Unit – III

Software Project Management

3.1 Responsibility of Software Project Manager

- Responsible for **accomplishing the stated project objectives**.
- The **job responsibility** → Planning to Deployment
- **Bridging gap** between the production team and client.
- **Manage constraints of project management triangle** which are Cost, Time, Scope and Quality.
- **General activities:** like project proposal writing, project cost estimation, scheduling, project staffing, software process tailoring, project monitoring and control, software configuration management, risk management, interfacing with clients, managerial report writing and presentations, etc.

- Take the **overall responsibility** of project success.
- **Above activities mainly classified in** → project planning, monitoring and controlling.
- **Risk management.**
- **Time and cost estimation.**

Skills needed for software project manager

- Project manager **take the decision** that directly affects the benefits of the projects.
- He **sets up development milestones** and entry or exit criteria.

The skills required in project manager to manage the project, are:

- Must have **theoretical-Technical knowledge**.
- A good **decision making** capabilities.
- He should be **client representative**.
- Should have **management skill**.
- Focuses on **risk management**.
- He should have **team leadership skill**.
- He should have the **experience** in the related area.

- Monitoring and scheduling capability
- **Evaluating performance** of each milestone
- Some skills like tracking and controlling the progress of the project, customer interaction, managerial presentations, and team building are **acquired through experience.**

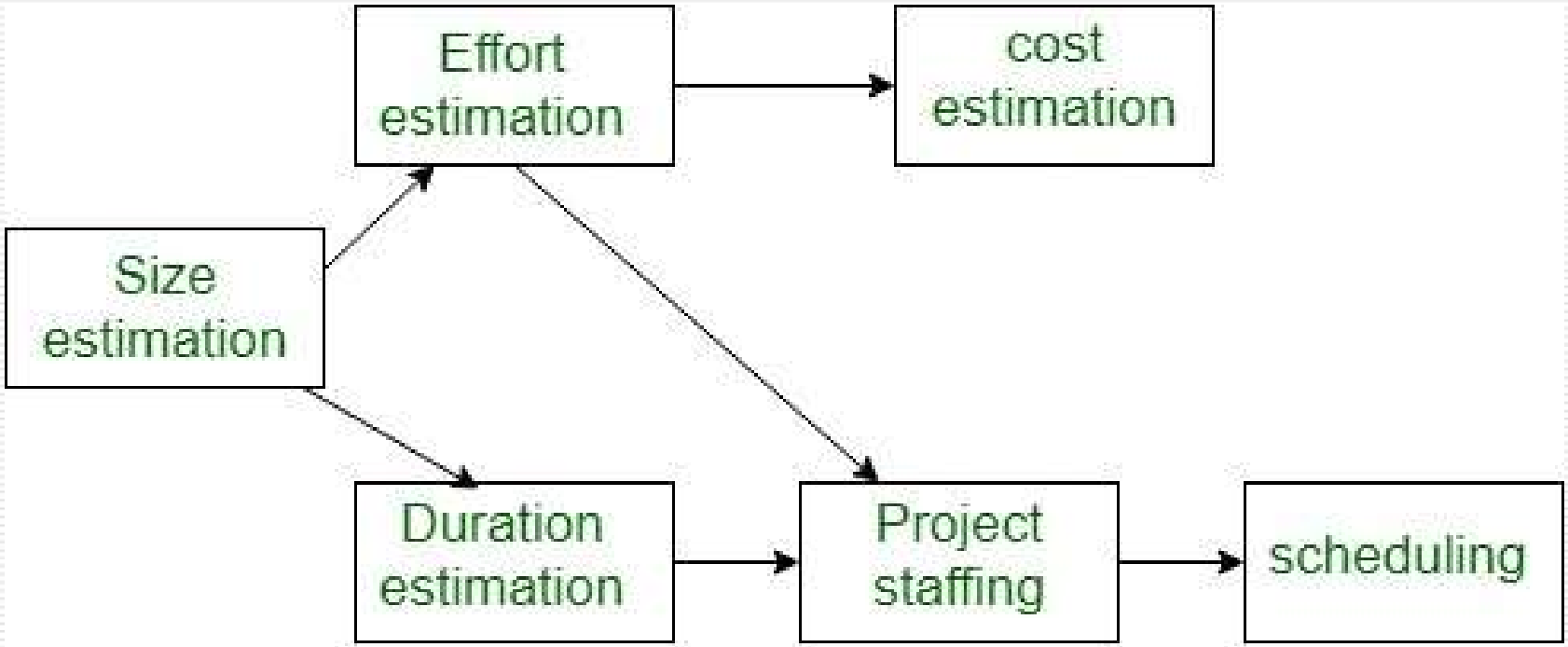
3.2 Metrics for project size estimation

Project Planning Activities

- Estimating the following attributes of the project:
  - **Project size**: What will be **problem complexity** in terms of the effort and time required to develop the product?
  - **Cost**: How much is it going to cost to develop the project?
  - **Duration**: How long is it going to take to complete development?
  - **Effort**: How much effort would be required?

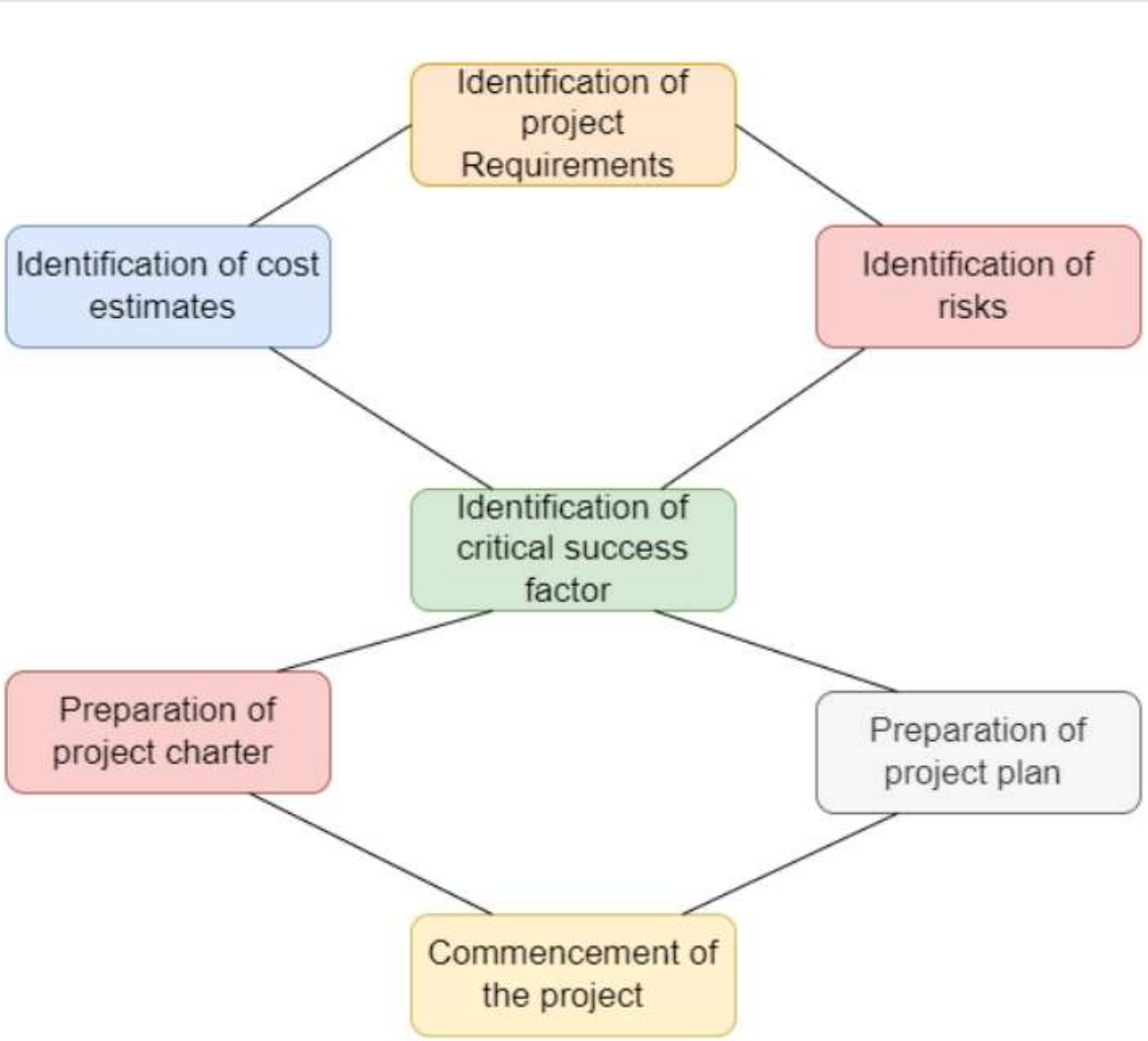
  The **effectiveness of the subsequent planning activities** is based on the accuracy of these estimations.

