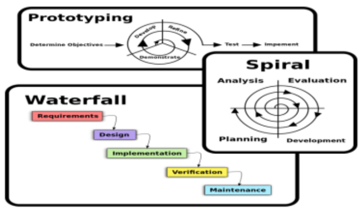
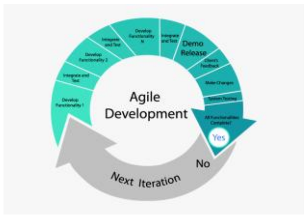
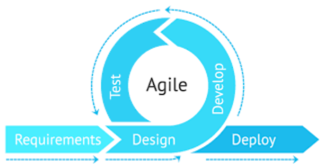


Software Engineering And Testing

Software Project Management





Analytical Estimation Techniques [Halstead's Software Science].

- The metrics are a set of measures the complexity of a software program.
- These metrics are based on the number of distinct operators and operands in the program and are used to estimate the effort required to develop and maintain the program.
- **Field of Halstead Metrics**
  - **Program length (N):** This is the total number of operator and operand occurrences in the program.
  - **Vocabulary size (n):** This is the total number of distinct operators and operands in the program.
  - **Program volume (V):** This is the product of program length (N) and the logarithm of vocabulary size (n), i.e., V = N*log2(n).

2

Analytical Estimation Techniques [Halstead's Software Science].

- **Field of Halstead Metrics**
  - **Program level (L):** This is the ratio of the number of operator occurrences to the number of operand occurrences in the program, i.e., L = n1/n2, where n1 is the number of operator occurrences and n2 is the number of operand occurrences.
  - **Program difficulty (D):** This is the ratio of the number of unique operators to the total number of operators in the program, i.e., D = (n1/2) * (N2/n2).
  - **Program effort (E):** This is the product of program volume (V) and program difficulty (D), i.e., E = V*D.

Analytical Estimation Techniques [Halstead's Software Science].

- **Field of Halstead Metrics**
  - **Time to implement (T):** This is the estimated time required to implement the program, based on the program effort (E) and a constant value that depends on the programming language and development environment.
- Limitations, such as the assumption that all operators and operands are equally important, and the assumption that the same set of metrics can be used for different programming languages and development environments.
- This software metrics can be a useful tool to estimate the effort required to develop and maintain software programs.

Analytical Estimation Techniques [Halstead's Software Science].

- **Notations**
  - n1 = Number of distinct operators.
  - n2 = Number of distinct operands.
  - N1 = Total number of occurrences of operators.
  - N2 = Total number of occurrences of operands.
- **Program Length** :The length of a program is total usage of operators and operands in the program.
  - Length (N) = N1 + N2
- **Program vocabulary:** The Program vocabulary is the number of unique operators and operands used in the program.
  - Vocabulary (n) = n1 + n2

Analytical Estimation Techniques [Halstead's Software Science].

- **Program Volume:** The Program Volume can be defined as minimum number of bits needed to encode the program.
  - Volume (V) = N log2 n
- **Length estimation:** N = n1 log2 n1 + n2 log2 n2
- **Guideline for calculating operands and operators:**

1. All the variables and constants are considered as operands.
2. Local variables with same name, if occurring in different functions are counted as unique operand.
3. Function calls are considered as operators.
4. The looping statements, do … while, while, for, are operators. The statements if, if …else, are operators. The switch … case statements are considered as operators.

