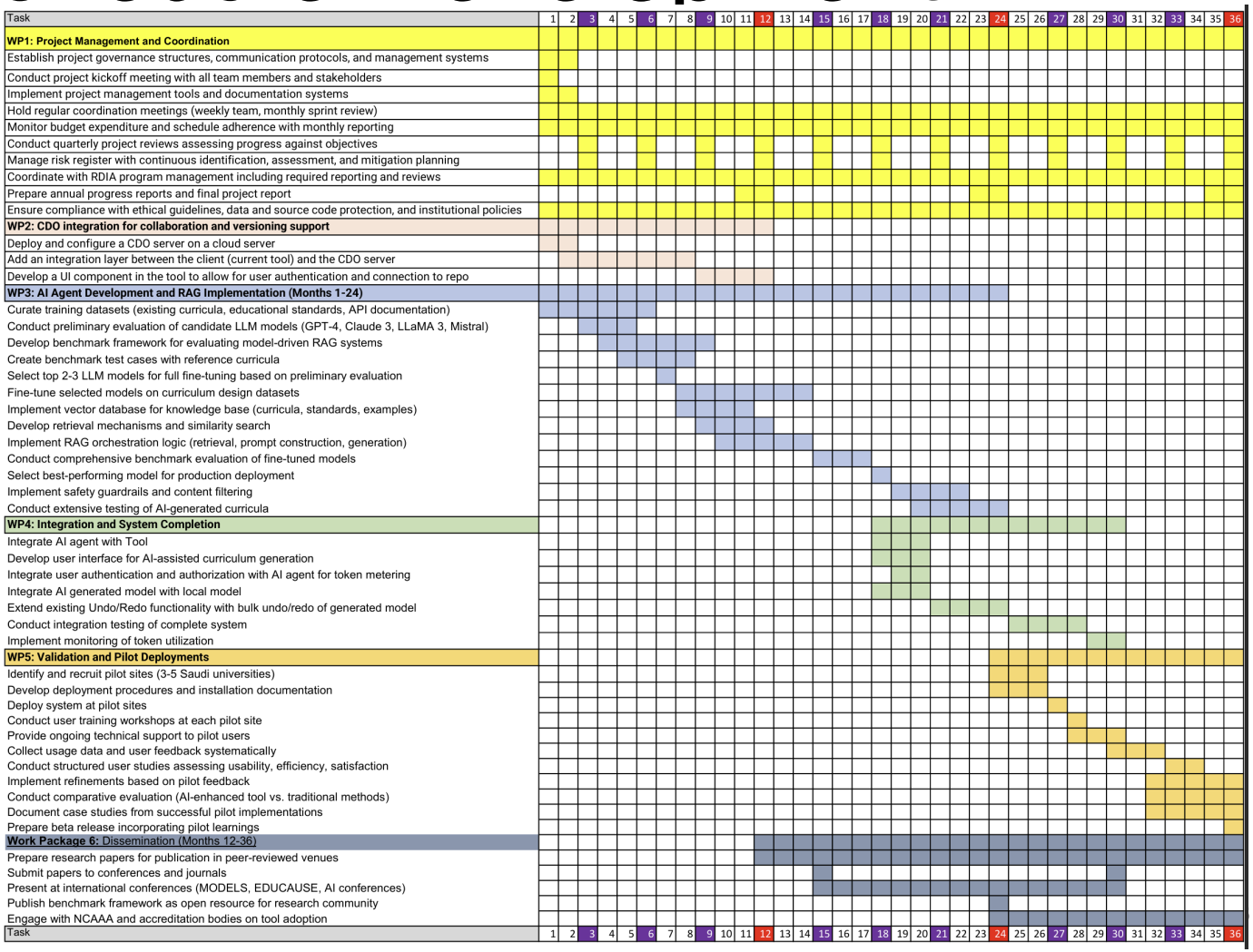
Scheduling and Tracking

SE423: Software Project Management

Outline

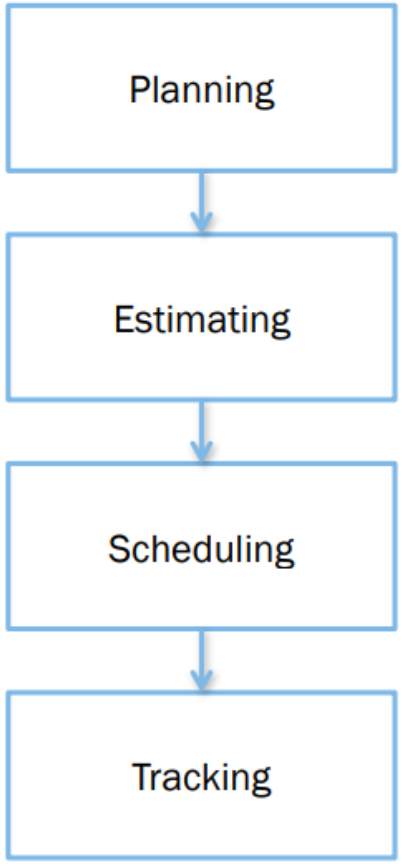
- Schedule Development
- Work Breakdown Structure (WBS)
- Network Analysis
- Scheduling Workflow
- PERT Estimation Technique
- Project Tracking

Schedule Development



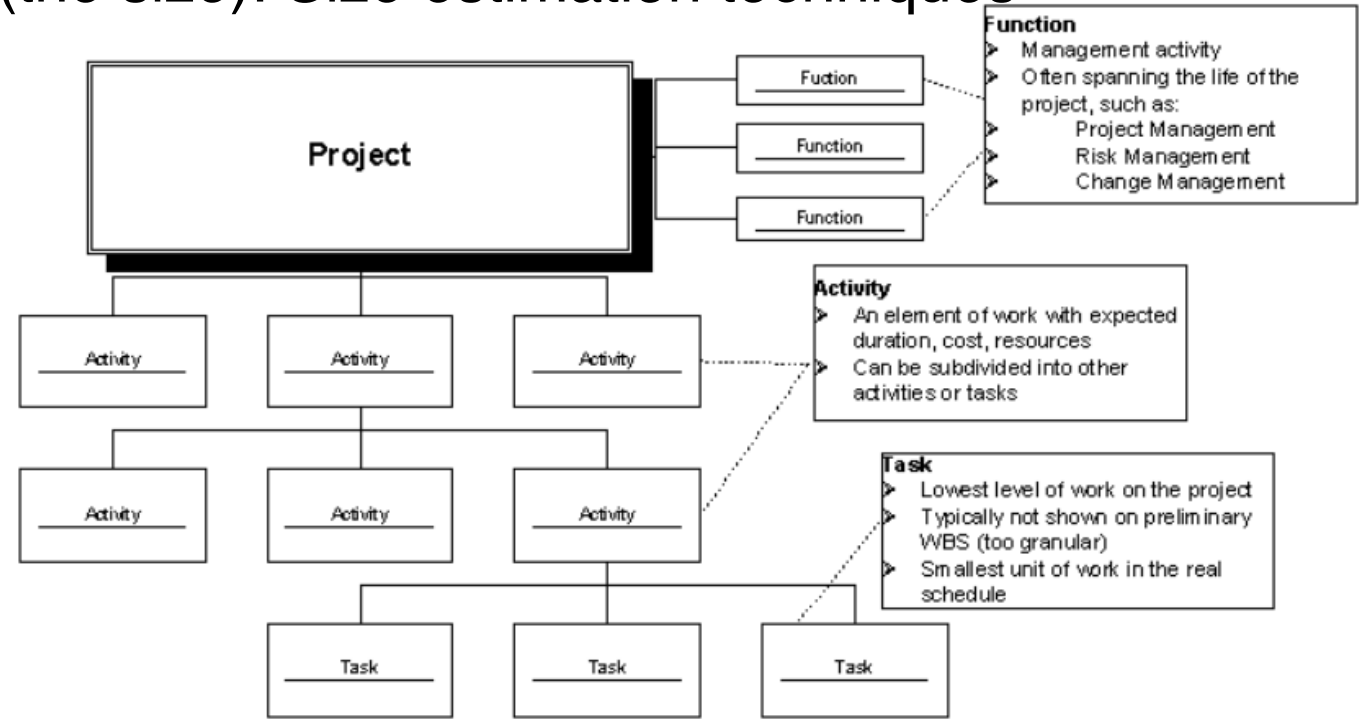
Planning, Estimation, Scheduling, Tracking

- Plan: Identify activities. No specific start and end dates.
- Estimating: Determining the size & duration of activities.
- Scheduling: Adds specific start and end dates, dependencies, and resources.
- Tracking: Uses monitoring and tools to determine if plans, estimates, and schedules are accurate



How to Schedule?

1. Identify “what” needs to be done: Work Breakdown Structure (WBS)
2. Identify “how much” (the size): Size estimation techniques
3. Identify the dependency between tasks
   - Dependency graph, network diagram
4. Estimate total duration of the work to be done
   - The actual schedule



Definition

- Schedule development is the culmination of the other Project Time Management processes we have discussed:
  - Activity definition
  - Activity sequencing
  - Activity resource estimating
  - Activity duration estimating
- It is an iterative process to determine planned start and finish dates for activities
- It is a continuous process throughout project, addressing approved changes and risks

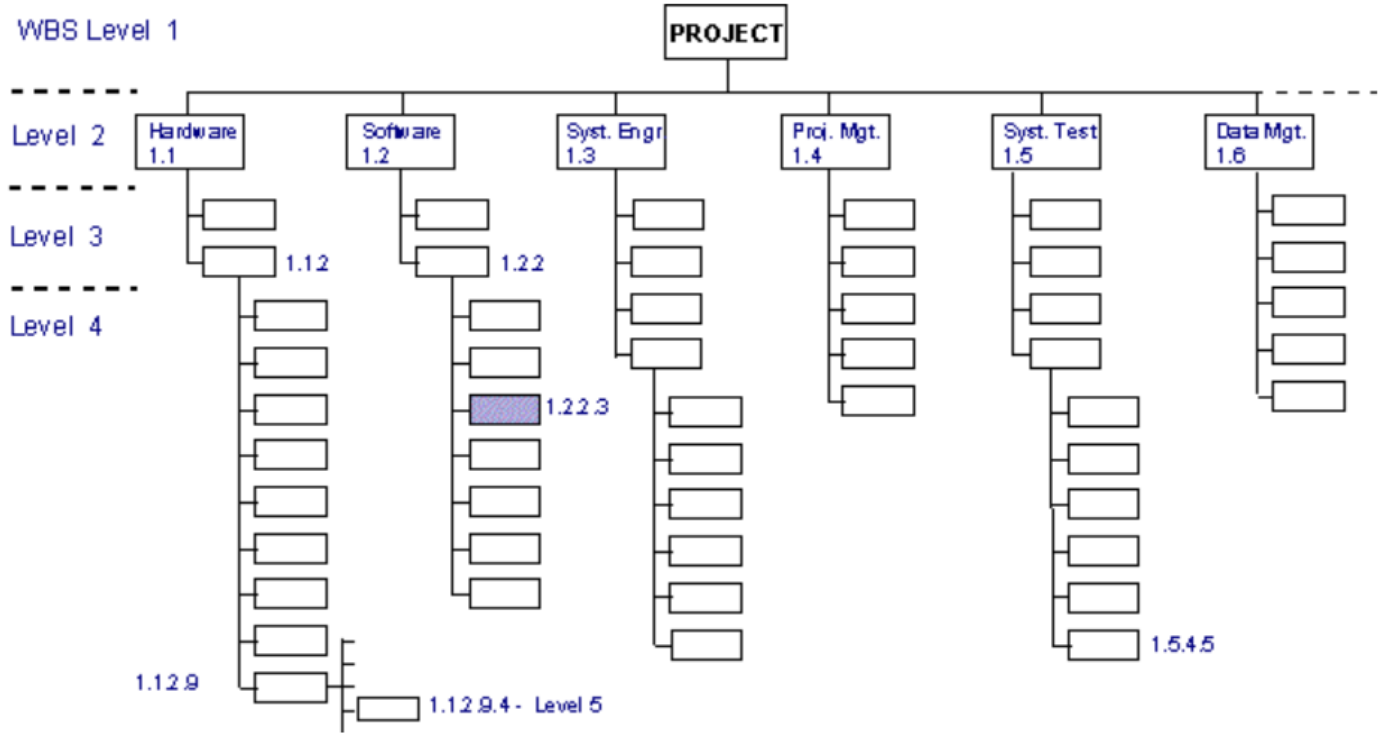
Controlling factors in schedule development