Project Planning



Precedence ordering among planning activities

Project Planning



Metrics for project size estimation

- The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.
- There are several metrics to measure problem size. Each of them has its own advantages and disadvantages.
- Two important metrics to estimate size:
  - ***Lines of code (LOC)***
  - ***Function point***

Lines of Code (LOC)

- Simplest and most widely used metric.
- The project size is estimated by counting the number of source instructions/lines in the developed program.
- Comments and blank lines should not be counted.
- To estimate LOC, project manager divides the problem into modules and sub modules, until leaf level modules can be estimated.

Line of Code (LOC)

- Disadvantage:
  - Size can vary with coding style.
  - Focuses on coding activity alone.
  - Correlates poorly with quality and efficiency of code.
  - Penalizes higher level programming languages, code reuse, etc.
  - Measures lexical/textual complexity only.
    - does not address the issues of structural or logical complexity.
  - Difficult to estimate LOC from problem description.

Function Point Metric (FP)

- It was first proposed by Albrecht in 1979 and internationally approve modified model in 1983.
- This metric overcomes many of the shortcomings of the LOC metric.
- Function point metrics, measure functionality from the users' point of view.
- FP is directly dependent on the number of different functions or features it supports.

Function Point Metric (FP)

- A software supporting many features or functions should have larger in size.
- FP considers five different characteristics of the product to calculate the size. The function point (FP) of given software is the weighted sum of these five items, and it will give Unadjusted Function Point (UFP).

Function Point Metric (FP)

- Once the unadjusted function point (UFP) is computed, the **Technical Complexity Factor (TCF)** is computed next.
- TCF refines the UFP by considering 14 factors which assigns 0 (no influence) to 5 (strong influence). These numbers are summed and yielding DI (Degree of Influence).
- Now TCF is computed as

TCF = (0.65 + 0.01 * DI)

- As DI can vary from 0 to 70, TCF can vary from 0.65 to 1.35.
- Finally: **FP = UFP * TCF**

Step 1

Step 1: Calculating UFP - Unadjusted Function Point

$$F.P = UFP \ X \ CAF$$

Measurement Parameter	Counts		Weighting Factor Low	Average	High		
External Inputs (EI)		X	3	4	6	=	
External Outputs (EO)		X	4	5	7	=	+
External Enquired (EQ)		X	3	4	6	=	+
Internal Logic Files (ILF)		X	7	10	15	=	+
External Interface Files (EIF)		X	5	7	10	=	+

Unadjusted Function Points (UFP) = Total Count =

UFP = Sum of all the Complexities of all the EI's, EO's, EQ's, ILF's, and EIF's

Example: SafeHome

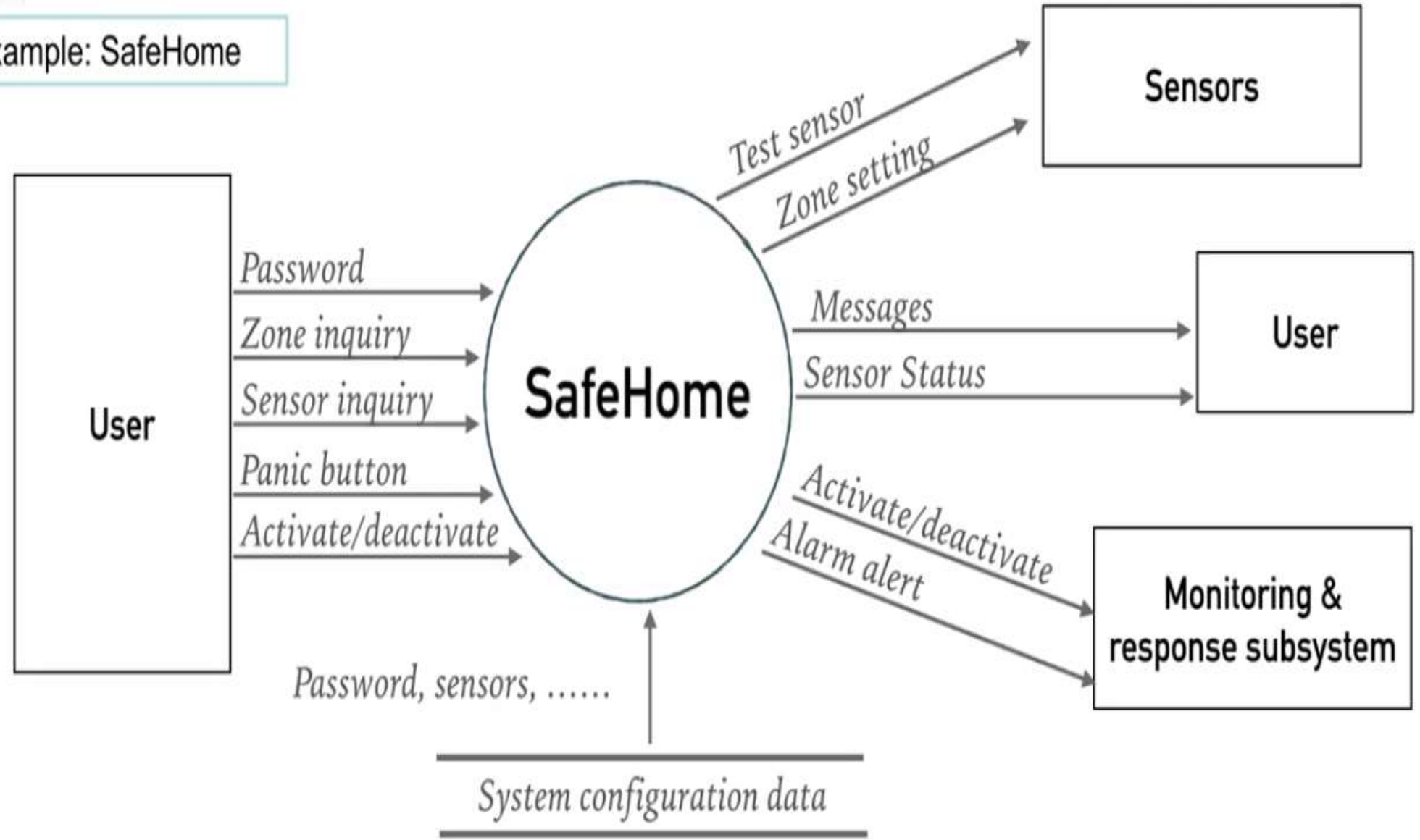


Fig: A flow model for SafeHome user interaction funcation

- *number of user inputs* - password, panic button, and activate/deactivate
- *number of user outputs* - messages and sensor status
- *number of user inquiries* - zone inquiry and sensor inquiry
- *number of files* - system configuration file
- *number of external interfaces* - test sensor, zone setting, activate/deactivate, and alarm alert

Step 1: Calculating UFP

$$F.P = UFP \times CAF$$

Measurement Parameter	Counts		Weighting Factor: Low	Average	High		
External Inputs (EI)	3	X	3			=	9
External Outputs (EO)	2	X	4			=	8
External Enquired (EQ)	2	X	3			=	6
Internal Logic Files (ILF)	1	X	7			=	7
External Interface Files (EIF)	4	X	5			=	20

Unadjusted Function Points (UFP) = Total Count = 50

UFP = Sum of all the Complexities of all the EI's, EO's, EQ's, ILF's, and EIF's

$$F.P = UFP \ X \ CAF$$

$$CAF = 0.65 + (0.01 \times \Sigma Fi)$$ Moderately complex product

Then,

$$\Sigma Fi = 14 \times 2 = 28$$ 2 - Moderate

$$CAF = 0.65 + (0.01 \times 28)$$

$$CAF = 0.93$$

$$F.P = UFP \ X \ CAF$$

$$F.P = 50 \ X \ 0.93 = 46.5$$

Step 2 CAF

Step 2: Calculating CAF - Complexity Adjustment Factor

1. Data Communication
2. Distributed Data Processing
3. Performance
4. Heavily Used Configuration
5. Transaction Role
6. Online Data Entry
7. End-User Efficiency
8. Online Update
9. Complex Processing
10. Reusability
11. Installation Ease
12. Operational Ease
13. Multiple Sites
14. Facilitate Change

Complexity Adjustment Factor is calculated using 14 aspects of processing complexity and these 14 questions answered on a scale of 0 – 5

0 - No Influences or No Important

1 - Incidental

2 - Moderate

3 - Average

4 - Significant

5 - Essential

Function Point Metric (FP)

- Advantages:
  - More accurate than LOC
  - We can measure from specification

- Disadvantages:
  - It ignores quality of output.
  - It is fully oriented to traditional data processing system.
  - Major shortcoming: it doesn't take algorithmic complexity into account.
  - To overcome this problem, an extension of the function point metric called ***feature point metric*** is proposed, which incorporates an extra parameter for algorithm complexity.
  - Not applicable universally, not good for real time software.

3.3 Project Estimation Techniques

- The **important project parameters** that are estimated include: project size, effort required to develop the software, project duration, and cost.
- There are **three main categories** of project estimation techniques:

1. **Empirical estimation techniques**
2. **Heuristic techniques**
3. **Analytical estimation techniques**

Empirical estimation techniques

- It is based on **making an educated guess** of the project parameters.
- The project managers use their **past experiences** to make guess.

