule, if it is positive then project is ahead of schedule. If SV is equal to zero, then project is on schedule.
6) **Cost Variance (CV) :** It is the difference between earned value and actual cost.

   If CV = 0 then project is on budget.

   If CV < 0 then project is over budget

   If CV > 0 then project is under budget
7) **Schedule Performance Index (SPI) :** It is a measure of schedule efficiency on a project. If SPI = 1.0, project is on schedule.

   If SPI is greater than 1 then project is ahead of the schedule.

   IF SPI is less than 1 then project is getting delayed and it is behind the schedule.
8) **Cost Performance Index (CPI) :** It is a measure of cost efficiency on a project. The value 1.0 represents that project is within the given budget.

   If CPI < 1.0 then that means project is over budgeted.

   If CPI > 1.0 then that means project is under budgeted and we require most cost to accomplish the project.

Formula used during (EVA) :

PV = Planned Completion (%) * Budget At Completion (BAC)

EV = Actual Completion (%) * BAC

SV = EV – PV

CV = EV – AC

SPI = EV/PV

CPI = EV/AC

Example 5.13.2 *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 $ 50,000.000, What is the Earned value ?*

AU : Dec.-15, Marks 2

Solution : The formula for Earned value is -

$$\text{Earned value} = \% \text{ of work} \times \text{Budget}$$

$$= 27\,\% \times 50000$$

$$= \frac{27}{100} \times 50000 = 13,500$$

∴ The earned value is **$ 13,500.**

Example 5.13.3 *Suppose you have a budget cost of a project as ₹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 ₹ 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.*

AU : Dec.-16, Marks 8

Solution : The abbreviations that we will use are as follows

EV = Earned Value

PV = Planned Value

BAC = Budget At Completion

AC = Actual Cost

Following formula will be used.

PV = Planned Completion(%) * BAC

EV = Actual Completion(%) * BAC

CPI = EV/AC

SPI = EV/PV

Given that :

BAC = ₹ 9,00,000

AC = ₹ 1,00,000

1) Planned Value (PV) = Planned Completion (%) * BAC

$$= 15\,\% * 900000$$

$$= \frac{15}{100} * 900000$$

∴ PV = ₹ 135000

2) Earned Value (EV) = Actual Completion (%) * BAC

$$= 10\,\% * 900000$$

$$= \frac{10}{100} * 900000$$

∴ EV = ₹ 90000

3) Cost Performance Index (CPI) = $\frac{EV}{AC}$

$$= \frac{900000}{100000}$$

CPI = 0.90

As CPI < 1 means project is **over budget.**

4) Schedule Performance Index (SPI) = $\frac{EV}{PV}$

$$= \frac{900000}{135000}$$

SPI = 0.67

As SPI < 1 means project is **behind schedule.**

This indicates project is in real trouble.

Example 5.13.4 *Suppose you are managing a software development project. The project is expected to be completed 8 months at a cost ₹10000 per month. After 2 months, you realize that project is 30 percent completed at a cost of ₹40,000. You need to determine whether the project is on-time and on-budget after 2 months.*

Solution : Given that

- Budget at completion (BAC) = 10,000 * 8

  = ₹ 80,000

- Actual Cost (AC) = ₹ 40,000
- Planned Completion = 2/8 = 25 %

- Actual Completion = 30 %

We will compute Planned Value (PV) and Earned Value (EV)

i) Planned Value (PV) = Planned Completion (%) * BAC

= 25% * 80,000

$= \frac{25}{100} * 80{,}000$

PV = ₹ 20,000

ii) Earned Value (EV) = Actual Completion (%) * BAC

= 30% * 80,000

$= \frac{30}{100} * 80{,}000$

EV = ₹ 24,000

Now we will compute the Cost Performance Index (CPI) and Schedule Performance Index (SPI).

iii) Cost Performance Index (CPI) $= \frac{EV}{AC}$

$= \frac{24{,}000}{40{,}000}$

CPI = 0.6

iv) Schedule Performance Index (SPI) $= \frac{EV}{PV}$

$= \frac{24{,}000}{20{,}000}$

SPI = 1.2

The CPI < 1, then that means the project is over budget. For every ₹ 1 spent we are getting 60 paisa worth performance.