- Project scope statement
  - Scope statement is the source of assumptions and constraints on the project
  - (PMI) "*Assumptions* are those documented, schedule-related factors that, for schedule development purposes, are considered to be true, real, or certain."
  - (PMI) "*Constraints* are factors that will limit the project management team's options when performing schedule network analysis." (ex: "only use internal HR")
  - Date constraints (contract dates, market windows, external deliveries) and milestones (deliverable dates) are of greatest importance in schedule development
- Project management plan (document)
  - Almost any of the many sub-plans (resource management plan, quality assurance plan, etc.) and other elements in the Project Management Plan may exert an influence on schedule development
  - One of the most critical elements for schedule development in the PM Plan is the ***risk register*** and risk-associated plans

Work Breakdown Structure (WBS)

Work Breakdown Structure (WBS)

- Work Break Down Structure (WBS)
  - a check list of the work that must be accomplished to meet the project objectives.
- The WBS lists the major project outputs and those departments or individuals primarily responsible for their completion.

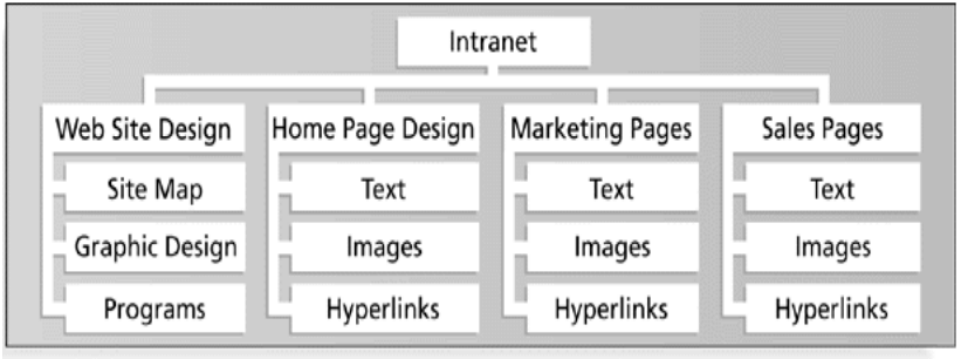


WBS Outline Example

- 0.0 Retail Web Site
- 1.0 Project Management
- 2.0 Requirements Gathering
- 3.0 Analysis & Design
- 4.0 Site Software Development
  - 4.1 HTML Design and Creation
  - 4.2 Backend Software
    - 4.2.1 Database Implementation
    - 4.2.2 Middleware Development
    - 4.2.3 Security Subsystems
    - 4.2.4 Catalog Engine
    - 4.2.5 Transaction Processing
  - 4.3 Graphics and Interface
  - 4.4 Content Creation
- 5.0 Testing and Production

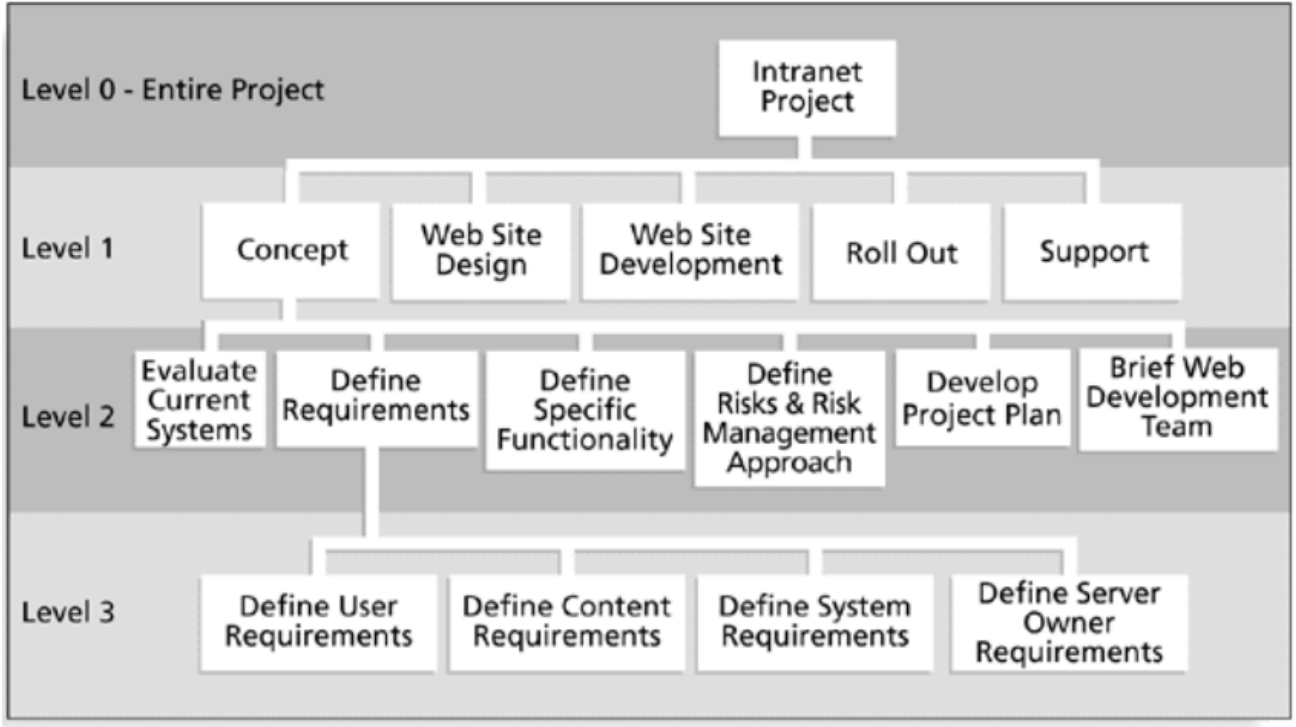
WBS Types

- Process WBS
  - a.k.a Activity-oriented
  - Ex: Requirements, Analysis, Design, Testing
  - Typically used by PM
- Product WBS
  - a.k.a. Entity-oriented
  - Ex: Financial engine, Interface system, D
  - Typically used by engineering manager
- Hybrid WBS: both above
  - This is not unusual
  - Ex: Lifecycle phases at high level with component or feature specifics within phases
  - Rationale: processes produce products



Process WBS

- List of Activities, not Things
- List of items can come from many sources: SOW (statement of work), Proposal, brainstorming, stakeholders, team
- Describe activities using "bullet language"
  - Meaningful but brief labels
- All WBS paths do not have to go to the same level
- Do not plan more detail than you can manage



* SOW: statement of work, Defines contractual obligations and major deliverables

Work Packages (Tasks)

- Generic term for discrete unit of work with definable end results
- The “one-to-two” rule
  - Often sized for: 1 or 2 persons for 1 or 2 weeks
- Basis for monitoring and reporting progress
  - Can be tied to budget items (charge numbers)
  - Resources (personnel) assigned
- Ideally shorter rather than longer
  - Longer makes in-progress estimates needed
  - These are more subjective than “done”
  - “4/40” or “8/80” rule (shortest/longest duration): sizing from 4 to 40 hours to avoid micromanagement, or 8 to 80 max to ensure visibility.
  - Not so small as to micro-manage

WBS and Methodology

- PM must map activities to chosen lifecycle
- Each lifecycle has different sets of activities
- Common process activities occur for all lifecycle models (waterfall, agile, etc.): Planning, configuration, testing
- Operations and maintenance phases are not normally in plan (considered post-project)
- Some models are “straightened” for WBS
  - Spiral and other iterative models -> Linear sequence several times (design Iteration 1 -> Design iteration 2, etc.)
- Deliverables of tasks vary by methodology (waterfall: requirements doc, for Agile: User sories)

WBS Techniques

- Top-Down: Start with the overall project goal or deliverable, then break it down into major components and subcomponents.
- Bottom-Up: Start by identifying detailed tasks or activities, then group them into higher-level categories.
- Analogy: Use a WBS from a similar past project as a template, adapting it to the current context.
- Rolling Wave: Plan at a high level initially, then add detail as more information becomes available:
  - 1st pass: go 1-3 levels deep
  - Gather more requirements or data
  - Add more detail later

- All WBS Techniques rely upon **Expert Judgment**!

WBS Techniques

- Top-down
  - Start at highest level
  - Systematically develop increasing level of detail
  - Best if
    - The problem is well understood
    - Technology and methodology are not new
    - Project is similar to an earlier project or problem
  - But is also applied in majority of situations

WBS Techniques

- Bottom-up
  - Start at lowest level tasks
  - Aggregate into summaries and higher levels
  - Cons
    - Time consuming
    - Needs more requirements complete
  - Pros
    - Detailed

WBS Techniques

- Analogy
  - Base WBS upon that of a “similar” project
  - Use a template
  - Pros
    - Based on past actual experience
  - Cons
    - Needs comparable project

WBS Techniques

- Brainstorming
  - Generate all activities you can think of that need to be done
  - Group them into categories
- Both Top-down and Brainstorming can be used on the same WBS
- Remember to get the people who will be doing the work involved (buy-in matters!)

