SOFTWARE PROJECT MANAGEMENT

CHAPTER 01

FUNDAMENTALS OF SOFTWARE PROJECT MANAGEMENT (SPM)

Introduction:

1. Software project management is a type of project management that focuses specifically on creating or updating software.
2. It is process of getting work done in time and within budget while meeting customer expectations.
3. Project management involves coordinating people, vendors, and resources.
4. Project management requires excellent communication skills, a strong will to protect the project scope, and leadership skills to enforce quality throughout the project work.
5. Project management involves coordinating people, vendors, and resources.
6. The project manager constantly makes decisions about the project. If those decisions are based on real information that's gathered by the team and trusted by management then the project would become a success.
7. Managing a project is all about forming a team and making sure that it is productive. The best way to do that is to rely on the expertise of the team members.
8. Every single role in a software project requires expertise, skill, training, and experience.
9. A software project has two main activity dimensions: engineering and project management. The engineering dimension deals with building the system and focuses on issues such as how to design, test, code, and so on. The project management dimension deals with properly planning and

controlling the engineering activities to meet project goals for cost, schedule, and quality.

Capability Maturity Model (CMM)

1. The CMM for software is a framework that was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University by observing the best practices in software and other organizations.
2. The CMM is one of the most popular frameworks for software process improvement
3. One objective of the CMM is to distinguish mature processes from immature ones.
4. **Immature software processes** imply that projects are executed without many guidelines, and the outcome of a project depends largely on the capability of the team and the project leader.
5. In **mature processes**, a project is executed by following defined processes.
6. Maturity of a process is directly proportional to more predictable results and more controlled projects.
7. **Process capability** is the range of results that can be expected from a project when it is executed using that process.
8. **Process performance** is the actual result achieved in a project which is executed using that process.
9. To consistently **improve process performance** on projects, we must **enhance the process capability**.
10. CMM has Five Maturity levels :

    (1) Level 1:-,

    - The Initial level
    - The project is executed in a manner that the team and project manager find it ok.

    (2) Level2:-

    - The Repeatable level.

- When standard project management practices are employed only for the projects and not for the organizational structure.

(3) Level3:-

- The Defined level
- Organization-wide processes have been defined and are regularly followed.

(4) Level4:-

- The Managed level
- Quantitative understanding of the process capability makes it possible to quantitatively predict and control the process performance on a project.

(5) Level5:-

- The Optimizing level
- The process capability is improved in a controlled manner and the improvement is evaluated quantitatively.

11. Each maturity level (except level 1) is characterized by key process areas (KPAs), which specify the areas on which the organization should focus to elevate its processes to achieve that maturity level.

Building a Software

Software is typically built by a team of software engineers, which includes:

- Business analysts or requirements analysts, who talk to the users and stakeholders (anyone who has an interest (or stake) in the software being completed), plan the behavior of software and write software requirements.
- Designers and architects who plan the technical solution.
- Programmers who write the code.
- Testers who verify that the software meets its requirements and behaves as expected.

Project Manager and Project Management

- The project manager plans and guides the software project.
- The project manager is responsible for identifying the users, stakeholders, engineers and determining their needs.
- The project manager coordinates the team, ensuring that each task is assigned to appropriate software engineer and that each engineer has sufficient knowledge to perform it.
- The project manager must be familiar with every aspect of software engineering.
- The project manager must have a **work breakdown structure (WBS), an effort estimate for each task, and a resource list** with availability status of each resource.
- He should understand the concepts behind the WBS, dependencies, resource allocation , critical paths, Gantt charts, and earned value.
- Must identify one or more people on the resource list capable of doing that task and assign it to them.
- Should be familiar with the relative expertise of each team member.
- Should also pay attention to professional development.
- Must make sure that the agenda at every meeting includes a discussion of whether the project is still on track.
- Should make sure that the representatives from the engineering team and stakeholders attend all of those meetings.
- Maintain a baseline schedule and track revisions to it.

Scope of Project

- The project manager drives the scope of the project.
- The project manager should identify and talk to the main stakeholder.
- Using the **vision and scope document** the project manager puts forward the intention of the project to satisfy the stake holders.

Vision and Scope Document

A typical vision and scope document has following points:

- **Problem Statement**
  - Project background (Source of Inspiration for the project).
  - Stakeholders.
  - Users.
  - Risks.
  - Assumptions.
- **Vision of the Solution**
  - Vision statement (Purpose of the project).
  - List of features.
  - Scope of phased release (optional).
  - Features that will not be developed.

Project Plan

The project plan defines the work that will be done on the project, the way to do things in the project and the persons who will do it. It consists of following points:

1. **Statement Of Work (SOW):**
   a. It describes all work products that will be produced and a list of people who will perform that work.
   b. It has a detailed description of all of the work products that will be created over the course of the project.
   c. A list of features that will be developed.
   d. A description of each intermediate deliverable or work product that will be built.
   e. The estimated effort involved for each work product to be delivered.
2. **Resource List**:
   a. A resource is a person, hardware, room or anything else that is necessary for the project but with limited availability.
   b. A resource list that contains a list of all resources that will be needed for the product and their availability (source from which the resource would come).

c. The resource list should give each resource a name, a brief description, and list the availability and cost (if applicable) of the resource.

3. **Work Breakdown Structure (WBS) and a set of Estimates:**
   a. A work breakdown structure (WBS) is a list of tasks that will generate all the work products needed to build the software.
   b. An estimate of the effort required for each task in the WBS is generated.
4. **Project schedule:**
   a. It contains the assignment of resources to each task.
   b. The calendar time required for each task.
5. **Risk plan:**
   a. It identifies any risks that might be encountered and indicates how those risks would be handled if the risk becomes reality.
   b. It is a list of all risks that threaten the project, along with a plan to mitigate some or all of those risks.
   c. The project manager selects team members to participate in a risk planning session
   d. The team members brainstorm potential risks.
   e. The probability and impact of each risk is estimated.
   f. A risk plan is constructed to encounter the situation if the risk becomes reality.
   g. It contains:
      i. Description of the Risk.
      ii. Probability factor.
      iii. Possible Impact factor.
      iv. Priority of the Risk situation.
      v. Possible Actions to be taken.

Project vs. Program Management

- Project Management includes Program Management.
- Program Management includes managing sub-projects that consist in the main project.
- Program management contributes to Project management.
- Both cannot exist individually.

Project Management Tools

Project Management tools include:

1. Financial tools
2. Cause and effect charts
3. Online Project Health Check tool (Ph-Check tool)
4. PERT charts
5. Gantt charts
6. Event Chain Diagrams
7. RACI diagram
8. Run charts
9. Project Cycle Optimization (PCO)
10. Participatory Impact Pathways Analysis

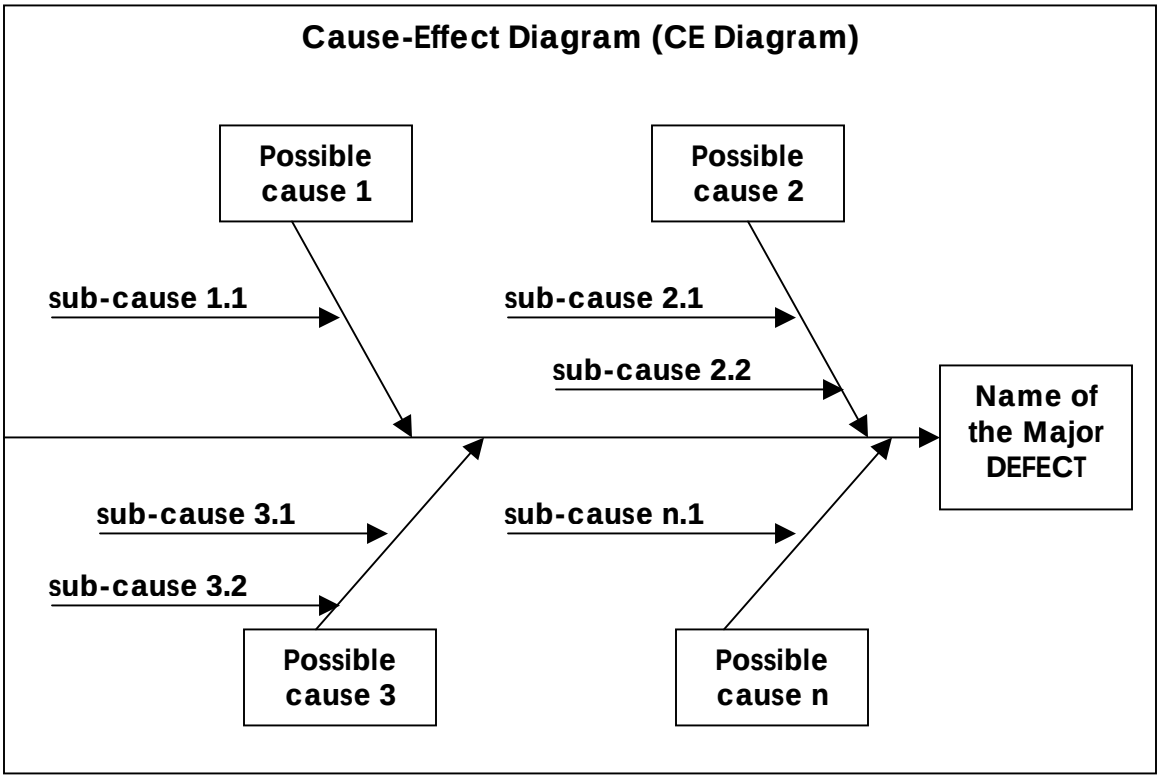
1. Financial tools :

- Analogous Estimating: Using the cost of similar project to determine the cost of the current project.
- Determining Resource Cost rates: The cost of goods and labor by unit gathered through estimates or estimation.
- Bottom Up estimating: Using the lowest level of work package detail and summarizing the cost associated with it. Then rolling it up to a higher level aimed and calculating the entire cost of the project.
- Parametric Estimating: Measuring the statistical relationship between historical data and other variable or flow.
- Vendor Bid Analysis: taking the average of several bids given by vendors for the project.

- Reserve Analysis: Aggregate the cost of each activity on the network path then add a contingency or reserve to the end result of the analysis by a factor determined by the project manager.
- Cost of Quality Analysis: Estimating the cost at the highest quality for each activity.

2. Cause and effect charts:

- The main purpose of the CE diagram is to graphically represent the relationship between an effect and its various possible causes. Understanding the causes helps to identify solutions to eliminate them.
- The main steps in drawing a cause-effect diagram are as follows:
    a. Clearly define the problem (the effect) to be studied. Here we can find the problems of partcular types that are occuring in the system.
    b. We Draw an arrow from left to right with a box containing the effect drawn at the head. This is the backbone of the diagram.
    c. We Determine the major categories of causes. These could be the standard categories or some variation to suit the problem.
    d. We then Write these major categories in boxes and connect them with diagonal arrows to the backbone.
    e. Then Brainstorm for the sub-causes of the major causes by asking repeatedly, for each major cause, "Why does this major cause produce the effect?"
    f. Then we Add the sub-causes to the diagram clustered around the major cause. If necessary, further we subdivide these causes and Stop when no worthwhile answer to the question can be found.



3. Online Project Health Check tool (Ph-Check tool) :

Here we try to find the answers to following attributes that are usually common to every project and then decide whether the project is in right path or not:

1. Scope
2. Cost
3. Time
4. Quality
5. Resources
6. Communication
7. Procurement
8. Risks
9. Contingency Planning
10. Benefits
11. Business Process

12. Training
13. Implementation
14. Governance
15. Roles and Responsibilities
16. Documentation
17. Requirements

1) Scope:

a. Was the scope well defined at the start of the project?
b. Is there a scope variation process in place?
c. Are there variations that have not been subject to that process?
d. Can the team identify the cumulative impact in time and cost of the variations?
e. Are actuals tracked against estimated impacts for variations?
f. Has the project plan been adjusted to cater for variations?

2) Cost:

a. Is there a cost tracking system in place?
b. Are costs being tracked?
c. Are any costs not being tracked properly?
d. Is the cost tracking up to date?
e. Are costs being reconciled against corporate accounts?
f. Are cost projections in place for the completion of the project?
g. Have any variances been approved in writing?

3) Time:

a. Is a suitably detailed schedule in place?
b. Is it up to date?
c. Does it reflect the current scope?
d. Are milestones being used to manage progress?
e. Are there regular enough milestones to track progress or are there weeks of the project that have no milestones?
f. Is time being monitored through a time sheeting system?
g. Is there a solid and realistic plan for the remaining work?

4) Quality:

a. Is there a quality plan in place?
b. Are deliverables subject to a quality plan?
c. What actions happen after a quality review, and are they monitored?
d. Does the plan have time allocated for rework after quality review?
e. Is the test plan realistic (test strategy, test plan and test cases in place)?
f. Are the appropriate people allocated to testing?
g. Is testing inclusive of system, performance and interface testing?

5) Resources:

a. Are there sufficient resources?
b. Are they the right resources?
c. Are they able to allocate sufficient time to the project?
d. Is there a sense of cooperation within the project?
e. Are the team being well managed?
f. Are the resources being used efficiently?
g. What is morale like in the team?

6) Communication:

a. Is there a communications plan in place?
b. Is it being implemented?
c. Have all key stakeholders been identified?
d. Do the stakeholders feel they are being kept up to date?
e. Are there any general areas of concern in communication?
f. Is the project likely to deliver corporate change and is there a plan to manage expectations?
g. Is that plan being implemented and expectations monitored?

7) Procurement:

a. Are external resources being used?
b. Are up to date contracts in place to manage those resources through to the end of the project?
c. Is there a contingency to retain those resources beyond the scheduled end of the project if required?

d. Does the project involve purchase of plant and equipment or software, and if so is there a plan in place for contract negotiation?
e. Are the authorities and timeframes for contract negotiation understood and built into the plan?

8) Risks:

a. Is there an up to date risk plan in place?
b. Have key stakeholders been involved in developing the plan?
c. Are there mitigation strategies in place, and are they being monitored?
d. Are regular risk reviews undertaken?
e. Is there an issues log?
f. Are issues managed to resolution?
g. Is there an issue escalation process in place? Does it work?

9) Contingency Planning:

a. Is there a contingency plan in place if the implementation fails to meet the deadline?
b. Has the plan been signed off by all concerned?
c. Has the plan been tested?
d. Are the impacts of a contingency plan understood by management?
e. What is the timeframe to implement the plan?
f. How long can the contingency arrangements last? What will happen then?

10) Benefits:

a. Were benefits identified at the start of the project?
b. Have they been reviewed recently?
c. Are they still relevant?
d. Have new benefits been identified and are they built into the project outcome?
e. Are the benefits being, or will they be, measured?
f. Have benefit delivery owners been identified?

11) Business Process:

a. Will there be an impact on business processes?
b. Are they being planned for by the business?
c. How and when will they be implemented?
d. Do we know they will work?
e. What impacts are likely to occur outside the company?
f. Are they being planned for?

12) Training:

a. Is there a training plan in place?
b. Is there time to produce training materials?
c. Are the trainers identified?
d. Will staff be available for training?
e. Is there a plan to pilot the training?

13) Implementation:

a. Is there an implementation plan in place (assuming the project is well progressed)?
b. What support will be provided during and after launch?
c. Who will authorise 'go live'?
d. Are there criteria for 'go live'?
e. If the implementation involves data conversion, is the state of the data known?
f. Will the data require manipulation, and has this been tested?

14) Governance:

a. Are there checkpoints for review by a management group to determine if the project is meeting organisational standards? Obviously a health check is one such process but typically a project should have some gates to pass through where certain criteria will need to be met.
b. Do the project team have the necessary tools, skills and processes to undertake the project successfully?
c. Who is monitoring compliance?
d. If the company has a methodology, is it being adhered to?

15) Roles and Responsibilities:

a. Are roles and responsibilities clearly defined?
b. Are they an accurate reflection of what is really happening?
c. Is the business providing the necessary level of support?
d. Is there sufficient executive support for the project?
e. Are any responsibilities not covered in the definition of the R&R?

16) Documentation:

a. Are documents held in a central project location?
b. Is the location well organised to enable documents to be easily located?
c. Is proper version control in place?
d. Are the major documents generally well constructed?
e. Are agendas and minutes kept of key meetings?
f. Are proper signed authorities in place for key decisions?
g. Is there a project glossary?
h. Is there a decision register?

17) Requirements:

a. Are requirements well documented?
b. Is there a tracking system for requirements and changes to requirements?
c. Do the deliverables reflect the requirements documents or did they morph during development?
d. Are reasons for changes to requirements documented along with who approved the change?