The SPI > 1, then that means the project is ahead of the schedule. That means project can be delivered before the given schedule.

Example 5.13.5 *Given the following project plan of tables Table 5.13.2 and Table 5.13.3 :*

Table 5.13.2

ID	Task	Immediate predecessor (*)	Expected duration (days)	Budget ($)
A	Meet with client		5	500
B	Write SW	A	20	10000
C	Debug SW	B	5	1500
D	Prepare draft manual	B	5	1000
E	Meet with clients	D	5	1000
F	Test SW	C, E	20	2000
G	Make modifications	F	10	8000
H	Finalized manual	G	10	5000
I	Advertise	C, E	20	8000

(*) all dependencies are assumed to be FS-Finish to Start and the following progress status :

Table 5.13.3

ID	Task	Status	Actual start (days)	Actual duration (days)	Actual costs ($)
A	Meet with client	100 %			1500
B	Write SW	100 %	+ 5 days	+ 10 days	9000
C	Debug SW	100 %	+ 5 days	+ 5 days	2500
D	Prepare draft manual	100 %	As per other delays		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 week 13, using EVA. Use the CPI and SPI to determine project efficiency. Explain the process involved. **AU : May-17, Marks 6**

Solution : To perform analysis of the project we will compute PV, AC and EV.

- **PV** is the sum of planned cost. It is computed by determining for each reporting period, the cost associated with each activity and by summing and cumulating them over time. The cumulative cost can be computed by summing planned expenditure week by week.

		W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	Total
A	Meet with client	500													500
B	Write SW		2500	2500	2500	2500									10000
C	Debug SW						1500								1500
D	Prepare draft manual						1000								1000
E	Meet with clients							1000							1000
F	Test SW								500	500	500	500			2000
G	Make modifications												4000	4000	8000
H	Finalize manual														0
I	Advertise								2000	2000	2000	2000			8000
	Total	500	2500	2500	2500	2500	2500	1000	2500	2500	2500	2500	4000	4000	
	Planned Value	500	3000	5500	8000	10500	13000	14000	16500	19000	21500	24000	28000	32000	

Note that each week is of 5 days.

$\therefore$ PV = 32000

- **AC** is the sum of the actual cost. The summing of actual cost is as shown below.

		W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	Total
A	Meet with client	1500													500
B	Write SW			1500	1500	1500	1500	1500	1500						9000
C	Debug SW									1250	1250				2500
D	Prepare dragt manual									1000					1000
E	Meet with clients										1000				1000
F	Test SW											250	250	250	750
G	Make modifications														0
H	Finalize manual														0
Z	Advertise												500	500	1000
	Total	1500	0	1500	1500	1500	1500	1500	1500	2250	2550	250	750	750	
	Planned Value	1500	1500	3000	4500	6000	7500	9000	10500	12750	15000	15250	16000	16750	

$\therefore$ AC = 16750

- EV is the earned value which is sum of the planned cost on actual schedule. The actual result is as shown below

		W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	Total
A	Meet with client	500													500
B	Write SW			5000	0	0	0	0	5000						10000
C	Debug SW									750	750				1500

D	Prepare dragt manual									1000					1000
E	Meet with clients										1000				1000
F	Test SW											1000	0	0	1000
G	Make modifications														0
H	Finalize manual														0
I	Advertise												4000	0	4000
	Total	500	0	5000	0	0	0	0	5000	1750	1750	1000	4000	0	
	Planned Value	500	500	5500	5500	5500	5500	5500	10500	12250	14000	15000	19000	19000	

$\therefore$ EV = 19000

- From above data, at W13 the following things are observed

i) PV > AC indicates that project is under budget.

ii) EV < PV means project is late.

- $CPI = \frac{EV}{AC} = \frac{19000}{16750}$

≈ 1.13

The CPI is cost performance index which is > 1. It indicates that project is within the given budget.