WBS basis for many things

- Network scheduling
- Costing
- Risk analysis
- Organizational structure
- Control
- Measurement

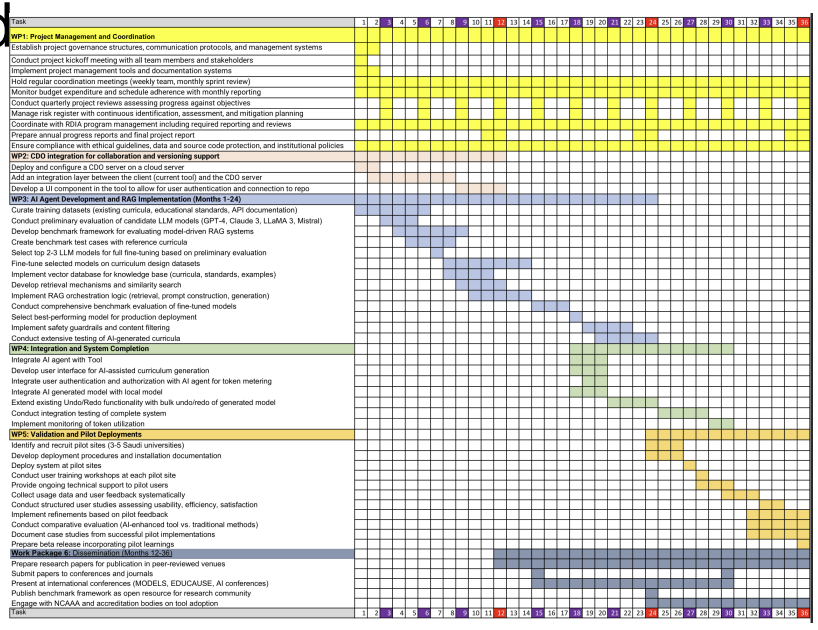
WBS Guidelines

- Should be easy to understand
- Some companies have corporate standards for these schemes
- Some top-level items, like Project Mgmt. are in WBS for each project
  - Others vary by project
- What often hurts most is what's missing
- Break down until you can generate accurate time & cost estimates
- Ensure each element corresponds to a deliverable

Network Analysis

Network Analysis

- *Network analysis* is the technique that generates the project schedule
- Network analysis may use several different analysis methods to calculate the early and late start dates for project activities. These methods may be combined and includ
  - Gantt Charts
  - Critical Path Method (CPM)
  - Critical Chain Method (CCM)
  - What-if analysis
  - Resource leveling



Network Analysis

- There are a number of network analysis techniques available – we will concentrate on the *Critical Path Method* (CPM)
- The Precedence Diagram Method (PDM) is a graphical network technique that establishes activity sequencing
- Network analysis may also make use of parallelism in project activities to allow schedule compression

Network Analysis Terminology

g

- **Activity.** An activity always consumes time and may also consume resources. I use task and activity equivalently
- **Critical.** A critical activity or event is one that must be achieved by a certain time, having no latitude (slack or float)
- **Critical path.** The critical path is the longest path through a project network. Because it has no slack, all activities on the critical path must be completed as scheduled, or the end date will slip
- **Events.** Beginning and ending points of activities are known as events. An event is a specific point in time
- **Milestone.** An event representing a point in a project of special significance. Usually the completion of a major phase of the work. Project reviews are often conducted at milestones

Creating a precedence table

- A **Precedence table** documents task durations and interdependencies
- Include duration estimation of each task
  - Any potentially risky task (technology, skills, dependencies) should include contingency factor
  - Include contingencies to mitigate potential resource shortages
- Determining task interdependencies
  - Goal is to determine predecessor/successor relations
  - Understanding interdependencies allows proper ordering in scheduling tasks
  - Understanding interdependencies also helps in finding possible parallel tasks, which can shorten schedule
  - Determining task interdependencies must be a team effort to avoid unpleasant surprises

Sample precedence table

	WBS	Task Name	Duration	Predecessors
1	**1**	**- Automatic Toll System**	**65 days?**	
2	**1.1**	**+ Management**	**1 day?**	
7	**1.2**	**- Environment**	**10 days?**	
8	1.2.1	Inception phase environment	10 days?	
9	1.2.2	Elaboration phase environment	10 days?	
10	1.2.3	Construction phase environment	5 days?	
11	1.2.4	Transition phase environment	1 day?	
12	**1.3**	**+ Requirements**	**1 day?**	
17	**1.4**	**- Design**	**40 days?**	
18	1.4.1	Inception phase architecture prototyping	10 days?	8
19	**1.4.2**	**- Elaboration phase design**	**16.67 days?**	
20	1.4.2.1	Architecture design	10 days?	9
21	1.4.2.2	Preliminary component design	6.67 days?	20
22	**1.4.3**	**- Construction phase design modeling**	**20 days?**	
23	1.4.3.1	Architecture design maintenance	20 days?	20,10
24	1.4.3.2	Component design	6.67 days?	21
25	1.4.4	Transition phase architecture design maintenance	1 day?	
26	**1.5**	**- Implementation**	**65 days?**	
27	1.5.1	Inception phase implementation	15 days?	18
28	1.5.2	Elaboration phase implementation	15 days?	21,27
29	1.5.3	Construction phase implementation	15 days?	24,28
30	1.5.4	Transition phase implementation	1 day?	
31	**1.6**	**+ Assessment**	**1 day?**	
36	**1.7**	**+ Deployment**	**1 day?**	

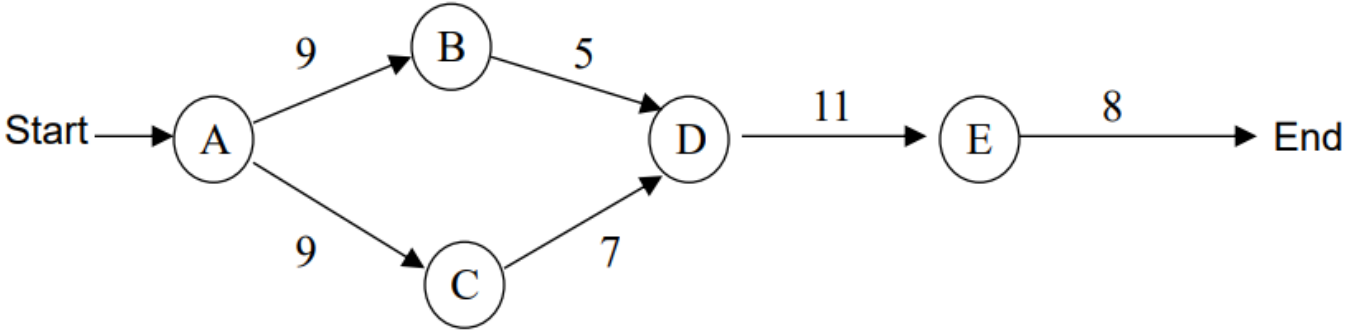
Reminder (from SE201): In the rational unified process (RUP) 4 phases are **iteratively** applied:

- **inception** is the initial phase where project scope, goals, and feasibility are defined, and a high-level business case is created.
- The **elaboration** phase follows, focusing on building a stable architecture, mitigating major risks, and refining the detailed requirements through activities like creating use cases and preliminary designs.
- The **construction** phase is where the bulk of the system is built. The main focus is on developing and testing components and integrating them into the product's architecture.
- The **transition** phase is the final phase, it focuses on delivering the system to the end-user community and achieving formal user acceptance.

Precedence Network Diagram

- A precedence network diagram is a graphic model portraying the sequential relationship between key events in a project.
- The network diagram clearly and precisely communicates the plan of action to the project team and the client.

Task	Duration	Dependencies
A - Architecture & design strategy	9	start
B - Decide on number of releases	5	A
C - Develop acceptance test plan	7	A
D - Develop customer support plan	11	B,C
E - Final sizing & costing	8	D



PERT Estimation Technique

What is a PERT?

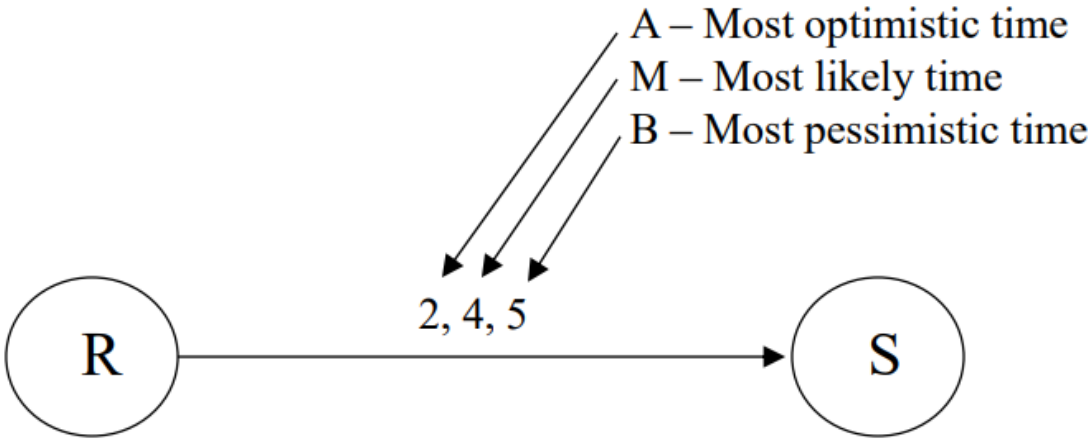
Program Evaluation & Review Technique or PERT developed to handle uncertainty in activity durations

- Identify the tasks (or activities) required to complete the given project and their dependencies.
- Estimate durations using three-point estimation: PERT uses optimistic, pessimistic, and most likely time estimates to calculate expected durations using the formula:

$$\text{Expected Time(TE)} = (O+4M+P) / 6$$

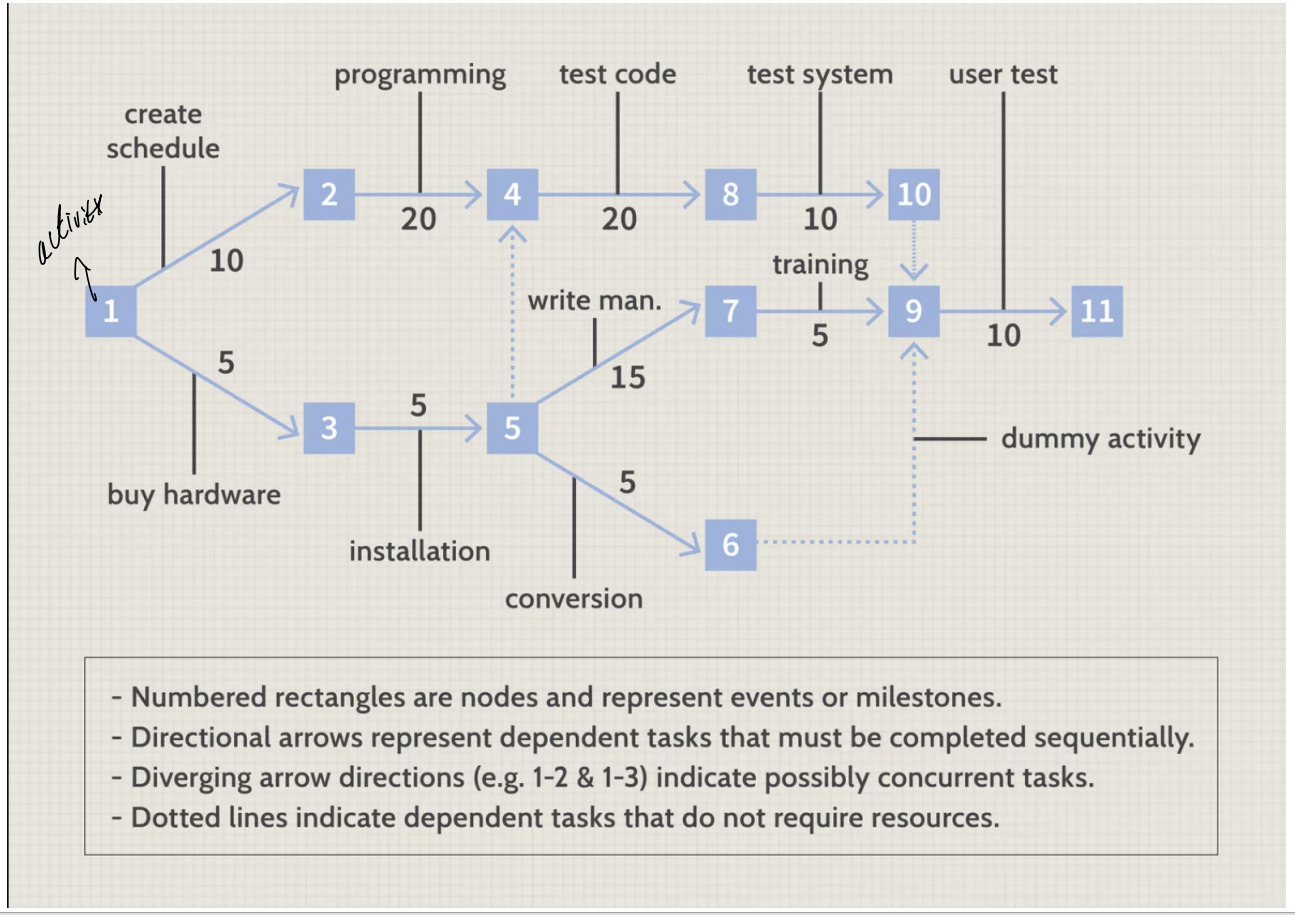
- Construct "network diagram" of the activities and their dependencies, each event (milestone) is represented by a node, arrows represent activities