4. PERT charts (diagrams):

- The Program (or Project) Evaluation and Review Technique, commonly abbreviated PERT, is a model for project management designed to analyze and represent the tasks involved in completing a given project, especially the time needed to complete each task, and identifying the minimum time needed to complete the total project.
- PERT was developed primarily to simplify the planning and scheduling of large and complex projects. It was able to incorporate uncertainty by making it possible to schedule a project while not knowing precisely the details and durations of all the activities.
- It is more of an event-oriented technique rather than start- and completion-oriented, and is used more in R&D-type projects where time, rather than cost, is the major factor.
- **Conventions used in PERT:**
  1) A PERT chart is a tool that facilitates decision making; The first draft of a PERT chart will number its events sequentially in 10s (10, 20, 30, etc.) to allow the later insertion of additional events.
  2) Two consecutive events in a PERT chart are linked by activities, which are represented as arrows in the diagram.
  3) The events are presented in a logical sequence and no activity can complete until its immediately preceding event(s) is completed.
  4) The planner decides which milestones should be PERT events and also decides their "proper" sequence.
  5) A PERT chart may have multiple pages with many sub-tasks.

- **Terminologies used in PERT:**

1) **A PERT event:** is a point that marks the start or completion of one or more tasks. It consumes no time, and uses no resources. It marks the completion of one or more tasks, and is not "reached" until all of the activities preceding that event have been completed.
2) **A predecessor event:** an event(s) that immediately precedes some other event without any intermediate events.
3) **A successor event:** an event(s) that immediately follows some other event without any intermediate events.
4) **A PERT activity:** is the actual performance of a task. It consumes time, it requires resources (such as labor, materials, space, machinery). A PERT activity cannot be completed until the event preceding it has occurred.
5) **Optimistic time (O):** the minimum possible time required to accomplish a task, assuming nothing goes wrong.
6) **Pessimistic time (P):** the maximum possible time required to accomplish a task, assuming everything goes wrong.
7) **Most likely time (M):** the best estimate of the time required to accomplish a task, assuming everything goes normal.
8) **Expected time (TE):** the best estimate of the time required to accomplish a task, assuming everything goes normal.

   TE = (O + 4M + P) / 6
9) **Critical Path:** the longest possible continuous pathway taken from the initial event to the terminal (ending) event. It determines the total calendar time required for the project to complete successfully. Any delay in this path would definitely delay the project completion.
10) **Critical Activity:** An activity that has total float equal to zero. Activity with zero float (critical activity) does not mean it is on critical path.
11) **Lead time:** the time by which a predecessor event must be completed in order to allow sufficient time for the activities that must complete before a specific PERT event is reached for completion.

12) **Lag time:** the earliest time by which a successor event can follow a specific PERT event.
13) **Slack:** the slack of an event is a measure of the excess time and resources available in achieving this event. Positive slack would indicate ahead of schedule; negative slack would indicate behind schedule; and zero slack would indicate on schedule.
14) **Fast tracking:** performing more critical activities in parallel.
15) **Crashing critical path:** Shortening duration of critical activities.
16) **Float:** is the amount of time that a task in a project network can be delayed without causing a delay.
    a. **Free float:** extra time available for a subsequent task.
    b. **Total float:** cumulative of all the free floats in the project.

- **The most common information shown in a PERT diagram are:**
  1) The activity name
  2) The normal duration time
  3) The early start time (ES)
  4) The early finish time (EF)
  5) The late start time (LS)
  6) The late finish time (LF)
  7) The slack
- **Example:**

In the following example there are seven tasks, labeled A through G. Some tasks can be done concurrently (A & B) while others cannot be done until their predecessor task is complete (C cannot begin until A is complete). Additionally, each task has three time estimates: the optimistic time estimate (a), the most likely or normal time estimate (m), and the pessimistic time estimate (b). The expected time (TE) is computed using the formula

TE=(a + 4m + b) /6.

Activity	Predecessor	Optimistic a	Normal m	Pessimistic b	TE = (a + 4m + b) /6.
A	--	2	4	6	4.00
B	--	3	5	9	5.33
C	A	4	5	7	5.17
D	A	4	6	10	6.33
E	B,C	4	5	7	5.17
F	D	3	4	8	4.50
G	E	3	5	8	5.17

PERT Activity Table

The Next step is to determine the ES and EF. The ES is defined as the maximum EF of all predecessor activities, unless the activity in question is the first activity, for which the ES is zero (0). The EF is the ES plus the task duration (EF = ES + duration).

- ✓ The ES for start is zero since it is the first activity. Since the duration is zero, the EF is also zero. This EF is used as the ES for A and B.
- ✓ The ES for A is zero. The duration (4 work days) is added to the ES to get an EF of four. This EF is used as the ES for C and D.
- ✓ The ES for B is zero. The duration (5.33 work days) is added to the ES to get an EF of 5.33.

- ✓ The ES for C is four. The duration (5.17 work days) is added to the ES to get an EF of 9.17.
- ✓ The ES for D is four. The duration (6.33 work days) is added to the ES to get an EF of 10.33. This EF is used as the ES for F.
- ✓ The ES for E is the greatest EF of its predecessor activities (B and C). Since B has an EF of 5.33 and C has an EF of 9.17, the ES of E is 9.17. The duration (5.17 work days) is added to the ES to get an EF of 14.34. This EF is used as the ES for G.
- ✓ The ES for F is 10.33. The duration (4.5 work days) is added to the ES to get an EF of 14.83.
- ✓ The ES for G is 14.34. The duration (5.17 work days) is added to the ES to get an EF of 19.51.
- ✓ The ES for finish is the greatest EF of its predecessor activities (F and G). Since F has an EF of 14.83 and G has an EF of 19.51, the ES of finish is 19.51. Finish is a milestone (and therefore has a duration of zero), so the EF is also 19.51.

Barring any unforeseen events, the project should take 19.51 work days to complete.

The next step is to determine the late start (LS) and late finish (LF) of each activity. This will eventually show if there are activities that have slack. The LF is defined as the minimum LS of all successor activities, unless the activity is the last activity, for which the LF equals the EF. The LS is the LF minus the task duration (LS = LF - duration).

- ✓ The LF for finish is equal to the EF (19.51 work days) since it is the last activity in the project. Since the duration is zero, the LS is also 19.51 work days. This will be used as the LF for F and G.
- ✓ The LF for G is 19.51 work days. The duration (5.17 work days) is subtracted from the LF to get an LS of 14.34 work days. This will be used as the LF for E.
- ✓ The LF for F is 19.51 work days. The duration (4.5 work days) is subtracted from the LF to get an LS of 15.01 work days. This will be used as the LF for D.

- ✓ The LF for E is 14.34 work days. The duration (5.17 work days) is subtracted from the LF to get an LS of 9.17 work days. This will be used as the LF for B and C.
- ✓ The LF for D is 15.01 work days. The duration (6.33 work days) is subtracted from the LF to get an LS of 8.68 work days.
- ✓ The LF for C is 9.17 work days. The duration (5.17 work days) is subtracted from the LF to get an LS of 4 work days.
- ✓ The LF for B is 9.17 work days. The duration (5.33 work days) is subtracted from the LF to get an LS of 3.84 work days.
- ✓ The LF for A is the minimum LS of its successor activities. Since C has an LS of 4 work days and D has an LS of 8.68 work days, the LF for A is 4 work days. The duration (4 work days) is subtracted from the LF to get an LS of 0 work days.
- ✓ The LF for start is the minimum LS of its successor activities. Since A has an LS of 0 work days and B has an LS of 3.84 work days, the LS is 0 work days.

The next step is to determine the critical path and if any activities have slack. The critical path is the path that takes the longest duration to complete.

To determine the path times, we add the task durations for all available paths. Activities that have slack can be delayed without changing the overall time of the project. Slack is computed in one of two ways,

slack = LF - EF or slack = LS - ES.

Activities that are on the critical path have a slack of zero (0).

- ✓ The duration of path ADF is 14.83 work days.
- ✓ The duration of path ACEG is **19.51** work days.
- ✓ The duration of path BEG is 15.67 work days.

The critical path is ACEG and the critical time is 19.51 work days. It is important to note that there can be more than one critical or that the critical path can change.

The slack for each activity can now be determined.

- ✓ Start and finish are milestones and by definition have no duration, therefore they can have no slack (0 work days).
- ✓ The activities on the critical path by definition have a slack of zero; however, it is always a good idea to check the math anyway when drawing by hand.
  - $LF_A – EF_A = 4 - 4 = 0$
  - $LF_C – EF_C = 9.17 - 9.17 = 0$
  - $LF_E - EF_E = 14.34 - 14.34 = 0$
  - $LF_G - EF_G = 19.51 - 19.51 = 0$
- ✓ Activity B has an LF of 9.17 and an EF of 5.33, so the slack is 3.84 work days.
- ✓ Activity D has an LF of 15.01 and an EF of 10.33, so the slack is 4.68 work days.
- ✓ Activity F has an LF of 19.51 and an EF of 14.83, so the slack is 4.68 work days.

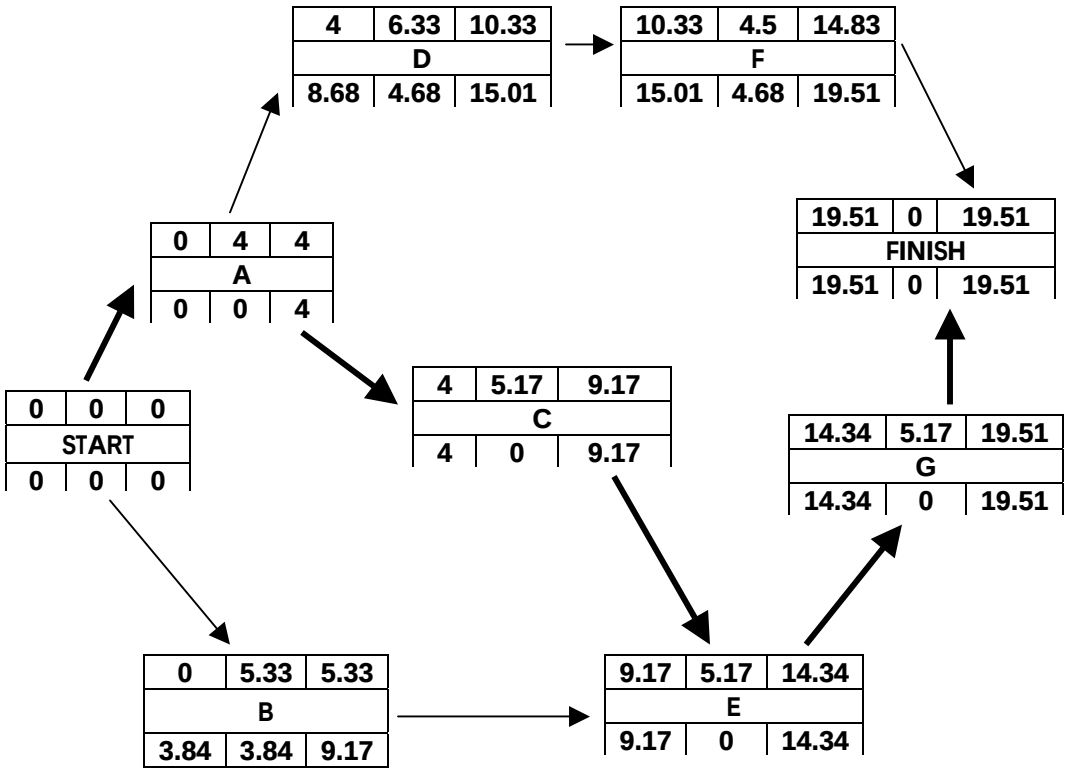
Therefore, activity B can be delayed almost 4 work days without delaying the project. Likewise, activity D or activity F can be delayed 4.68 work days without delaying the project (alternatively, D and F can be delayed 2.34 work days each).

PERT Chart

Legend:

Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish

Chart:



The bold lines represent the Critical Path.

5. Gantt charts:

a. A Gantt chart is a graph of activities against time. It is the most effective way to present a project's plan and its progress.

b. A Gantt chart is a type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the tasks (activity) in a project.

c. The Task list comes from the work breakdown structure of the project.

d. Some Gantt charts also show the dependency relationships between activities.

e. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line.

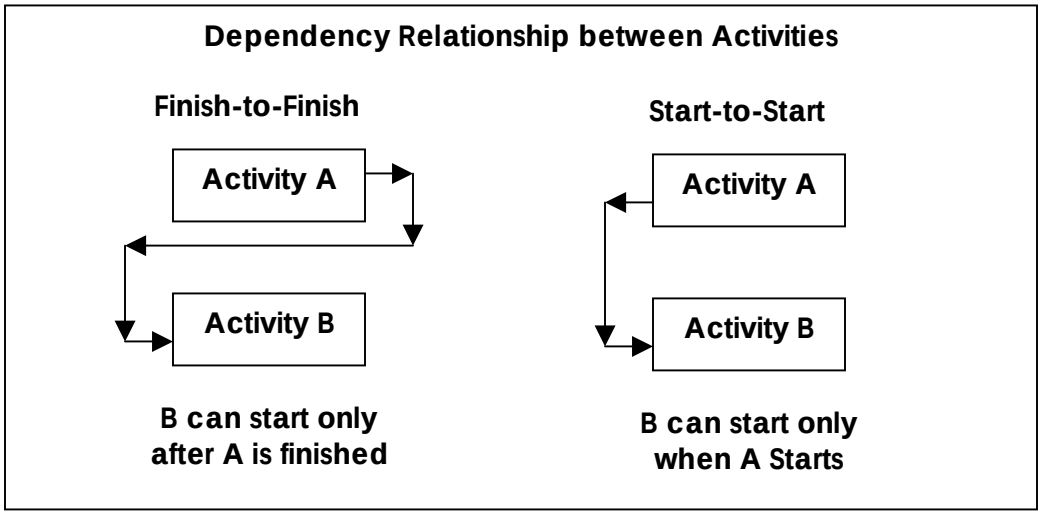
f. A Gantt chart can indicate:

   i. Milestones
   ii. Slack time
   iii. Activities on the critical path

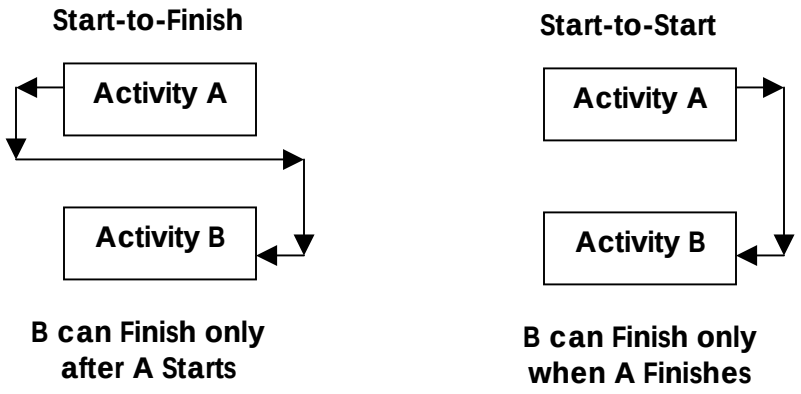
g. Once the project has started, the Gantt chart can also show:

   i. Activities that have started and their actual start dates
   ii. Activities that have finished and their actual completion dates
   iii. Revised milestone and activity dates
   iv. Revised slack time and critical path
   v. Actual performance compared to the original plan

h. Dependency Relationship between activities is represented as follows

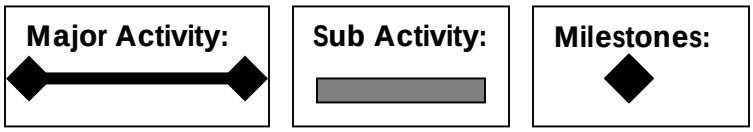


Dependency Relationship between Activities contd...

