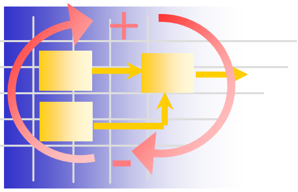
ESD.36J System & Project Management

Lecture 2



Network Planning Techniques: CPM-PERT

Instructor(s)

Prof. Olivier de Weck

09 Sept 2003

Massachusetts Institute of Technology

Today's Agenda

- Overview of PM methods and tools
- CPM 101
- Critical Paths, Slack
- Probabilistic Task Times
- Task "Crashing" and Cost
- Conclusions and Class Discussions
- Introduce HW1

History of Project Management

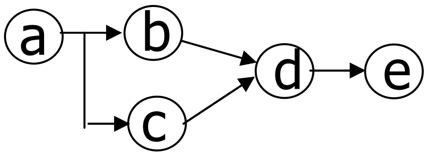
- Big Projects since antiquity
  - Pyramids (Egypt), Great Wall (China)
  - Enormous workforce, but no documented evidence of formal project management
- Formal Project Management
  - Henry Gantt (1861-1919) → bar chart
  - 1957 Sputnik Crisis → revival of "scientific management"
  - Polaris (1958) → Project Evaluation and Review Technique (PERT)
  - DuPont Company (1960) → Critical Path Method (CPM)
  - 1960's NASA projects: Mercury, Gemini, Apollo
    - Work Breakdown Structures (WBS)
    - Cost and Schedule Tracking, Configuration Management

Comments about early PM

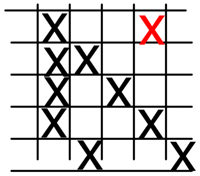
- Project decomposition necessary due to complexity
- Resource allocation and workload smoothing
- Schedule urgency .."before the decade is out" *JFK*
- Circumstances
  - Complex Relations between Government and Contractors
  - "Shielded" from Society, Competition, Regulations
  - Cold War Pressures for Nuclear Power, Space Race ..
- Other Innovations
  - Project Manager as a central figure
  - Beginnings of Matrix Organization
  - "Earned Value" – adopted by USAF (1963)
- Professionalization since 1969
  - Diffusion into other industries: computers, automotive …
  - Project Management Institute (PMI) founded – PMBOK
  - ISO 10006:1997 Quality in Project Management
  - Recent criticism about PM standards as "bureaucratic"

Fundamental Approaches

- How to represent task relationships?
- Network-based (graph theory) methods
    - CPM, PERT, ….
    - Task is a node or an arc

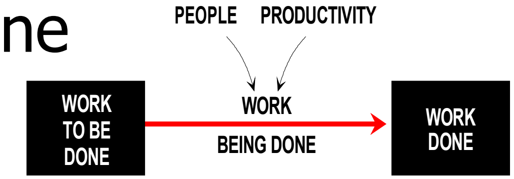


- Matrix-based methods
    - DSM - Tasks are columns and rows
    - Interrelationships are off-diagonal entries



X			X	
X	X			
X		X		
X			X	
	X			X

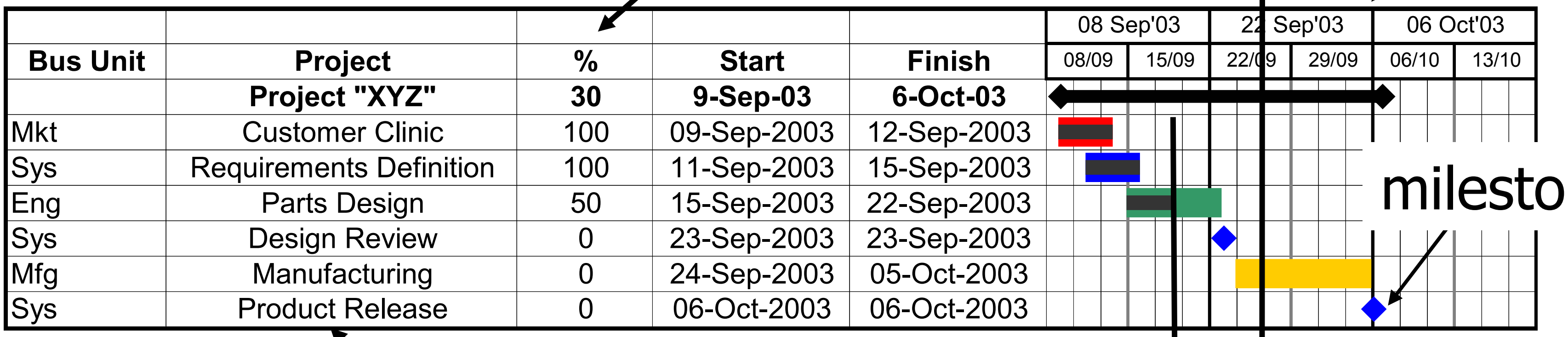
- System Dynamics
    - Feedback loops, causal relationships
    - Stocks and flows simulation
    - Tasks that are done or waiting to be done are stocks – "amount of work"
    - Doing project work causes a "flow"



Gantt Charts

- Attributed to Henry Gantt – most popular PM tool (80%)
- Used to plan big shipbuilding projects (cargo ships WWI)
- Graphical way of showing task durations, project schedule
- Does not explicitly show relationships between tasks
- Limited use for project tracking
- Easy to understand