Project Estimation Techniques

I. Expert judgment

- **Process:**

1. **Identify Experts:** Select individuals with relevant expertise.
2. **Consultation:** Engage in discussions or interviews to gather their insights on cost estimates.
3. **Synthesis:** Combine expert inputs to form a coherent estimate.

Project Estimation Techniques

I. Expert judgment

- **Expert makes an educated guess** of the problem size after analyzing the problem in detail.
- Expert **estimates the costs of different modules (or units)** then combined them for overall estimation.
- But in some situation, an **expert may not have knowledge** of all aspects of project or he may overlook some factors.

Project Estimation Techniques

- So, the **chance of human error or individual bias** should be there.
- **To solve the above problem** →a more refined form of expert judgment is the estimation made by **group of experts**. It should minimize the factors of above problem.
- Even after this, **chance of biasing is there.**

Project Estimation Techniques

I. Expert judgment

- **Advantages:**
- **Quick Implementation:** Can be conducted rapidly without extensive processes.
- **Flexibility:** Adaptable to various project contexts and stages.
- **Disadvantages:**
- **Subjectivity:** Estimates may be influenced by individual biases or limited perspectives.
- **Lack of Anonymity:** Experts may be influenced by group dynamics or dominant opinions.

Project Estimation Techniques

II. Delphi cost estimation

- It provides solution to the shortcomings of the expert judgment approach.
- Delphi estimation is carried out by a **team having a group of experts (estimators) and a coordinator.**
- In it, the coordinator provides each estimator with a copy of the software requirements specification (SRS) document and a form for recording his cost estimate.
- Estimators complete their individual estimates anonymously (unidentified) and submit to the coordinator.

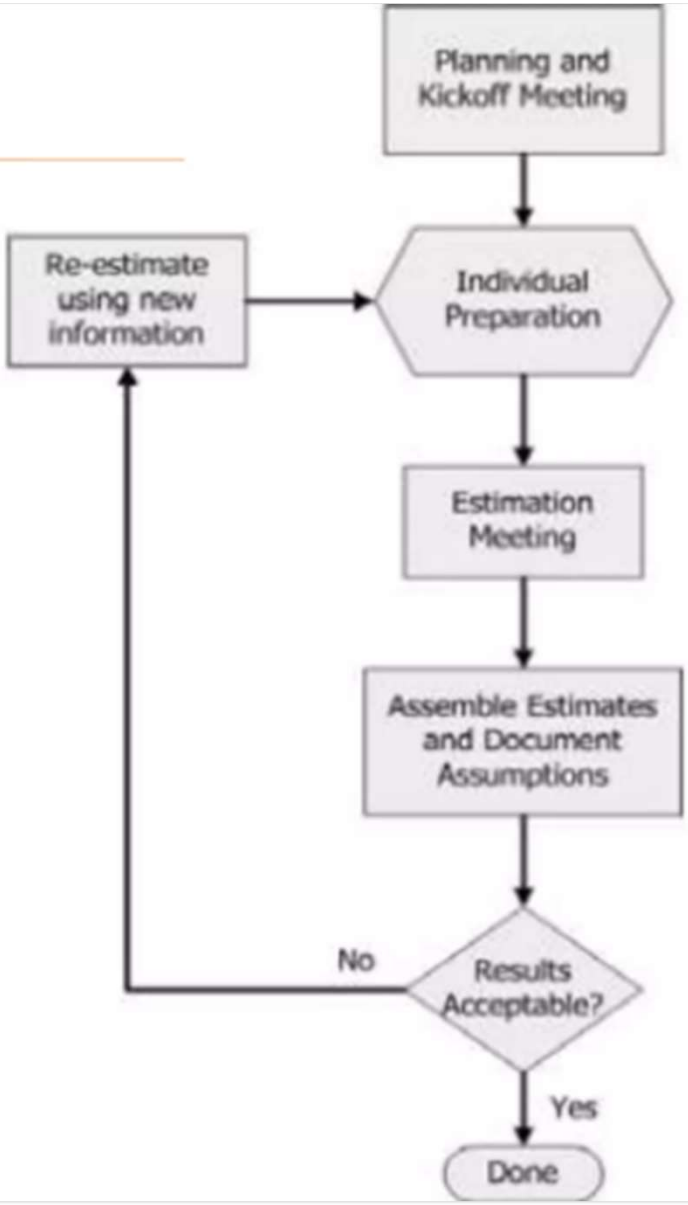
Project Estimation Techniques

- The estimators mention any unusual characteristic of the product/project in their estimation report.
- The **coordinator prepares summary** and allocate to estimators.
- Based on this summary, the estimators re-estimate.
- This process is **iterated for several rounds**.
- **Discussion** among the estimators is **not allowed** during the entire estimation process.
- After the completion of several iterations, coordinators prepares the final estimate.

Project Estimation Techniques

Wideband Delphi Steps

- *Team Selection:* Project Manager selects a moderator & an estimation team with 3 to 7 members.
- *Kickoff Meeting:* The first meeting during which estimation team creates a WBS and discusses assumptions.
- *Individual Preparation:* After the meeting, each team member creates an effort estimate for each task.
- *Estimation Session:* The second meeting in which the team revises the estimates as a group and achieves consensus.
- *Assemble Tasks:* After the estimation session, the project manager summarizes results and reviews them with team
- *Review Results:* Review the results that have come out from the Estimation Session.



Project Estimation Techniques

Team Selection

- Picking qualified team is important part of generating accurate estimates.
- TMs must be willing to estimate each task honestly, and should be comfortable working with rest of the team.
- Should be knowledgeable about organization's needs and past engineering projects to make educated estimates.
- Team should include representatives from each areas of development team: managers, devs, designers, architects, QA, analysts, technical writers, etc
- Moderator should be familiar with the Delphi process, but should not have a stake in the outcome of the session
- PM should avoid Moderator role - should be part of estimation team
- One or more observers - selected stakeholders, users, and managers should be encouraged to attend the meeting

Project Estimation Techniques

Kickoff Meeting

- TMs are given vision, scope & other docs.
- A goal statement for estimation session should be agreed upon by project manager and moderator and distributed to the team before the session.
- Should be no more than a few sentences that describe the scope of the work that is to be estimated
- Ex: Generate estimates for programming and testing the first phase of RedRock Project
- Moderator leads the meeting

Project Estimation Techniques

Kickoff Meeting (Cont'd)

- Meeting consists of these activities -
  - Moderator explains the Wideband Delphi method to any new estimators.
  - If any TM has not yet read vision & scope and supporting docs, the moderator reviews it with the team.
  - Moderator reviews goal of estimation session with team, and checks that TMs are knowledgeable to contribute.
  - Discusses product being developed & brainstorms assumptions.
  - Generates a task list consisting of 10 – 20 major tasks. These tasks represent the top level of work breakdown structure.
  - Agrees on the units of estimation (days, weeks, pages)

Project Estimation Techniques

Individual Preparation

- After kickoff meeting, moderator writes down assumptions and tasks generated by team & distributes them.
- TMs independently generates a set of *preparation results*, that contains
  - Estimate for each of the tasks
  - Any additional tasks that should be included in WBS missed by Team during kickoff meeting.
  - Any assumptions that TM made in order to create the estimates
  - Any effort related to project overhead (status meetings, reports, vacation, etc) should NOT be taken into account. Should be added to the "Project overhead tasks" section.
  - Potential delays, (like certain tasks can't start until after specific dates) NOT be taken into account. Should be added to the "Calendar waiting time" section.
- Each estimate should be made in terms of effort, not calendar time.

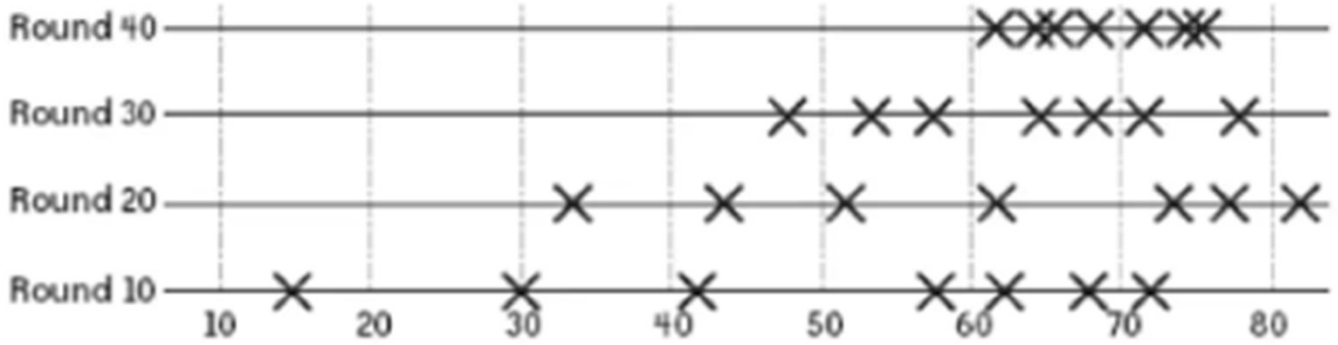
Estimation Form

Name	Mike			Date 4/3/2004			Estimation form 1/1
Goal statement	To estimate the time to develop prototype for customers A & B						Units days
Category	☑ goal tasks ☑ quality tasks ☐ waiting time ☐ project overhead						