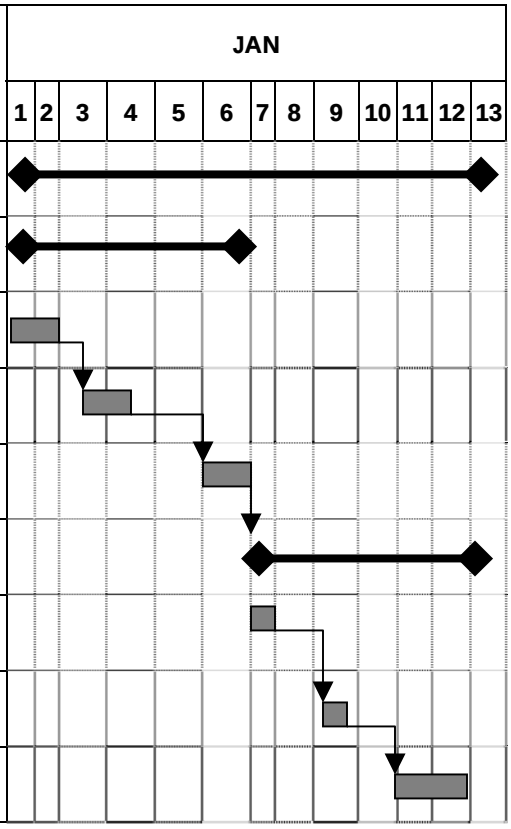


Sample Gantt chart

Legends used:



Id No	Task Name	Duration (days)
1	**Software Selection and Acquisition**	13
2	**Selection**	6
3	Define Requirements	2
4	Identify vendors	2
5	Issue Requisitions	1
6	**Acquisitions**	5
7	Place order	1
8	Install hardware and Software	1
9	Test	2



6. Event Chain Diagrams:

a. Event Chain Diagrams are visualizations that show the relationships between events and tasks and how the events affect each other.

b. Events can cause other events, which will create event chains. These event chains can significantly affect the course of the project.

c. It is focused on identifying and managing events and event chains that affect project schedules.

d. Event chain diagrams are presented on the Gantt chart according to the specification given below:

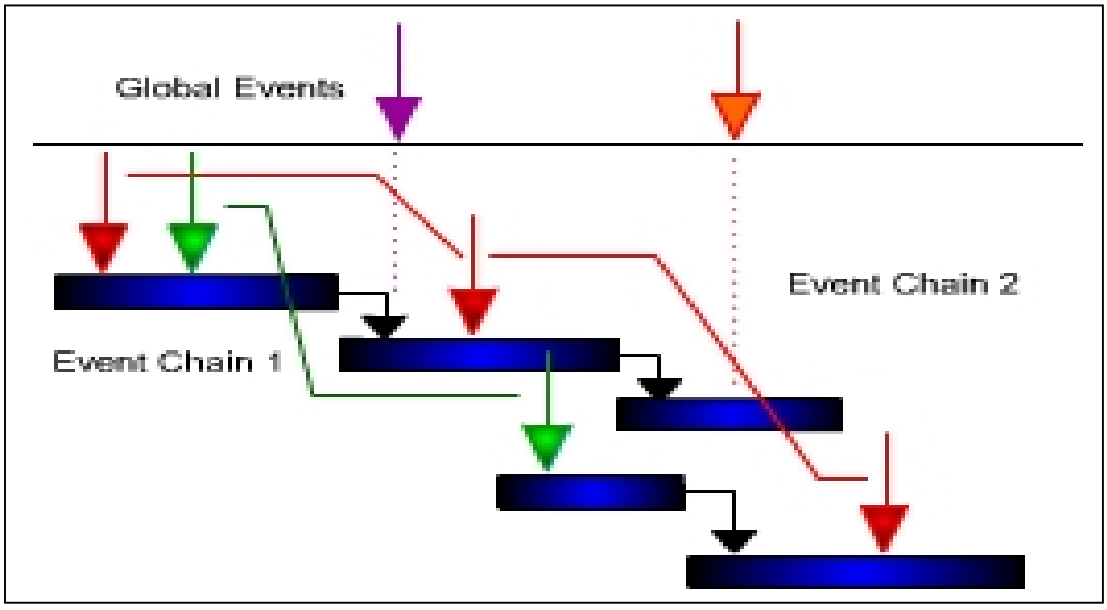
i. All events are shown as arrows. Names and/or IDs of events are shown next to the arrow.

ii. Events with negative impacts (risks) are represented by down arrows or different colour; events with positive impacts (opportunities) are represented by up arrows or different colour.

iii. Individual events are connected by lines representing the event chain.

iv. A sender event with multiple connecting lines to receivers represents multicasting.

v. Events affecting all activities (global events) are shown outside Gantt chart.



7. RACI (Responsible Accountable Consulted Informed) diagram:

a. It is a way to examine a process step, task, activity, effort, decision or inspection to determine who is **Accountable, Responsible, Informed or Consulted.**

b. Sometimes RACI is know as RASCI where the additional S stands for **Support.**

c. **R = Responsible:** The person who performs the action/task.

d. **A = Accountable:** The person who is held accountable that the action/task is completed.

e. **C = Consulted:** The person(s) who is consulted before performing the action/task.

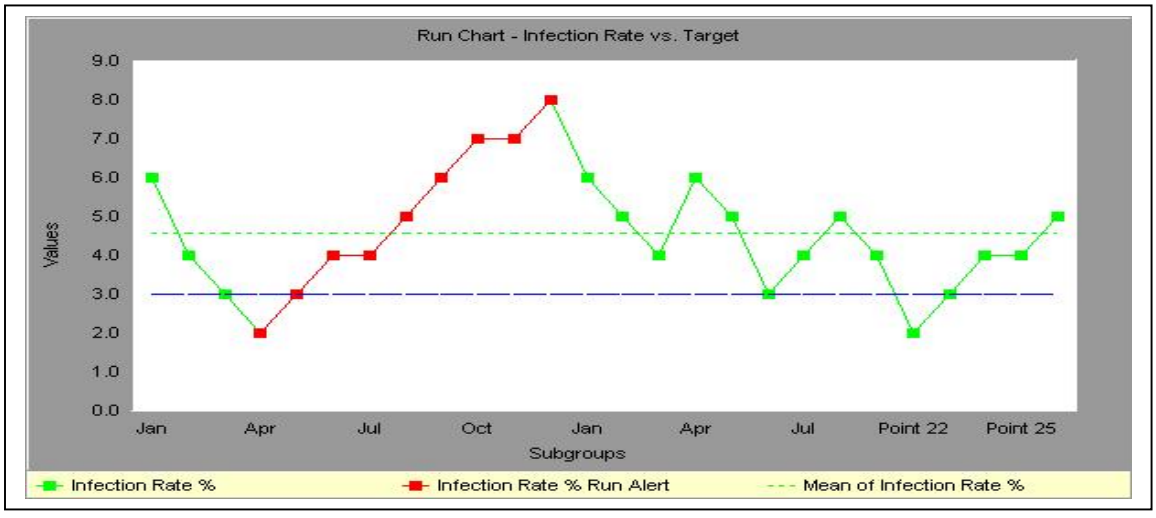
f. **I = Informed:** The person(s) who is informed after performing the action/task.

8. Run charts:

a. Run charts are line graphs, which display process performance over time.

b. The events are shown on the y-axis and are graphed against a time period on the x-axis.

c. Upward and downward trends, cycles, and large aberrations may be spotted and investigated further for gaining insight about the process under review.



9. Project Cycle Optimization (PCO) :

a. It is set of procedures followed to streamline software development projects, assisting and enabling corporate and public sector organizations to deliver their systems on time, within budget and requirements matched.

b. Software specification is comprehensive with accurate contingency plans for changing project factors.

c. The team is motivated, happy to work together and has a clear understanding of their pay and potential incentives.

d. Each team member understands their individual role as well as the roles of colleagues and appreciates their importance.

e. Ordered and documented communication, resulting in team-wide appreciation of changing needs and accurate responses to developments.

f. The production chain is linear (without overlapping with other teams) with transparency.

g. The entire team is specifically trained to be comprehensively skilled and integrated to fulfill the projects goals.

h. Teams are constantly motivated with new thinking, incentives, in project training and a valuing of their role to maintain project enthusiasm.

i. Innovation is recognized in and outside of the process to produce exceptional products with time and cost savings.

10. Participatory Impact Pathways Analysis (PIPA):

a. It is a practical planning, monitoring and evaluation approach.
b. PIPA begins with a participatory workshop where stakeholders present their assumptions about how their project will achieve an impact.
c. Participants construct problem trees, carry out a visioning exercise and draw network maps to help clarify the imagined 'impact pathways'.
d. These ideas are then articulated in two logic models. The **outcomes logic model,** which describes the project's medium term objectives in the form of hypotheses having details about which actors need to change, what are those changes and which strategies are needed to realize these changes. **The impact logic model** describes how after achieving the expected outcomes, the project will impact on people's livelihoods.
e. The Participants derive outcome targets and milestones, which are regularly revisited and revised as part of project monitoring and evaluation.

CHAPTER 02

PROJECT INTEGRATION MANAGEMENT

Project Phases:

I. Initiation,
II. Planning or development,
III. Production or execution,
IV. Monitoring and controlling, and
V. Closing.

1.. Initiation:

1) The initiation stage determines the nature and scope of the development.
2) Study and Analysis of the business needs with measurable goals.
3) Review of the current operations.
4) Conceptual design of the operation of the final product.
5) Equipment and contracting requirements.
6) Financial analysis of the costs and benefits.
7) Stakeholder analysis, including users, and support personnel for the project.
8) Project charter including costs, tasks, deliverables, and schedule.
9) Various tasks included are as follows:

a) Developing Business Case:

1. Performing Problem analysis.
2. Developing solution strategies.
3. Performing cost and benefit analysis of the solutions.
4. Performing Risk analysis.
5. Adopting the optimum solution.

b) Performing Feasibility Study:

1. Technical feasibility: determining the current hardware and software resources including the technical skills of the team members for implementing the project.
2. Operational feasibility: determining to which extent the product would operate when delivered to the customer.
3. Economic feasibility: determining the costs of developing and maintaining the product compared with the profits the product would generate for the company.
4. Managerial feasibility: determining the ease with which the product could be marketed and sold in the market. Also to determine whether the production can be managed successfully.

c) Establish the Project Charter:

1. Vision
2. Objectives.
3. Scope.
4. Implementation.
5. Deliverables.
6. Risks, Issues and Assumptions.

d) Appointing members of Project Team:

1. List of persons involved in the project.
2. Roles and Responsibilities of every one involved in the project.

e) Reviewing:

1. To determine that all the above activities are preformed properly.
2. Create documents to support the other phases.

2.. Planning and design:

1) After the initiation stage, the system is designed.
2) Occasionally, a small prototype of the final product is built and tested.
3) Testing is generally performed by a combination of testers and end users, and can occur after the prototype is built or concurrently.
4) Controls should be in place that ensure that the final product will meet the specifications of the project charter.
5) Following Plan documents are created:
   a. **Project Plan:**
      i. Statement of Work.
      ii. Resource List.
      iii. Work Breakdown Structure.
      iv. Cost estimates.
      v. Risk List.
   b. **Resource Plan:**
      i. People.
      ii. Equipment.
      iii. Materials.
   c. **Financial Plan:**
      i. Cost estimation:
         1. People
         2. Equipment.
         3. Materials.
         4. Maintenance.
      ii. Budgeting.
   d. **Quality Plan:**
      i. Software Quality Assurance.
      ii. SQA group.
      iii. SQA schedule.
      iv. SQA milestones and reviews.

   e. **Risk Plan:**

i. Risk list.
ii. Risk impact.
iii. Risk mitigation plan.

f. Acceptance Plan:

i. Operational and Quality acceptance criteria for the product.
ii. Scheduled reviews by customer to accept the product development.

g. Communication Plan:

i. Schedule of communication events.
ii. Procedure of communication (formal & informal).
iii. Information type during communication.

6) The results of the design stage should include a product design that:
   a. Satisfies the project sponsor, end user, and business requirements.
   b. Functions as it was intended.
   c. Can be produced within quality standards.
   d. Can be produced within time and budget constraints.

3.. Executing:

1) Executing consists of the processes used to complete the work defined in the project management plan to accomplish the project's requirements.
2) Execution process involves coordinating people and resources.
3) Integrating and performing the activities of the project in accordance with the project management plan.
4) The intended deliverables are produced as defined in the project management plan.

4.. Monitoring and Controlling:

1) It consists of those processes performed to observe project execution so that potential problems can be identified in a timely manner and corrective action can be taken.
2) The project performance is observed and measured regularly to identify variances from the project management plan, It includes:

a. Measuring the ongoing project activities (where we are).
b. Monitoring the project variables (cost, effort, ...) against the project management plan.
c. Monitoring the project performance baseline (where we should be).
d. Identify corrective actions to properly address issues and risks (How can we get on track again);

5.. Project Maintenance:

1) Continuing support of end users
2) Correction of errors
3) Updates of the software over time

6.. Closing:

1) It includes the formal acceptance of the project by the customer.
2) Archiving of the project documents.
3) Documenting the lessons learned.
4) Finalize all activities across all of the process groups to formally close the project.
5) Closing each contract applicable to the project.

Scope:

1. A scope definition specifies what the project will do as well as what it will not.
2. Scope statement must specify what activities are not in scope.
3. Scope change cannot be recognized until the scope baseline is established.
4. The Scope statement includes all the work that the project will be required to carry out.
5. There are two types of scopes **(1) the scope of the product and (2) the scope of the project.** The scope of the product defines the work that the

project will carry out, the scope of the project defines how that work will be executed.

Scope Change Management :

1. When we identify a change of scope then we submit it to the technical people for estimates, and calculate the effect of the change on the project schedule and costs.
2. The changes and impacts are documented in a change request form.
3. The change request form is submitted to the client for approval within the date specified in the form i.e "Date Required"
4. After this, the change becomes a matter of client approval. It may require negotiations or modifications of the change, but ultimately, the client must either approve or reject the change.
5. If the change is approved then the project plan is revised by including the change.

A sample Change Request Form

Change Request Form

Project: _____ **Date:** _____

Manager: _____

Requested by: _____

Description of the Change:

Justification for the Change:

Impacts on the Project:

Impacts on the Schedule:

Impacts on the Costs:

Resolution: _____ Date Required: _____

Approved/Rejected: _____ Date: _____

Signed: _____

Time:

1. **The Estimate at Completion (EAC):** It is the amount of work done to date plus the team member's estimate of the amount of work remaining.
2. **The Time Sheets:** are used that shows the time spent on each activity. Team members record their time against activity numbers (which ideally are the WBS numbers).
3. **Progress Reporting:** is a formal mechanism in which each team member prepares a weekly progress report. This information is also used for finding the current status and the amount of pending work.
4. **Milestones:** are key points in the project at which certain deliverables will be ready or a set of activities completed. Milestones are easy to track because they occur on specific dates and mark the delivery of specific products. If the products are ready on or before that date, then the project is on schedule. If not, it is late.
5. **Gantt Charts:** are used for time scheduling of the activities in the project.
6. **PERT:** is a model for project management designed to analyze and represent the tasks involved in completing a given project, especially the time needed to complete each task, and identifying the minimum time needed to complete the total project.

Cost:

1. **Systems Development Costs:**
    a. Labor costs to define system requirements
    b. Labor costs to design the system
    c. Labor costs to code, unit test, integrate, and systems testing
    d. Labor costs for documentation including training materials
    e. Labor costs for project management including functions such as configuration management, quality control, and support
2. **Hardware Costs:**
    a. Capital costs to purchase hardware
    b. Labor costs to install hardware
    c. Labor costs to maintain hardware
3. **Software Costs:**

a. Purchase costs for operating systems software.
b. Purchase costs for infrastructure software such as communications, performance enhancement, or performance statistics
c. Purchase costs for applications support software such as database management, graphical user interface.
d. Labor costs to install and configure software.

4. **Project Execution Costs:**
   a. Project costs for travel and living
   b. Project costs for external consulting
   c. Project costs for training
   d. Project costs for supplies and materials
5. **Project Client Costs:**
   a. Operating costs for staff attendance at training programs for clients
   b. Operating costs for client involvement in the project
6. **Implementation Costs:**
   a. Operating costs for additional staff effort during implementation
   b. Operating costs for travel and living during implementation
7. **System Operations Costs:**
   a. Hardware and software maintenance contracts.
   b. Projected costs for hardware and software upgrades.

Quality:

It's a measurable quantity which can be compared with known standards for any product or process. It specifies the degree to which a product or process maintains the known standards. Quality can be classified in to two groups – a) Quality of design and b) Quality of conformance.

a) Quality of design :- It refers to the standards maintained during design processes which include the grade of - i) Materials, ii) Tolerances and iii) Performance specifications.

b) Quality of conformance :- It refers to the degree to which the design specifications are followed during manufacturing.

Quality Control

a) It's a series of inspections, reviews and tests used throughout the development cycle to ensure that each work product meets the requirements placed up on it.
b) It includes a feedback loop to the process that created the work product. The feedback loop helps tune up the process during the creation of the product.
c) It should be a part of the manufacturing process.

Quality Assurance

It's the auditing and reporting functions of the management. If the data provided through quality assurance identify problems then it's the management's responsibility to solve the problems and apply the necessary resources to resolve the quality issues. The goal of quality assurance is to provide management to provide information and data related to product quality so that they can evaluate whether or not the manufacturing process is maintaining the product quality.