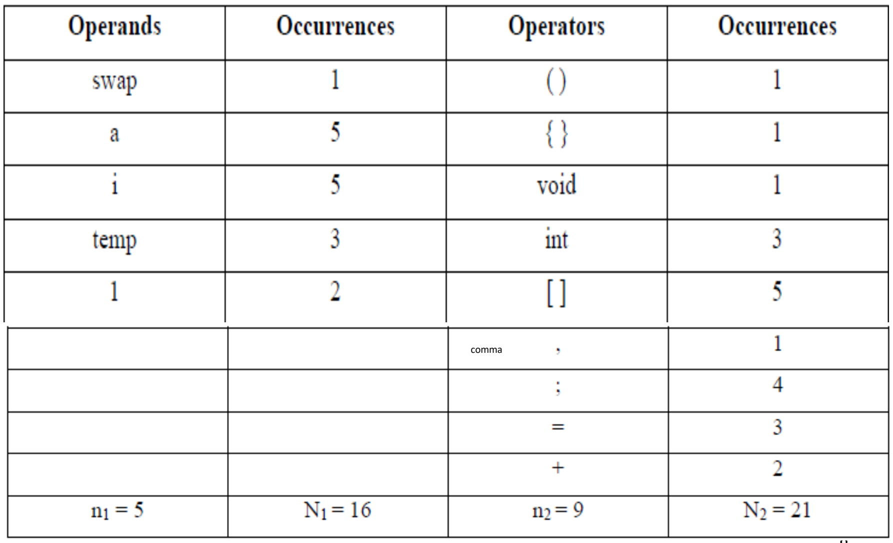
Analytical Estimation Techniques [Halstead's Software Science].

- **Example: Obtain Halstead's length and volume measure for following C function.**

```
void swap (int a[ ], int i)
{
        int temp;
       temp = a[i];
       a[i] = a[i+1];
       a[i+1] = temp;
}
```

- **Solution :**
  - We first find out the operands and operators from above function along with their occurrences.

Analytical Estimation Techniques [Halstead's Software Science].



Operands	Occurrences	Operators	Occurrences
swap	1	( )	1
a	5	{ }	1
i	5	void	1
temp	3	int	3
1	2	[ ]	5
		comma ,	1
		;	4
		=	3
		+	2
$n_1 = 5$	$N_1 = 16$	$n_2 = 9$	$N_2 = 21$

Analytical Estimation Techniques [Halstead's Software Science].

- **N = N1 + N2** = 16 +21 = 37 ∴ **N = 37**
- **n = n1 + n2** = 5 + 9 = 14 ∴ **n = 14**
- **Estimated length = n1 log n1 + n2 log n2**

  = 5 log 5 + 9 log 9

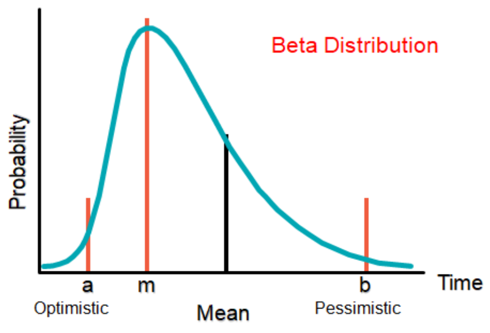
  = 5 * 2.32 + 9 * 2.19 = 31.37
- **Estimated length = 31.37**
- **Volume = N * log n**
  - = 37 * log (14)
  - 37 * 2.63 = 97.64
- **Volume (V) = 97.64**

9

PERT Chart

- Activity time estimates usually cannot be made with certainty.
- ***PERT used for probabilistic*** activity times.
- In PERT, three time estimates are used: ***most likely*** time (m), the ***optimistic*** time (a), and the ***pessimistic*** time (b); using Beta Distribution.
- These provide an estimate of the ***mean and variance*** of a beta distribution:
- variance: $v = \left(\frac{b - a}{6}\right)^2$
- mean (expected time):

  $$t = \frac{a + 4m + b}{6}$$

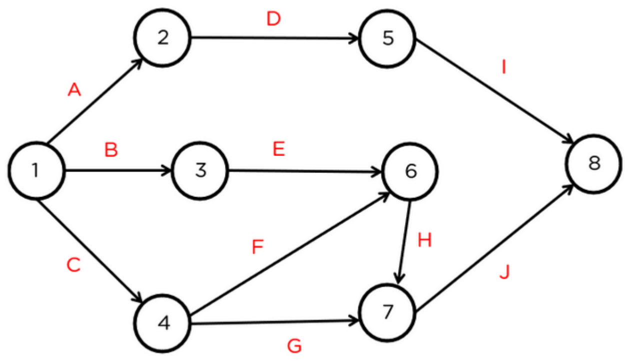


PERT Chart

- For the following project, draw the network diagram.
- Find the mean and variance.
- Find the critical path and estimated time of completion.