Gantt Chart Builder System (Excel) 1.6



Bus Unit	Project	%	Start	Finish
	Project "XYZ"	**30**	**9-Sep-03**	**6-Oct-03**
Mkt	Customer Clinic	100	09-Sep-2003	12-Sep-2003
Sys	Requirements Definition	100	11-Sep-2003	15-Sep-2003
Eng	Parts Design	50	15-Sep-2003	22-Sep-2003
Sys	Design Review	0	23-Sep-2003	23-Sep-2003
Mfg	Manufacturing	0	24-Sep-2003	05-Oct-2003
Sys	Product Release	0	06-Oct-2003	06-Oct-2003

completion

today

calendar

milestone

tasks

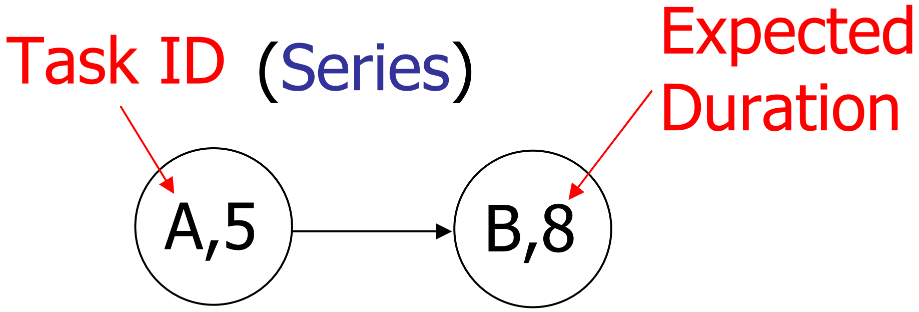
actual

planned

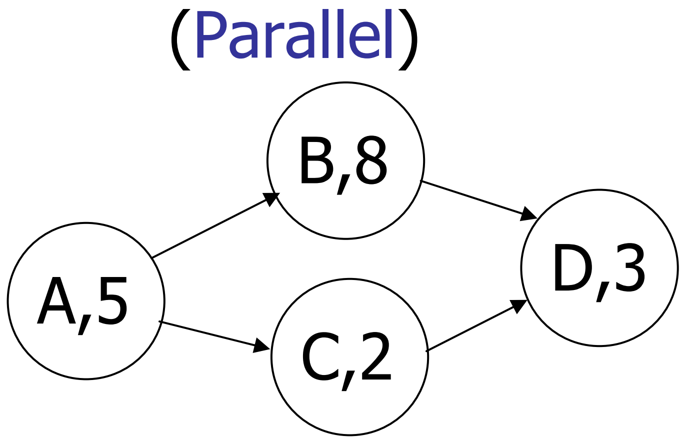
Δ

CPM 101

- Represent a project (set of task) as a network using graph theory
  - Capture task durations
  - Capture task logic (dependencies)



"B can only start after A is completed"



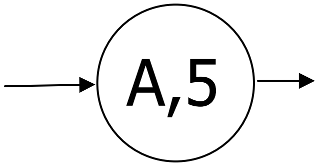
"B and C do not depend on each other"

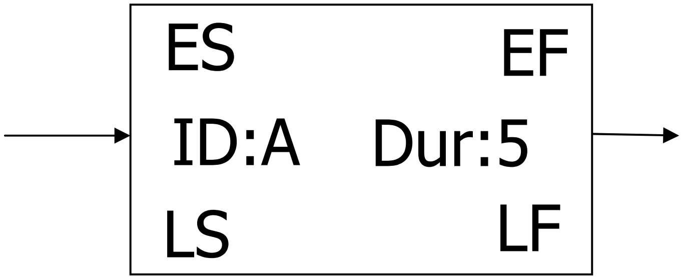
CPM Assumptions

- Project consists of a collection of well defined tasks (jobs)
- Project ends when all jobs completed
- Jobs may be started and stopped independently of each other within a given sequence (no "continuous-flow" processes)
- Jobs are ordered → "technological sequence"

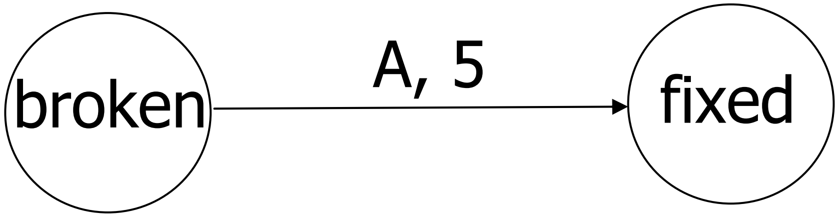
Task Representations

- Tasks as Nodes of a Graph
  - Circles
  - Boxes



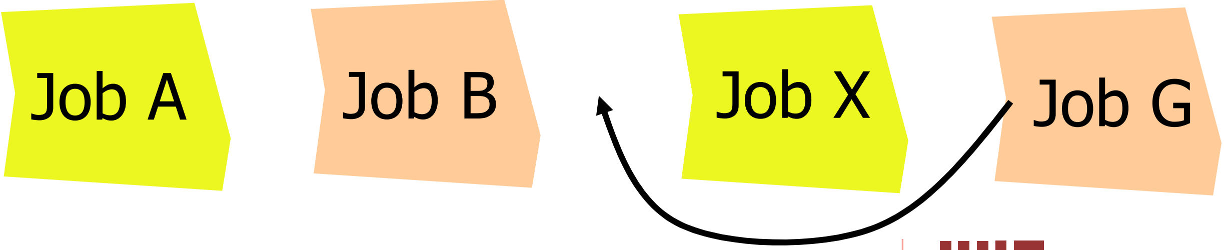


- Tasks as Arcs of a Graph
  - Tasks are uni-directional arrows
  - Nodes now represent “states” of a project
  - Kelley-Walker form



Work Breakdown Structure

- Used to create the task (job) list
- Tree-decomposition of project tasks
- WBS identifies "terminal elements"
- The key starting point for project planning
- Required by US Govt as part of SOW
- Can be activity-oriented or deliverable- oriented
- Use "sticky-notes" method early on
- Carl L. Pritchard. *Nuts and Bolts Series 1: How to Build a Work Breakdown Structure*. ISBN 1890367125