WBS# or priority	Task name	Est.	Delta 1	Delta 2	Delta 3	Delta 4	Total	Assumptions
1	Interview customers (A+B)	3	+2	+1				Needs off-site tri
2	Develop requirements docs	6	+5	-2	+1			Start from scratc
3	Inspect requirements docs	1	+2	+2	-2			Team of 4 BSAs
4	Do rework	1	+4					
5	Prototype design	20	-3	4	-2			Includes DB
6	Test design	5	+3					20% exists now

Estimation Session

- Meeting consists of these activities -
  - Moderator collects all estimate forms. Estimates are tabulated on whiteboard by plotting the totals on a line
  - Estimators read out clarifications & changes to task list written on estimation form. New or changed tasks, discovered assumptions, or questions are raised. Specific estimate times are NOT discussed.
  - Team resolves issues or disagreements. Since individual estimate times are not discussed, these disagreements are usually about the tasks themselves, and are often resolved by adding assumptions.
  - Estimators revise their individual estimates by filling in "Delta" column on their forms.



Assemble Tasks

- Project Manager works with Moderator to gather all results from individual preparation and estimation session.
- PM removes redundancies and resolves remaining estimate differences to generate a final task list, with effort estimates
- The assumptions are summarized and added to list. The Visio doc and other docs are updated with assumptions.
- PM should create spreadsheet that lists final estimates that each person came up with. The spreadsheet should indicate the best-case and worst-case scenarios,
- Any task with an especially wide discrepancy should be marked for further discussion.
- Final task list should be in same format as individual preparation results.

Project Estimation Techniques

Summarized Estimation Results

Goal statement	To estimate the time to develop prototype for customers A & B								
Estimators	Mike, Quentin, Jill, Sophie							**Units**	days
									Shaded items must be discussed
WBS# or priority	**Task name**	**M.**	**Q.**	**J.**	**S.**	**Best-case**	**Worst-case**	**Avg.-hi & lo**	**Notes**
1	Interview customers (A+B)	6	4	3	3	3	6	3.5	
2	Develop requirements docs	5	10	2	5	2	10	5	Discrepancy between Q. and J.
3	Inspect requirements docs	7	5	6	5	5	7	5.5	
4	Do rework	8	7	9	7	7	9	7.5	
5	Prototype design	28	23	31	25	23	31	26.5	
6	Test design	9	7	6	6	6	9	6.5	
	Total	63	56	57	51	46	72	54.5	

Project Estimation Techniques

Review Results

- Once results are ready PM calls a final meeting to review the estimation results with the team.
- Goal of meeting is to determine whether the results of the session are sufficient for further planning. The team should determine whether the
- Estimates make sense and if the range is acceptable. They should also examine the final task list to verify that it's complete. There may be an area that needs to be refined:
  - For example, a task might need to be broken down into subtasks. In this case, the team may agree to hold another estimation session to break down those tasks into their own subtasks and estimate each of them.
  - This is also a good way to handle any tasks that have a wide discrepancy between the best-case and worst-case scenarios.

Project Estimation Techniques

Pros of Delphi Cost Estimation

1.Reduces Bias – Since experts provide anonymous estimates, there's less influence from dominant personalities or organizational politics.
2.Encourages Expert Input – Uses the knowledge of multiple experts, increasing the accuracy of estimates.
3.Iterative Refinement – Multiple rounds of estimation allow refinement based on feedback, leading to better accuracy.
4.Applicable to Complex Projects – Works well for projects with high uncertainty, where historical data is limited.
5.Encourages Consensus – The process naturally drives experts toward agreement, improving reliability.

Project Estimation Techniques

Cons of Delphi Cost Estimation

1.Time-Consuming – Multiple rounds of discussion and revision can take a long time.
2.Resource-Intensive – Requires participation from multiple experts, which can be costly.
3.Dependence on Expert Availability – Accuracy depends on the knowledge and experience of selected experts.
4.Lack of Quantitative Rigor – The method relies on expert judgment rather than strict mathematical models, which can lead to subjectivity.
5.Difficulty in Handling Large-Scale Estimates – Managing many experts and rounds of estimation can become complicated for very large projects.

Heuristic estimation techniques

- Mathematical techniques are used in this techniques.
- Independent and dependent variables.
- **Static single variable model**:
  - A single variable estimation model takes the following form:

$$\text{Estimated Parameter} = c1 * e^{d1}$$

  - Example: basic COCOMO model.
- **Static multi variable model:**
  - The multivariable cost estimation model takes the following form:

$$\text{Estimated Resource} = c1*e1^{d1} + c2*e2^{d2} + \ldots$$

  - It gives more accurate estimation compare to single variable model.
  - Example: intermediate COCOMO model.

Heuristic estimation techniques

- Boehm classified a software into one of the following three categories based on the development complexity:
  - Organic
  - Semi detached
  - Embedded
- **Organic**
  - Small group of team.
  - Develop well understood application programs and having experienced developers.
  - Little or no innovation.
  - Project size should be 2-50 KLOC.
  - For example data processing programs, payroll, inventory management system.

Heuristic estimation techniques

- **Semidetached**
  - Consists of a mixture of experienced and inexperienced staff.
  - Team members may have limited experience.
  - Medium innovation.
  - Project size should be 50-300 KLOC.
  - For example compilers, interpreters, assemblers.

- **Embedded:**
  - Strongly coupled to complex hardware.
  - Tight or inflexible (rigid) regulations.
  - Great deal with innovation and requirements constraints, and strict deadlines.
  - Large project size and development team, few experience.
  - Project size is approximately over 300 KLOC.
  - Example: air traffic control, ATMs

Heuristic estimation techniques

Development Mode	Project Characteristics			
	Size	**Innovation**	**Deadline**	**Development Environment**
Organic	Small	Little	Not tight	stable
Semi-Detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex h/w

COCOMO (A heuristic estimate techniques)

- COCOMO (COnstructive COst estimation MOdel) was proposed by Boehm.
- It is regression based model provides hierarchy of software cost estimation models.
- Three stages:
  - Basic COCOMO,
  - Intermediate COCOMO, and
  - Complete COCOMO.

Basic COCOMO

- Single valued model, computing s/w development efforts.
- It is quick and rough estimation technique.
- It gives an approximate estimate of the project parameters.

- The basic COCOMO estimation model is given by the following expressions:

$$\textbf{Effort (E)} = a1 * (KLOC)^{a2} \quad PM$$

$$\textbf{Tdev} = b1 * (Effort)^{b2} \quad Months$$

Basic COCOMO

- Estimation of development effort in basic COCOMO model.

Organic	:	Effort (E) = 2.4 $(KLOC)^{1.05}$	PM
Semidetached	:	Effort (E) = 3.0 $(KLOC)^{1.12}$	PM
Embedded	:	Effort (E) = 3.6 $(KLOC)^{1.20}$	PM

- Estimation of development time in basic COCOMO model

Organic	:	T_{dev} = 2.5 $(effort)^{0.38}$	Months
Semidetached	:	T_{dev} = 2.5 $(effort)^{0.35}$	Months
Embedded	:	T_{dev} = 2.5 $(effort)^{0.32}$	Months

Basic COCOMO

- Example: Assume that the size of an organic type software product has been estimated to be 32,000 lines of source code. Assume that the average salary of software engineers be Rs. 15,000/- per month. Determine the effort required to develop the software product and the nominal development time.

Effort = $2.4 * (32)^{1.05}$ PM

Development Time = $2.5 * (38.05)^{0.38}$ = 3.98

= 4 Months

Person needed=Effort/D.Time=38.05/3.98=9.56

Cost of Development = 9.56 * 15,000 =1,43,400 Rs.

Intermediate COCOMO

- The basic COCOMO model considers efforts and time development are the alone functions of product size.
- But some factors from other projects may also affect the efforts and time.
- The intermediate COCOMO model doing this by using a set of **15 cost drivers (multipliers)** based on various attributes of software development.
- The Intermediate COCOCMO model consumes these "cost drivers"
- Project manager doing the task of rating these 15 parameters in scale 1 to 3 and then cost driver values are estimated.

Intermediate COCOMO

- The cost drivers can be grouped into four categories.

(i) Product attributes –
Required software reliability extent
Size of the application database
The complexity of the product

(ii) Hardware attributes –
Run-time performance constraints
Memory constraints
The volatility of the virtual machine environment
Required turnabout time

(iii) Personnel attributes –
Analyst capability
Software engineering capability
Applications experience
Virtual machine experience
Programming language experience

(iv) Project attributes –
Use of software tools
Application of software engineering methods
Required development schedule

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

Type 2: Intermediate COCOMO Model Example

❑Problem Statement:

- For a given Semidetached project was estimated with a size of 300 KLOC. Calculate th Effort, Scheduled time for development by considering developer having high applicatio experience and very low experience in programming.