PERT



- Expected Time = (a + 4m + b)/6
- Expected Time = 3.8

PERT Chart


programming
test code
test system
user test
create schedule
activity
2
4
8
10
20
20
10
10
1
training
write man.
7
9
11
5
10
5
15
3
5
5
dummy activity
buy hardware
installation
conversion
6
- Numbered rectangles are nodes and represent events or milestones.
- Directional arrows represent dependent tasks that must be completed sequentially.
- Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks.
- Dotted lines indicate dependent tasks that do not require resources.

PERT estimation technique

- Widely used three-point estimate with PERT (*Program Evaluation and Review Technique*)
- Defines three estimate points as:
  - *Most likely*: estimate that occurs with greatest frequency
  - *Optimistic*: shortest duration, taken as 10th percentile value
  - *Pessimistic*: longest duration, taken as 90th percentile value
  - PERT activity **duration estimate** T_E and its **standard deviation** (s_E or σ) are calculated according to:

$$T_E = (E_{optimistic} + 4E_{most\_likely} + E_{pessimistic}) / 6$$

$$s_E = (E_{pessimistic} - E_{optimistic}) / 6$$; measures the **uncertainty or variability** in the estimate.

* Standard deviation is a statistical metric that measures how spread out data points are from the mean (average) of a dataset

Slack Time

- Slack time, also known as float, is the amount of delay expressed in units of time that could be tolerated in the starting time or completion time of an activity without causing a delay in the completion of the project.
- Slack time is the difference between the **late finish** and the **early finish** (LF-EF). If the result is greater than zero, then the activity has a range of time in which it can start and finish without delaying the project completion date.
- If an activity has zero slack, it determines the project completion date. In other words, all the activities on the critical path must be done on their earliest schedule or the project completion date will suffer.
- If an activity with total slack greater than zero was delayed beyond its late finish date, it becomes a critical path activity and causes the completion date to be delayed.
- The sequence of activities that has zero slack is defined as the critical path
- In general, the critical path is the path that has 0 slack.

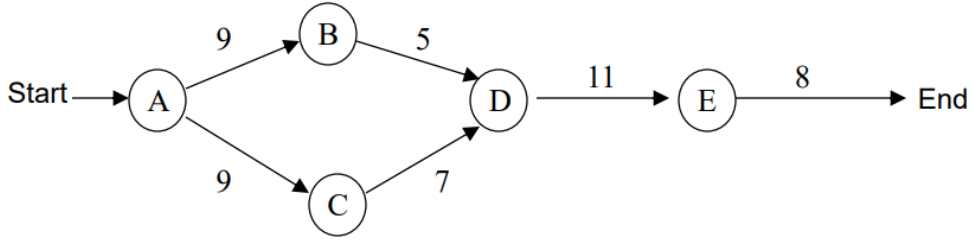
Critical Path Method

- The *critical path method* (CPM) focuses on calculating theoretical start and finish dates for every activity in the project.
- CPM performs these calculation without regard for resource limitations, which can lead to resource over-allocation or inefficient multitasking
- A *forward pass* analysis performs schedule calculations that identify the early start and finish dates of activities and the project
- A *backward pass* analysis performs schedule calculations that identify the late start and finish dates of activities and the project, as well as total and free float
- *Total float* (TF) is the amount of time an activity can be delayed without delaying the project as a whole
- *Free float* (FF) is the amount of time an activity can be delayed without delaying its successor (dependent) activities
- Note that total float is global to the project, while free float is local to the neighborhood of the activity

Critical Path Method

- Critical Path Method (CPM) tries to answer the following questions:
  - What is the duration of the project?
  - By how much will the project be delayed if any one of the activities takes N days longer?
  - How long can certain activities be postponed without increasing the total project duration?

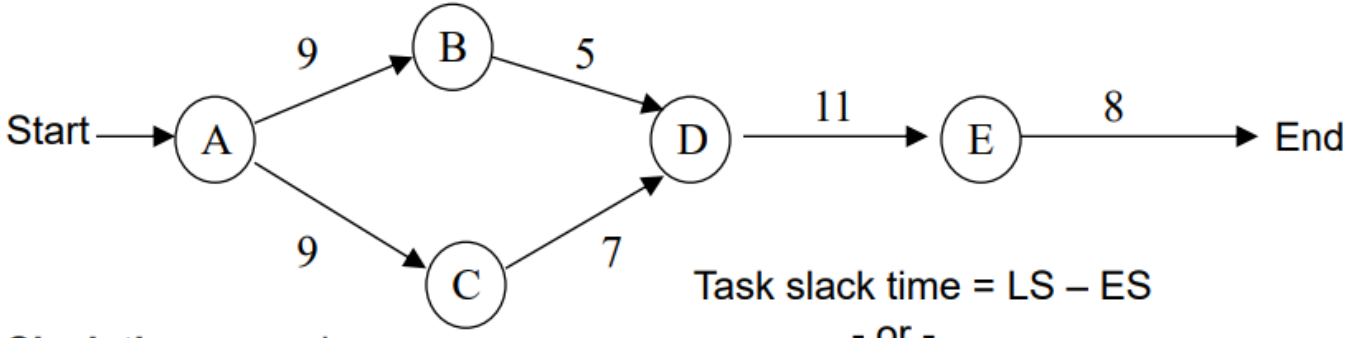
- Critical Path = A – C – D – E (35 time units)
- Critical Tasks = A,C,D,E
- Non-Critical Path = A-B-D-E
- Non-Critical Tasks = B (only)



Critical Path Example

(see SE201 for revision)

Task	Duration	Depend	Earliest Start	Earliest Finish	Latest Start	Latest Finish
A	9	none	0	9	0	9
B	5	A	9	14	11	16
C	7	A	9	16	9	16
D	11	B,C	16	27	16	27
E	8	D	27	35	27	35



Slack time – maximum allowable delay for a non-critical activity.

Task slack time = LS – ES

- or -

Task slack time = LF - EF

Task B has 2 time units of ***slack time***

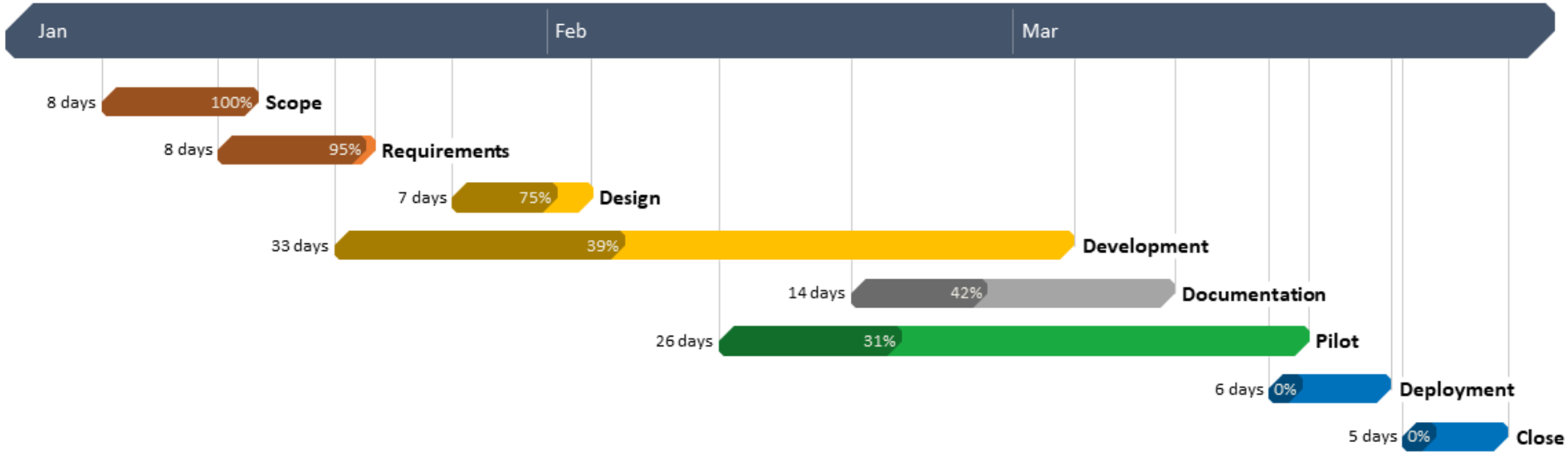
What is a Gantt chart?

A **Gantt Chart** (named for Henry Laurence Gantt) consists of a table of project task information and a bar chart that graphically displays project schedule, depicting progress in relation to time and often used in planning and tracking a project.