WBS – Painting a Room

- 1 Prepare materials
  - 1.1 Buy paint
  - 1.2 Buy brushes/rollers
  - 1.3 Buy wallpaper remover
- 2. Prepare room
  - 2.1 Remove old wallpaper
  - 2.2 Remove detachable decorations
  - 2.3 Cover floor with old newspapers
  - 2.4 Cover electrical outlets/switches with tape
  - 2.5 Cover furniture with sheets
- 3. Paint the room
- 4. Clean up the room
  - 4.1 Dispose or store left over paint
  - 4.2 Clean brushes/rollers
  - 4.3 Dispose of old newspapers
  - 4.4 Remove covers

http://www.wikipedia.org

WBS Guidelines

- No more than 100-200 terminal elements, if more → use subprojects
- Can be up to 3-4 Levels deep
- Not more than 5-9 jobs at one level
  - Human cognitive "bandwidth" only 3 bits=2^3=8
  - Short term memory for most people 5-9 items
  - Poorer planning if "too-fine grained" – dilution of attention
  - The more tasks there are, the more intricate dependencies there will be to keep track of
- Jobs should be of similar size/complexity
- Manageable chunks → sense of progress
- Level of graininess → very difficult to answer

Task List

- List all tasks in a table with
  - Identifying symbol (tag)
  - Task description
  - Immediate prerequisite jobs
  - Expected task duration
- Arrange jobs in "technological order"
  - No job appears in the list until all its predecessors have been listed
  - Iterations are NOT allowed → "cycle error"
  - Job **a** precedes **b** precedes **c** precedes **a**
  - We will discuss <u>iterations</u> a lot in this class !!!

Simple Example: Job List

- Two Parts X and Y: Manufacture and Assembly

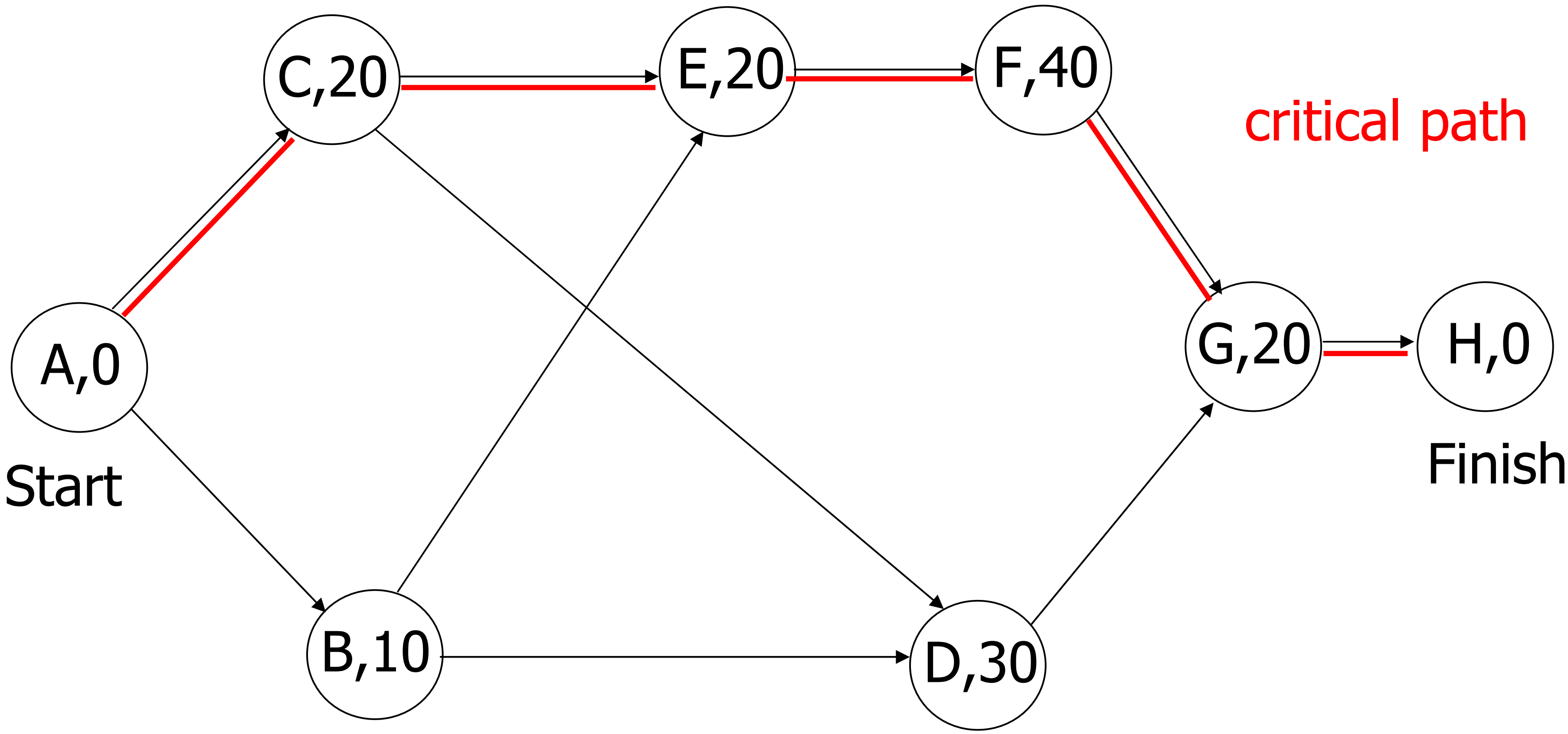
Job #	Description	Immediate Predecessors	Time [min]
A	Start		0
B	Get materials for X	A	10
C	Get materials for Y	A	20
D	Turn X on lathe	B,C	30
E	Turn Y on lathe	B,C	20
F	Polish Y	E	40
G	Assemble X and Y	D,F	20
H	Finish	G	0

Project Graph

- Each job is drawn on a graph as a circle*
- Connect each job with immediate predecessor(s) – unidirectional arrows "→"
- Jobs with no predecessor connect to "start"
- Jobs with no successors connect to "finish"
- "start" and "finish" are pseudo-jobs of length 0
- A finite number of "arrow paths" from "start" to "finish" will be the result
- Total time of each path is the sum of job times
- Path with the longest total time → **"critical path"**
- There can be multiple critical paths → minimum time to complete project

* or other symbol, see before

Project Graph



4 unique paths: $\underbrace{A,C,E,F,G,H}_{100}$; $\underbrace{A,C,D,G,H}_{70}$; $\underbrace{A,B,D,G,H}_{60}$; $\underbrace{A,B,E,F,G,H}_{90}$

Critical Path

- CP is the "bottleneck route"
- Shortening or lengthening tasks on the critical path directly affects project finish
- Duration of "non-critical" tasks is irrelevant
- "Crashing" all jobs is ineffective, focus on the few % of jobs that are on the CP
- "Crashing" tasks can shift the CP to a different task
- Shortening tasks – technical and economical challenge
  - How can it be done?
- Previously non-critical tasks can become critical
- Lengthening of non-critical tasks can also shift the critical path (see HW1).