Complete COCOMO

- It is also known as detailed COCOMO model.
- A major shortcoming of both the basic and intermediate COCOMO models is that they consider a software product as a single homogeneous entity.
- But most large systems are made up several smaller sub-systems.
- For example, some subsystems may be considered as organic type, some semidetached, and some embedded.
- And these subsystems may be differ in requirements, development complexities and may not have previous experience of similar developing.

Complete COCOMO

- Detailed COCOMO model - incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.
- It estimates the effort and development time as the sum of the estimates for the individual subsystems.
- The cost of each subsystem is estimated separately.
- This approach reduces the margin of error in the final estimate.
- For example a distributed Management Information System (MIS) consists of different categories of the software.

COCOMO Model

- Advantages:
  - It is easy to use.
  - It is repeatable process.
  - It is versatile enough.
  - It works well on projects that are not so different in size , process and complexity.
  - It is highly standardized, based on previous experience.
  - It can be detailed documented.

- Disadvantages:
  - Not accurate and not good for scientific justification.
  - It ignores software safety issues.
  - It ignores personal turnover levels.
  - All the levels of COCOMO are dependent on size estimates.
  - It also ignores many hardware issues.

Scheduling

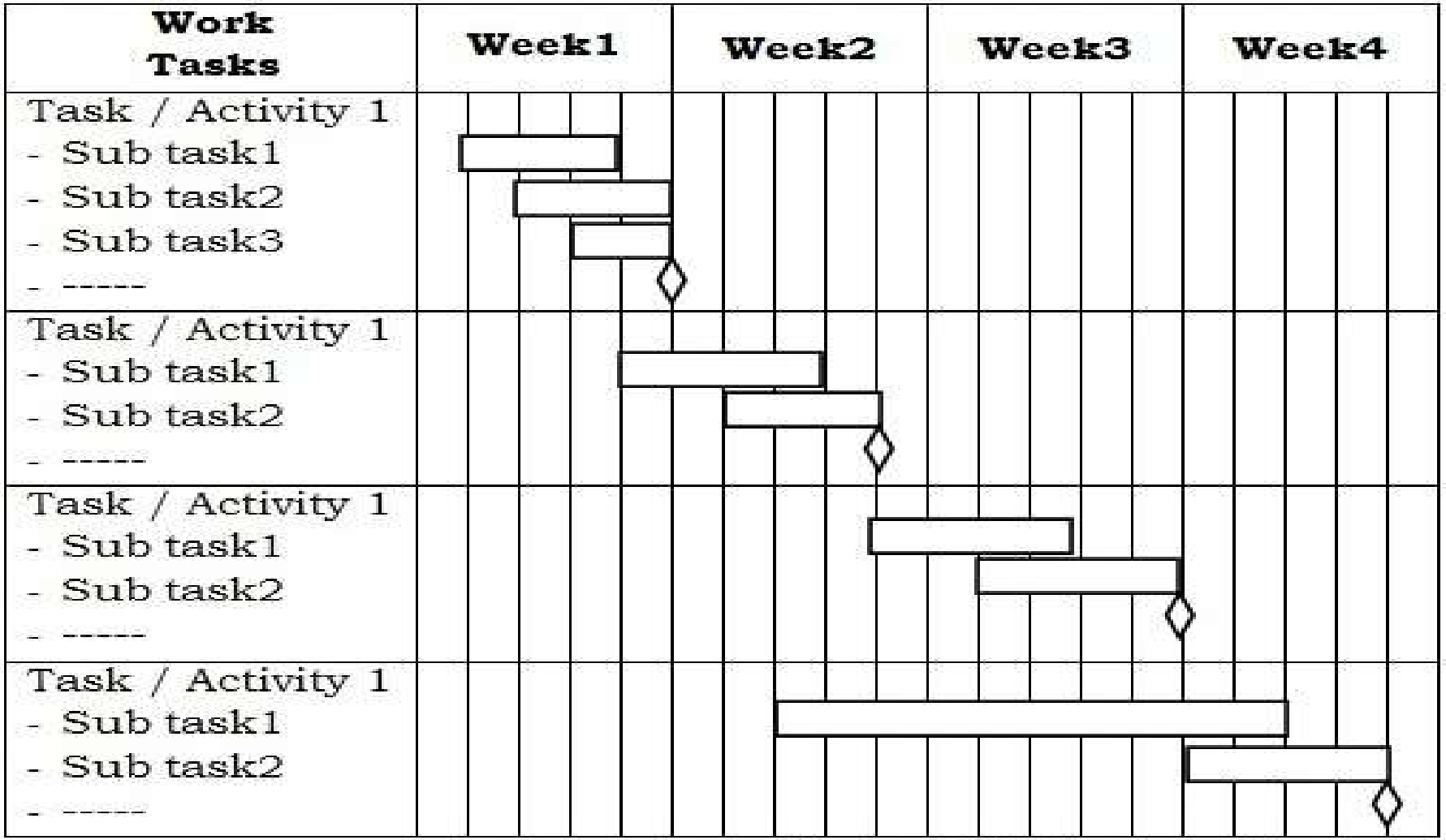
- It is important project planning activity.
- In order to schedule the project activities, a software project manager needs to do the following:
  - Identify all the tasks needed to complete the project.
  - Break down large tasks into small activities.
  - Determine the dependency among different activities.
  - Establish the most likely estimates for the time durations necessary to complete the activities.
  - Allocate resources to activities.
  - Plan the starting and ending dates for various activities.
  - Determine the critical path.

Project Scheduling:GANTT Chart

- It was proposed by Henry Gantt in 1914, also called **time line chart.**
- It is mainly used to show allocation of resources to activities (Resource Planning).
- The resources allocated to activities include staff, hardware, and software etc.
- A Gantt chart is a special type of bar chart where each bar represents an activity. Bars are drawn along time line. Length of bar proportional to the duration of time planned.
- It shows activities against time.
- Slack time is also shown in the bars.
- The chart is prepared by the project manager.

GANTT Chart

- GANTT Chart example:



Diamond indicates milestones.

GANTT chart example.

GANTT Chart

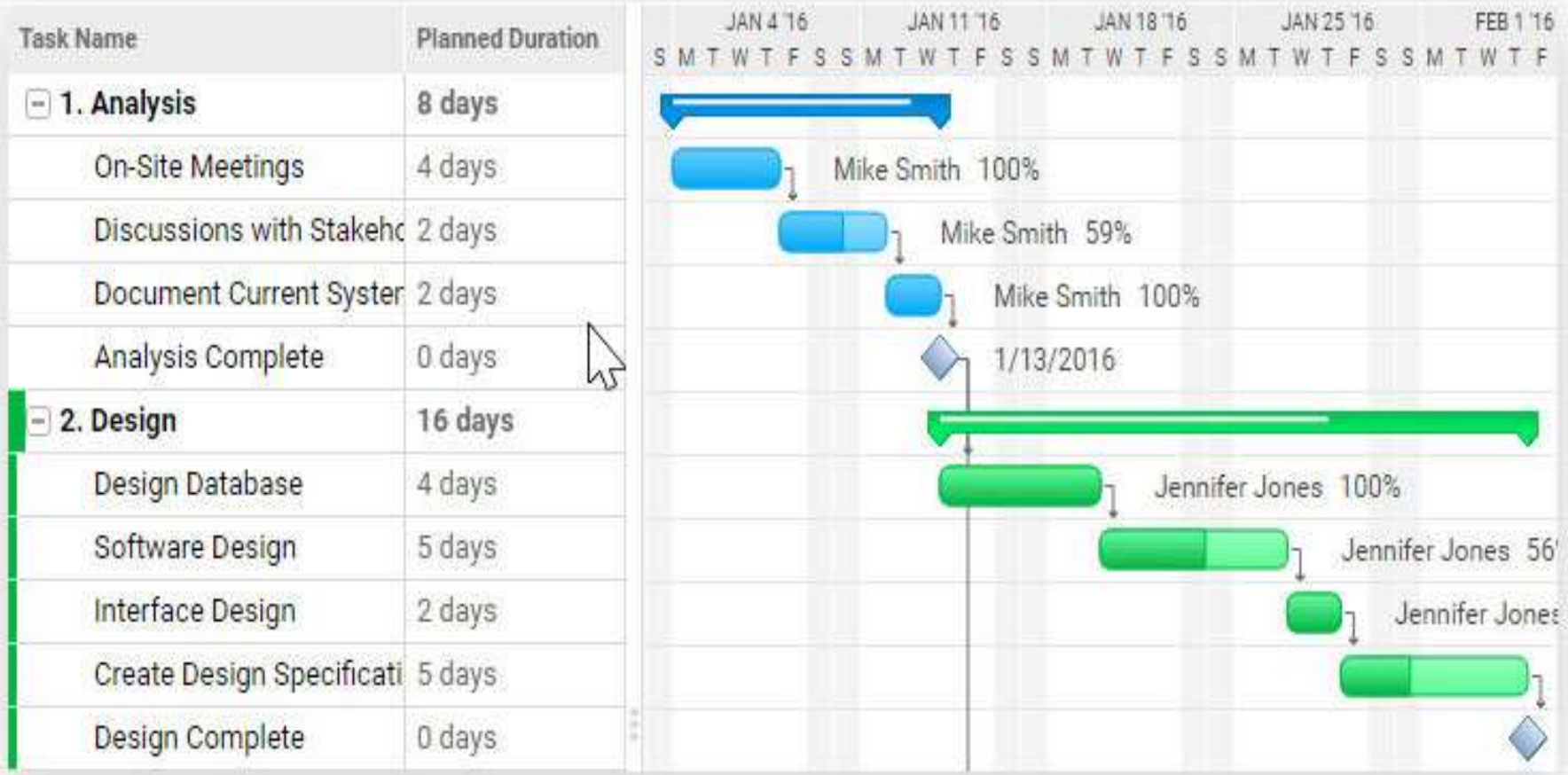
- How to plan GANTT chart:
  - Identify all the tasks.
  - If possible, break down the tasks into smaller tasks.
  - Determine the total estimated completion time for each task.
  - Plot activities on GANTT chart. Draw milestones at applicable places.