- Horizontal bar chart format, with bars representing the phases and activities of the WBS
- Time extends along the horizontal axis
- Able to show planned and actual progress on tasks as well as task dependencies
- Effective communication tool but with limitations
  - Very low information density: lots of wasted space
  - Not very useful for large projects

Gantt Chart

- Gantt chart is a means of displaying simple activities or events plotted against time.
- Most commonly used for exhibiting program progress or for defining specific work required to reach an objective
- Gantt charts may include listing of activities, activity duration, scheduled dates, and progress-to-date



Gantt Chart

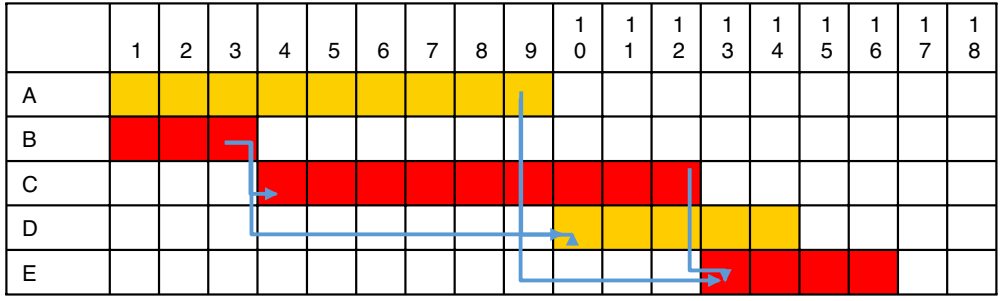
- Advantages:
  - Easy to understand
  - Easy to change
- Disadvantages:
  - Only a vague description of the project
  - Does not always show interdependency of activities
  - May not show results of an early or late start of an activity

Gantt Example

Suppose a project comprises five activities: A,B,C,D, and E. A and B have no preceding activities, but activity C requires that activity B be completed before it can begin. Activity D cannot start until both activities A and B are complete. Activity E requires activities A and C to be completed before it can start. If the activity times are A: 9 days; B: 3 days; C: 9 days; D: 5 days; and E: 4 days, determine the shortest time necessary to complete this project.

Identify those activities which are critical in terms of completing the project in the shortest possible time. [Critical path]

Time is 16 days



Without graphical representation of the dependencies, we cannot identify the critical path.

Critical Chain Method

- The critical chain method (CCM) is a project management technique that is an improvement of the Critical Path Method as it takes into account the availability of resources.
- In software engineering projects, CCM can be used to identify and manage critical tasks, reduce project risk, and improve project delivery time.
- CCM focuses on the **resources** required for project activities, attempting to keep them leveled throughout the project
- CCM, assumes that all activities have a statistically-probable *range* of durations and uses this assumption to create a more flexible and resilient schedule
- Resources are assigned to the activities using the *most likely* durations
- The longest sequence of activities in the project is called the *critical chain*

Critical Chain Method

- Suppose a software development team is tasked with developing a new mobile application.
- The project timeline is six months, and the team must deliver a functional prototype at the end of the fourth month.
- To manage the project using CCM, the following steps could be taken:
    1. Identify the critical tasks: The team would begin by identifying the critical tasks that must be completed to deliver the functional prototype on time. These tasks might include developing the user interface, integrating with external APIs, and testing the application.

Critical Chain Method

2. Estimate task duration: Once the critical tasks have been identified, the team would estimate the duration of each task, taking into account any **potential delays** or obstacles that might arise during the development process.
3. Create a buffer: To account for potential delays and risks, the team would create a buffer at the end of the project timeline. This buffer would be used to ensure that the functional prototype is delivered on time, even if some tasks take longer than expected.
4. Manage the critical chain: Using CCM, the team would manage the critical chain of tasks by focusing on the critical tasks and ensuring that they are completed on time. This might involve reallocating resources, adjusting task priorities, or providing additional support to team members as needed.
5. Monitor progress: Throughout the project, the team would monitor progress and adjust the project plan as needed. This would help to ensure that the project stays on track and that the critical tasks are completed on time.

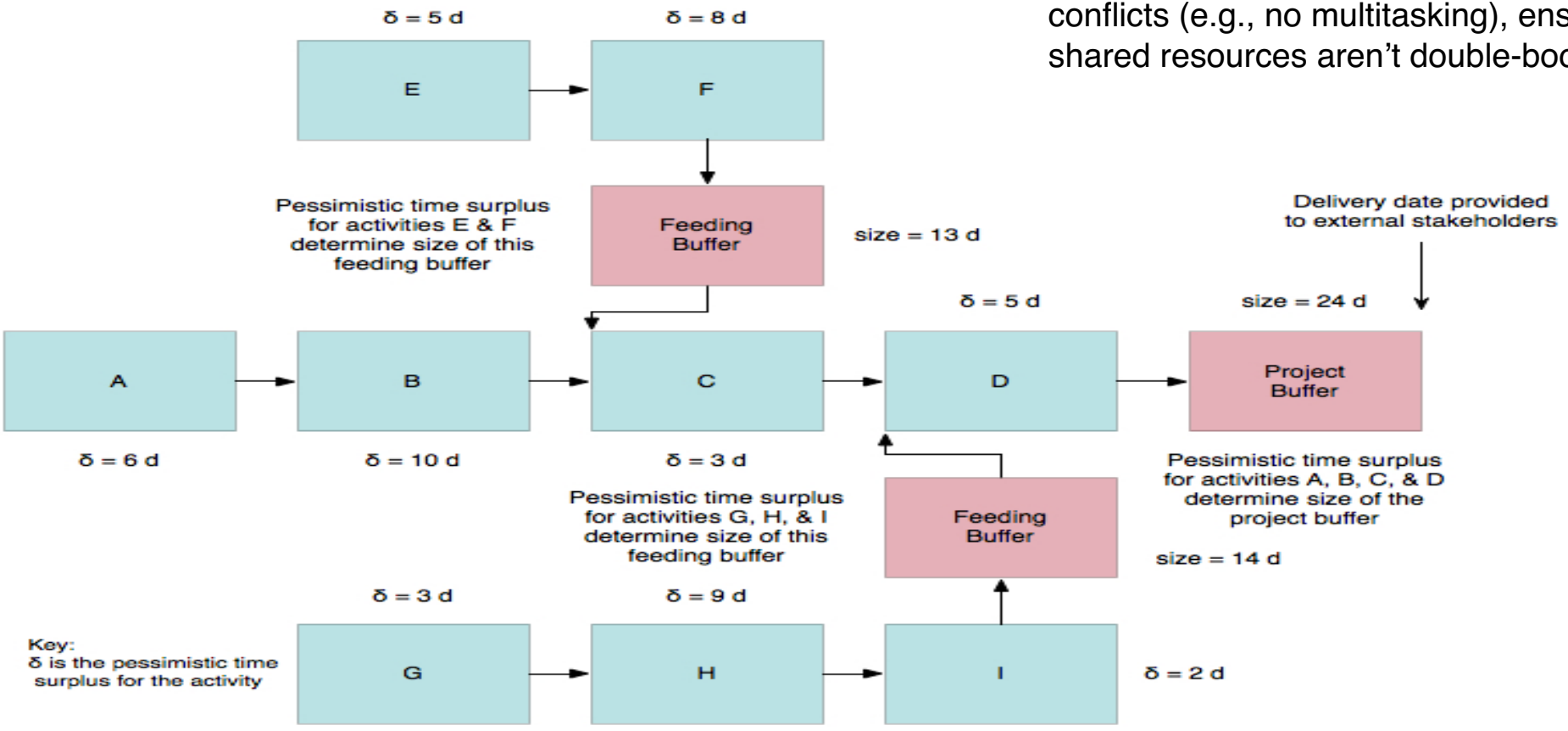
Critical Chain Method

- Buffer Types:
  - Project buffer
    - Contingency time to the whole project inserted at end of the project. Any delay in the Longest Path will consume some or all of the buffer.
  - Feeder buffers
    - inserted at the point where non-critical chains feed into the critical chain. This is to ensure that any tasks feeding into the Critical Chain may not delay the Critical Chain.
  - Resource buffers
    - Alerts or placeholders to ensure critical resources are ready when needed.

Critical Chain Method

g

Tasks are sequenced to avoid resource conflicts (e.g., no multitasking), ensuring that shared resources aren't double-booked.



What-if analysis

- It is a scenario-based analysis of schedule to determine effects of various scenarios on different aspects of the project
  - *Example:* Delay delivery of a critical component by various amounts to determine effect on schedule; *"COTS supplier is unable to provide a critical component at all"*
- ***What-if scenario analysis*** effectively tests the robustness of the project schedule in response to adverse circumstances
- Most common technique uses *Monte Carlo simulation* to generate a population of possible project schedule outcomes
  - Think of executing the same project 10,000 times with the same resources, different boundary conditions, and no memory between executions
- Very useful in preparing contingency and response plans for project risks

Resource leveling

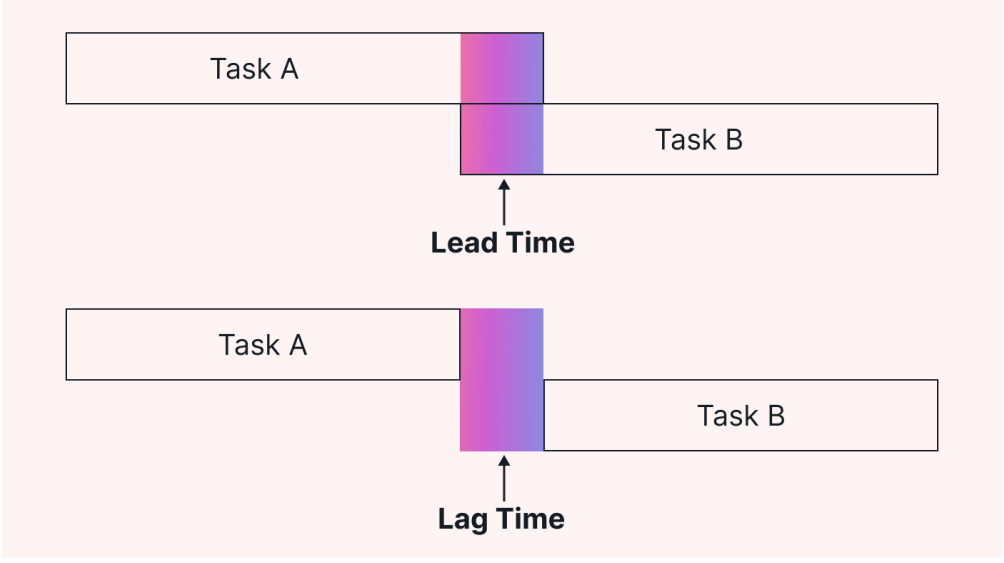
- Resource leveling (smoothing a surface) is applied to a schedule analyzed by CPM
- Addresses situation where resource availability is constrained by time or amount of the resource available
- May also be used to keep resource usage at a constant level during certain time periods in the project
- Resource leveling is needed when resources have been over-allocated or assigned to two or more activities in the same time period
- May change the critical path in the schedule model

Resource leveling

- Uses a number of different approaches, including:
  - Assign under-allocated resources to multiple tasks to keep them busy
  - Move key resources off of non-critical tasks
  - Delay start of task until required resources are available, possibly using lags
  - Split tasks into two or more subtasks so the subtasks can be assigned to different resources