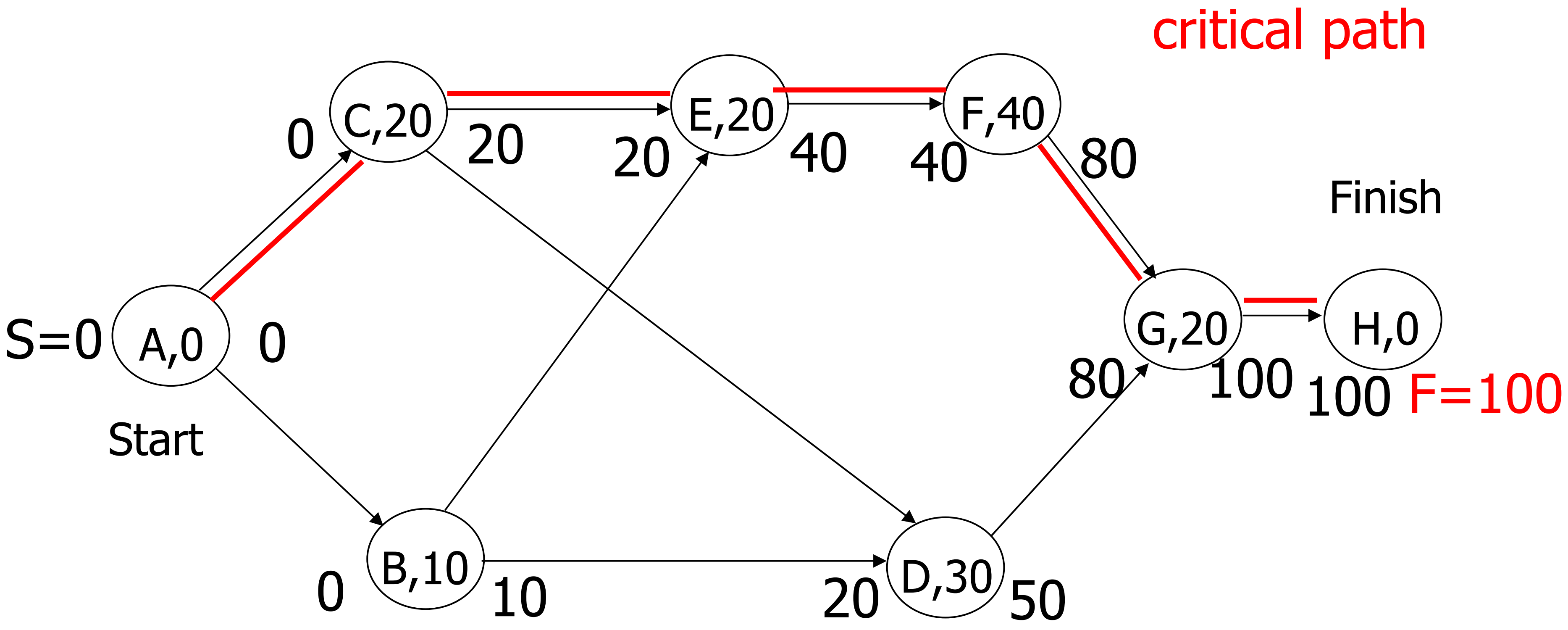
Critical Path Algorithm

- For large projects there are many paths
- Need a algorithm to identify the CP efficiently
- Develop information about each task in context of the overall project
- Times
  - Start time (S)
  - For each job: Earliest Start (ES)
    - Earliest start time of a job if all its predecessors start at ES
  - Job duration: t
  - Earliest Finish (EF)=(ES)+t
- Finish time (F) – earliest finish time of the overall project
- Show algorithm using project graph

CP Algorithm

1. Mark the value of **S** to left and right of Start
2. Consider any new unmarked job, all of whose predecessors have been marked. Mark to the left of the new job the largest number to the right of its immediate predecessors: (**ES**)
3. Add to **ES** the job time **t** and mark result to the right (**EF**)
4. Stop when **Finish** has been reached