On a Gantt chart you can easily see:

- The **start date** of the project
- What the **project tasks** are
- **Who is working** on each task
- When tasks **start and finish**
- How long each task will take**(Duration)**
- How tasks group together, **overlap and link** with each other
- The **finish date** of the project

GANTT Chart

- GANTT Chart example:



GANTT Chart

✅ **Advantages of Gantt Charts**

1️⃣ **Clear Project Visualization**

•Provides a **timeline-based view** of tasks, making it easy to track progress.

2️⃣ **Task Dependencies are Clear**

•Shows **which tasks depend on others**, helping teams avoid scheduling conflicts.

3️⃣ **Improves Time Management**

•Helps allocate **deadlines and milestones** efficiently.

4️⃣ **Resource Allocation**

•Teams can see **who is responsible** for each task and balance workloads.

5️⃣ **Easy Progress Tracking**

•**% Completion indicators** show how much work has been done.

6️⃣ **Enhances Communication**

•Stakeholders and team members get a **shared view** of the project status

❌ **Disadvantages of Gantt Charts**

1️⃣ **Complexity for Large Projects**

•As projects grow, **Gantt charts can become cluttered and hard to manage**.

2️⃣ **Time-Consuming to Update**

•Requires frequent updates to **reflect real-time progress**.

3️⃣ **Doesn't Show Detailed Task Breakdowns**

•Focuses more on **timelines** rather than deep task dependencies.

4️⃣ **Limited Adaptability in Agile Projects**

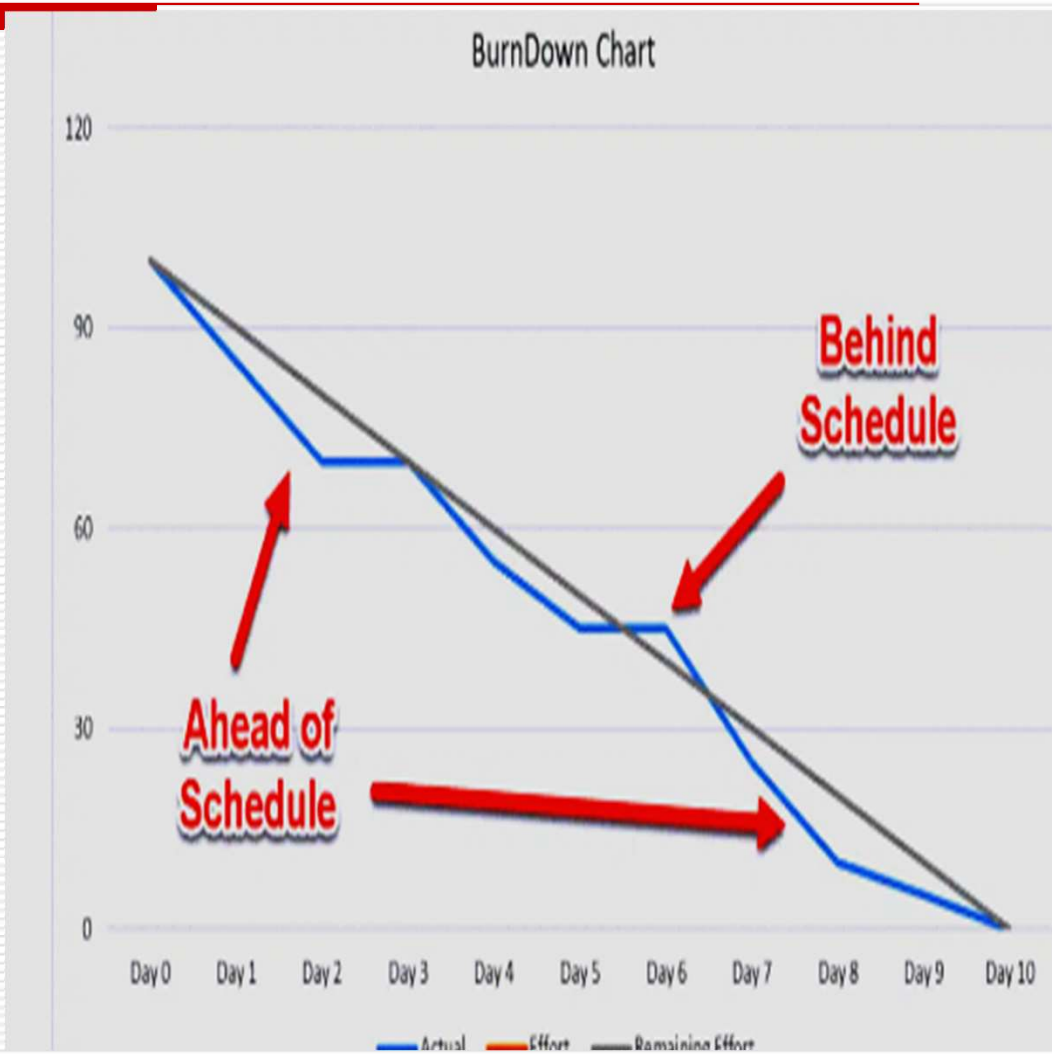
•Not ideal for **fast-changing** Agile environments

5️⃣ **Risk of Over-Reliance**

•Teams may become **too focused on timelines** instead of quality and innovation

Sprint BurntDown Chart

- **visual representation of the remaining work versus the time required** to complete it.
- used to efficiently calculate whether your team has enough time to complete their work and is commonly used while working in short iterations
- X-Axis: Sprints or time
- Y-Axis: Remaining effort
- Actual effort line
- Ideal effort line



Sprint BurntDown Chart

- **How to Draw:**
- Step 1: Estimate the effort required

  For example, let's say your ideal baseline is to complete your sprint in 5 days with 80 hours of work. That equates to 16 hours of work per day. You would then begin your effort trajectory at 80 (representing 80 hours) and track your effort for the remaining days. This would look something like this:

- Day 1: 80 hours of work
- Day 2: 64 hours of work
- Day 3: 48 hours of work
- Day 4: 32 hours of work
- Day 5: 16 hours of work

Sprint BurntDown Chart

- **How to Draw:**
- Step 2: Track daily progress and remaining work
- Once you have your estimates, you can begin tracking your daily progress. This can be done with a simple chart or timeline tool.
- Step 3: Calculate actual effort spent
- Step 4: Compile the final dataset
- Step 5: Plot the burndown chart

Sprint BurntDown Chart

Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
100	90	80	70	60	50	40	30	20	10	0

The daily planned progress is recorded in the table below.