- $SPI = \frac{EV}{PV} = \frac{19000}{32000}$

≈ 0.6

The SPI is schedule performance which is < 1. It indicates that the project is late.

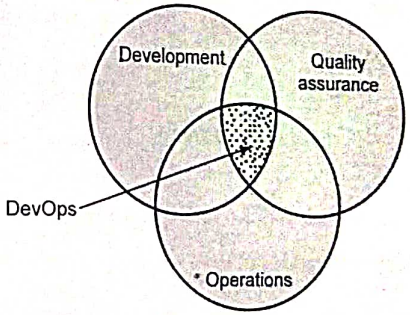
Review Questions

1. *How to track the schedule ? Give an illustration.*
2. *Write short notes on the following.* **AU : Dec.-06,07. Marks 8**
   i) Project scheduling.
   ii) Project timeline chart and task network.
3. *Discuss Putnam resources allocation model. Derive the time and effort equations.* **AU : May-15. Marks (8+8)**
4. *Discuss about any one project scheduling technique.* **AU : May-16. Marks 12** **AU : May-22. Marks 6**

5.14 DevOps

Definition : DevOps is a practice in which development and operation engineers participate together in the entire lifecycle activities of system development from design and implementation to product support.

- The term DevOps is derived from "Software **DEV**elopment " and "information technology **OPerationS**".
- DevOps promotes a set of processes and methods from the three department **Development**, IT **operations** and **Quality assurance** that communicate and collaborate together for development of software system.



5.14.1 Why DevOps ?

- DevOps enhances the organization's performance, improves the productivity and efficiency of development and operations teams.
- Bringing the two teams together centralizes the responsibility on the entire team and not specific individuals working.
- DevOps is more than just a tool or a process change. It inherently requires an organizational culture shift.

- This cultural change is especially difficult, because of the conflicting nature of departmental roles :
  1. **Operations -** seeks organizational stability;
  2. **Developers -** seek change;
  3. **Testers -** seek risk reduction.
- Adoption of DevOps is driven by various factors. These factors are -
  1. Demand for an increased rate of production releases - from application and business unit stakeholders.
  2. Increased usage of data center automation and configuration management tools.
  3. Use of agile and other development processes and methods.
  4. Increased focus on test automation and continuous integration methods.
  5. Wide availability of virtualized and cloud infrastructure.

5.14.2 Motivation

The Goals of DevOps are as follows -

1. To make simple processes increasing programmable and dynamic.
2. Fast delivery of product.
3. Lower failure rate of new releases.
4. Shortened lead time between fixes.
5. Faster mean time to recovery.
6. Increases net profit of organization.
7. To standardize the development environment.
8. To reduce work in progress.
9. To reduce operating expenses.
10. To set up the automated environment.

5.14.3 Benefits

Various benefits of DevOps are -

➢ **Technical Benefits**

1. Continuous software delivery is possible.
2. There is less complexity in managing the project.

- The problems in the project gets resolved faster.

➢ **Cultural benefits**

1. The productivity of teams get increased.
2. There is higher employee engagement.
3. There arise greater professional development opportunities.

➢ **Business benefits**

1. The faster delivery of the product is possible.
2. The operating environment becomes stable.
3. Communication and collaboration are improved among the team members and customers.
4. More time is available for innovation rather than fixing and maintaining.

5.14.4 Agility and DevOps

- Basically Agile and DevOps are similar. But there lies some differences.
- DevOps brings more flexibility than Agile. With Continuous Integration (CI) and Continuous Delivery (CD) , the release of software products is made often and it is ensured that these releases actually work and meet the customer needs.
- Thus in DevOps there are an increased number of releases.
- One goal of DevOps is to establish an environment where releasing more reliable applications, faster and more frequently, can occur. This actually brings the **continuous delivery approach.**
- DevOps is not a separate concept but a mere extension of Agile to include operations as well to collaborate different agile teams together and work as ONE team with an objective to deliver software fully to the customer.