Activities	Immediate Predecessor	Optimistic Time	Most Likely Time	Pessimistic Time
A	-	6	7	8
B	-	3	5	7
C	-	4	7	10
D	A	2	3	4
E	B	3	4	11
F	C	4	8	12
G	C	3	3	9
H	E, F	6	6	12
I	D	5	8	11
J	H, G	3	3	9

PERT Chart



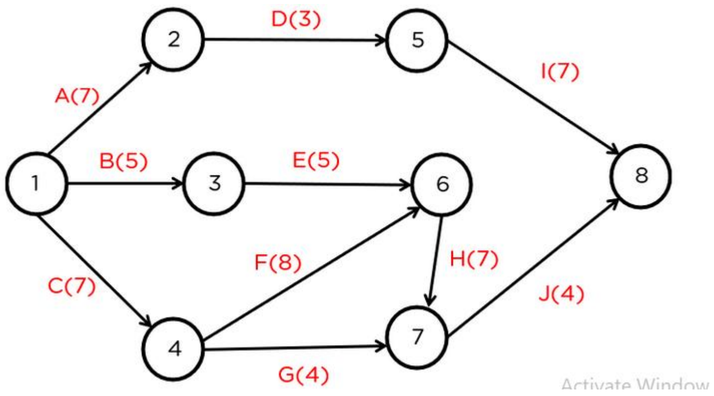
PERT Chart

$$T_e = \frac{T_0 + 4Tm + Tp}{6} \qquad \sigma^2 = \left(\frac{T_P - T_0}{6}\right)^2$$

For activity A,
The mean will be: (To + 4*Tm + Tp) /6
= (6 + 4*7 + 8) /6 = 7

Activities	Immediate Predecessor	Optimistic Time	Most Likely Time	Pessimistic Time	Mean	Variance
A	-	6	7	8	7	
B	-	3	5	7	5	
C	-	4	7	10	7	
D	A	2	3	4	3	
E	B	3	4	11	5	
F	C	4	8	12	8	
G	C	3	3	9	4	
H	E, F	6	6	12	7	
I	D	5	8	11	7	
J	H, G	3	3	9	4	

PERT Chart



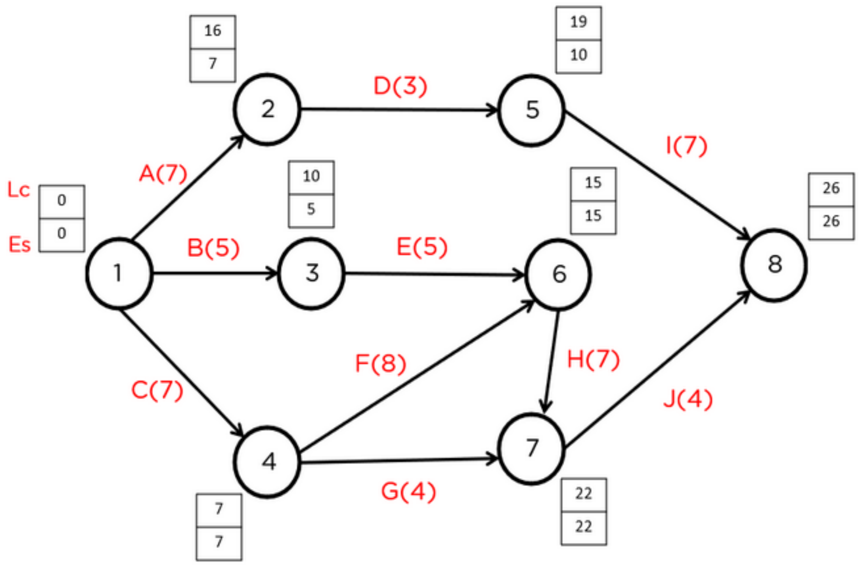
PERT Chart

$$T_e = \frac{T_0 + 4Tm + Tp}{6} \qquad \sigma^2 = \left(\frac{T_P - T_0}{6}\right)^2$$