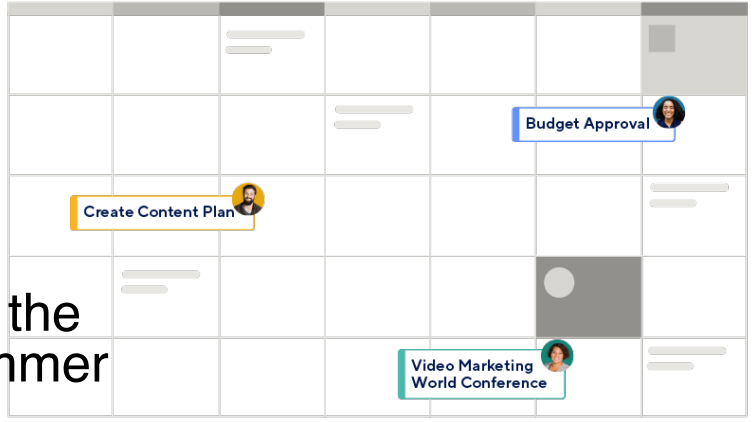
Applying leads and lags

- *Applying leads and lags* allows refinement of a schedule once the major schedule network analysis effort has been completed
- Use of leads and lags may be used to meet imposed constraints, help in resource leveling, or incorporate contingency reserves into a schedule



Calendars

- Identify days and dates when work can be performed
- *General project calendars* govern overall limitations on when project work may be performed
  - *Example:* Work is performed at a client site, and the client shuts down for three weeks during the summer
- *Resource calendars* govern limitations on when particular resources (or resource groups) may perform project work
  - *Example:* Individual project team member vacation schedules
  - *Example:* Development team training schedules



Schedule development output

- Project schedule network diagrams
  - Show both project network logic (sequencing) as well as critical path schedule activities
  - Usually displayed as an activity-on-node diagram
- Gantt charts
  - Specialized bar charts format to show activity start and end dates, along with durations
  - Easy to read but limited by low information density
- Milestone charts
  - 'Stripped-down' version of Gantt chart, showing only milestones

Reducing Project Duration

- How can you shorten the schedule?
- Via
  - Reducing scope (or quality) Less outputs.
  - Adding resources more people.
  - Concurrency (perform tasks in parallel) (Lead (?))
  - Substitution of activities

Schedule compression

- Shortens the project schedule without changing the project scope, to meet schedule constraints, imposed dates, or other schedule objectives
- There are two types of schedule compression, ***crashing*** and ***fast-tracking***

- ***Crashing***. Analyzes cost and schedule trade-offs to get the greatest amount of compression with the least cost:
  - Add resources to critical path tasks, be more efficient, changing approach used to perform work, work overtime
  - Limit or reduce requirements (scope)
  - Changing the sequence of tasks

Schedule compression

- ***Fast-tracking***. Activities that would normally be done sequentially are done in parallel.
  - Fast-tracking can lead to rework and increased risks due to unforeseen dependencies: Involves some risk
  - Fast-tracking only works when activities can be overlapped to shorten the total duration

For both schedule compression approaches, you cannot compress more than 25% (According to Boehm)

Mythical Man-Month

- Book: "The Mythical Man-Month"
  - Author: Fred Brooks
- "***The*** classic book on the human elements of software engineering"
- First two chapters are full of terrific insight (and quotes)

Sample Quotes

- ***"Cost varies as product of men and months, progress does not.":*** progress **does not scale linearly**. Adding more people doesn't necessarily mean faster progress.

- ***"Hence the man-month as a unit for measuring the size of job is a dangerous and deceptive myth":*** A unit that assumes one person working for a month is equivalent to two people working for half a month

Mythical Man-Month

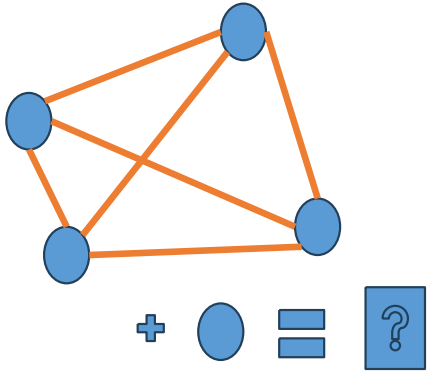
g

- Why is software project disaster so common?
  1. Estimation techniques are poor & assume things will go well (a very optimistic 'unvoiced' assumption)
  2. Estimation techniques falsely confuse effort with progress, hiding the false assumption that (You can just throw more people at a problem and expect linear acceleration.)
  3. Managers often don't push back against unrealistic deadlines or vague estimates: Good managers defend the integrity of the schedule, even when it's unpopular.
  4. Schedule progress is poorly monitored: Without clear metrics, slippage goes unnoticed until it's too late.
  5. When schedule slippage is recognized, the natural response is to add manpower. Which, is like dousing a fire with gasoline: Brooks's Law: **"Adding manpower to a late software project makes it later."**

Mythical Man-Month

- **Over-Optimism**
  - "All programmers are over-optimists"
  - 1st false assumption: "all will go well" or "each task takes only as long as it 'ought' to take"
  - The Fix: Consider the larger probabilities
- **Cost (overhead) of communication (and training)**
  - nb_communication_channels = n(n-1)/2, $O(n^2)$, n: number of team members

$$\frac{n(n-1)}{2}$$



- **How long does a 12 month project take?**
  - 1 person: 12 months
  - 2 persons = 7 months (2 man-months extra)
  - 3 persons = 5 months (3 man-months extra)
  - Fix: don't assume adding people will solve the problem

Mythical Man-Month

- **What is the most mis-scheduled part of process?**
  - Testing (the most linear process)
  - delays in earlier phases **compress** the time available for testing

  - Why is this particularly bad?
    - Occurs late in process and without warning
    - Higher costs:
      - **Primary cost**: Time and effort to fix bugs.
      - **Secondary cost**: Rework, missed deadlines, reputational damage, or even system failure if bugs escape into production.

  - Fix: Allocate more test time
    - Understand task dependencies (testing depends on completed, integrated components)

Mythical Man-Month

- **Reliance on intuition and guesses**
  - What is ‘gutless estimating’? Estimators **don’t push back** on unrealistic deadlines.
  - Q: “How does a project get to be a year late”?
    - A: “One day at a time”: Missed check-ins (*Regular project reviews or sync points that are skipped, delayed, or done superficially*), Untracked slippage, Quiet scope creep
  - Studies show that:
    - Each task: twice as long as estimated
    - Only 50% of work week was programming
  - Fixes
    - No “fuzzy” milestones: avoid vague goals like “almost done”: use measurable deliverables.
    - Reduce the role of conflict: Encourage honest dialogue between developers and managers.
    - Identify the “true status”: Use metrics, demos, and progress tracking to know where things really stand.

Quick Summary of the Scheduling Workflow

Scheduling Workflow

- Define activities
  - Use of WBS template helps guide definition process and organize activities
- Perform activity sequencing
  - Develop schedule framework according to what is *logically possible* – perform resource allocation later
- Estimate effort – the total number of labor units (*e.g.* staff-days) for each activity
- Identify resources for each activity
- Apply calendars to schedule framework

Scheduling Workflow

- Estimate activity duration based on resources for activity
- Perform forward pass and backward pass critical path analysis to generate schedule model
- Perform ‘what-if’ scenario analysis to identify contingency and risk response needs
- Apply resource leveling to schedule model
- Apply schedule compression, if needed

WBS template

Component groups with a '+' in front of them are 'rolled up' – subcomponents are hidden to reduce clutter

	WBS	Name
1	**1**	− **Project**
2	**1.1**	+ **Management**
20	**1.2**	+ **Environment**
33	**1.3**	+ **Requirements**
42	**1.4**	− **Design**
43	1.4.1	Inception phase architecture prototyping
44	**1.4.2**	− **Elaboration phase architecture baselining**
45	1.4.2.1	Architecture design modeling
46	1.4.2.2	Design demonstration planning and conduct
47	1.4.2.3	Software architecture description
48	**1.4.3**	+ **Construction phase design modeling**
51	1.4.4	Transition phase architecture design maintenance
52	**1.5**	− **Implementation**
53	1.5.1	Inception phase component prototyping
54	**1.5.2**	− **Elaboration phase component implementation**
55	1.5.2.1	Critical component coding demonstration integration (architectural baseline)
56	**1.5.3**	+ **Construction phase implementation**
61	1.5.4	Transition phase component maintenance
62	**1.6**	− **Assessment**
63	1.6.1	Inception phase assessment planning
64	**1.6.2**	− **Elaboration phase assessment**
65	1.6.2.1	Test modeling
66	1.6.2.2	Architecture test scenario implementation
67	1.6.2.3	Demonstration assessment and release descriptions
68	**1.6.3**	+ **Construction phase assessment**
72	**1.6.4**	+ **Transition phase assessment**
74	**1.7**	+ **Deployment**

Activity sequencing

	WBS	Name	Predecessors
1	**1**	**− Automatic Toll Booth System**	
2	**1.1**	**+ Management**	
20	**1.2**	**+ Environment**	
33	**1.3**	**+ Requirements**	
42	**1.4**	**− Design**	
43	1.4.1	Inception phase architecture prototyping	
44	**1.4.2**	**− Elaboration phase architecture baselining**	
45	**1.4.2.1**	**− System architecture definition**	
46	1.4.2.1.1	Define high-level hardware architecture	43
47	1.4.2.1.2	Define high-level communications architecture	43,46SS+5 days
48	**1.4.2.2**	**− Architecture design modeling**	
49	1.4.2.2.1	Perform ABD (architecture-based design)	45
50	1.4.2.2.2	Perform quality attribute analysis (ATAM)	49
51	1.4.2.2.3	Perform cost/benefit analysis (CBAM)	50
52	1.4.2.2.4	Document software architecture	51
53	**1.4.2.3**	**− Design demonstration planning and conduct**	
54	1.4.2.3.1	Select architectural baseline critical use-cases	48
55	1.4.2.3.2	Select architectural baseline critical functionality	54
56	1.4.2.3.3	Design architectural baseline components	55
57	**1.4.3**	**+ Construction phase design modeling**	
60	1.4.4	Transition phase architecture design maintenance	
61	**1.5**	**− Implementation**	
62	1.5.1	Inception phase component prototyping	
63	**1.5.2**	**− Elaboration phase component implementation**	
64	**1.5.2.1**	**− Critical component coding demonstration integration (architectural baseline)**	
65	1.5.2.1.1	Implement architectural baseline components	56,62
66	1.5.2.1.2	Integrate architectural baseline components	65FF
71	**1.5.3**	**+ Construction phase implementation**	
72	1.5.4	Transition phase component maintenance	
73	**1.6**	**− Assessment**	
74	1.6.1	Inception phase assessment planning	
75	**1.6.2**	**− Elaboration phase assessment**	
76	1.6.2.1	Select and define architectural baseline tests	53
77	1.6.2.2	Implement architecture test scenario	76,66
78	1.6.2.3	Test architectural baseline	77

Note dual predecessors. Default relationship is Finish-to-Start. Here, we have defined a Start-to-Start relationship with an added lag of 5 days