Sr.No.	Agile	DevOps
1.	The idea in Agile is to develop software in small iterations and be thus able to adapt to the changing customer needs.	DevOps is to deliver technology to business units in a timely fashion and ensures the technology runs without interruption or disruption.
2.	It adopts a rapid development approach.	This is not a rapid development approach.
3.	The focus of agile development is merely on software development and release.	The focus of DevOps is not only on software development and its release but on it safest deployment in the working environment.

4.	In agile development, every team member has a skill of design, development and coding. Any available team member should be able to do what's required for progress.	DevOps, on the other hand, assumes there will be development teams and operational teams, the two will be separate. These teams can communicate between themselves on a frequent and regular basis.
5.	Communication in agile development is informal and in the form of daily meetings.	In DevOps communication, specifications, and documents are involved and it is formal. It does not occur on a daily basis.
6.	The team is small in nature.	Large team sizes and multiple teams are required in DevOps.
7.	Agile is about software development.	DevOps is about software development and management.
8.	Documentation is not very important in Agile development.	Documentation is very important in DevOps.
9.	Agile development teams may choose to use certain automation tools. However, there are no specific tools required for an agile team.	DevOps absolutely depends on automated development to make everything happen smoothly and reliably. Certain tools are an integral part of DevOps.
10.	Agile is less flexible.	DevOps is more flexible.
11.	Agile has limited scope.	DevOps has a broader scope.

5.14.5 Deployment Pipeline

- A **Pipeline** is a set of automated processes that allow developers and DevOps professionals to reliably and efficiently compile, build and deploy their code to their production computing platform.
- Various components of a pipeline are -
  1) Build Automation
  2) Test Automation
  3) Deploy Automation
- Deployment pipeline is an important concept and practice in DevOps that involves automating and Streaming the process of delivering the software application from development to production.

- There are two important concepts used in the Deployment pipeline -
  1. Continuous Integration(CI) and
  2. Continuous Delivery(CD)/Continuous Deployment.

1. Continuous Integration :

- The continuous integration in DevOps is a practice where developers regularly merge their code changes into a central repository or a database after which automated builds and tests are run.
- Continuous Integration (CI) is the practice of **automating the integration of code changes** from multiple developers or testers into a single software project.
- Automated tools are used to assert the new code's correctness before integration.
- Continuous integration serves as a prerequisite for the testing, deployment and release stages of continuous delivery.
- The main benefit of performing continuous integration regularly and testing each integration is that we can detect errors more quickly and locate them easily.

2. Continuous Delivery or Continuous Deployment :

- Continuous Delivery means a developer's changes to an application are automatically bug-tested and uploaded to a repository.
- Continuous Deployment (CD) refers to the final stage in the pipeline that refers to the automatic releasing of any developer changes from the repository to the production.

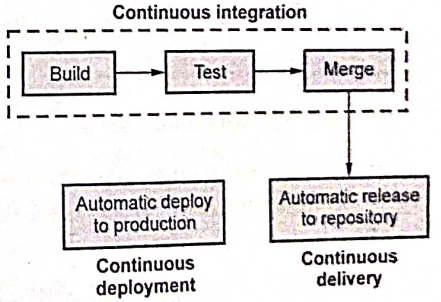


Fig. 5.14.1 Continuous Integration (CI) and Continuous Delivery / Deployment (CD)

- The key components of the deployment pipeline are -
  - **Source Code Management :** This is the first step of the deployment pipeline in which source code is stored in a version control system like Git and GitHub. Developers commit the changes (upload final changes) to this repository and pipeline is triggered when there are new commits.
  - **Build :** In this process, the code is built into executable entities. During the build process, the code is compiled, dependencies are packaged and binaries are created.
  - **Automated Testing :** The pipeline runs a suite of automated tests in order to test the software system. It includes unit testing, integration testing and performance testing.
  - **Deployment :** Once the code passes testing it is deployed to a staging or testing environment that closely resembles the production environment. This allows for further testing and validation in a controlled setting.
  - **User Acceptance Testing (UAT) :** In this stage, the software is tested by a group of users or stakeholders to ensure it meets their expectations.
  - **Final Deployment :** If all tests and checks are successful, the software is deployed to the production environment. This can be done manually, automatically or with a combination of both.
  - **Monitoring and Feedback :** The application is continuously monitored, if any issues are raised then these are fixed immediately. Feedback from the production environment is used to inform future development and improvements.
  - **Documentation :** Finally the comprehensive documents and reports are prepared.