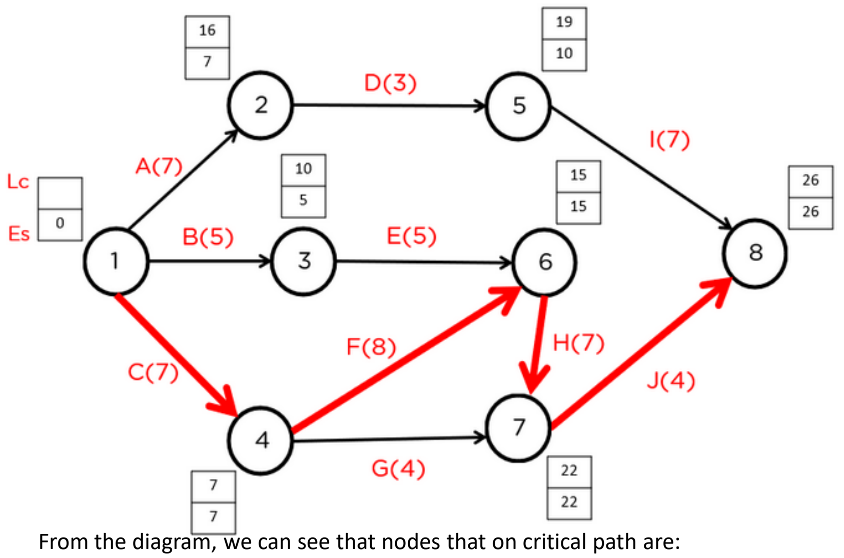
For activity A:
σ = [(Tp - To) /6]^2= [(8 - 6) /6]^2= 0.11

Activities	Immediate Predecessor	Optimistic Time	Most Likely Time	Pessimistic Time	Mean	Variance
A	-	6	7	8	7	0.11
B	-	3	5	7	5	0.44
C	-	4	7	10	7	1
D	A	2	3	4	3	0.11
E	B	3	4	11	5	1.77
F	C	4	8	12	8	1.77
G	C	3	3	9	4	1
H	E, F	6	6	12	7	1
I	D	5	8	11	7	1
J	H, G	3	3	9	4	1

PERT Chart



PERT Chart



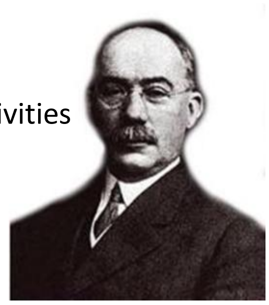
From the diagram, we can see that nodes that on critical path are:
1 - 4 - 6 - 7 - 8 or C - F - H – J
The estimated project time is: 7 + 8 + 7 + 4 = 26 days.

Comparison Between CPM and PERT

	CPM	PERT
1	Uses network, calculate float or slack, identify critical path and activities, guides to monitor and controlling project	Same as CPM
2	Uses one value of activity time	Requires 3 estimates of activity time Calculates mean and variance of time
3	Used where times can be estimated with confidence, familiar activities	Used where times cannot be estimated with confidence. Unfamiliar or new activities
4	Minimizing cost is more important	Meeting time target or estimating percent completion is more important
5	Example: construction projects, building one off machines, ships, etc	Example: Involving new activities or products, research and development etc