Software Quality Assurance (SQA) Activities

The activities performed in software quality assurance involves two groups –

a) **The software engineers doing the technical work :-** Their work includes – 1) Applying technical methods and measures for ensuring high quality, 2) Conducting formal reviews and 3) Performing well planned software testing. and
b) **The SQA group :-** That has the responsibility for – Quality assurance planning, to assist the software engineering team in achieving high standards of production, record keeping, analysis and reporting.

Human Resource:

1. The planner begins by evaluating scope and selecting the skills required to complete the development tasks. Both Organizational positions (e.g., manager, senior software engineer) and Specialty positions (e.g., telecommunications, database, client/server) are identified.
2. The number of people required for a software project can be determined only after an estimate of development effort (e.g., person-months) is made.

Communications:

1. Characteristics of modern software include scale, uncertainty, and interoperability. To deal with them effectively, a software engineering team must establish effective methods for coordinating the people who do the work. To accomplish this, mechanisms for formal and informal communication among team members and between multiple teams must be established.
2. **Formal communication:** is accomplished through "writing, structured meetings, and other relatively non-interactive and impersonal communication channels".
3. **Informal communication:** is more personal. Members of a software team share ideas on an ad hoc basis, ask for help as problems arise, and interact with one another on a daily basis.
4. **Five points of effective Communication:**
    1) **Transmitted:** Information is sent to the appropriate person in a format that the receiver can accept it.
    2) **Received:** An acknowledgement should be received from the receiver, that the receiver has read the information.
    3) **Understood:** The information should be easy enough so that the intended receiver can understand its meaning and can analyze what to do next.

4) **Agreed:** The receiver should be convinced enough to agree up on the information sent to him and becomes ready to participate further in the communication process.
5) **Converted to useful action:** The communication succeeds only when the intention of the communication is fulfilled i.e the intended result in the form of actions/tasks to be done by the receiver is accomplished.

The project manager prepares a communications plan. This document lists each of the communications types intend to be used during the project, then, for each type, it describes the purpose of the communication, specifies the target audience, identifies who is responsible for preparing and issuing the communication, and gives the frequency (how often) of communications.

Risk:

General categorization of risk is as follows :

1) **Known risks :-** These can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is been developed. Other reliable factors can also be considered, like, unrealistic schedule, poor development environment etc.
2) **Predictable risks :-** These can be predicted form past project experiences (e.g., staff turnover, poor communication with customers etc.) and a plan can be designed long before the risk becomes reality.
3) **Unpredictable risks :-** These very difficult to identify and often become reality. For these kinds of risks, reactive risk management strategy needs to be taken.

PMI Fundamentals:

The Nine areas of Project Management outlined by Project Management Institute (PMI) are:

1. **Project Scope Management:**
    a. Controlling the planning, execution, and content of the project.
    b. We need to pay special attention to both project and product scope so that the software we end up with is what we intended to make in the first place.
2. **Project Time Management:**
    a. Managing everything that affects the project's schedule.
    b. Building the product at right time for maximum customer acceptance.
3. **Project Cost Management:**
    a. Cost estimating, budgeting, and controlling the costs involved in building and maintaining the project.
4. **Project Quality Management:**
    a. Ensuring that the product we are producing is a quality product and that it meets customer expectations.
5. **Project Human Resources Management:**
    a. Hiring and managing the competent people to work for the project.
6. **Project Communications Management:**
    a. Making sure that the people who need information get it — when they need it.
7. **Project Risk Management:**
    a. Anticipating and handling risks, as well as taking advantage of opportunities that can help a project.
8. **Project Procurement Management:**
    a. Creating vendor contracts and purchasing goods and services.
9. **Project Integration Management:**
    a. Ensuring perfect coordination between all the knowledge areas.

Project Dimensions:

Project Dimensions include People, Product, Process, and Technology

1. **People:**

People management includes:

a. Recruiting.
b. Selection.
c. Performance management.
d. Training.
e. Compensation.
f. Career development.
g. Work design.
h. Team-culture.
i. Development.

2. Product:

a. Before a project can be planned, product objectives and scope should be established
b. Alternative solutions should be considered
c. Technical and management constraints should be identified.
d. The software developer and customer must meet to define product objectives and scope.
e. Scope identifies the primary data, functions and behaviors that characterize the product.
f. Once the product objectives and scope are understood, alternative solutions are considered.

3. Process:

a. A software process provides the framework from which a comprehensive plan for software development can be established.
b. A number of different activities like tasks, milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.
c. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement are carried out during the process follow-up.

4. Technology:

a. Analysis of presently available technology to implement the project.
b. Assessing the Risks associated with the implementation of the technology, especially while implementing new technology.
c. Adopting technology that would reduce the cost of the finally built product.
d. Imparting training about using the technology for team members who are new to the technology.
e. Upgrading the implemented technology from time to time.

CHAPTER 03

PLANNING

Software Project Planning:

Following are the various project planning activities

1) Defining the problem:

a)Develop a definitive statement of the problem to be solved. Including a description of the present situation , problem constraints and a statement of the goals to be achieved. The problem statement should be formulated in the customer's terminology.

b)Justify a computerized solution strategy for the problem.

c)Identify the various functions provided by the hardware subsystem, software subsystem and people subsystem. The constraints thereof also should be identified.

d)Determining system level goals and requirements for the development process and the work products.

e)Establish high-level acceptance criteria for the system.

2) Developing a solution strategy:

a)Create multiple solution strategies, with out considering the constraints.

b)Conduct a feasibility study for each strategy.

c)Recommend a solution strategy, indicating why other strategies were rejected.

d)Develop a list of the characteristics of the product priority wise.

3) Planning the development process:

a) Define a life-cycle model and an organizational structure for the project.

b) Planning of -

I) Configuration management activities ,

II) Quality assurance activities and

III) Validation activities.

c) Determine phase-dependent factors –

I) Tools,

II) Techniques ,

III) Notations.

d) Establish preliminary cost estimates for the system development.

e) Establish a preliminary development schedule.

f) Establish preliminary staffing estimates.

g) Preliminary estimation of the required computing resources and maintenance of those systems.

h) Preparation of glossary of terms.

i) Identification of information sources for reference during the project development.

40-20-40 Rule:

1. Each of the software project estimation techniques usually lead to estimates of work units (e.g., person-months) required to complete software development.
2. A recommended distribution of effort across the definition and development phases is often referred to as the 40–20–40 rule.
3. 40 percent of all effort is allocated to front-end analysis and design.
4. 20 percent of all effort is applied to Coding.
5. 40 percent of all effort is applied to back-end Testing.
6. This effort distribution should be used as a guideline only. The characteristics of each project must dictate the distribution of effort.

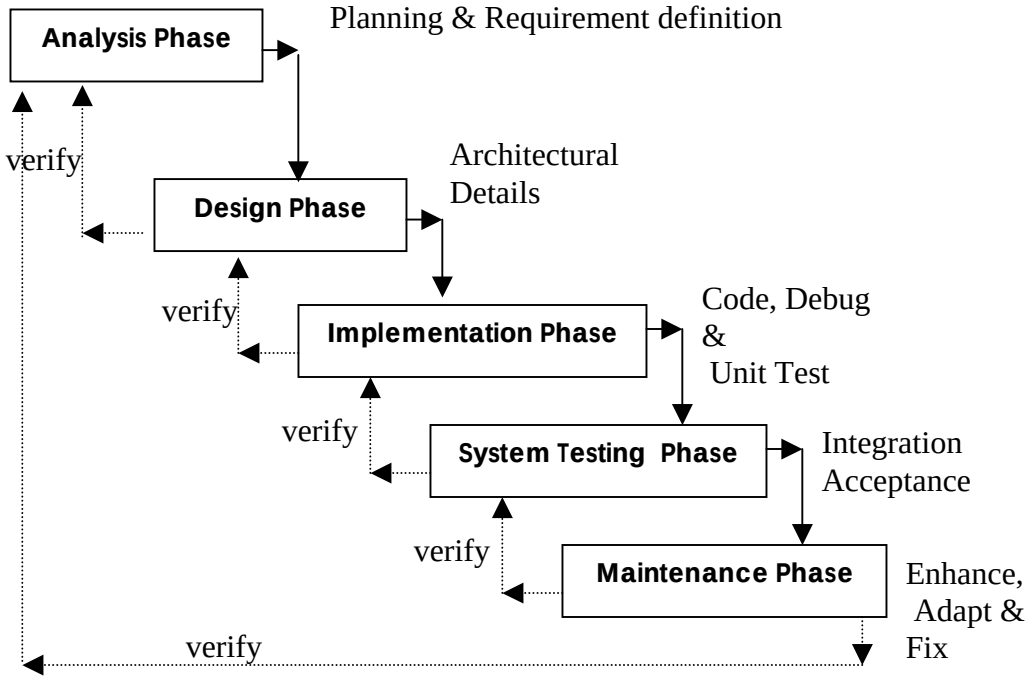
7. Work expended on project planning rarely accounts for more than 2–3 percent of effort.
8. Requirements analysis may comprise 10–25 percent of project effort. Effort expended on analysis or prototyping should increase in direct proportion with project size and complexity.
9. A range of 20 to 25 percent of effort is normally applied to software design. Time expended for design review and subsequent iteration must also be considered.
10. A range of 15–20 percent of overall effort can be achieved during Coding.
11. Testing and subsequent debugging can account for 30–40 percent of software development effort.
12. Today, more than 40 percent of all project effort is often recommended for analysis and design tasks for large software development projects. Hence, the name 40–20–40 rule no longer applies in a strict sense.

Software Development Models or Software Life Cycle Models

There are various types of models that have been engineered for creating various types of software products. Every software process model has some specialties that make it appropriate for the development of certain category of software products. Some of the basic models are :-

1) Waterfall model or Linear Sequential model.
2) Prototyping model.
3) Rapid Application Development (RAD) model.
4) Incremental model.
5) Spiral model.
6) Component Assembly model.
7) WINWIN Spiral Model.

2) Waterfall Model (Linear Sequential Model) :-



Waterfall Model

Analysis Phase :- It consists of Planning and Requirements definition activities. The end products of planning are - a) System Definition and b) Project Plan.

a) System Definition :- is typically expressed in English. It incorporates charts, figures, graphs, tables and equations of various kinds. The notations used here are highly dependent up on the problem area.

a) Project Plan :- contains the life-cycle model to be used, the preliminary development schedule, preliminary development schedule, preliminary cost estimates and resource estimates, tools & techniques to be used and standard practices to be followed.

Design Phase :- This phase is concerned with - a) identifying software components like functions, data streams and data stores, b) specifying software structure, c) maintaining a record of design decisions and providing blue prints for the implementation phase.

There are three basic categories of design -

a. Data design: It involves :-
   - ✓ Identifying data stores.
   - ✓ Identifying the type of data to be stored in the data store.
   - ✓ Identifying data grouping.
   - ✓ Identifying relationships between data.

b. Architectural design: It involves :-
   - ✓ Identifying the software components.
   - ✓ Decoupling and Decomposing the software components into modules and conceptual data structures.
   - ✓ Specifying the interconnections between the various component.

c. Detailed design:
   1. Its concerned with details of the implementation procedures to process the algorithms, data structures and interaction between the modules & data structures.
   2. The various activities that this phase includes are –

      I) adaptation of existing code,

      II) modification of existing algorithms,

      III) design of data representation and

      IV) packaging of the software product. This process is highly influenced by the programming language under which the implementation would be carried out. This stage is different from implementation.

Implementation Phase :- It involves the translation of the design specifications into source cod. It also involves activities like debugging, documentation and unit testing of the source code. In this stage various styles of programming can be followed like built-in and user-defined data types, secure type checking, flexible scope rules, exception handling, concurrency constructs etc.

System Testing Phase :- it involves three kinds of activities –

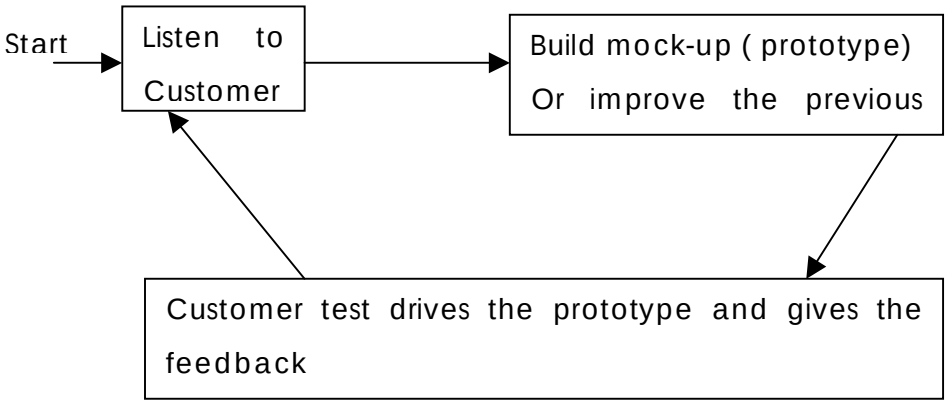
1) Unit testing,
2) Integration testing and
3) Acceptance testing.

Maintenance Phase :- In this phase the activities include a) enhancement of capabilities, b) adaptation of software to new processing environments and c) correction of software bugs.

Best Suited Projects :-

1. Where requirements are completely known beforehand and are baselined before starting the project.
2. For converting any existing manual system to an automated system.

3) Prototyping Model



Prototyping Model

Listen to Customer :- This is the starting step, where the developer and customer together a) Define the overall objectives for the software, b) Identify the known requirements and c) The analyst then outline those factors

and requirements that are not visible normally but are mandatory from development point of view.

Build prototype :- After the identification of the problem a quick design is done which will cause the software show the output that the customer wants. This quick design leads to the development of a prototype (a temporary working model).

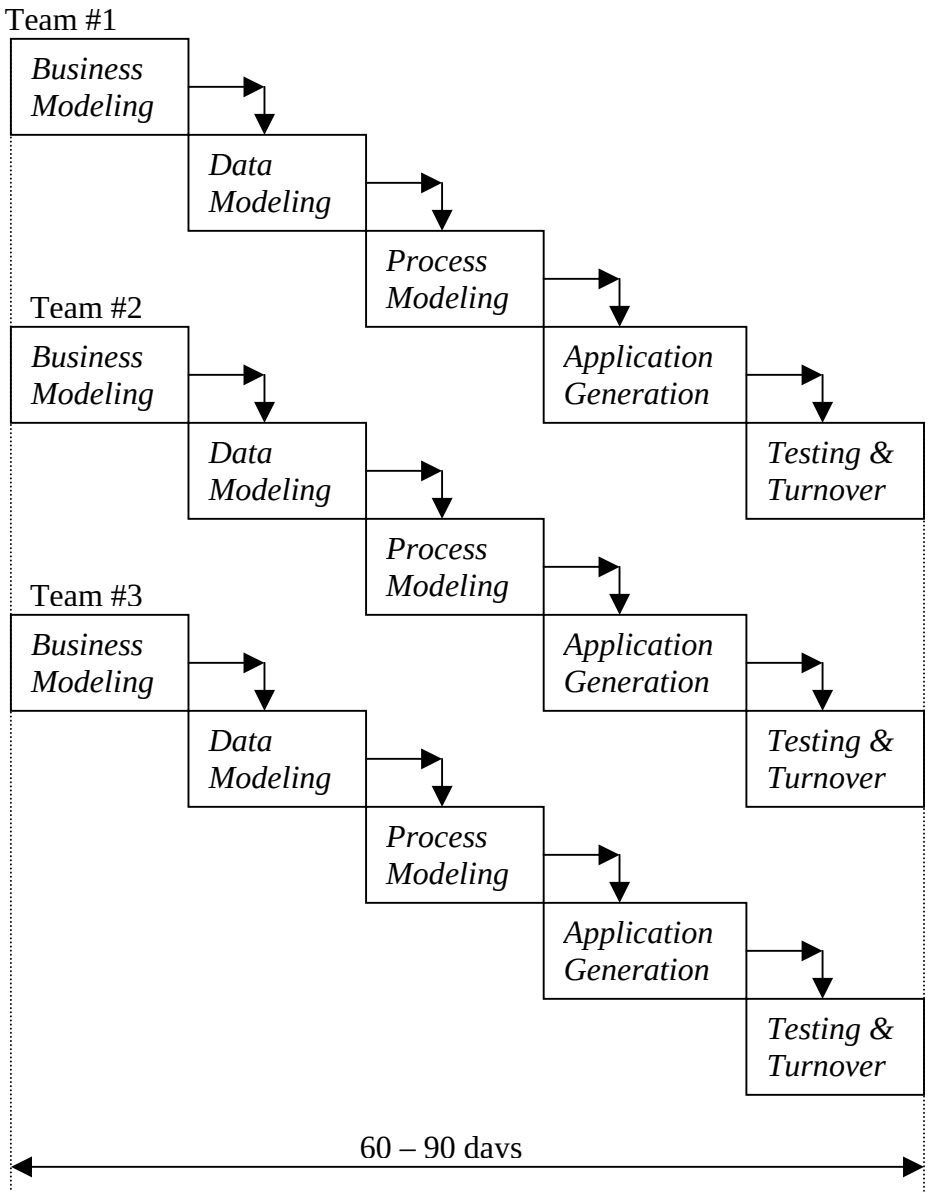
Customer test drives the prototype :- The customer runs and checks the prototype for its perfection. The prototype is evaluated by the customer and further improvements are made to the prototype unless the customer is satisfied.