Fig. 5.14.2 Deployment pipeline

Deployment pipelines are typically designed as automated services. This approach reduces the manual errors, accelerates the delivery process of the software system.

5.14.6 Overall Architecture

- DevOps is a practice of operations and development.

There are different phases of DevOps architecture -

1) **Plan :** In this phase, all the requirements of the project are gathered. The schedule and cost of the project is estimated approximately.
2) **Code :** In this phase the code is written as per the requirements. Entire project is divided into smaller units. Each unit can be coded as a module.
3) **Build :** In this phase, the building of all the units is done using tools such as Maven, Gradle to submit the code to a common code source.

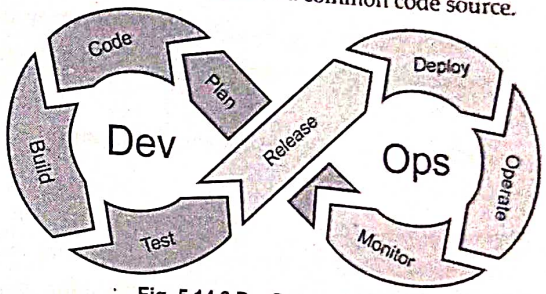


Fig. 5.14.3 DevOps architecture

4) **Test :** At this stage, all the units are tested to find if there exists any bug in the code. The testing can be done using tools like Selenium, JUnit, PYtest. Some important testing techniques such as acceptability testing, safety testing, integration checking, performance testing are carried out.
5) **Integrate :** In this phase, a new feature is added to the existing code and testing is performed. Continuous Development is achieved only because of continuous integration and testing.
6) **Deploy :** In this stage, the code is deployed in the client's environment. Some of the examples of the tools used for Deployment are AWS, Docker.
7) **Operate :** At this stage, the version can be utilized by the users. Operations are performed on the code if required. Some of the examples of the tools used are Kubernetes, open shift.
8) **Monitor :** At this stage, the monitoring of the version at the client's workplace is done. During this phase, developers collect data, monitor each function and spot

errors like low memory or server connection are broken. The DevOps workflow is observed at this level depending on data gathered from consumer behavior, application efficiency and other sources. Some of the examples of the tools used are Nagios, elastic stack for monitoring.

5.14.7 DevOps Lifecycle

The DevOps lifecycle phases are as follows -

1) **Continuous development :** In this phase, the planning and coding of software is done. Version control mechanism is used during this phase.
2) **Continuous integration :** In this phase, developers are required to commit changes in the source code frequently. The code supporting new functionality is continuously integrated with the existing code. Therefore, there is continuous development of software.
3) **Continuous testing :** In this phase, the software is continuously tested for bugs. Many times automation testing is preferred.
4) **Continuous monitoring :** By continuous monitoring, we can get notified before anything goes wrong. We can gather many performance measures, including CPU and memory utilization, network traffic, application response times, error rates and others.
5) **Continuous feedback :** In this DevOps stage, the software automatically sends out information about performance and issues experienced by the end-user. It's also an opportunity for customers to share their experiences and provide feedback.
6) **Continuous deployment :** In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers. The deployment process takes place continuously in this DevOps life cycle phase.
7) **Continuous operations :** It is the last phase which involves automating the application's release and all these updates that help you keep cycles short and give developers more time to focus on developing.

5.14.8 Tools

Various tools used in DevOps are as follows -

1. **Nagios :** It is a monitoring solution that gives new features and a modern user experience.