Task	Hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Total
Task 1	10	2	1	1	0	2	1	0	1	0	2	10
Task 2	10	2	1	1	0	2	1	0	1	0	2	10
Task 3	10	2	1	1	0	2	1	0	1	0	2	10
Task 4	10	2	1	1	0	2	1	0	1	0	2	10
Task 5	10	2	1	1	0	2	1	0	1	0	2	10
Task 6	10	2	1	1	0	2	1	0	1	0	2	10
Task 7	10	2	1	1	0	2	1	0	1	0	2	10
Task 8	10	2	1	1	0	2	1	0	1	0	2	10
Task 9	10	2	1	1	0	2	1	0	1	0	2	10
Task 10	10	2	1	1	0	2	1	0	1	0	2	10

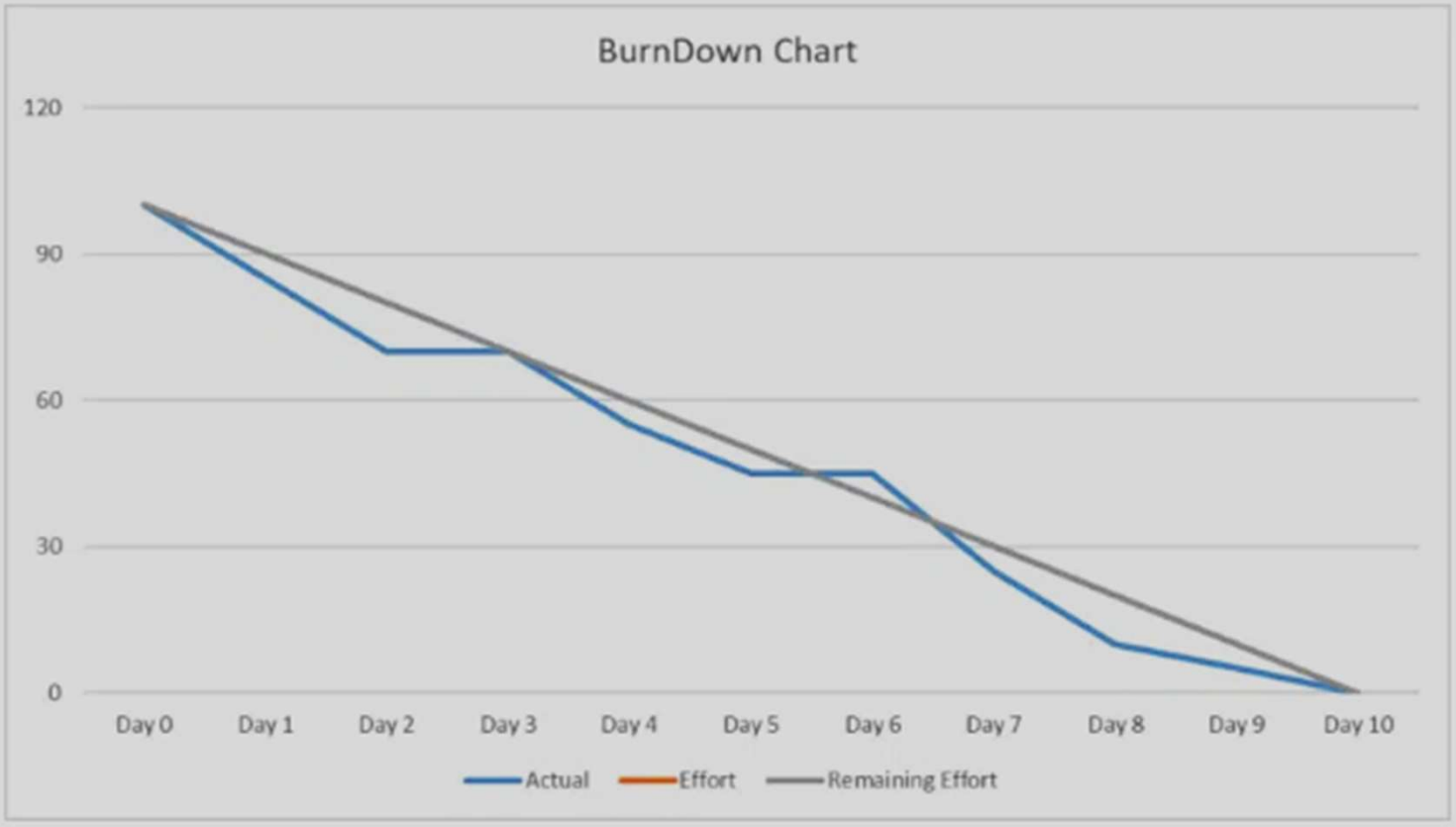
Sprint BurntDown Chart

The remaining effort is captured at the end of the day. The effort caught is the total (sum) of the calculated or estimated time remaining at the end of each day.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11
Actual Effort	100	90	80	70	60	50	40	30	20	10	0
Remaining Effort	95	85	70	65	55	45	30	20	15	5	0