All the stages are repeated until the customer gets satisfied. When the final prototype is fully accepted by the customer then final development processes like proper coding for attaining maintainability, proper documentation, testing for robustness etc. are carried out. And finally the software is delivered to the customer.

Best Suited Projects:-

1. When a new technology is to be adopted whose outcome is not certain.
2. When requirements are not completely known before starting the project.
3. When the project is new to the team members.
4. When the requirements are expected to change in future.
5. When quick outcomes are required to show the development in the project.

4) RAD (Rapid Application Development) Model



RAD Model

Business Modeling :- This phase involves identification of various information flow between various business functions that act as the driving force in the development process. It's a kind of problem identification task.

Data Modeling :- The relationship between various data elements are identified. Its like the planning phase where the data which drives the business functions are identified and planned out to establish synchronization so that the designing could be done.

Process Modeling :- Its like the design phase. Here the properly modeled data are put together such that actual coding to implement the functionalities for data manipulation could be done.

Application generation :- Its like the coding and implementation phase. Here the planned out processes are actually coded and executed. In this phase its taken in to consideration that the code should be made is such a manner that they become reusable.

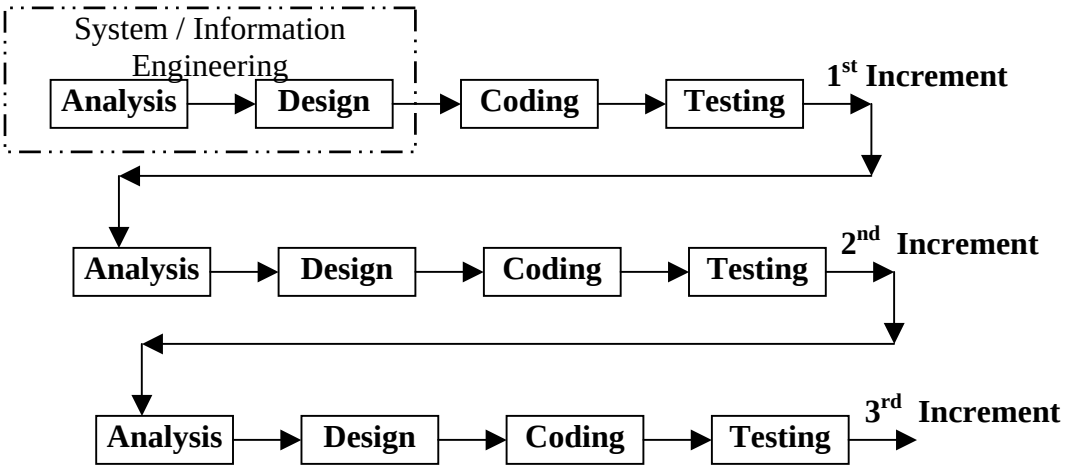
Testing and turnover :- Here the final code is repeatedly tested for reliability, robustness and maintainability. If some short comings are encountered then again the initial stages are repeated for identification and eradication of errors.

In this type of development process a problem is initially analyzed in details such that the problem could be divided or decomposed into manageable smaller problems. The analysis is done in such a way that the decomposed problem could be solved parallel and the solutions could be integrated to solve the entire problem.

Best suited Projects:-

1. When software needs to be developed in a record time.
2. When creating Large Information Systems like ERP or MIS software.
3. When the problem can be disintegrated in to parallely developable modules and finally integrated at the end.
4. When sufficient human resources is available.

5) Incremental Model



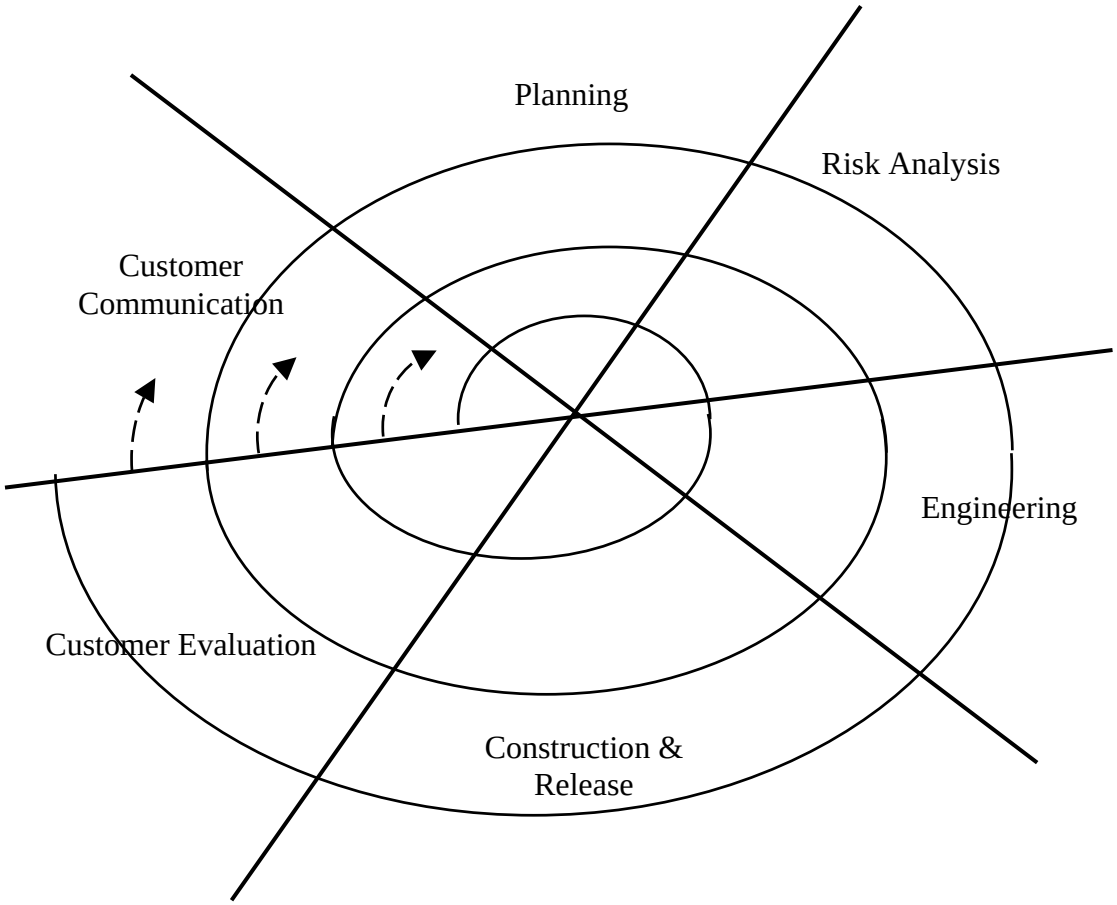
Incremental Model

1) The incremental model combines the elements of linear sequential model with the iterative property of prototyping.
2) The first increment is often a core product where only basic requirements are satisfied and supplementary features are added later in successive increments.
3) Prior to every increment the presently developed prototype or model is tested by the customer. If the customer is not satisfied with the present prototype then again the entire phases of development are repeated unless the prototype is fully accepted by the customer. If the prototype is fully accepted by the customer then the development proceeds towards the development of next increment and again the same process of customer evaluating the prototype is repeated.
4) With every increment additional features & functionalities are provided along with more refinements.
5) The final increment results in a fully furnished s/w product which fully accepted by the customer.

Best suited Projects:-

1. When enough staffing is not available immediately.
2. When the modules are flexible enough to incorporate changes to a very large extent.

6) Spiral Model



Spiral Model

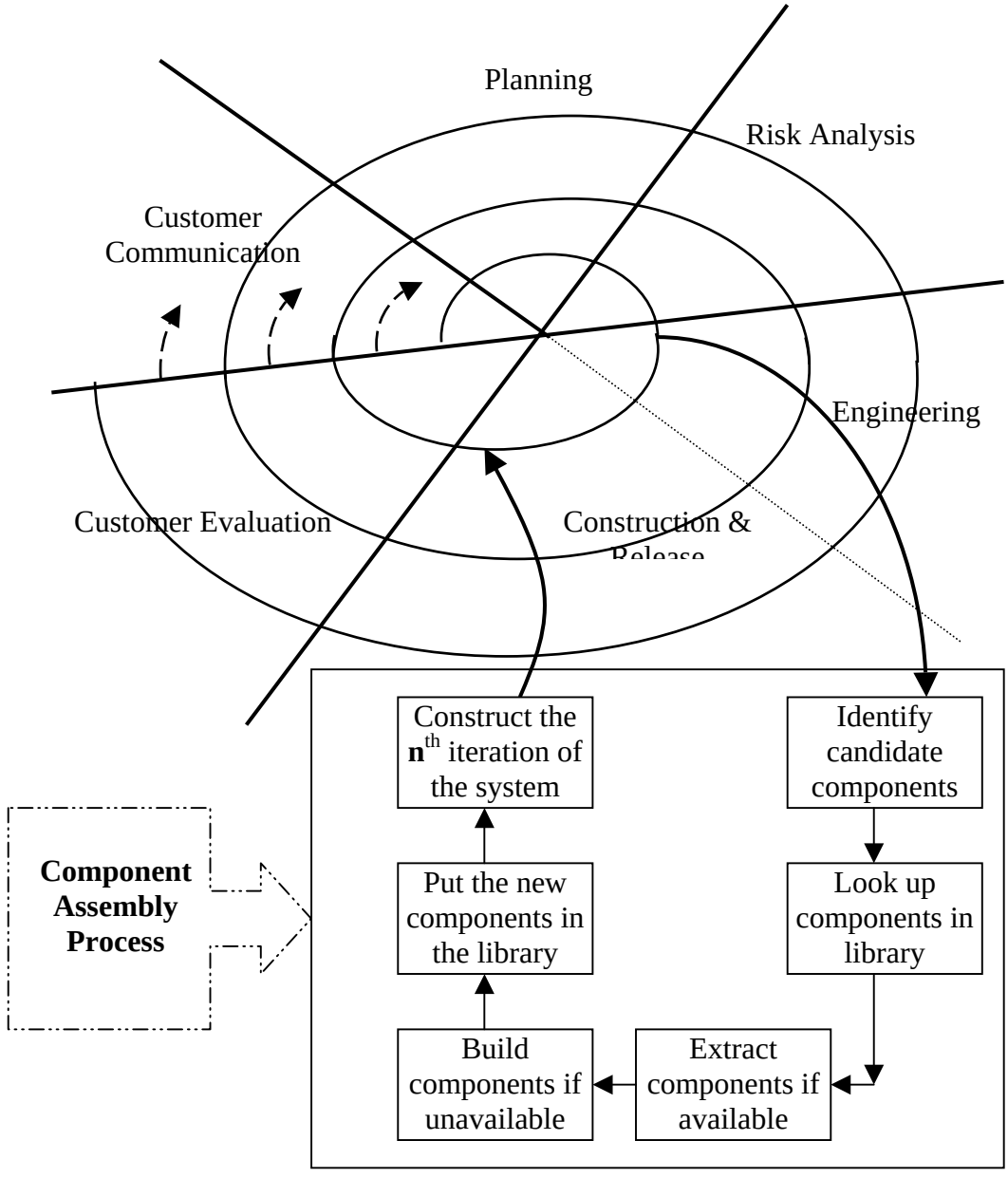
1) The spiral model is developed in a series of incremental releases. After each iteration the s/w product gets more refined.
2) The entire model is divided in to number of framework activities also called task regions. Following are the basic 6 regions :-
    i) Customer communication :- An effective communication is established between the developer and the customer in order to understand and refine the problem identification tasks.
    ii) Planning :- Planning for product development occurs in which time schedules, resource planning and other project related factors are included.
    iii) Risk analysis :- The assessment of technical and managerial risks are done.
    iv) Engineering :- The technical designs are prepared so that construction process could be started.
    v) Construction & release :- The tasks included here are coding, testing, installation and providing technical support (eg. Documentation & training).
    vi) Customer evaluation :- Here the exercises for obtaining feedback from the customer are carried out and if required further enhancements are done to the s/w repeating again the previous phases.

3) The s/w engineering team moves around the spiral in a clock wise direction starting from the core. The first circuit around the spiral generally results in development of product specifications. Subsequent passes through the spiral generally results in the preparation of a prototype and then progress is made is developing more sophisticated versions of the s/w depending on the feedback from the customer.

Best Suited Projects:-

1. When a large scale system is to be created.
2. When the risk factor is to be controlled.

7) Component Assembly (Object Oriented) Model



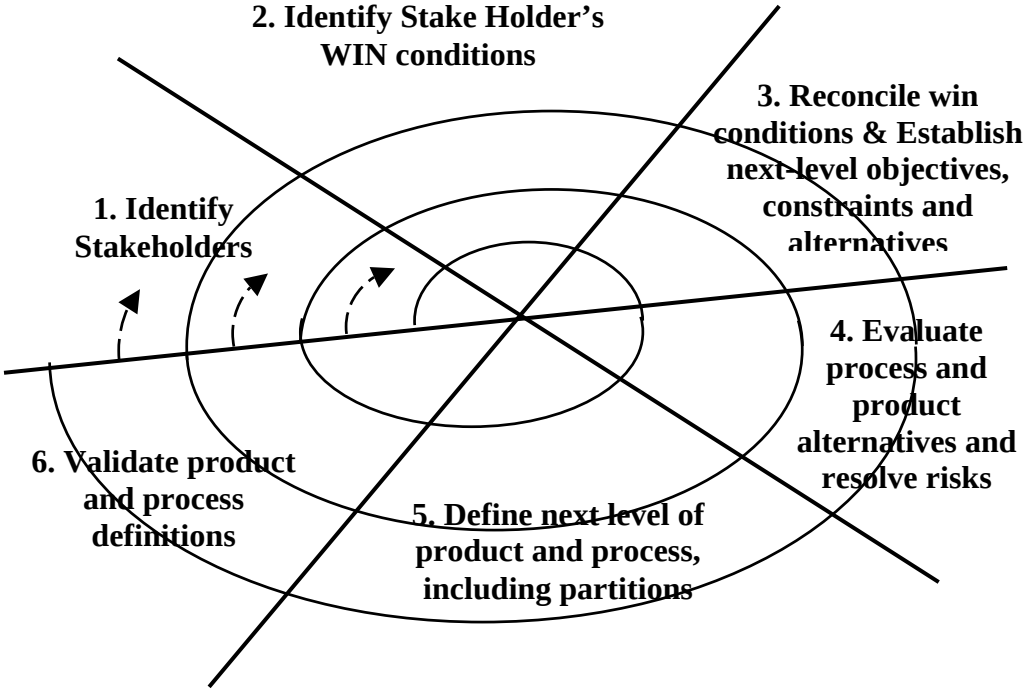
Component Assembly Model

1) This kind of model is used for object based (oriented) development process.
2) Here the emphasis is on the creation of reusable classes that encapsulate both data and functions, where the functions can to manipulate the data.
3) This model incorporates many of the characteristics of spiral model and iterative approach towards software development.
4) During the Engineering phase and Construction & release phase , the data is examined and accordingly algorithms applied for data manipulation is also decided. Corresponding data and algorithms are encapsulated into a Class.
5) Classes created in past are stored in a class library. Once the required classes are identified, the class libraries are searched, if they are found then the are reused. In case where the required classes are not found in the library then they are engineered using object oriented methods.
6) When the process of recognition and usage of classes is done then the development process returns back to the spiral path and subsequently if required reenter in the component assembly process during successive iterations, unless fully acceptable product is made.

Best Suited Projects:-

1. Where reusable components can be implemented.
2. Where the development is to be done fast and high accuracy and reliability is required in the software.

8) WINWIN Spiral Model



WINWIN Spiral Model

1. Here the customer and the developer enter into a process of negotiation, where the customer wants to reduce cost and time required for development then the developer would suggest to balance functionality, performance, or system characteristics against cost and time.
2. The best negotiations results in a "win-win" solution where the customer wins by getting a product that satisfies the majority of his needs and the developer wins by working in such a way that realistic budgets and deadlines are achieved.
3. This model defines a set of negotiation activities at the beginning of each pass around the spiral.
4. Successful completion of the negotiation activities achieves a win-win result for both the customer and developer.