2. **ELK Stack :** This tool is used for collecting logs from all services, applications, networks, tools, servers and more in an environment into a single, centralized location for processing and analysis.
3. **Docker :** It eases configuration management and control issues.
4. **Jenkins :** Jenkins is a top tool for DevOps engineers who want to monitor executions of repeated jobs.
5. **Puppet :** It handles configuration management and software while making rapid, repeatable changes in it.
6. **Ansible :** Ansible is a configuration management tool or DevOps tool that is similar to Puppet and Chef.
7. **God :** It is a monitoring tool used in DevOps.
8. **Monit :** Monit has everything DevOps engineers need for system monitoring and error recovery.
9. **Consul.io :** This tool is used for service discovery and configuration management activities.
10. **Loggly :** This tool is used for log management in DevOps.

Review Questions

1. *What is DevOps ? What are the benefits of DevOps ?*
2. *Compare agility with DevOps.*
3. *Explain in detail the deployment pipeline.*
4. *Explain the overall architecture of DevOps.*
5. *Explain DevOps lifecycle.*
6. *What are different tools used in DevOps ?*

5.15 Cloud as a Platform

- "Cloud as a platform" is a concept that refers to using cloud computing infrastructure and services as the foundation for developing, deploying and running applications and services.
- Cloud computing is a term that refers to storing and accessing data over the internet. It doesn't store any data on the hard disk of our personal computer. In cloud computing, we can access data from a remote server.
- It includes services for storage, databases, analytics, networking, mobile, development tools and enterprise applications.

- AWS manages and maintains hardware and infrastructure, saving organizations and individuals the cost and complexity of purchasing and running resources on site. These resources may be accessed for free or on a **pay-per-use** basis.
- Popularly used cloud platforms are - **Amazon Web Services (AWS), Microsoft Azure** and **Google Cloud Platform (GCP).**

➢ **Benefits of using Cloud as a Platform**

1) Flexibility :

- The cloud platform always allows to use the operating system, programming languages and web application platforms that user is comfortable with.
- Flexibility means that migrating legacy applications to the cloud should be easy.
- Instead of re-writing the applications to adopt new technologies, we just need to move the applications to the cloud and tap into advanced computing capabilities.

2) Cost Effective :

- Instead of purchasing and creating our own expensive servers, we can use cloud services where we need to pay only for the tools and services that we use.
- Cloud platform offers a pay-as-you-go pricing method, which means that we only pay for the services that are needed and have been used for a period of time.

3) Scalable and Elastic :

- Cloud platform is scalable because its Auto Scaling service automatically increases the capacity of constrained resources as per requirements so that the application is always available.
- Elasticity is one of the advantages. If we use fewer resources and don't need the rest of them, then cloud platform itself shrinks the resources to fit our requirements. In short, upsizing and downsizing of resources is possible with AWS.

4) Secure :

- Cloud computing maintains confidentiality, integrity and availability of the user's data.
- Each service provided by the cloud is secure.
- Personal and business data can be encrypted to maintain data privacy.

5) High Performance :

- High-performance computing is the ability to process massive amounts of data at high speed.
- Cloud computing offers a high-performance computing service so that the companies need not worry about the speed.

5.15.1 Operations

- It provides different services of the cloud such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and packaged Software as a Service (SaaS).
  - **IaaS :** The infrastructure as a service means delivering computing infrastructure on demand. Under this service, the user purchases the cloud infrastructure including servers, networks, operating systems and storage using virtualization technology. These services are highly scalable. IaaS is used by network architects. Examples of cloud services : AWS and Microsoft Azure.
  - **PaaS :** The Platform as a Service means it is a service where a third-party provider provides both hardware and software tools to the clients. It provides elastic scaling of your application which allows developers to build applications and services over the internet and the deployment models include public, private and hybrid. PaaS is used by developers. Examples of cloud services - are Facebook and Google Search Engine.
  - **SaaS :** Software as a Service model that hosts software to make it available to clients. It is used by the end users. Examples of Cloud services : Google Apps.