GANTT Charts

- Named after Henry Gantt (1861 - 1919) an American mathematical engineer.
- **A Gantt chart is a type of bar chart:**
  - Captures start and finish dates of the activities
- **Elements of a GANTT chart:**
  - Task names
  - Start and finish dates of each tasks (graphically)
  - Dependency relationships
  - Task duration in an additional column
  - Name of the project worker responsible for the task or Resource specifications
  - other



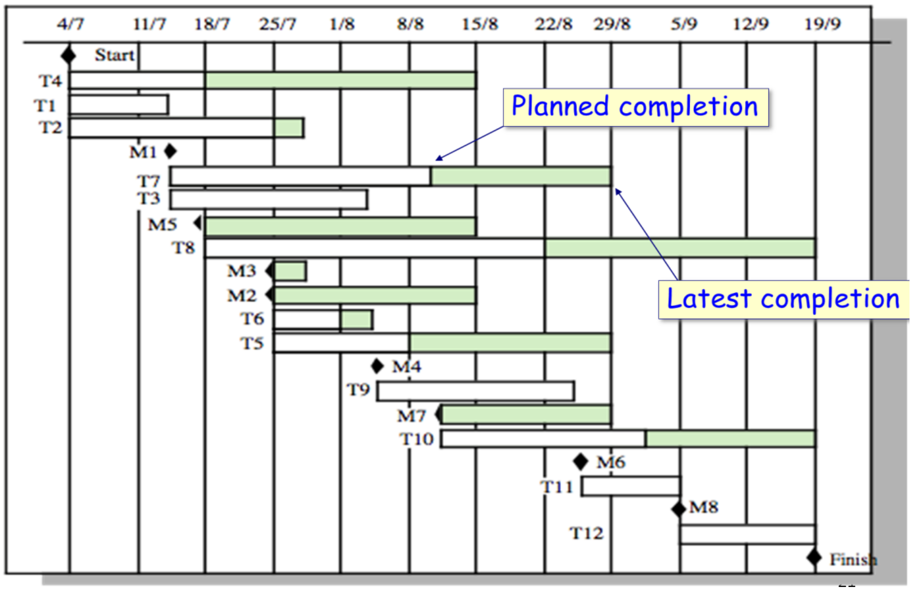
19

GANTT Charts

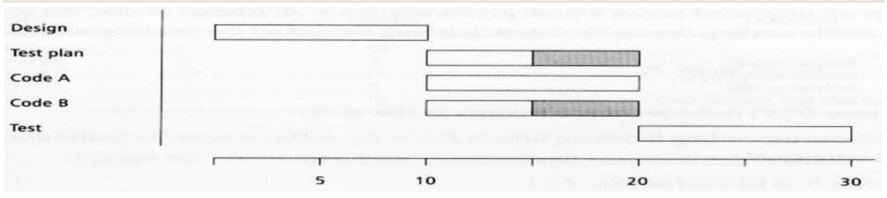
- **Characteristics:**
  - The bar in each row identifies the corresponding task
  - The horizontal position of the bar identifies start and end times of the task
  - Bar length represents the duration of the task
  - Task durations can be compared easily
  - Good for allocating resources and re-scheduling
  - Precedence relationships can be represented using arrows
  - Critical activities are usually highlighted
  - Slack times are represented using bars with doted lines
  - Bar of each activity begins at the activity earliest start time (ES)
  - The bar of each activity ends at the activity latest finish time (LF).