CP Algorithm - Graphical



Latest Start and Finish Times

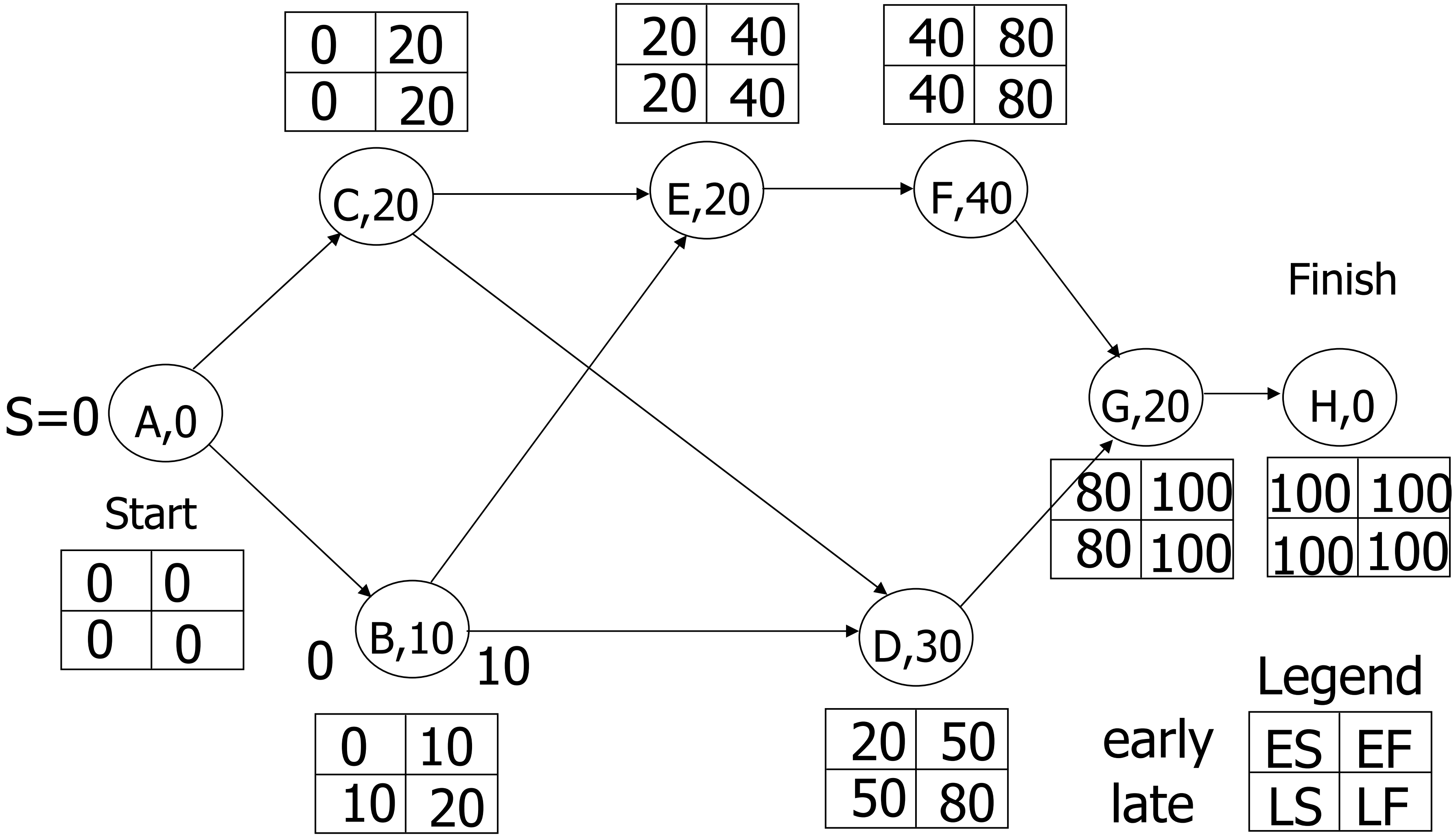
- Set target finish time for project: **T** >= **F**
- Usually target is a specific calendar date, e.g. October 1, 2007
- When is the latest date the project can be started?
- Late Finish (**LF**) - latest time a job can be finished, without delaying the project beyond its target time (**T**)
- Late Start: **LS** = **LF-t**

Determine LF and LS

- Work from the end of the project: **T**

1. Mark value of **T** to left and right of Finish
2. Consider any new unmarked job, all of whose successors have been marked - mark to the right the smallest **LS** time marked to the left of any of its immediate successors
3. Subtract from this number, **LF**, the job time **t** and mark result to the left of the job: **LS**
4. Continue upstream until Start has been reached, then stop