Difference between IaaS, PaaS and SaaS

Sr. No.	IaaS	PaaS	SaaS
1.	IaaS is an acronym for Infrastructure as a Service.	PaaS is an acronym for Platform as a Service.	SaaS is an acronym for Software as a Service.
2.	IaaS allows suppliers to share their physical servers, virtual machines or virtual storage to the users or customers.	It provides an environment to allow users to create their own web applications. PaaS provides a runtime environment for applications and the development of applications.	It allows the use of software applications as a service to end users. It allows users to access specific applications online.

3.	It is most expensive.	It has mid-level costs.	It is the cheapest.
4.	The network architects primarily use the IaaS.	Developers mainly make use of PaaS.	End-users generally use SaaS.
5.	Services available are - Virtual machines, operating systems, network, storage, backup services.	PaaS provides all the facilities required to build and develop web applications.	SaaS model provides services to deploy applications online.
6.	For example - AWS EC2.	For example - Microsoft Azure.	For example - Salesforce, DropBox.

Review Questions

1. *What is cloud computing ? Explain the benefits of using cloud computing.*
2. *Differentiate between various services offered by cloud computing platform.*

5.16 Two Marks Questions with Answers

Q.1 Write short notes on empirical estimation models. **AU : IT, May-04, 05**

Ans. : 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.

Examples of such models calculate effort as,

1. Walston-Felix model

$$E = 5.2 \times (KLOC)^{0.91}$$

2. Bohem simple model

$$E = 3.2 \times (KLOC)^{1.05}$$

Q.2 What is the standardization for the software metrics ? **AU : IT, Dec.-04**

Ans. : The standardization of software metrics indicates the attributes of effective software metrics. The software metrics should follow following standards.

1. Simple and computable - The software metrics should be easy to learn and derive.
2. Empirically and intuitively persuasive - The software metrics should satisfy the intuitive notions about the product attribute.
3. Consistent and objective - The software metrics should give the unambiguous results.

4. Consistent in its use of units and dimensions - The units and dimensions used in the software metrics should be consistent.
5. 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.
6. 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.

Q.3 What are project indicators and how do they help a project manager ?

AU : CSE, May-05, 06

Ans. : 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.

Q.4 Distinguish between direct and indirect measures of metrics. **AU : CSE, Dec.-04**

Ans. : 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.

Q.5 Differentiate between size oriented and function oriented metrics.

AU : IT, Dec.-06, May-13

Ans. :

Sr. No.	Size oriented metrics (LOC based)	Function oriented metrics
1.	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.
2.	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.

Q.6 Define basic equation for the effort estimation models. **AU : IT, May-07**

Ans. : 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.

Q.7 List down few process and product metrics. AU : CSE, Dec.-06

Ans. : 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

- 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.

Q.8 Define software measure. AU : CSE, May-08

Ans. : 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.

Q.9 How to measure the function point (FP) ? *(Refer section 5.8.5)* AU : IT,Dec.-07

Q.10 List out the different approaches to size of the software. AU : CSE, Dec.-08

Ans. : The two approaches used for estimating the size of the software are -

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

Q.11 Mention difference between organic mode and embedded mode in cocomo model. AU : IT, May-09

Sr. No.	Organic mode	Embedded mode
1.	The small size projects are handled using organic mode.	The large size projects are handled using embedded mode.
2.	The experienced developers develop such projects.	The embedded mode projects are handled by little previous experienced people.
3.	Deadline of the projects under organic mode is not tight.	The embedded mode projects are carried out with tight deadline.
4.	Example - Payroll system	Example - Whether forecasting system.

Q.12 Name the metrics for specifying non-functional requirements. AU : Dec.-11

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

Q.13 An organic software occupies 15,000 LOC. How many programmers are needed to complete ? AU : Dec.-12

Ans. : Given that :

System = Organic

Lines of Code = 15KLOC

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

$= 2.4(15)^{1.05}$

= 41 persons-month

$D = c_b(E)d_b$

$= 2.5(41)^{0.38}$

= 10 months

P = 41/10

P = 4 persons.

Approximately 4 programmers are required to complete the work.