Sprint BurntDown Chart

Plotting BurnDown Chart



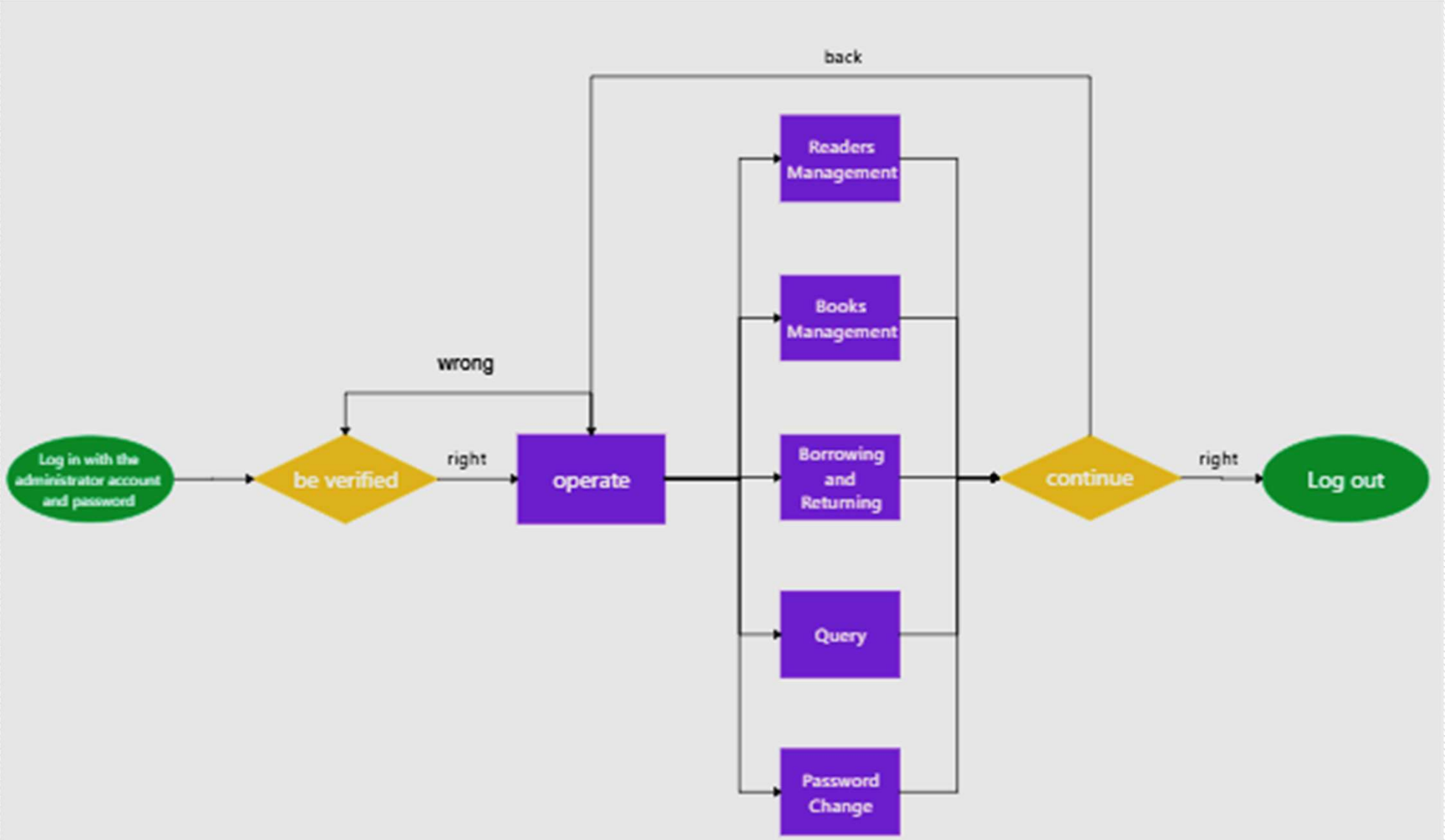
Sprint BurntDown Chart:Advantages

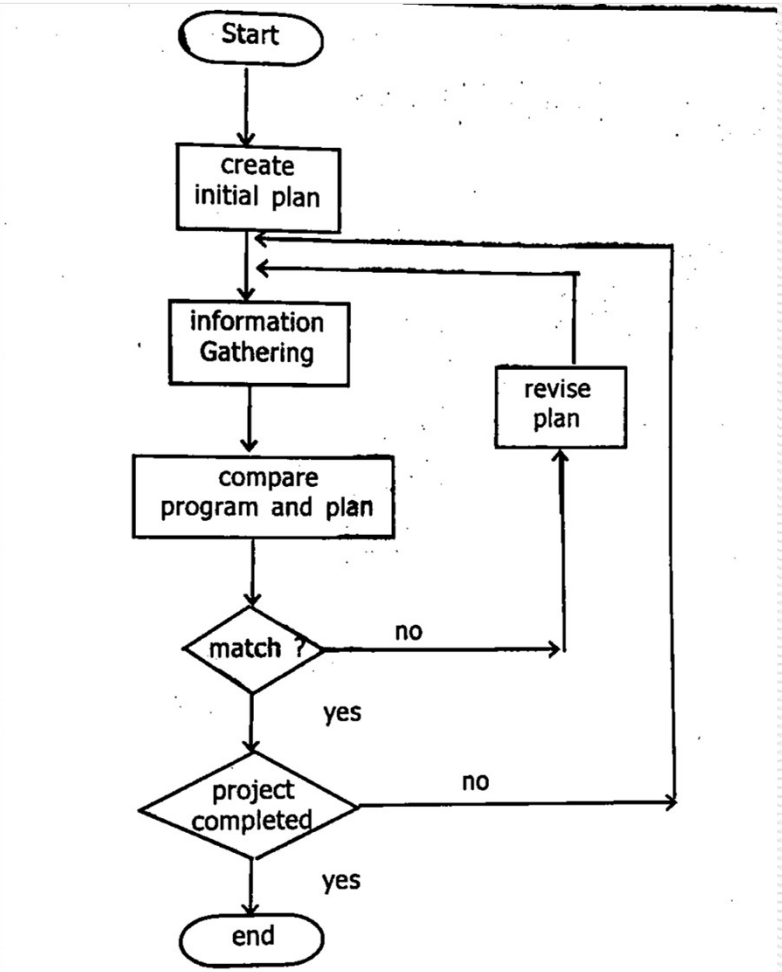
- **Clear Progress Tracking** – Shows how much work is completed and what remains, giving teams a clear picture of sprint status.
- **Motivates the Team** – Seeing progress visually can boost team morale and encourage focus on completing tasks.
- **Helps in Sprint Planning** – Provides insights into team velocity, helping improve future sprint planning.
- **Encourages Transparency** – Stakeholders and team members can easily understand sprint progress without needing long explanations.
- **Early Problem Detection** – Identifies if the team is behind schedule, allowing adjustments before the sprint ends.
- **Identifies Scope Creep** – Any unexpected increase in work (upward trend) signals scope creep that needs to be managed.

Sprint BurntDown Chart:Disadvantages

- **Doesn’t Show Task Complexity** – Only tracks work completed, not how difficult or time-consuming the remaining tasks are.
- **No Quality Indicator** – Completing tasks quickly doesn’t mean they’re done well; technical debt isn’t visible.
- **Doesn’t Show Individual Contributions** – It focuses on overall team progress, not individual performance, which may be a drawback for some managers.
- **Potential for Misinterpretation** – Stakeholders unfamiliar with Agile might panic if they see little progress initially, even though sprints are often front-loaded with research and planning.
- **Can Be Misleading** – A flat line early in the sprint might indicate slow progress, but work could be getting done in larger chunks later.
- **Can Create Unnecessary Pressure** – Teams may rush to "burn down" points rather than focusing on delivering value properly.

Flowchart


back
Readers Management
Books Management
wrong
Log in with the administrator account and password
be verified
right
operate
Borrowing and Returning
continue
right
Log out
Query
Password Change


Start
create initial plan
information Gathering
compare program and plan
match ?
no
revise plan
yes
project completed
no
yes
end

Flowchart:Advantages

- **Clear Visualization** – Helps stakeholders easily understand the project flow, dependencies, and task order.
- **Identifies Dependencies** – Shows which tasks depend on others, preventing scheduling conflicts.
- **Enhances Communication** – Team members and managers can quickly grasp the scheduling process without reading long documents.
- **Aids in Problem Solving** – Helps identify bottlenecks and optimize resource allocation.
- **Improves Efficiency** – A well-structured flowchart can streamline project execution by reducing confusion.
- **Better Monitoring & Control** – Makes it easy to track progress and adjust schedules dynamically.
- **Supports Automation** – Can be integrated into project management tools like Gantt charts, PERT charts, and scheduling software.

Flowchart:Disadvantages

- **Time-Consuming to Create** – Designing a detailed flowchart takes effort, especially for large projects.
- **Complexity in Large Projects** – As projects grow, flowcharts can become too detailed, making them difficult to read and manage.
- **Not Easily Modifiable** – If schedules change frequently, updating a flowchart can be tedious compared to using dynamic scheduling tools like Jira or MS Project.
- **Limited Detail on Time & Resources** – Flowcharts focus on process flow but may not effectively show exact time estimates or resource utilization.
- **Doesn't show milestones**– Unlike gantt Chart it does not show milestones
- **Does not show responsibilities**

3.1 Risk Management

Risk Management

- *Today's risk is Tomorrow's problem.*
- Software risk is a problem that could cause some loss or threaten the success of software project, but which hasn't happened yet.
- This risk may affect negatively to the cost, schedule, technical success or quality of the project.
- **Risk management is the process of identifying, addressing and eliminating these problems before they can damage the project.**
- Objectives of the risk management:
  - Identify potential problems and deal with them when they are easier to handle before they become critical.
  - Focus on the project's objectives and consciously look after.
  - Allow early identification of risks and provide management.
  - Increase the chance of project success.

Risk Management

- Risk Management Activities:

Risk Assessment

- It is the process of examining a project and identifying areas of potential risk.
- It includes the following activities:
  - Risk identification
  - Risk analysis
  - Risk prioritization
- **Risk identification**
  - It is a systematic attempt to specify threats of the project plan.
  - **Purpose**: to develop list of risk items also called risk statement
  - Some common risks areas are found and **checklist is prepared**.
  - Categorize risks into different classes.
  - The project manager can then examine which risks from each class are relevant to the project.

Risk Assessment

- There are **three main categories of risks**:
  1. **Project risks**: budgetary, schedule, personnel, resource etc. Schedule slippage.
  2. **Technical risks**: Risks that threaten the quality of the product. design, implementation, interfacing, testing, and maintenance problems etc. Most technical risks occur due to improper knowledge.
  3. **Business risks:** Risks that threaten the development (or client) organization. Include risks of building an excellent product that no one wants, losing budgetary or personnel commitments, etc.

Risk Assessment

- **Risk analysis**
  - When the risks have been identified, all risks are analyzed using different criteria.
  - The purpose of this activity is to examine how project outcomes might change with modification of risk input variables.
  - The input of this activity is the list of risks developed in risk identification and output is the ranking of the risks.
  - Main activities of this process are:
    - Group similar risks
    - Determine risk drivers
    - Determine source of risks
    - Estimate risk exposure
    - Evaluate against criteria

Risk Assessment

- **Risk prioritization**
  - It focuses on its most severe risks by assessing the risk.
  - Risks are estimated in probability of (0.1 – 1.0).
  - The higher the exposure, the more the risk should be tackled.
  - Another way is: risk avoidance (don’t do risky things).

Risk Control

- It is the process of managing risks to achieve the desired outcomes.
- It includes:
  - Risk management planning
  - Risk monitoring
  - Risk resolution
- **Risk management planning.**
  - Provides plan to deal with risk.
  - Identify different strategies, like:
    - Risk avoidance
    - Risk minimization (risk reduction)
    - Risk contingency plan

Risk Control

- **Risk monitoring.**
  - It is the **continuous process** of reassessing risks when project get changed.
  - Projects can be **evaluated periodically** to manage the risks.
  - Each key risk should be discussed at top level.

- **Risk resolution.**
  - Risk resolution is the **execution of the plans** for dealing with each risk.
  - Decided by project manager.
  - The input of this activity is risk action plan
  - And out puts are:
    - Risk status
    - Acceptable risks
    - Reduced rework
    - Corrective actions
    - Problem prevention

Example:

1 Risk Identification

Definition: This step involves recognizing potential risks that can affect the project's timeline, budget, performance, or security.
Example:

A team is developing an e-commerce website. Possible risks include:

- **Technical Risks:** The payment gateway API may not integrate properly.
- **Schedule Risks:** Key developers may leave before project completion.
- **Security Risks:** Customer data could be exposed to cyber threats.
- **Operational Risks:** Server downtime may impact user experience.

✅ **Outcome:** A list of risks is documented in a **Risk Register** for further analysis

2 Risk Assessment

Definition: This step evaluates the identified risks based on **likelihood (probability)** and **impact (severity)** to prioritize them.

Example: Risk Prioritization for the E-commerce Website

Example: Risk Prioritization for the E-commerce Website

Risk	Likelihood (1-5)	Impact (1-5)	Risk Level (L × I)	Priority
Payment gateway failure	4 (High)	5 (Critical)	20	🔴 High
Developer resignation	3 (Medium)	4 (Major)	12	🟡 Medium
Security breach	5 (Very High)	5 (Critical)	25	🔴 High
Server downtime	2 (Low)	3 (Moderate)	6	🟢 Low

✅ **Outcome:** High-risk areas (🔴) are prioritized for **immediate mitigation.**

3 Risk Control

Definition: This step involves implementing strategies to **reduce, transfer, accept, or eliminate** risks.

- **Example: Risk Control Strategies**

Risk	Control Strategy	Action Plan
Payment Gateway Failure	Mitigation	Use multiple payment providers (PayPal, Stripe) as backups.
Developer Resignation	Transfer	Document key processes and cross-train developers.
Security Breach	Avoidance & Mitigation	Implement strong encryption, regular security audits, and penetration testing.
Server Downtime	Acceptance	Use auto-scaling cloud infrastructure with failover mechanisms.

✅ **Outcome:** Risk control actions are **monitored and adjusted** throughout the project.

Thank You