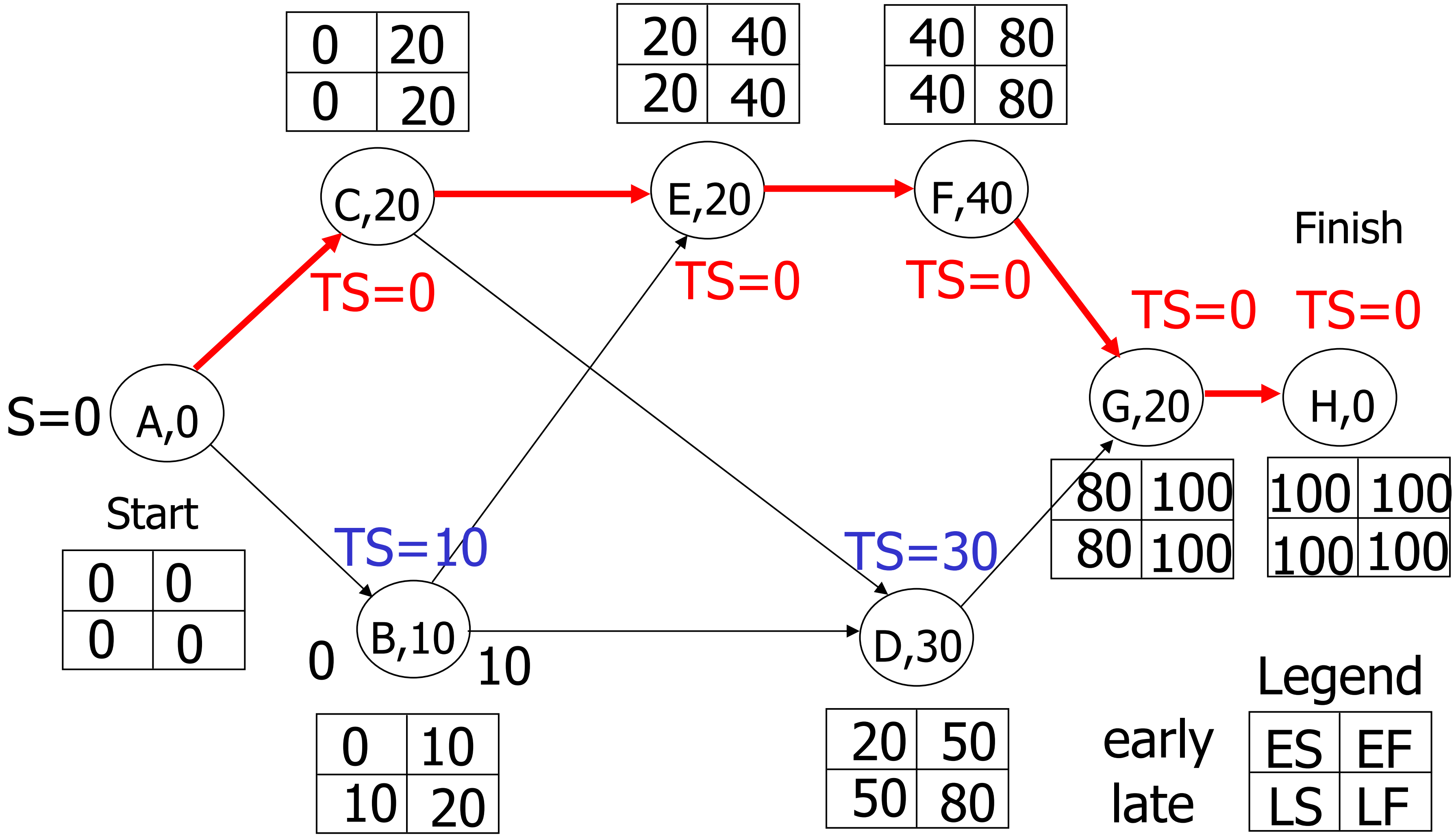
LS and LF : Project Graph



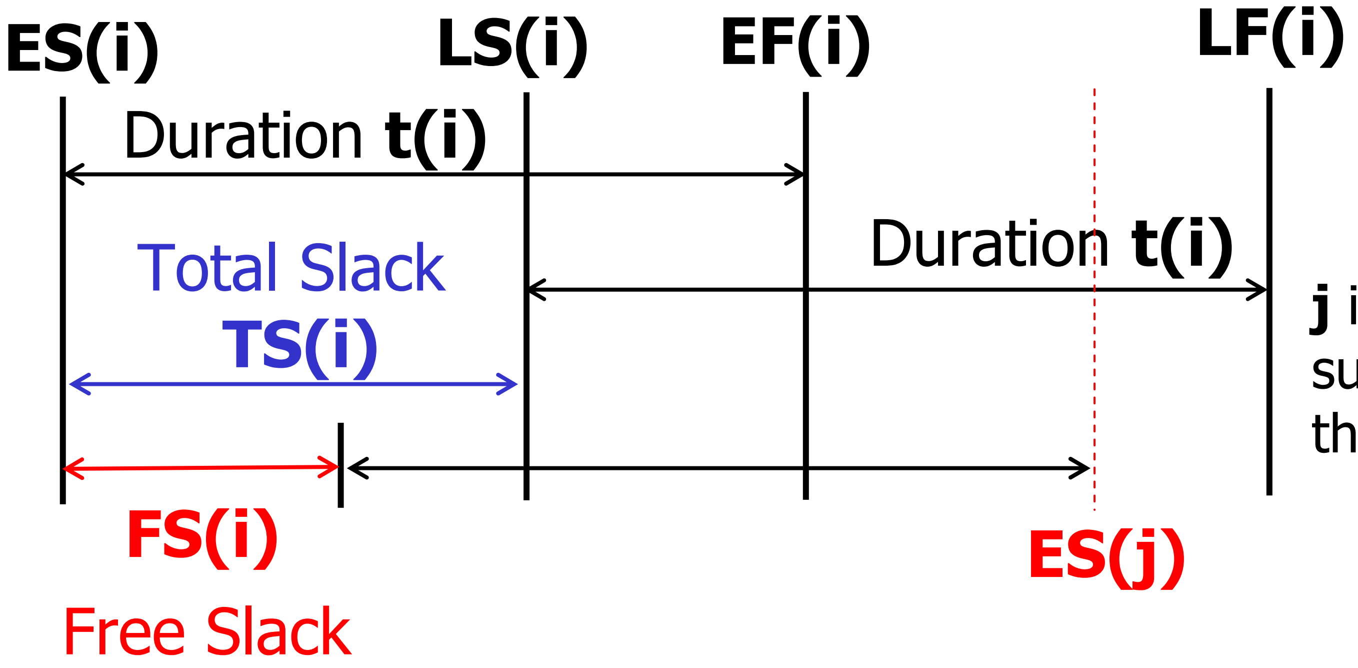
Slack

- Some tasks have **ES**=**LS** --> no slack
- Total Slack of a task **TS**=**LS-ES**
- **Maximum amount of time a task may be delayed beyond its early start without delaying project completion**
- Slack time is precious … managerial freedom, don't squander it unnecessarily
  - e.g. resource, work load smoothing
- When **T**=**F** then all critical tasks have **TS=0**
- At least one path from Start->Finish with critical jobs only
- When **T**>**F**, then all critical jobs have **TS**=**T-F**

Project Graph - Slack



Task Times Detail - Task i



ES(i)

LS(i)

EF(i)

LF(i)

Duration **t(i)**

Total Slack

TS(i)

Duration **t(i)**

FS(i)

Free Slack

ES(j)

j is the immediate successor of **i** with the smallest ES

j>i

- Free Slack (**FS**) is the amount a job can be delayed with delaying the Early Start (ES) of any other job.

FS<=TS always

Main CPM Errors

- Estimated job times are wrong
- Predecessor relationships may contain cycles → "cycle error"
- List of prerequisites contains more than the immediate predecessors, e.g. **a→b**, **b→c** and **a,b→c**
- Overlooked some predecessor relationships
- Some predecessor relationships may be listed that are spurious
- and …. Some tasks/jobs may be missing !!!