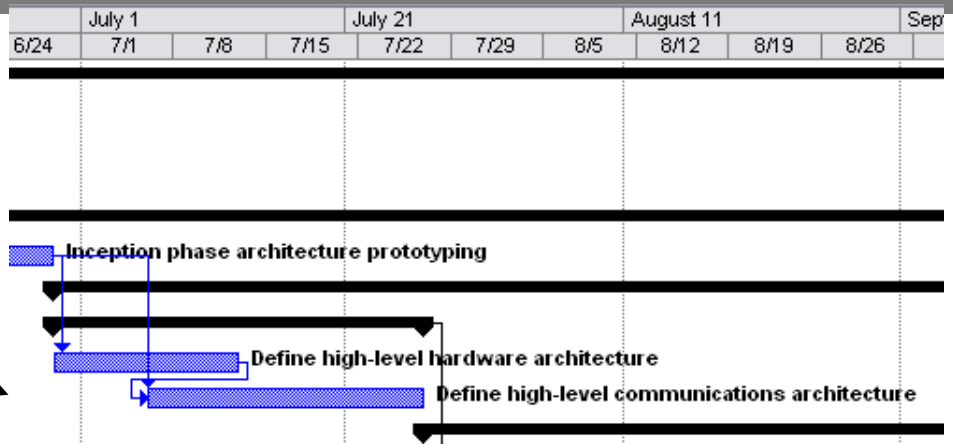
Here, we have defined a Finish-to-Finish relationship: this is common for implementation/integration task pairs

Critical path: early/late start and finish with float (slack)

	Name	Duration	Early Start	Early Finish	Late Start	Late Finish	Free Slack
1	**- Automatic Toll Booth System**	**104.5 days?**	**Fri 6/15/07**	**Thu 11/8/07**	**Fri 6/15/07**	**Thu 11/8/07**	**0 days?**
2	**+ Management**	**1 day?**	**Fri 6/15/07**	**Fri 6/15/07**	**Wed 11/7/07**	**Thu 11/8/07**	**103.5 days?**
20	**+ Environment**	**1 day?**	**Fri 6/15/07**	**Fri 6/15/07**	**Wed 11/7/07**	**Thu 11/8/07**	**103.5 days?**
33	**+ Requirements**	**1 day?**	**Fri 6/15/07**	**Fri 6/15/07**	**Wed 11/7/07**	**Thu 11/8/07**	**103.5 days?**
42	**- Design**	**90 days?**	**Fri 6/15/07**	**Thu 10/18/07**	**Fri 6/15/07**	**Thu 11/8/07**	**0 days?**
43	Inception phase architecture prototyping	10 days?	Fri 6/15/07	Thu 6/28/07	Fri 6/15/07	Thu 6/28/07	0 days?
44	**- Elaboration phase architecture baselining**	**80 days?**	**Fri 6/29/07**	**Thu 10/18/07**	**Fri 6/29/07**	**Thu 10/18/07**	**0 days?**
45	**- System architecture definition**	**20 days?**	**Fri 6/29/07**	**Thu 7/26/07**	**Fri 6/29/07**	**Thu 7/26/07**	**0 days?**
46	Define high-level hardware architecture	10 days?	Fri 6/29/07	Thu 7/12/07	Fri 6/29/07	Thu 7/12/07	0 days?
47	Define high-level communications architecture	15 days?	Fri 7/6/07	Thu 7/26/07	Fri 7/6/07	Thu 7/26/07	0 days?
48	**- Architecture design modeling**	**40 days?**	**Fri 7/27/07**	**Thu 9/20/07**	**Fri 7/27/07**	**Thu 9/20/07**	**0 days?**
49	Perform ABD (architecture-based design)	10 days?	Fri 7/27/07	Thu 8/9/07	Fri 7/27/07	Thu 8/9/07	0 days?
50	Perform quality attribute analysis (ATAM)	8 days?	Fri 8/10/07	Tue 8/21/07	Fri 8/10/07	Tue 8/21/07	0 days?
51	Perform cost/benefit analysis (CBAM)	7 days?	Wed 8/22/07	Thu 8/30/07	Wed 8/22/07	Thu 8/30/07	0 days?
52	Document software architecture	15 days?	Fri 8/31/07	Thu 9/20/07	Fri 8/31/07	Thu 9/20/07	0 days?
53	**- Design demonstration planning and conduct**	**20 days?**	**Fri 9/21/07**	**Thu 10/18/07**	**Fri 9/21/07**	**Thu 10/18/07**	**0 days?**
54	Select architectural baseline critical use-cases	2.5 days?	Fri 9/21/07	Tue 9/25/07	Fri 9/21/07	Tue 9/25/07	0 days?
55	Select architectural baseline critical functionality	2.5 days?	Tue 9/25/07	Thu 9/27/07	Tue 9/25/07	Thu 9/27/07	0 days?
56	Design architectural baseline components	15 days?	Fri 9/28/07	Thu 10/18/07	Fri 9/28/07	Thu 10/18/07	0 days?
57	**+ Construction phase design modeling**	**1 day?**	**Fri 6/15/07**	**Fri 6/15/07**	**Wed 11/7/07**	**Thu 11/8/07**	**103.5 days?**
60	Transition phase architecture design maintenance	1 day?	Fri 6/15/07	Fri 6/15/07	Wed 11/7/07	Thu 11/8/07	103.5 days?
61	**- Implementation**	**100 days?**	**Fri 6/15/07**	**Thu 11/1/07**	**Fri 10/5/07**	**Thu 11/8/07**	**4.5 days?**
62	Inception phase component prototyping	10 days?	Fri 6/15/07	Thu 6/28/07	Fri 10/5/07	Thu 10/18/07	80 days?
63	**- Elaboration phase component implementation**	**10 days?**	**Fri 10/19/07**	**Thu 11/1/07**	**Fri 10/19/07**	**Thu 11/1/07**	**0 days?**
64	**- Critical component coding demonstration integration (architectural baseline)**	**10 days?**	**Fri 10/19/07**	**Thu 11/1/07**	**Fri 10/19/07**	**Thu 11/1/07**	**0 days?**
65	Implement architectural baseline components	10 days?	Fri 10/19/07	Thu 11/1/07	Fri 10/19/07	Thu 11/1/07	0 days?
66	Integrate architectural baseline components	5 days?	Fri 10/26/07	Thu 11/1/07	Fri 10/26/07	Thu 11/1/07	0 days?
67	**+ Construction phase implementation**	**1 day?**	**Fri 6/15/07**	**Fri 6/15/07**	**Wed 11/7/07**	**Thu 11/8/07**	**103.5 days?**
72	Transition phase component maintenance	1 day?	Fri 6/15/07	Fri 6/15/07	Wed 11/7/07	Thu 11/8/07	103.5 days?
73	**- Assessment**	**104.5 days?**	**Fri 6/15/07**	**Thu 11/8/07**	**Fri 10/26/07**	**Thu 11/8/07**	**0 days?**
74	Inception phase assessment planning	5 days?	Fri 6/15/07	Thu 6/21/07	Thu 11/1/07	Thu 11/8/07	99.5 days?
75	**- Elaboration phase assessment**	**14.5 days?**	**Fri 10/19/07**	**Thu 11/8/07**	**Fri 10/26/07**	**Thu 11/8/07**	**0 days?**
76	Select and define architectural baseline tests	5 days?	Fri 10/19/07	Thu 10/25/07	Fri 10/26/07	Thu 11/1/07	5 days?
77	Implement architecture test scenario	2.5 days?	Fri 11/2/07	Tue 11/6/07	Fri 11/2/07	Tue 11/6/07	0 days?
78	Test architectural baseline	2 days?	Tue 11/6/07	Thu 11/8/07	Tue 11/6/07	Thu 11/8/07	0 days?

Schedule Compression

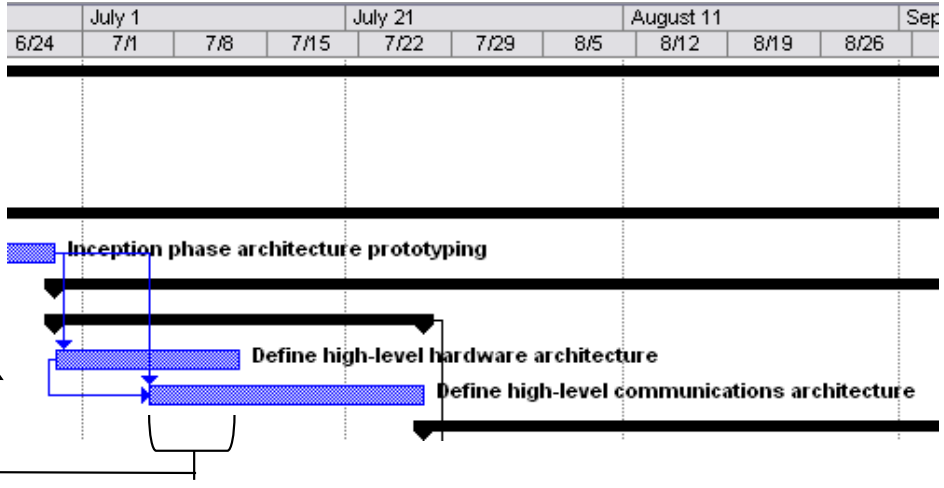
Use Finish-to-Start dependency with 5 day *negative lag* (-5) for successor to get reasonable relationship between two tasks. Equivalent to a 5 day lead for successor



Alternatively:

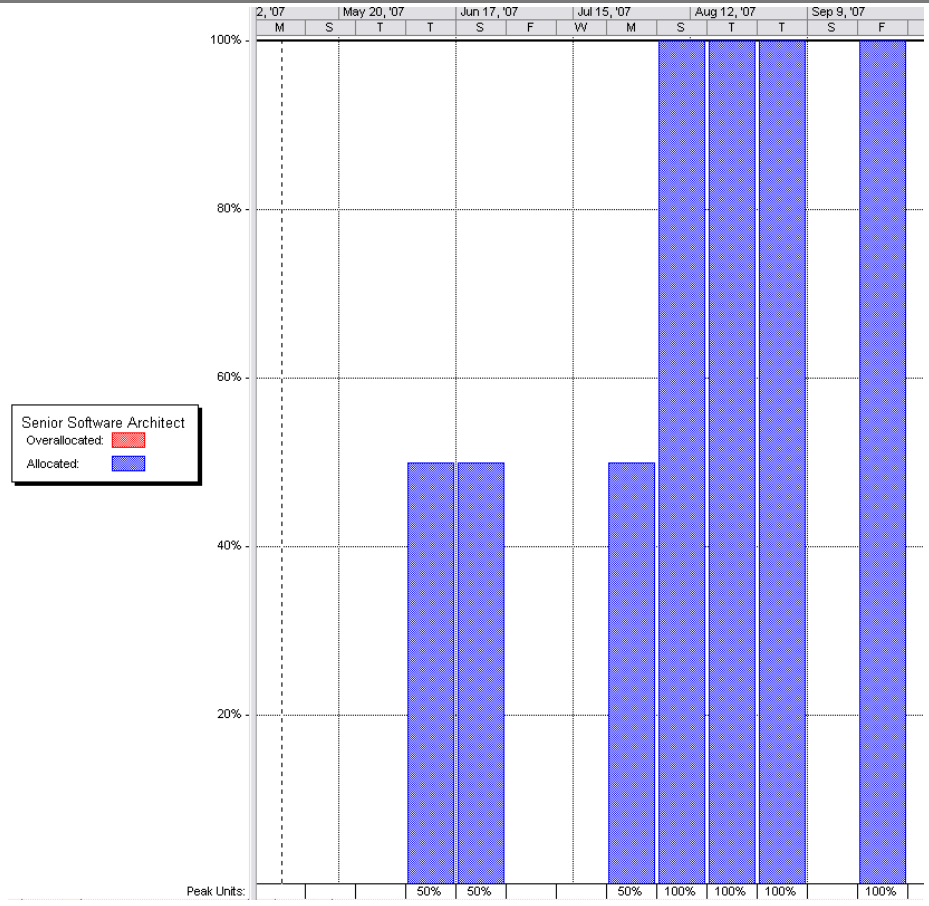
Use Start-to-Start dependency with 5 day *lag* for successor to get reasonable relationship between two tasks