Best Suited Projects:-

1. When a large scale costly system is to be created.
2. When the customer can change his requirements at any time.
3. When there are budget and time constraints from customer as well as developers sides.

Dynamic Systems Development Method (DSDM):

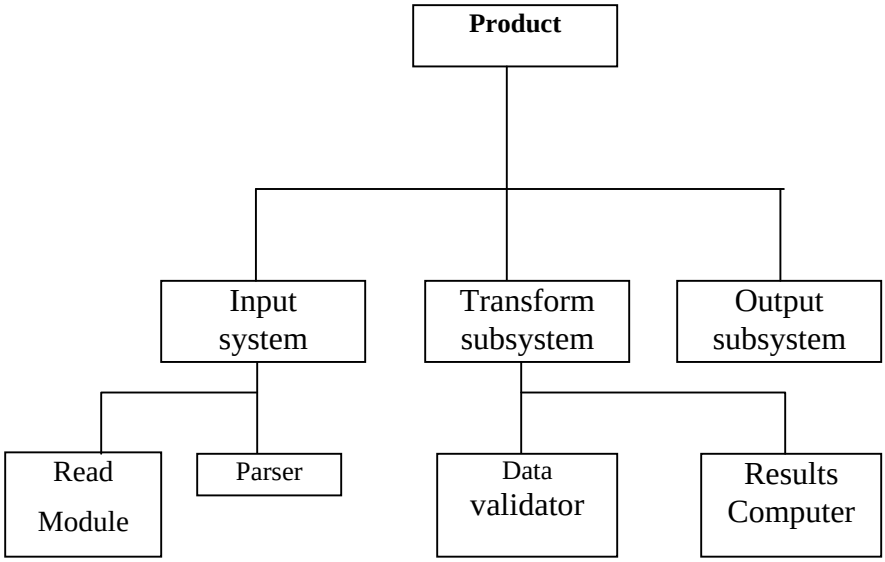
1. It is a software development methodology originally based upon the Rapid Application Development methodology.
2. DSDM is an iterative and incremental approach that emphasizes continuous user involvement.
3. Its goal is to deliver software systems on time and on budget while adjusting for changing requirements along the development process.
4. DSDM consists of following Activities and Sub-Activities:

Activity	Sub activity
Study	Feasibility Study
	Business Study
Functional Model Iteration	Identify functional prototype
	Agree schedule
	Create functional prototype
	Review functional prototype
Design and Build Iteration	Identify design prototype
	Agree schedule
	Create design prototype
	Review design prototype
Implementation	User approval and guidelines
	Train users
	Implement
	Review business

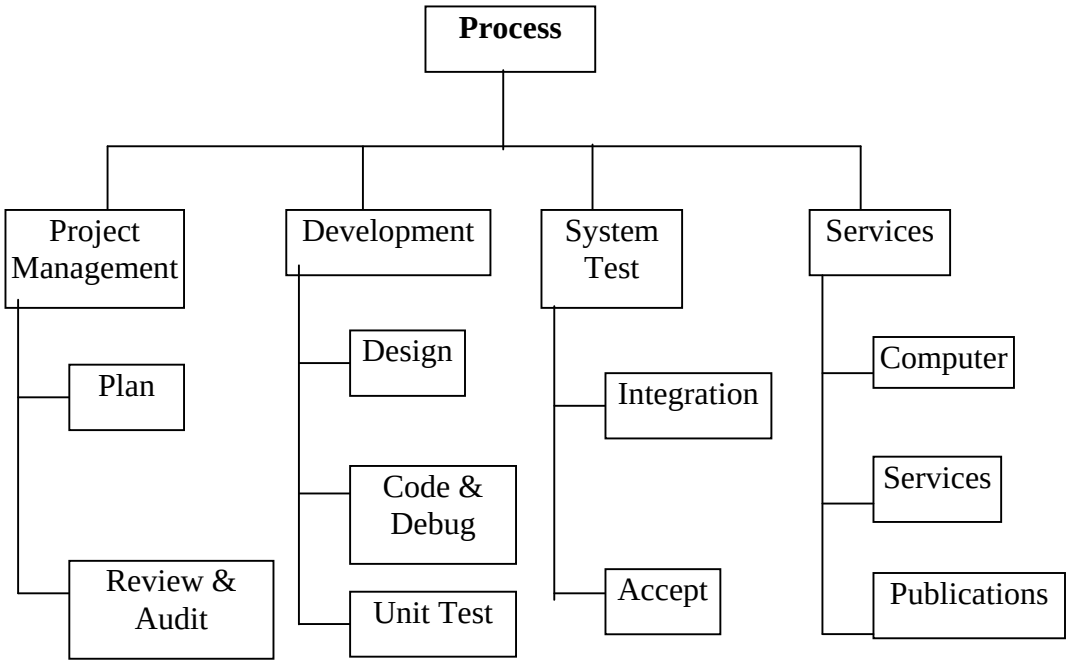
5. User involvement is the main key in running an efficient and effective project, where both users and developers share a workplace, so that the decisions can be made accurately.
6. Development is iterative and incremental and driven by users' feedback to converge on an effective business solution.
7. All changes during the development are reversible.
8. The high level scope and requirements should be base-lined before the project starts.
9. Testing is carried out throughout the project life-cycle. (See Test-driven development for comparison).

Work Breakdown Structures (WBS) (Software Product and Process Metric):

1) The WBS method is a bottom-up estimation tool.
2) It's a hierarchical chart that accounts for the individual parts of system.
3) There are two main classifications - a) Product hierarchy , b) Process hierarchy.
   a) Product hierarchy :- It identifies the product components and indicates the manner in which the components are interconnected.
   b) Process hierarchy :- It identifies the work activities and the relationship between them.
4) The primary advantage of the WBS technique are it helps in identifying and accounting the various process & product factors and in identifying the exact costs included in the estimates



Example of a Product Hierarchical Structure



Example of a Process Hierarchical Structure

Software Cost Estimation

The major factors that influence software cost are: Development Costs and Maintenance Costs. Further it can be categorized as :-

1) Programmer ability
2) Product complexity
3) Product size
4) Available time
5) Required reliability
6) Level of technology

Programmer Ability

It had been observed that on very large projects the differences in individual programmer ability will tend to average out and would not effect the project adversely. But on projects utilizing 5 or lesser programmers the factor of individual ability is significant and can affect the project positively or negatively. Hence individual programmer ability governs programmer productivity that affects the cost of the project.

Software Product Complexity

(Metric for Product Complexity, Programmer Productivity and Staffing level)

A software product can be classified on the basis of their complexity in to basic three types – a) Application programs , b) Utility programs , c) Systems program. Through extensive studies it had observed that Utility programs are 3 times as difficult to write as Application programs and that Systems programs are three times as difficult to write as Utility programs. So the levels of complexity can be stated as 1 - 3 - 9 for Application – Utility – Systems programs. The following equations were derived from the extensive research –

For calculating Programmer Months (PM) :-

Application programs :- $PMap = 2.4 * (KDSI)^{1.05}$

Utility programs :- $PMup = 3.0 * (KDSI)^{1.12}$

Systems programs :- $PMsp = 3.6 * (KDSI)^{1.2}$

For calculating Development time for a program (TDEV):-

Application programs :- $TDEVap = 2.4 * (PM)^{1.05}$

Utility programs :- $TDEVup = 3.0 * (PM)^{1.12}$

Systems programs :- $TDEVsp = 3.6 * (PM)^{1.2}$

For calculating Average Staffing Level (SL) :-

Application programs :- SLap= PMap / TDEVap

Utility programs :- SLup= PMup / TDEVup

Systems programs :- SLsp= PMsp / TDEVsp

Where , PM – Programmer Months , KDSI – Thousands of delivered source instructions.

A major consideration that needs to be taken in to account is extra coding required for housekeeping purpose. Housekeeping codes include that portion of coding that handles a) input/output , b) interactive user communication , c) human interface engineering and d) error checking & error handling. So the estimation of cost on lines of code should include housekeeping codes.

Product Size

A large software product is obviously more expensive to develop than a small one. The equations derived for Programmer months (PM) and Development time (TDEV) show that the rate of increase in effort required grows with the number of source instructions at an exponential rate slightly greater than 1.

Available Time

Total project effort is sensitive to the calendar time available for the project completion. Software projects require more effort if the development time is compressed or expanded from the optimal time. After extensive research it had been observed that there is limit beyond which a software project cannot reduce its schedule even by buying more personnel and equipment. The limit occurs roughly at 75% of the nominal schedule.

Required Reliability

It includes the capacity give to the software for

1. Exception handling.
2. Error handling.
3. Accuracy.

Level of Technology

The level of technology provided by latest software products help reducing overall cost. Concurrency, type-checking and self-documentation aspects of high-level languages improve the reliability and modifiability of the programs created using these high-level languages. The familiarity of the programmer with the latest technology is also a contributing factor in effort reduction and cost reduction.

Software Cost Estimation Techniques

Cost estimates can be made either top-down or bottom-up. Top-down estimation first focuses on system level costs, such as the computing resources and personnel required to develop the system, as well as the costs of a) Configuration management, b) Quality assurance, c) System integration, d) Training, and e) Publications. Personnel costs are estimated by examining the cost of similar past projects.

Bottom-up cost estimation first estimated the costs to develop each module or sub-system, those costs are combined to arrive at an overall

estimate. Bottom-up estimation emphasizes the costs associated with developing individual system components, but may fail to account for system level costs, such as configuration management and quality control.

1) Expert Judgment

The most widely used cost estimation technique is Expert judgment. Inherently it's a **top-down estimation technique**. Expert judgment relies on the experience background and business sense of one or more key people in the organization.

Groups of experts sometimes prepare a consensus estimate(collective estimate) , this tends to minimize individual oversights and lack of familiarity with particular projects and neutralizes personal biases and the desire to win the contract through an overly optimistic estimate as seen in individual judgment style.

The major disadvantage of group estimation is the effect that interpersonal group dynamics may have on individuals in the group. Group members may not be candid enough due to political considerations. The presence of authority figures in the group or the dominance of an overly assertive group member.

2) Delphi cost estimation (Recursive Estimation)

This technique was developed to gain expert consensus without introducing the adverse side effects of group meetings. The Delphi can be adapted to software cost estimation in the following manner :-

1) A coordinator provides each estimator with the system definition document and a form for recording the cost estimation.
2) Estimators study the definition and complete their estimates anonymously. They may ask questions to the coordinator, but they do not discuss their estimates with one another.
3) The coordinator prepares and distributes a summary of the estimator's responses any unusual rationales noted by the estimators.

4) Estimators complete another estimate again anonymously , using the result from the previous estimate. Estimators whose estimates differ sharply from the group may be asked, anonymously, to provide justification for their estimates.
5) The process id iterated for as many rounds as required. No group discussion is allowed during the entire process.

3) The COCOMO Model

(Software Cost Metric, Mix of BottomUp and TopDown approach)

The Constructive Cost Model(COCOMO) is an empirical estimation model i.e., the model uses a theoretically derived formula to predict cost related factors. This model was created by "Barry Boehm" . The COCOMO model consists of three models :-

1) The Basic COCOMO model :- It computes the effort & related cost applied on the software development process as a function of program size expressed in terms of estimated lines of code (LOC or KLOC).
2) The Intermediate COCMO model :- It computes the software development effort as a function of – a) Program size, and b) A set of cost drivers that includes subjective assessments of product, hardware, personnel, and project attributes.
3) The Advanced COCOMO model :- It incorporates all the characteristics of the intermediate version along with an assessment of cost driver's impact on each step of software engineering process.

The COCOMO models are defined for three classes of software projects, stated as follows:-

1) Organic projects :- These are relatively small and simple software projects which require small team structure having good application experience. The project requirements are not rigid.
2) Semi-detached projects :- These are medium size projects with a mix of rigid and less than rigid requirements level.

3) Embedded projects :- These are large projects that have a very rigid requirements level. Here the hardware, software and operational constraints are of prime importance.

Basic COCOMO Model (Bottom Up)

The effort equation is as follows :-

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

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

Where E – effort applied by per person per month, D – development time in consecutive months, KLOC – estimated thousands of lines of code delivered for the project. The coefficients a_b, c_b, and the coefficients b_b, d_b are given in the Table below.

Table

Coefficients & exponents used in the Basic (_b) COCOMO Model

Software project type	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

Intermediate COCOMO Model (Mix of TopDown and BottomUp)

The effort equation is as follows :-

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

Where E – efforts applied by per person per month, KLOC – estimated thousands of lines of code delivered for the project, and the coefficients a_i and exponent b_i are given in the Table below. EAF is Effort Adjustment Factor whose typical ranges from 0.9 to 1.4 and the value of EAF can be determined by the tables published by "Barry Boehm" for four major categories – product attributes, hardware attributes, personal attributes and project attributes.

Table

Coefficients & exponents used in the Intermediate (_i) COCOMO Model

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

The effort adjustment factor (EAF) is calculated using 15 cost drivers. The cost drivers are grouped into four categories: *product*, *computer*, *personnel*, and *project*. Each cost driver is rated on a six-point ordinal scale ranging from low to high importance. Based on the rating, an effort multiplier is determined using Table provided by Barry Boehm (Boehm, 1981). The product of all effort multipliers is the EAF.

Payback Analysis:

1. Payback period in business and economics refers to the period of time required for the return on an investment to "repay" the sum of the original investment. For example, a $1000 investment which returned $500 per year would have a two year payback period.
2. It intuitively measures how long something takes to "pay for itself"; shorter payback periods are obviously preferable to longer payback periods.
3. **Basic formula:**

   Payback = Days/Weeks/Months x Initial Investment / Total cash received

Earned Value Analysis:

1. One approach to measuring progress in a software project is to calculate how much has been accomplished. This is called earned value analysis.
2. It is basically the percentage of the estimated time that has been completed.
3. This is based on estimated effort, it could be based on any quantity that can be estimated and is related to progress.
4. **Basic Measures used:**
   - **Budgeted Cost of Work (BCW):** The estimated effort for each work task.
   - **Budgeted Cost of Work Scheduled (BCWS):** The sum of the estimated effort for each work task that was scheduled to be completed by the specified time.
   - **Budget at Completion (BAC):** The total of the BCWS and thus the estimate of the total effort for the project.
   - **Planned Value (PV):** The percentage of the total estimated effort that is assigned to a particular work task; **PV = BCW/BAC**.
   - **Budgeted Cost of Work Performed (BCWP):** The sum of the estimated efforts for the work tasks that have been completed by the specified time.
   - **Actual Cost of Work Performed (ACWP):** The sum of the actual efforts for the work tasks that have been completed.

5. **Progress Indicators for EVA :**
   - **Earned Value (EV)** = BCWP/BAC
     = The sum of the PVs for all completed work tasks
     = PC = Percent complete
   - **Schedule Performance Index(SPI**) = BCWP/BCWS
   - **Schedule Variance (SV)** = BCWP - BCWS
   - **Cost Performance Index(CPI)** = BCWP/ACWP
   - **Cost Variance (CV)** = BCWP – ACWP

Scheduling techniques :

1. **Task networks (activity networks)** are graphic representations that can be of the task interdependencies and can help define a rough schedule for particular project Scheduling tools should be used to schedule any non-trivial project.
2. **PERT (program evaluation and review technique) and CPM (critical path method)** are quantitative techniques that allow software planners to identify the chain of dependent tasks in the project work breakdown structure that determine the project duration time.
3. **Timeline (Gantt) charts** enable software planners to determine what tasks will be needed to be conducted at a given point in time (based on estimates for effort, start time, and duration for each task).
4. **Time-boxing** is the practice of deciding in advance the fixed amount of time that can be spent on each task. When the task's time limit is exceeded, development moves on to the next task (with the hope that a majority of the critical work was completed before time ran out).