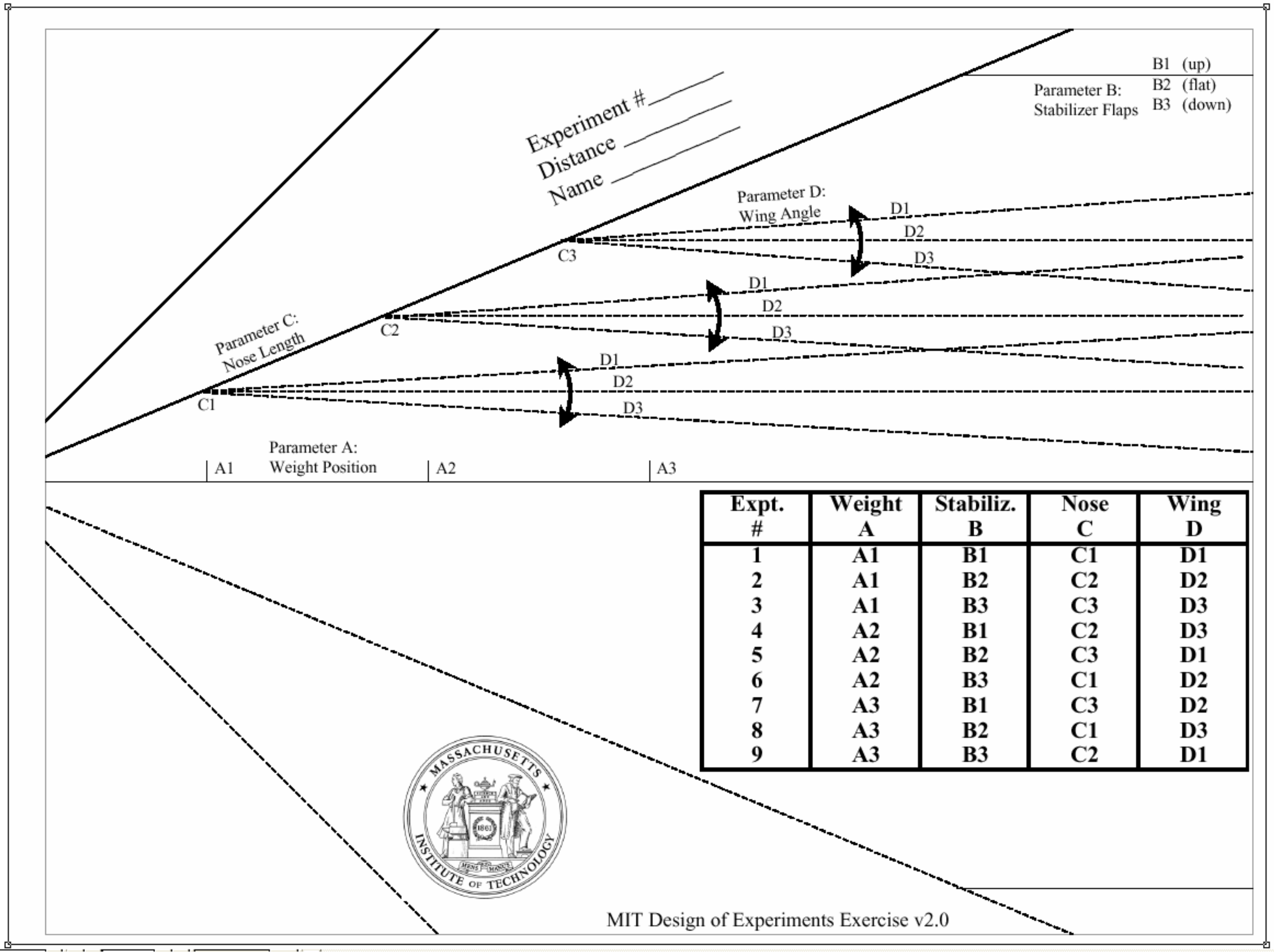
Gradual Refinement of CPM

- Job Times
  - Given rough time estimates construct CPM chart
  - Re-estimate times for **CP** and those with very small **TS**
  - Iterate until the critical path is stable
  - Focus attention on a subset of tasks
- Predecessor Relationships
  - Check algorithmically for cycle errors and pre-predecessor errors
  - Cancel all except immediate predecessor relationships
- Wrong or Missing Facts
  - Cannot be detected by computers!

How long does a task take?

- Conduct a small in-class experiment
- Fold MIT paper airplane
  - Have sheet & paper clip ready in front of you
  - Paper airplane type will be announced, e.g. A1-B1-C1-D1
  - Build plane, focus on quality rather than speed
  - Note the completion time in seconds +/- 5 [sec]
- Plot results for class and discuss
  - Call out your time aloud
  - We will build a histogram and display results in real-time

MIT Paper Airplane



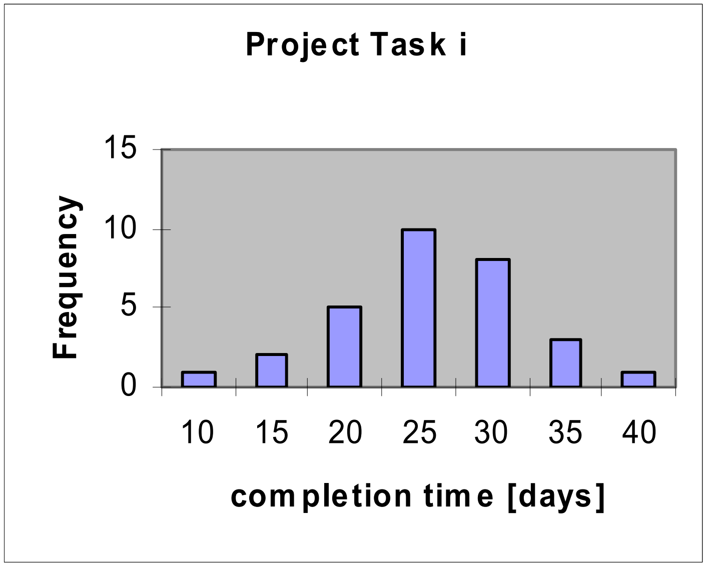
Expt. #	Weight A	Stabiliz. B	Nose C	Wing D
1	A1	B1	C1	D1
2	A1	B2	C2	D2
3	A1	B3	C3	D3
4	A2	B1	C2	D3
5	A2	B2	C3	D1
6	A2	B3	C1	D2
7	A3	B1	C3	D2
8	A3	B2	C1	D3
9	A3	B3	C2	D1

MIT Design of Experiments Exercise v2.0

Credit:
S. Eppinger

Job Duration Distribution

- Job task durations are stochastic in reality
- Actual duration affected by
  - Individual skills
  - Learning curves … what else?