20

GANTT Charts: Activity TimeLine



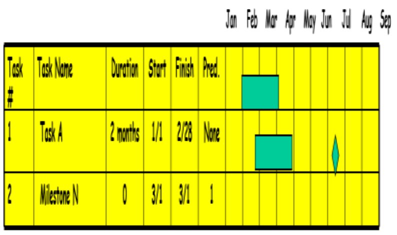
GANTT Charts



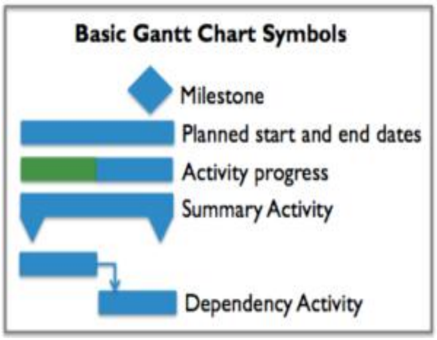
- **A graphical visualization of a *schedule*, where the time span for each activity is depicted by the length of a segment drawn on an adjacent calendar**
  - Generally does not show task decomposition
  - Does not show duration, only the time span over which the task is scheduled
  - Can show activity of multiple developers in parallel
  - Makes it easy to monitor a project's progress and expenditures

GANTT Charts

- **Horizontal bar on the calendar indicates duration of the task**
  - Multiple bars occurring at the same time interval on the calendar, implies task concurrency
  - A diamond in the calendar area of a specific task indicates that the task is a milestone; a milestone has a time duration of zero



Task #	Task Name	Duration	Start	Finish	Pred.
1	Task A	2 months	1/1	2/28	None
2	Milestone N	0	3/1	3/1	1



GANTT Charts

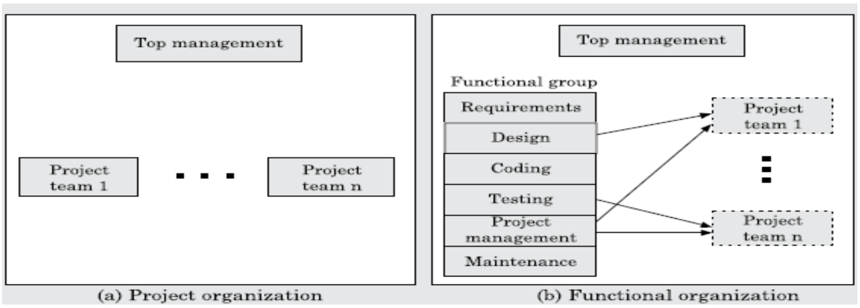
- **Limitations**
  - Can become quite unwieldy for projects with too many activities.
  - Projects are often too complex for a Gantt chart.
  - Gantt charts represent only a part of the project constraints.
- **Benefits**
  - Easy to read and comprehend
  - Easy to create
  - Identify the project network coupled with its schedule baseline
  - Supports updating and project control
  - Useful for resource planning

Organization and Team Structures

- Software development organisation handles several projects at any time.
- Software organisations assign different teams of developers to handle different software projects.
- Two important issues—How is the organisation as a whole structured (organisation structure)?
- And, how are the individual project teams structured (team structure)?

Organization and Team Structures

- **Organization Structure :** functional format, project format, and matrix format
- **Functional Format :** In the functional format, the development staff are divided based on the specific functional group to which they belong to.
- **Project Format:** In the project format, the development staff are divided based on the project for which they work.



Organization and Team Structures

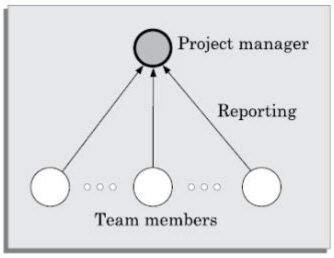
- **Matrix format :** A matrix organisation is intended to provide the advantages of both functional and project structures. In a matrix organisation, the pool of functional specialists are assigned to different projects as needed.