Q.14 What is scheduling ?

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

Q.15 What is error tracking ? AU : May-14

Ans. : 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.

Q.16 Differentiate between size oriented and function oriented metrics

Ans. :

Sr. No.	Size oriented metrics	Function oriented metrics
1.	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.
2.	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.

Q.17 State the advantages and disadvantages in LOC based cost estimation **AU : May-15**

Ans. : Advantages

1. Artifact of software development can be 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.

Q.18 State the importance of scheduling activity in project management. **AU : May-15**

Ans. : Scheduling is carried out to assess progress of a software project. In order to build a complex system, many software engineering tasks occur in parallel. The result of work performed during one task may have a certain effect on work to be conducted in another task.

These interdependencies are very difficult to understand without a schedule.

Q.19 What are the issues in measuring the software size using LOC as metric ?

AU : Dec.-15

Ans. : 1) This measure is based on lines of code computation.
2) The LOC is not universally accepted metric.
3) This measure is extremely dependent upon the programming language.
4) This measure does not accommodate non procedural languages.

Q.20 List a few process and project metrics. **AU : May-16**

Ans. : 1) Size oriented metrics 2) Function oriented metrics
3) Object oriented metrics 4) Software quality metrics

Q.21 Differentiate between size oriented and function oriented metrics.

Ans. :

Sr. No.	Size oriented metrics	Function oriented metrics
1.	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.
2.	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.

Q.22 How is productivity and cost are related to function points ? **AU : Dec.-16**

Ans. : Using Function Point (FP) one can compute both productivity and overall cost of the project.

Productivity = FP/persons-months

Cost = Rupees/FP

Q.23 What are the different types of productivity estimation measures ? **AU : May-17**

Ans. : i) LOC based Estimation is used for productivity estimation as Productivity = KLOC/person-month.

ii) Function Point based estimation can be used for productivity estimation as

Productivity = FP/person-month.

Q.24 List out the principles of project scheduling. **AU : Dec.-17**

Ans. : 1) Compartmentalize : Project must be compartmentalized into a number of manageable activities and tasks.

2) Interdependency : The interdependency of each compartmentalized activity or task must be determined.

3) Time allocation : Each task to be scheduled must be allocated some number of work units.

4) Effort validation : The project manager must ensure that no more than the allocated number of people have been scheduled at any given time.

5) Defined responsibilities : Every task that is scheduled should be assigned to a specific team member.

Q.25 What is EVA ? **AU : May-18**

Ans. : EVA stands for Earned Value Analysis. This analysis is made to find out project scope, schedule and resource characteristics. It acts as a measure for software project progress.

Q.26 List two advantages of COCOMO model. AU : May-19

Ans. : The two advantages of COCOMO model are - 1) It predicts the efforts and schedule of software product based on size of the software.

2) It can work on historical data and is more predictable and accurate.

Q.27 Bring the importance between COCOMO I and II. AU : May-22

Ans. : COCOMO I model is useful in the waterfall models of software development cycle. COCOMO II is useful in non-sequential rapid development and reuse models of software.

The size of software is stated in terms of Lines of Code in case of COCOMO I model. The size of software is stated in terms of object points, function points and lines of code in COCOMO II model.

Q.28 State any two project scheduling techniques. AU : May-22

Ans. :

1) Taks network 2) Tracking schedule 3) Earned value analysis.

Q.29 If team A found 342 errors prior to the 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 ? AU : Dec.-22

Ans. : Following are some additional measures and metrics that are needed to find out if the teams have removed the errors effectively or not -

1) **Defect Density :** Calculate the defect density for each team, which is the number of defects found per unit of code, here the unit of code can be per thousand Lines of Code (LOC). This metric can help you compare the effectiveness of error removal relative to the size of the software.
2) **Defect Removal Efficiency (DRE) :** DRE is a measure of how effectively teams are removing defects. It's calculated as (Total defects found prior to release - Total defects found after release) / Total defects found prior to release. A higher DRE indicates better defect removal effectiveness.

❑❑❑