Note schedule compression of 5 days



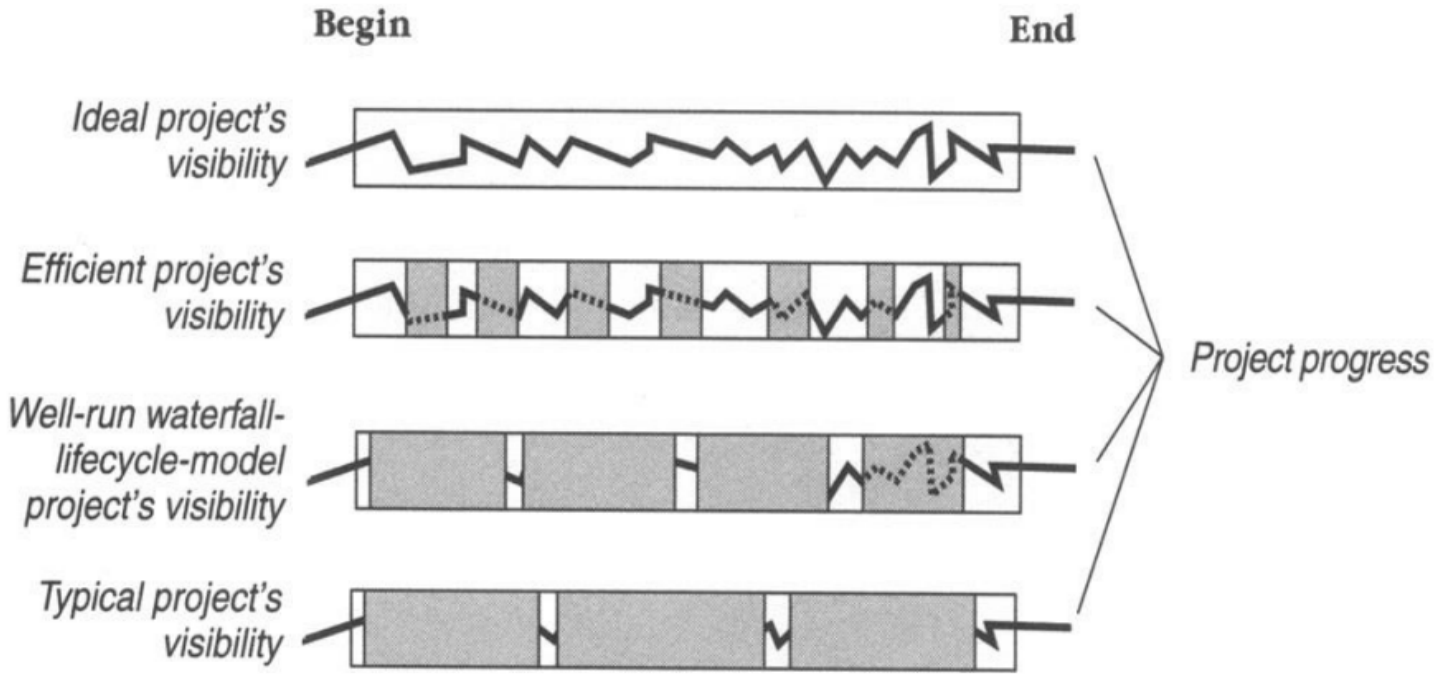
Determining resource utilization

- Resource graphs show the utilization of individual resources as a function of time
- Resource graphs can be used to identify over- and under-utilized resources
- Resource leveling can be used to balance utilization and/or to shift tasks to lower-cost resources



Project Tracking

Tracking and Visibility



Percent Complete

- **Illusion of Progress:** The early phases (design/specification) are done, which can give a **false sense of completion**. The reader of the table may get a feeling that around 50% of the project is done, but the most effort-intensive and risk-prone phases, **coding and testing,** are still underway or not even started: What it the actual percentage?

Task	Complete?
Conceptual Design	Complete
Program Specification	Complete
Coding	In Progress
Documentation	In Progress
User Manual Production	Not Started
Testing	Not Started

Percent Complete

660 / 2100 =

Task	How Complete?
Conceptual Design	200/200
Program Specification	300/300
Coding	150/600
Documentation	10/100
User Manual Production	0/400
Testing	0/500

660 / 2100 * 100 = 31.4% complete

Earned Value

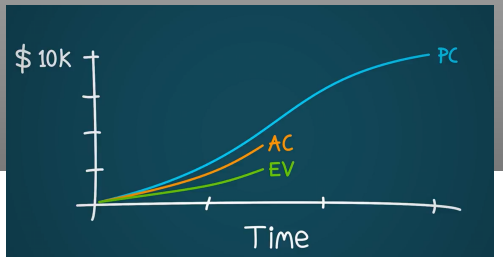
* They might give you some values and you'll need to calculate.

- EVM is a project **control** methodology that integrates **scope**, **schedule**, and **cost**.
- It provides a **quantitative** **basis (طريقة حسابية)** for measuring project performance.
- Enables early detection of deviations and supports **forecasting**.
- Provides a consistent method of project progress and performance

Metric	Definition	Formula
PV (Planned Value)	Budgeted cost of work scheduled	% planned × total budget
EV (Earned Value)	Budgeted cost of work performed	% complete × total budget
AC (Actual Cost)	Actual cost incurred for work performed	Tracked from financials
CPI (Cost Performance Index)	Cost efficiency	EV ÷ AC
SPI (Schedule Performance Index)	Schedule efficiency	EV ÷ PV

Fig. Key EVM Metrics and Definition

Using Earned Value Tracking



Setup:

1. Establish a WBS to divide the project into manageable components
2. Identify the activities required under each WBS element
3. Estimate the effort required for each activity (assign labor hours or cost)
4. Schedule tasks: start/end dates and resources
5. Assign planned value to each task

Execution:

6. Update the schedule by reporting activity progress (% complete for each task)
7. Record actual costs on the activities (real expenditures)
8. Calculate EV metrics: compute EV, PV, AC, CPI, SPI
9. Analyze the data (variance): CV = EV – AC ; SV = EV - PV
10. Take corrective action: Reallocate resources, adjust scope, revise schedule, etc.

Earned Value Example

$\frac{100}{1000} \times Budget$

- Total Budget: 1000 person-hours
- Task A: Estimated 100 hours
  - Planned Value (PV) = 100
  - Actual Cost (AC) = 120
  - % Complete = 100% → Earned Value (EV) = 100
- Cost Performance Index: CPI = 100 ÷ 120 = 0.83 → Over budget
- Schedule Performance Index: SPI = 100 ÷ 100 = 1.0 → On schedule

Earned Value Tracking

- Avoid “fuzzy” milestones:
  - Earned value credit is given only when the task is 100% complete
  - Partially completed tasks are NOT given partial credit (in most software projects)
  - Large tasks can/must be broken into subtasks
- Size tasks to allow frequent completions (e.g., 8–80 hour rule).
- Use objective progress measures (e.g., demos, deliverables).
- Encourage honest reporting and team buy-in.

Project Velocity (PV)

$$\text{PV in story weeks} \equiv \sum_{\text{All stories completed in the iteration}} \text{Time estimated for the story}$$

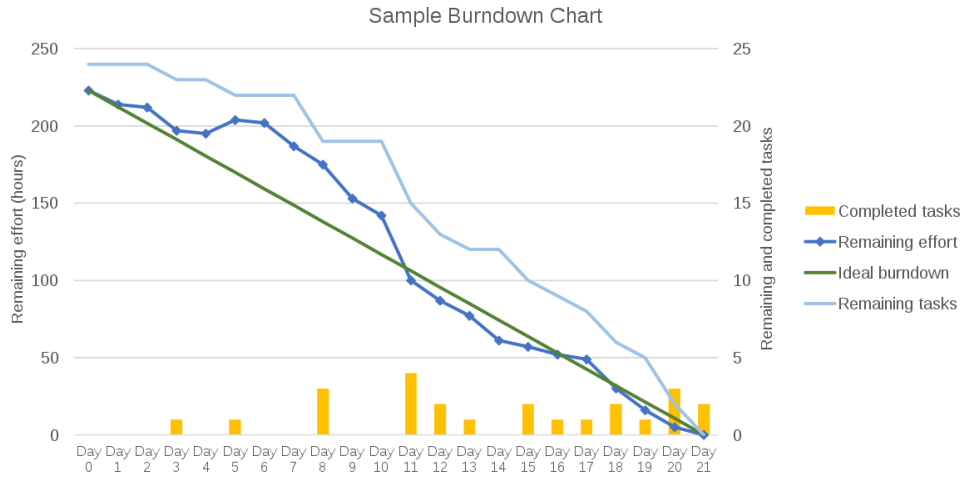
- In Agile (iterative) context: Use story points as effort units.
- Track velocity to estimate future capacity: Project velocity tells you how many story points you can allocate to the next iteration.
- Combine EV with burn-down charts for visual tracking.
- The customer gets to pick stories that equal the project velocity.

Project Velocity (PV)

- The metric is calculated by reviewing work the team successfully completed during previous sprints; for example, if the team completed 10 stories during a two-week sprint and each story was worth 3 story points, then the team's velocity is 30 story points per sprint.
- Generally, velocity remains somewhat constant during a development project, which makes it a useful metric for estimating how long it will take a team to complete a software development project.
- If the product backlog has 300 story points, and the team is averaging 30 story points per sprint, it can be estimated that team members will require 10 more sprints to complete work. If each sprint lasts two weeks, then the project will last 20 more weeks.

Burn down chart

- A burn down chart is a visual representation of tracks how much work remains in a project or sprint over time.
- The chart typically starts with a total amount of work to be done and a deadline, and as work is completed, the chart shows the remaining amount of work decreasing.
- Burn down charts offer **real-time visibility** into progress. They:
  - Encourage **daily accountability**
  - Help **forecast completion**
  - Reveal **early warning signs** of slippage
  - Support **data-driven retrospectives**



Summary

- Planning, Estimating, Scheduling, and Tracking are a continuum
- Projects need to be partitioned for manageability
  - Work Breakdown Structures are a great way to do this (or stories for agile)
- Sequencing Tasks & Activities is vital
  - Gantt Charts allow quick reference
  - Network Techniques such as Precedence Network Diagrams, the Critical Path Method, and PERT Charts are useful tools
- Project Tracking is important for project visibility
  - The Earned Value Technique is a key tool in this
- The project charter typically includes information such as the project's objectives, scope, timelines, budget, risks, and assumptions. It also identifies the project team members and their roles and responsibilities.