	Project			
Functional group	#1	#2	#3	
#1	2	0	3	Functional manager 1
#2	0	5	3	Functional manager 2
#3	0	4	2	Functional manager 3
#4	1	4	0	Functional manager 4
#5	0	4	6	Functional manager 5
	Project manager 1	Project manager 2	Project manager 3	

27

Team Structures

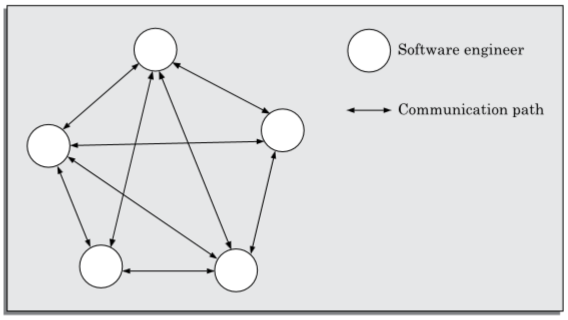
- Team structure addresses organisation of the individual project teams.
- Three formal team structures—democratic, chief programmer, and the mixed control team.
- **Chief programmer team** : In this team organisation, a senior engineer provides the technical leadership and is designated the chief programmer. The chief programmer partitions the task into many smaller tasks and assigns them to the team members.



28

Team Structures

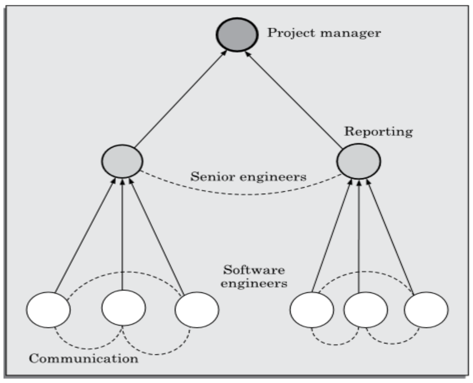
- **Democratic team :** The democratic team structure, as the name implies, does not enforce any formal team hierarchy . Typically, a manager provides the administrative leadership. At different times, different members of the group provide technical leadership.



29

Team Structures

- **Mixed control team organisation :** The mixed control team organisation, as the name implies, draws upon the ideas from both the democratic organisation and the chief-programmer organisation. This team organisation incorporates both hierarchical reporting and democratic set up



30

Risk Management

- A risk is any anticipated unfavourable event or circumstance that can occur while a project is underway.
- Risk management consists of three essential activities—risk identification, risk assessment, and risk mitigation.
- **Risk Management Approaches** : classified into reactive and proactive
- Reactive approaches : Reactive approaches take no action until an unfavourable event occurs. Once an unfavourable event occurs, these approaches try to contain the adverse effects associated with the risk and take steps to prevent future occurrence of the same risk events.

Risk Management

- Proactive approaches : The proactive approaches try to anticipate the possible risks that the project is susceptible to. After identifying the possible risks, actions are taken to eliminate the risks. If a risk cannot be avoided, these approaches suggest making plans to contain the effect of the risk.
- Risk Identification : The project manager needs to anticipate the risks in a project as early as possible. As soon as a risk is identified, effective risk management plans are made, so that the possible impact of the risks is minimised.
- There are three main categories of risks which can affect a software project: project risks, technical risks, and business risks.
- Project risks: Project risks concern various forms of budgetary, schedule, personnel, resource, and customer-related problems.

Risk Management

- Technical risks: Technical risks concern potential design, implementation, interfacing, testing, and maintenance problems.
- Business risks: This type of risks includes the risk of building an excellent product that no one wants, losing budgetary commitments, etc.
- Risk Assessment : The objective of risk assessment is to rank the risks in terms of their damage causing potential.
- Risk Mitigation : After all the identified risks of a project have been assessed, plans are made to contain the most damaging and the most likely risks first. Different types of risks require different containment procedures.
- There are three main strategies for risk containment: Avoid the risk, Transfer the risk, Risk reduction.

Software Configuration Management

- The configuration of the software is the state of all project deliverables at any point of time; and software configuration management deals with effectively tracking and controlling the configuration of a software during its life cycle.
- Necessity of Software Configuration Management
  - Inconsistency problem when the objects are replicated
  - Problems associated with concurrent access
  - Providing a stable development environment
  - System accounting and maintaining status information
  - Handling variants
- Configuration Management Activities : Configuration identification and Configuration control

35