Tracking Project Schedules :

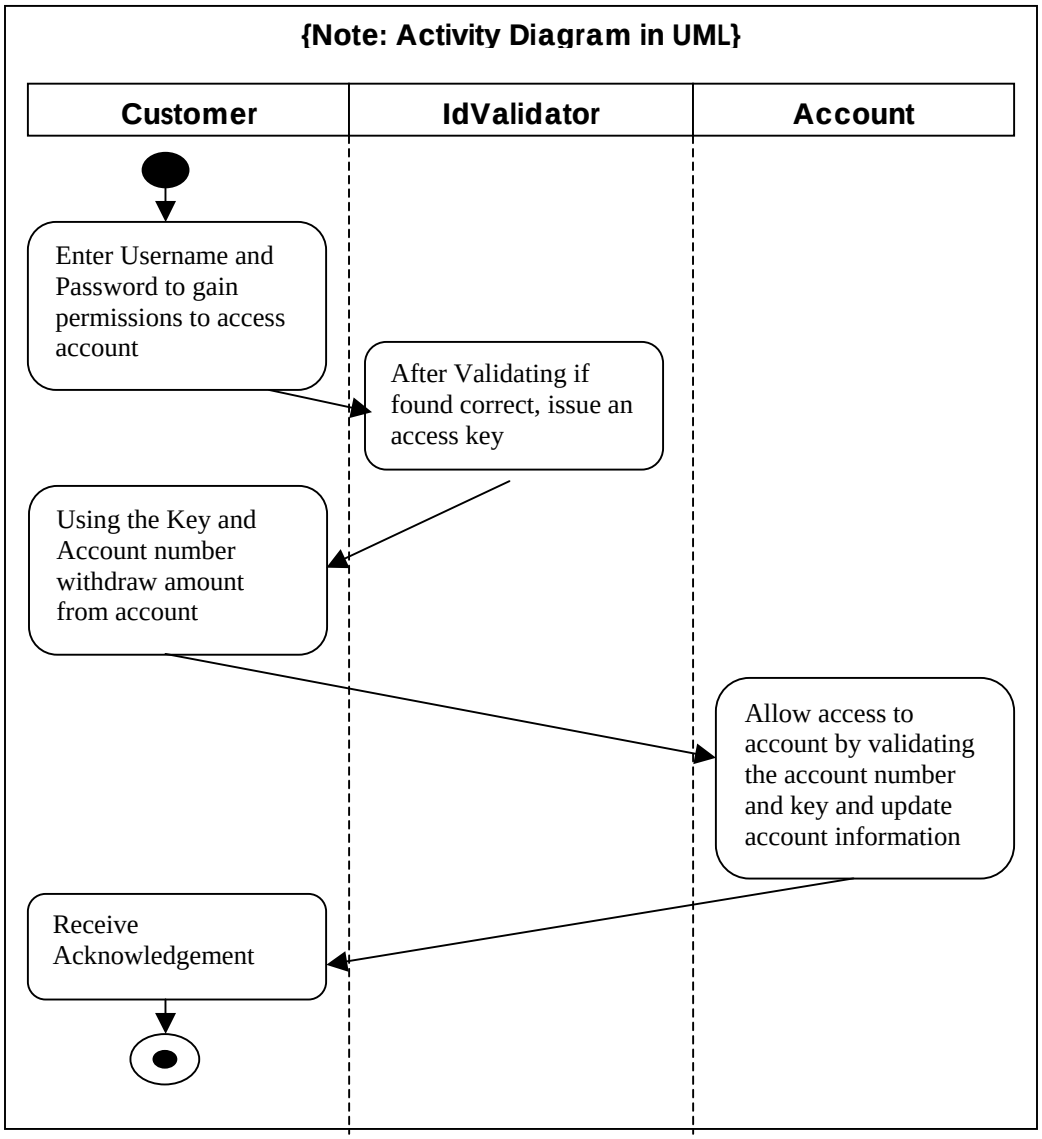
1. Periodic status meetings
2. Evaluation of results of all work product reviews
3. Comparing actual milestone completion dates to scheduled dates
4. Comparing actual project task start-dates to scheduled start-dates
5. Informal meeting with practitioners to have them assess subjectively progress to date and future problems
6. Use earned value analysis to assess progress quantitatively

Effort-Driven Scheduling

1. Assigning resources to tasks affects the schedule differently if the tasks are effort-driven.
2. Effort-driven scheduling means that as we add resources to a an effort-driven task, the work is redistributed among all the resources to maintain the same amount of work overall. Likewise, if we remove resources from an effort-driven task then the work is redistributed among the remaining resources, to maintain the same amount of work overall.
3. By default, tasks have fixed units and are effort-driven. This means that as more resources are assigned to a task, there is less work to be done. Adding resources reduces duration.
4. If we have a fixed-duration, effort-driven task then by assigning more resources, reduces the overall amount of work to be done and also reduces the overall duration of work.
5. Effort-driven scheduling only takes effect when resources are added to or removed from a task.
6. Effort-driven calculation rules are not applied when you change work, duration, and unit (units: The quantity of a resource assigned to a task) values for resources already assigned to a task.

Activity Diagram :

- Activity diagrams show the procedural flow of control between two or more class objects while processing an activity.
- Activity diagrams can be used to model higher-level business processes, or to model low-level internal class actions.
- Usually activity diagrams are best used to model higher-level processes.
- Here the diagram is divided into columns where every column header represents the name of the Class whose objects would interact with each other.
- Here also a specific Use-Case is explained.
- The start of the activity is denoted by a filled circle and end of the activity is denoted by a bulls eye.
- The activities that the Class Objects would perform is represented using circular edged rectangles in which the name of the activity is written.
- The activities are connected through transition lines.


{Note: Activity Diagram in UML}
Customer
IdValidator
Account
Enter Username and Password to gain permissions to access account
After Validating if found correct, issue an access key
Using the Key and Account number withdraw amount from account
Allow access to account by validating the account number and key and update account information
Receive Acknowledgement

CHAPTER 04

RISK AND CHANGE MANAGEMENT

Risk Management

A general categorization of risk can be done as follows

1. **Known risks :-** These can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is been developed. Other reliable factors can also be considered, like, unrealistic schedule, poor development environment etc.
2. **Predictable risks :-** These can be predicted form past project experiences (e.g., staff turnover, poor communication with customers etc.) and a plan can be designed long before the risk becomes reality.
3. **Unpredictable risks :-** These very difficult to identify and often become reality. For these kinds of risks, reactive risk management strategy needs to be taken.

Apart from above mentioned general categorization of risks there are some specific kinds of risks that are commonly seen during software projects :-

1. **Project risks :-** These are related to the project plan. They include – a) Project budget, b) Project schedule, c) Project personnel (staffing & organization), d) Project resources, e) Customer, f) Software requirements related problems.
2. **Technical risks :-** These are related to acceptance and quality of the software. These occur because during problem analysis the complexity of the problem is not understood to the maximum limits. They include – a) Implementation problems, b) Design problems, c) Verification deficiency, d) Maintenance problems, e)Interfacing problems, and f) Technical obsolescence.
3. **Business risks :-** These are related to general acceptability of the software. The factors that contribute to the business risks are – a) Market risk, where the product is not required in the market, b) Strategic risks, where the product does not fit the business strategy of the customer, c)

Management risks, where the management is no more interested in the project due to change in focus or people of management, d) Budget risks, where the management or control on budgetary is lost, and e) Sales risk, where the sales force does not find the appropriate & effective method to sell the product.

There two main categories of Risk Management strategies :-

1. **Reactive risk strategy :-** Here the software team does not worry about the risks becoming reality and if it does then the entire risk management force comes in to action, when they fail then the "crisis management" group takes over and tries to solve the problem.
2. **Proactive risk strategy :-** Here the preparations are made long before the risk can become reality. The potential risks are identified , their probability and impact are assessed. The risk information are prioritized. The team has a contingency plan for the situations when the risk becomes reality, otherwise risk avoidance procedures are maintained throughout the project.

Risk Management:

In ideal risk management, a prioritization process is followed whereby the risks with the greatest loss and the greatest probability of occurring are handled first, and risks with lower probability of occurrence and lower loss are handled in descending order.

Principles of risk management:

1. Risk management should create value.
2. Risk management should be an integral part of organizational processes.
3. Risk management should be part of decision making.
4. Risk management should explicitly address uncertainty.

5. Risk management should be systematic and structured.
6. Risk management should be based on the best available information.
7. Risk management should be tailored.
8. Risk management should take into account human factors.
9. Risk management should be transparent and inclusive.
10. Risk management should be dynamic, iterative and responsive to change.
11. Risk management should be capable of continual improvement and enhancement.

Potential risk treatments:

1. **Accept risk**; simply take the chance that the negative impact will be incurred
2. **Avoid risk**; changing plans in order to prevent the problem from arising
3. **Mitigate risk**; lessening its impact through intermediate steps
4. **Transfer risk**; outsource risk to a capable third party that can manage the outcome

Software Configuration Management (SCM)

1) It's an umbrella (commonly implemented in every phase) activity that is applied through out the development process.
2) The basic intention of SCM is :- a) Identify change, b) Control change, c) Ensuring proper implementation of change, d) Reporting the change to the concerned individuals.
3) SCM activities begin with the project from the initial phase and remain active till the software becomes obsolete.
4) **SCM terms** :-
   a) **Software Configuration Items (SCI)** :- are the items that are generated as a result of the software process. SCI can consist of other SCIs.
   b) **Baselines** :- A specification or product that has been formally reviewed and agreed such that it serves as the basis for further development

and that can be changed only through formal change control procedures. Before an SCI becomes a baseline (or baselined), the change can be made quickly and informally.

Types of baselines are as follows:-

1. **Functional Baselines**:- The requirement document which specifies the functional requirements of the software.
2. **Design Baselines**:- Designs of various components of the system.
3. **Product or System Baselines**:- The subsystems developed and the overall system developed.

5) **The five basic SCM tasks are :-**

i) Identification of objects,
ii) Version control,
iii) Change control
iv) Configuration auditing and
v) Reporting.

i) **Identification of objects :-** To control and manage SCIs, each of them must be separately named and then organized using object-oriented approach. The SCIs can be classified in to two kinds of objects – a) Basic objects and b) Aggregate objects.

a) Basic object :- It's a "unit of text" created by a software engineer during analysis, design, coding or testing phases.
b) Aggregate object :- It's a collection of basic objects and other aggregate objects.

Each object has a set of unique features for distinct identification. Every object should have – i) An unique name, ii) A description of the object including its SCI type eg. Document, program or data, a project identifier and version information, iii) A list of resources, which are entities that are processed, provided or required by the object.

ii) **Version control :-** It combines various tools and procedures to manage and control different versions of SCI objects (as a result of change) that are created during the software engineering process. Here some

standards are followed to give version numbers to the changing instances of the software.

iii) **Change control :-** For a large system development project , uncontrolled change rapidly leads to confusion and inconsistencies. Change control includes human procedures and automated tools to control the reasons for change. The following processes take place when a situation of change occurs (Standard Change Control Procedures) :-

a) Change request :- A change request consists of (i) The SCI object description, (ii) Suggestions for change in the object, (iii) Effected subsystems and (iv) estimated cost for carrying out the change.
The change request is submitted and then it is evaluated to asses its – i) Technical merit, ii) Potential side effects, subsystems and the cost for implementing the change. Then a change report is generated.

b) Change report :- After the evaluation is done, a change report is created that is submitted to the Change Control Authority/Board (CCA or CCB). CCA or CCB is a group who are responsible for evaluating the change report and makes the final decision on the status and priority of the change. This group generates a Engineering Change Order (ECO) for each approved change.

c) ECO (Engineering Change Order):- It consists of - i) The description of the change to be made, ii) Constraints that has to be taken care of and iii) Criteria for review and audit.

d) Check out & check in :- The object to be changed is "checked out" of the project database, the decided changes are made and appropriate SQA (Software Quality Assurance) activities are performed. The object is then "checked in" the project database and appropriate version control mechanisms are used to create the next version of the software.

iv) **Formal technical reviews (FTR) & Configuration audit :-** These two activities are required to ensure that the change made to the software is properly implemented. It's a kind of Quality Assurance activity.

e) **Formal technical reviews (FTR)** :- It's a part of Software Quality Assurance (SQA) procedures. The objectives of FTR are - i) To uncover errors in functions, logic or implementation, ii) To verify that the software under review should meet its requirement, iii) To ensure that the representation of the software is according to the standards, iv) To make the project more manageable v) It consists of walkthroughs, inspections and round-robin reviews.

f) **Software configuration audit** :- It consists of the following auditing procedures – i) To check whether the changes specified in the ECO (Engineering Change Order) has been properly made and to check if any additional modules are added, ii) To check whether formal technical reviews are conducted for the assessment of technical correctness, iii) To check whether SE standards are properly followed, iv) To check whether the change is highlighted in the SCI (Software Configuration Items) documentation and also the attributes of the configuration object should reflect the change, v) To check whether SCM(Software Configuration Management) procedures like noting change, recording it and reporting it has been properly done, vi) To check whether all the related SCIs are properly updated.

v) **Configuration Status reporting (CSR) :-** Its an SCM task which summarizes the activities done so far which includes the following – a) The report of all the activities done , b) The report on the persons involved in the above reported activities, c) The report on the time and date when the reported activities were accomplished, d) The report on various other factors or objects that will be effected due to the reported activity.

Following are the situations where the CSR needs updating - a) Each time when a SCI is assigned a new or updated identification, b) Each time ECO is issued i.e a when a change is approved by the CCA/CCB, c) Each time a configuration audit is conducted.

The CSR is made available online and is updated occasionally in order to keep the management and concerned individuals updated on the changes. It improves the communication among all the people involved.

Some ANSI/IEEE SCM standards are

1. MIL-STD-483, DOD-STD-480A, etc. for military purpose.
2. 828-1983, 1042-1987, etc. for non military purpose.

Reviews

1. Reviews are useful in finding and eliminating defects.
2. Reviews help to find defects soon after they are injected during the development phases making it less costly to fix compared to if they were found in later phases.
3. All work products like Software requirements specifications, schedules, design documents, code, test plans, test cases, and defect reports in a software project should be reviewed and tested.

Types of Reviews:

1. **Inspections**:
    a. They are moderated meetings in which reviewers list all issues and defects they have found during the inspections.
    b. The goal of the inspection is to repair all of the defects so that everyone on the inspection team can approve the work product.
    c. Commonly inspected work products include software requirements specifications and test plans.
2. **Deskcheck:**
    a. It is a simple review in which the author of a work product distributes it to one or more reviewers.

b. The author sends a copy of the work product to selected project team members. The team members read it, and then write up defects and comments and send it back to the author.
c. Deskchecks can be used as predecessors to inspections.

3. **Walkthrough**:
   a. It is an informal way of presenting a technical document in a meeting.
   b. Unlike other kinds of reviews, the author runs the walkthrough by calling the meeting, inviting the reviewers, soliciting comments and ensuring that everyone present understands the work product.
4. **Code Review**:
   a. It is a special kind of inspection in which the team examines a sample of code and fixes any defects in it.
5. **Pair programming**:
   a. It is a technique in which two programmers work simultaneously at a single computer and continuously review each others' work.
   b. Pair programming improves the project management by ensuring that at least two programmers are able to maintain any piece of the software.

Reports

1. Reports do not need to be of the same depth or at the same frequency for each level
2. Reports must contain data relevant to the control of specific tasks that are being carried out according to a specific schedule
3. The timing of reports should generally correspond to the timing of project milestones

Benefits of detailed, timely reports:

1. Creating awareness related to the progress of parallel activities
2. More realistic planning for the needs of all groups
3. Understanding the relationships of individual tasks to one another and the overall project
4. Early warning signals of potential problems and delays
5. Faster management action in response to unacceptable or inappropriate work
6. Higher visibility to top management

Three distinct types of reports:

1. **Routine:**
   - ✓ They are issued on a regular basis at regular intervals.
   - ✓ The information is gathered in specific schedule from the sources through which assessment can be done
2. **Exception:**
   - ✓ Generated only when some exception condition occurs.
   - ✓ Indicating the variance from normal.
3. **Special analysis:**
   - ✓ They are used to disseminate the results of special studies conducted as a part of the project.
   - ✓ These reports may also be used in response to special problems that arise during the project.

Effort-Driven Scheduling

1. Assigning resources to tasks affects the schedule differently if the tasks are effort-driven.
2. Effort-driven scheduling means that as we add resources to a an effort-driven task, the work is redistributed among all the resources to maintain the same amount of work overall. Likewise, if we remove resources from an effort-driven task then the work is redistributed among the remaining resources, to maintain the same amount of work overall.

3. By default, tasks have fixed units and are effort-driven. This means that as more resources are assigned to a task, there is less work to be done. Adding resources reduces duration.
4. If we have a fixed-duration, effort-driven task then by assigning more resources, reduces the overall amount of work to be done and also reduces the overall duration of work.
5. Effort-driven scheduling only takes effect when resources are added to or removed from a task.
6. Effort-driven calculation rules are not applied when you change work, duration, and unit (units: The quantity of a resource assigned to a task) values for resources already assigned to a task.

Scheduling techniques :

1. **Task networks (activity networks)** are graphic representations that can be of the task interdependencies and can help define a rough schedule for particular project Scheduling tools should be used to schedule any non-trivial project.
2. **PERT (program evaluation and review technique) and CPM (critical path method)** are quantitative techniques that allow software planners to identify the chain of dependent tasks in the project work breakdown structure that determine the project duration time.
3. **Timeline (Gantt) charts** enable software planners to determine what tasks will be needed to be conducted at a given point in time (based on estimates for effort, start time, and duration for each task).
4. **Time-boxing** is the practice of deciding in advance the fixed amount of time that can be spent on each task. When the task's time limit is exceeded, development moves on to the next task (with the hope that a majority of the critical work was completed before time ran out).

Tracking Project Schedules :

1. Periodic status meetings
2. Evaluation of results of all work product reviews
3. Comparing actual milestone completion dates to scheduled dates
4. Comparing actual project task start-dates to scheduled start-dates
5. Informal meeting with practitioners to have them assess subjectively progress to date and future problems
6. Use earned value analysis to assess progress quantitatively

CHAPTER 05

PROJECT TESTING & PROJECT SUCCESS

Verification and Validation

Verification & Validation (V & V) are the activities that encapsulate al kinds of testing. Verification is the set of activities that are carried out to ensure that the software implements a specific function correctly. Validation is the set of activities that are carried out to ensure that the software product that is built is in accordance with the requirements stated by the customer.

In simple words :-

Verification :- "Are we building the product right?", To check whether all the functionalities are working as per Goals set for each functionality.

Validation :- "Are we building the right product?", To check whether the customer's requirements are getting fulfilled by the product.

The goals of verification and validation activities are to asses and improve the quality of the work products generated during development and modification of software. The quality attributes include – a) Correctness, b) Completeness, c) Consistency, d) Reliability, e) Usefulness, f) Conformance to standards and g) overall cost effectiveness.

Verification :- Its the process of determining whether the product is built in a right manner or not. There are two categories of verification :- a) Life-cycle verification and b) Formal verification.

i) Life-Cycle verification :- It's the process of determining the degree to which the work products of a given phase of the development cycle fulfill the specifications established during prior phases.

ii) Formal verification :- It's a rigorous mathematical demonstration that whether the source code conforms to the specification.

Validation :- It's the process of evaluation of the software at the end of the software development process to determine compliance with the requirements. In other words it's the process of determining whether the correct product is built or not.

Verification and validation involve the assessment of work products to determine conformance to the specifications. Specifications include – a) Requirements specification, b) The design documentation, c) Various stylistic guidelines, d) Implementation language standards, d) Project standards, e) Organizational standards, and f)User expectations.

Software Testing:

1. Testing is a major quality control measure used during software development. Its basic function is to detect defects in the software.
2. Testing not only has to uncover errors introduced during coding, but also errors introduced during the previous phases. Thus, the goal of testing is to uncover requirement, design, and coding errors in the programs.

Basic Levels of Testing:

The starting point of testing is **unit testing**, where the different modules or components are tested individually. As modules are integrated into the system, **integration testing** is performed, which focuses on testing the interconnection between modules. After the system is put together, **system testing** is performed. Here the system is tested against the system requirements to see if all the requirements are met and if the system performs as specified by the requirements. Finally, **acceptance testing** is performed to demonstrate to the client the response of the created system by entering real data.

Testing Objectives :

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding an error that was not discovered yet.
3. A successful test is one that uncovers all the errors as well as those errors that were not discovered earlier.

There are four types of test a software product must satisfy :-

1. **Structural test (White-Box testing or Glass-Box testing):-** They are concerned with the internal processing logic of the software system. It examines the routines called and the logical path traversed through the routine and determining the efficiency of the logical path followed during processing.

   This type of testing is carried out in early stages of the entire testing process. It uses the control structure of the procedural design to derive the test cases. This method helps in the following way :-

   - ✓ To guarantee that all the independent paths within a module have been exercised at least once,
   - ✓ Checking all the logical decisions for true and false,
   - ✓ Checking all the loops for their maximum and minimum operational bounds, and
   - ✓ Exercise all the internal data structures to assure validity.
2. **Performance test :-** They are designed to verify response time, execution time, throughput, primary & secondary memory utilization, traffic rates on communication channels etc. All these test are performed under various loads. The results of these tests would indicate any processing bottlenecks if present.
3. **Stress test :-** They are designed to overload the system in various ways ex. to assign more than maximum allowed number of terminals, disconnecting a communication link etc. The purpose of this test is to determine the limitations of the system and if the system fails then to

determine the causes of failure and the manner in which the failure has taken place. It shows the strength and weaknesses of the system.

4. **Functional test (Black-Box testing) :-** They specify typical operating conditions, typical input values, and typical expected results. These tests should be designed to test boundary conditions just inside & just beyond the boundaries.

   This kind of testing focuses on the functional aspects of the software. This type of testing is carried out in the later stages of the entire testing procedure. It enables the software engineer to derive sets of input conditions that will fully exercise all the functional aspects of the software. Black-box testing can be used to find out the following errors:

   - ✓ Interface errors.
   - ✓ Missing or incorrect functions.
   - ✓ Performance errors.
   - ✓ Initialization & termination errors.
   - ✓ Errors in data structures

Equivalence Partitioning:

1) It's a type of Black-box testing. In equivalence partitioning a test case is defined (based on various observations) that discovers a various categories("classes") of errors which reduces the number of test cases to be developed for encountering the errors.
2) In this method various "equivalence classes" are evaluated for an input condition. An equivalence class represents a set of valid or invalid states for input condition. Usually an input condition can be any of the following :-
   - i) A specific numeric value.
   - ii) A range of values.
   - iii) A set of related values.
   - iv) A Boolean condition.

Equivalence classes may be defined according to the following guidelines.

1. For a specific numeric value, one valid and two invalid equivalence classes are defined. Example, for a 6 digit alphanumeric value as input for a password entry, the valid equivalence class would be the presence of correct password input and the two invalid equivalence classes would be, first if the input has less than 6 alphanumeric characters and second, if the input password string is incorrect.
2. For a range of values, one valid and two invalid equivalence classes are defined. Example, for a range of three digits or blank as input for an area code entry, here the valid equivalence class would be the input within the range and the first invalid equivalence class would be a range of values as area codes less than the lower bound and second invalid equivalence class would be for the entry more than the upper bound.
3. For a set of related values (member of a set), one valid and one invalid equivalence class are defined. Example, for a value from a predefined set like a set of commands fed as input for a commands entry, here the valid equivalence class would be the presence of input that is in the commands list and the invalid equivalence class would be invalid command input which is not present in the commands set.
4. For a Boolean value, one valid and one invalid equivalence class are defined. Here the presence of input will come under valid equivalence class and an absence of input will come under invalid equivalence class.

Regression Testing:

1. Each time a new module is added as part of integration process, the software changes. These changes may cause problems with functions that previously worked flawlessly. Regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.
2. Regression testing may be conducted manually, by re-executing a subset of all test cases to ensure that the new integration is flawless.
3. Following are the basic levels of tests performed during regression testing:

a. Tests that will exercise all software functions.
b. Tests that focus on software functions that are likely to be affected by the change.
c. Tests that focus on the software components that have been changed.

Unit Testing:

1. Unit testing focuses on the smallest unit of software design i.e. the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module.
2. The unit test is white-box oriented.
3. Selective testing of execution paths is an essential task during the unit test.
4. Boundary testing is the last and one of the most important tasks of the unit test.

Integration Testing :

1. Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing.
2. The objective is to take unit tested components and build a program structure strictly as per design.
3. The program is constructed and tested in small increments, where errors are easier to isolate and correct.

Validation Testing:

1. Validation succeeds when software functions in a manner as expected by the customer.
2. The SRS (Software Requirement Specification) contains a section called Validation Criteria. Information contained in that section forms the basis for a validation testing approach.
3. Following are the parts of Validation testing:

i) Software validation is achieved through a series of black-box tests that demonstrate the requirements are fulfilled.

ii) **Configuration Review or Audit** is an important element of the validation process. The intent of the review is to ensure that all elements of the software have been properly developed.

iii) The **alpha test** is conducted at the developer's site by a customer. The software is run by the user with the developer "looking over the shoulder" of the user and recording errors and usage problems. Alpha tests are conducted in a controlled environment.

iv) The **beta test** is conducted at one or more customer sites by the end-user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a "live" application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta tests, software engineers make modifications to the software.

System Testing:

1. System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. All the tests are done to verify that system elements have been properly integrated and perform allocated functions.
2. **Recovery testing** is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.
3. **Security testing** attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.
4. **Stress testing** executes a system in a manner that demands resources in abnormal quantity, frequency, or volume and observes how much stress the system can handle.

5. **Performance testing** is designed to test the run-time performance of software, in which both the software and hardware get tested for performance.

Debugging:

1. Debugging occurs as a consequence of successful testing. That is, when a test case uncovers an error, debugging is the process that results in the removal of the error.
2. Three categories for debugging approaches may be proposed:
   (1) Brute force, (2) Backtracking, and (3) Cause elimination.
   (1) The **Brute Force** category of debugging is probably the most common and least efficient method for isolating the cause of a software error. Using a "let the computer find the error" philosophy, memory dumps are taken, run-time traces are invoked, and the program is loaded with WRITE statements. We hope that somewhere we may find a clue that can lead us to the cause of an error.
   (2) **Backtracking** is a fairly common debugging approach that can be used successfully in small programs. Beginning at the site where a symptom has been uncovered, the source code is traced backward (manually) until the the cause is found.
   (3) In **Cause Elimination** a list of all possible causes is developed and tests are conducted to eliminate each. If initial tests indicate any positive clue finding then the process is continued further to reach to the cause of the bug.

Decision Making:

i. Decisions often arise in response to new information, and the initial way the issue is raised focuses on the acute and narrow aspects of the problem. So, the first thing is to ask probing questions and get down to the real decision that needs to be made.
ii. A big decision, such as the direction of the vision or the technology to use, will impact the entire project. If it's a long-term decision, and the impact is

deep, patience and rigor are required. It's best to make big decisions early on or in a given phase of a project, so they can be made with patient thought and consideration, instead of when time is running out.

iii. A small decision, such as what time to have a meeting or what the agenda should be, will impact a small number of people in a limited way. If it's a short-term decision with shallow impact, go for speed and clarity, based on a clear sense of the strategic decisions made in the vision.

iv. For aspects of projects such as usability or reliability, quality comes from many small decisions being aligned with each other.

v. If you wait too long to make the decision, routes will close and all options will go away eventually. Sometimes, you have to make tough strategic decisions quickly because of the limited window of opportunity you have. And sometimes, the speed of making a decision is more important than the quality of the decision itself.

vi. When we have no experience in taking decision related to any particular situation then it is better to take outside support and learn from the experience, rather than taking a wrong decision.

vii. There are political costs to decisions that have nothing to do with technology, business, or customer considerations, and the impact of a decision includes them. A seemingly trivial (small) decision can become complex when the politics and desires of stakeholders and partner organizations come into play.

viii. The decision maker needs to check periodically to make sure that the decision is understood properly and is being carried out properly.

ix. Use the lessons learnt from pervious projects while taking decisions in the present project.

Effective Communication:

1. **Transmitted:** The information is sent properly to the intended receipient, but it does not ensure that the receiver has read the information.
2. **Received:** Here the recipient has to acknowledge back to the sender that he has received and read the information, but it does not ensure that the receiver has properly understood the information.
3. **Understood:** Here the sender asks questions related to the information to ensure that the receiver has understood the information; also the receiver should ask questions to understand the information better.
4. **Agreed:** Understanding something doesn't mean a person agrees with it. So there should be an agreement between the sender and receiver that both agree upon the meaning expressed by the information.
5. **Converted to useful action:** even after formal approval the receiver might not follow the information completely, for that continuous reviews are needed.
6. Good communicators transmit information with the intention of it being understood and converted to action.

PMO (Project Management Office):

1. PMO is a corporate department responsible for the practice and discipline of project management within the organization.
2. They are usually departments with management control over project managers and responsibility for project success.
3. The functions of a PMO fall into three categories:
    a. **Development functions:** are those that build a cadre of effective project managers. They include activities such as:
        i. Recruiting staff.
        ii. Defining project management career and training paths.
        iii. Providing support and assistance to project managers.
        iv. Evaluating project managers at the end of a project.

b. **Support functions:** are those that help project managers become more effective in managing their projects. They include activities such as:
    i. Time gathering and reporting.
    ii. Defining standards for project documents.
    iii. Establishing priorities among projects.
    iv. Establishing procedures for issues such as:
        ✓ Scope control or review and approval.
        ✓ Creating standards for initiating and closing projects.
        ✓ Implementing project management methodologies and software.
        ✓ Providing a forum for resolving disputes.
c. **Control functions:** They include:
    i. Taking care of employee promotion.
    ii. Providing discipline and direction.
    iii. Defining mandatory standards such as the frequency of status reports or team meetings.
    iv. Reviewing projects in progress.

Resource leveling:

It is the process of smoothing out people's planned workloads to a realistic level. There are three steps in leveling resources:

1. **Assign specific people to activities:**
    a. By assigning the most experienced person to maximize your chances of success.
    b. If the company requires you to treat part of the project as a training exercise then assign an inexperienced person in order to expand the company's base of skilled people.
2. **Determine percent availabilities for each person**:

   a. While many people on the project will be available full-time, some, particularly those with specialized skills like network management, will be available on a more limited basis.
   b. The usual availability of any person is around 80%, as it includes general leaves, emergency leaves, training leaves, etc.
   c. Rarely overtime is expected from the employees.
3. **Smoothen the resource requirements**:
   a. The goal of smoothing resources is to plan activities so that all the members of the project team are busy to the extent of their availability while they are on the project.

Action Items:

1. All projects are the result of actions, and actions are carried out by individuals.
2. It is hard enough to get people to do work that is assigned to them.
3. There are two types of work to assign:
   a. project activities and
   b. action items.
4. Like project activities, an action item is a piece of work that is given to one or two people to be done by a specified date.
5. Action items are generally small, but they have the ability to cripple a project.
6. As soon as the issue is raised, the project manager must create an action item.
7. The project manager may assign the action item to the team member, or the technology architect, or a programmer or may do it by himself. It does not matter who handles it. The only things that matter is that somebody does it and that we make note of it so that we can follow up on that matter.
8. Action items arise from:
   a. Un documented Assumptions.

b. Doubts.
c. Unplanned Tasks.
d. Questions.
e. Requests.
f. Self-Thoughts.

Micro-planning:

1. Micro-planning is no different from normal planning. We decompose activities, determine dependencies, and level resources. The difference is only in the scale and degree of formality.
2. Micro-planning is not a replacement for regular planning. We are zooming in to a small part of the project plan, transforming the scale of planning from months or weeks to days or hours.
3. We Keep micro-planning informal. So we use hand-drawn schedules, or a daily activity list on a word processor or on an excel sheet.
4. We track a micro-plan the same way you track a normal project plan. However, just as the scale of a Micro-plan is of few days or hours, so is the frequency of its status reporting.
5. Micro-planning should not be done frequently and if done should be done when any critical situation is to be tackled.

Test metrics commonly used in software projects:

1. **The Defects per Project Phase metric** provides a general comparison of defects found in each project phase. This metric is usually shown as a bar graph, with the project phases on the X-axis, the number of defects on the Y-axis, and the data points being the number of defects found per phase.
2. **The Mean Defect Discovery Rate metric** weighs the number of defects found on a day-by-day basis over the course of the project. This is a standard and useful tool to determine whether a specific project is going according to a projected defect discovery rate (based on previous

projects and industry averages). This rate should slow down as the project progresses, so that far more defects are found at the beginning of software testing than at the end. If this metric remains constant or is increasing over the course of several test iterations, it could mean that there are serious scope or requirements problems, or that the programmers did not fully understand what the software was supposed to do.

3. **The Defect Resolution Rate** tracks the time taken to resolve defects, from the time that they are identified. This can be used to predict release dates by calculating the amount of time required to solve the remaining problems presently in the project.
4. **Defect Age** is used to measure how effective the review and inspection process is. A defect that was introduced in the scope or requirements phase but not discovered until testing is far more costly to fix than if it would have been caught in early phases. So the mean defect age should be as low as possible.
5. **The Percentage of Engineering Effort metric** compares how software effort (in man-hours) and actual calendar time break down into project phases. These two measurements will be useful in determining where additional training, staff, or process improvement is needed. This metric measures the average number of man-hours spent in each project phase. A project manager can use this measurement to gauge whether the effort is being used efficiently over the course of the project schedule.
6. **Defect Density** measures the number of defects per KLOC (thousand lines of code). It is often used to determine whether the software is ready for release. A test plan may have a maximum defect density listed in the acceptance criteria, which requires that the software should not contain any critical or high-priority defects, and contain less than a specific number of low-priority defects per KLOC.

Best of Luck