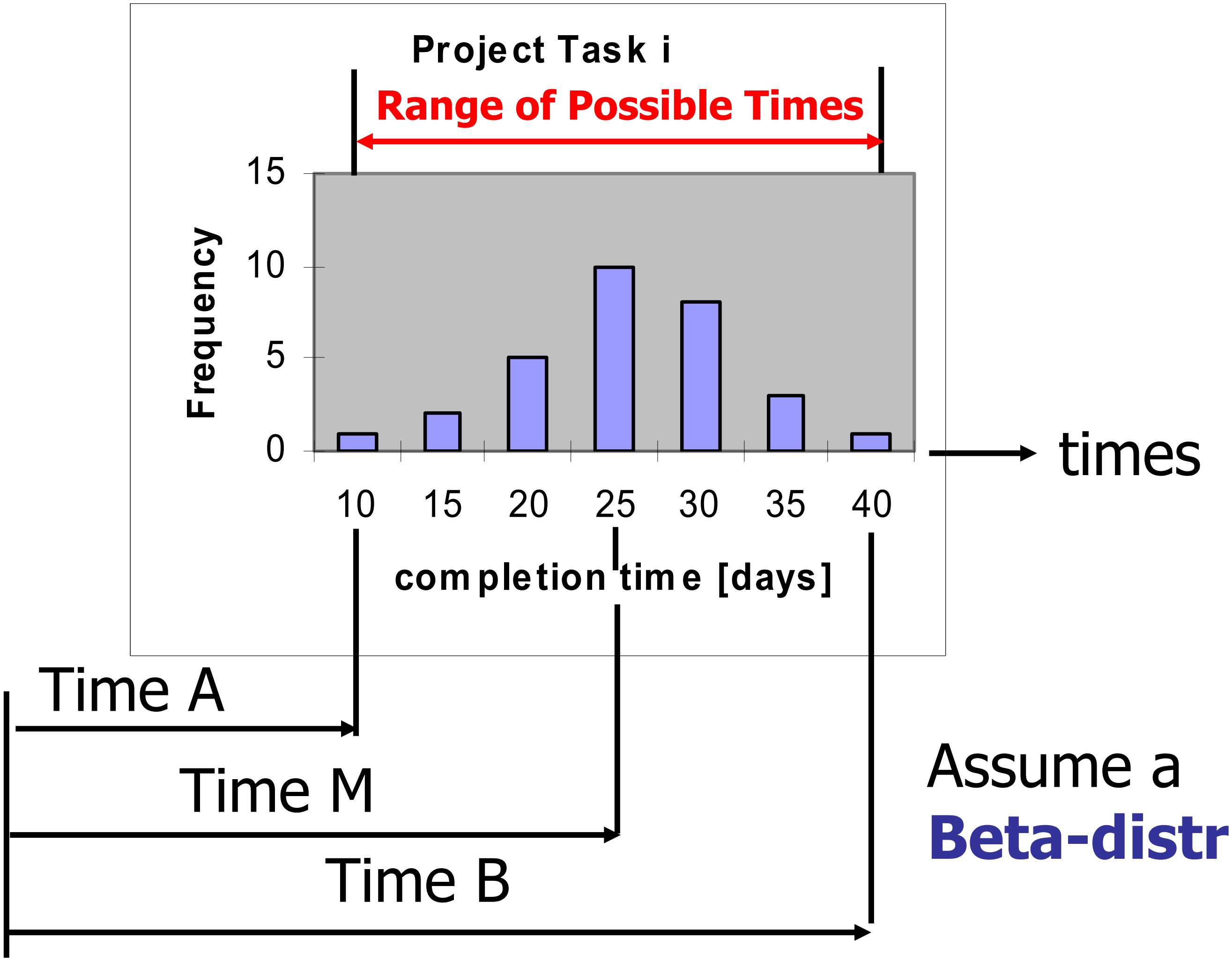
CPM vs PERT

- Difference how "task duration" is treated
- CPM assumes time estimates are <u>deterministic</u>
  - Obtain task duration from previous projects
  - Suitable for "construction"-type projects
- PERT treats durations as probabilistic
  - PERT = CPM + probabilistic task times
  - Better for R&D type projects
  - Limited previous data to estimate time durations
  - Captures schedule (and implicitly some cost) risk

PERT

- Project Evaluation and Review Technique
- Task time durations are treated as uncertain
  - **A** - optimistic time estimate
    - minimum time in which the task could be completed
    - everything has to go right
  - **M** - most likely task duration
    - task duration under “normal” working conditions
    - most frequent task duration based on past experience
  - **B** - pessimistic time estimate
    - time required under particularly “bad” circumstances
    - most difficult to estimate, includes unexpected delays
    - should be exceeded no more than 1% of the time

A-M-B Time Estimates



times

Time A

Time M

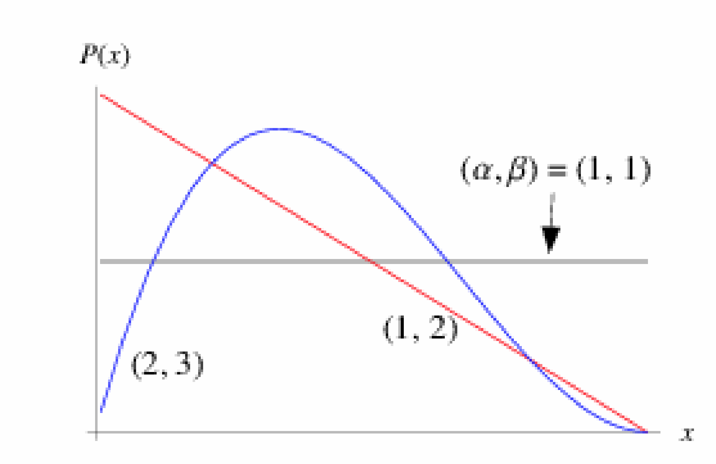
Time B

Assume a
Beta-distribution

Beta-Distribution

- All values are enclosed within interval $t \in [A, B]$
- As classes get finer - arrive at β-distribution
- Statistical distribution

pdf:



$$x \in [0,1]$$

$$\begin{aligned} P(x) &= \frac{(1-x)^{\beta-1} x^{\alpha-1}}{B(\alpha,\beta)} \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(1-x)^{\beta-1}x^{\alpha-1} \end{aligned}$$

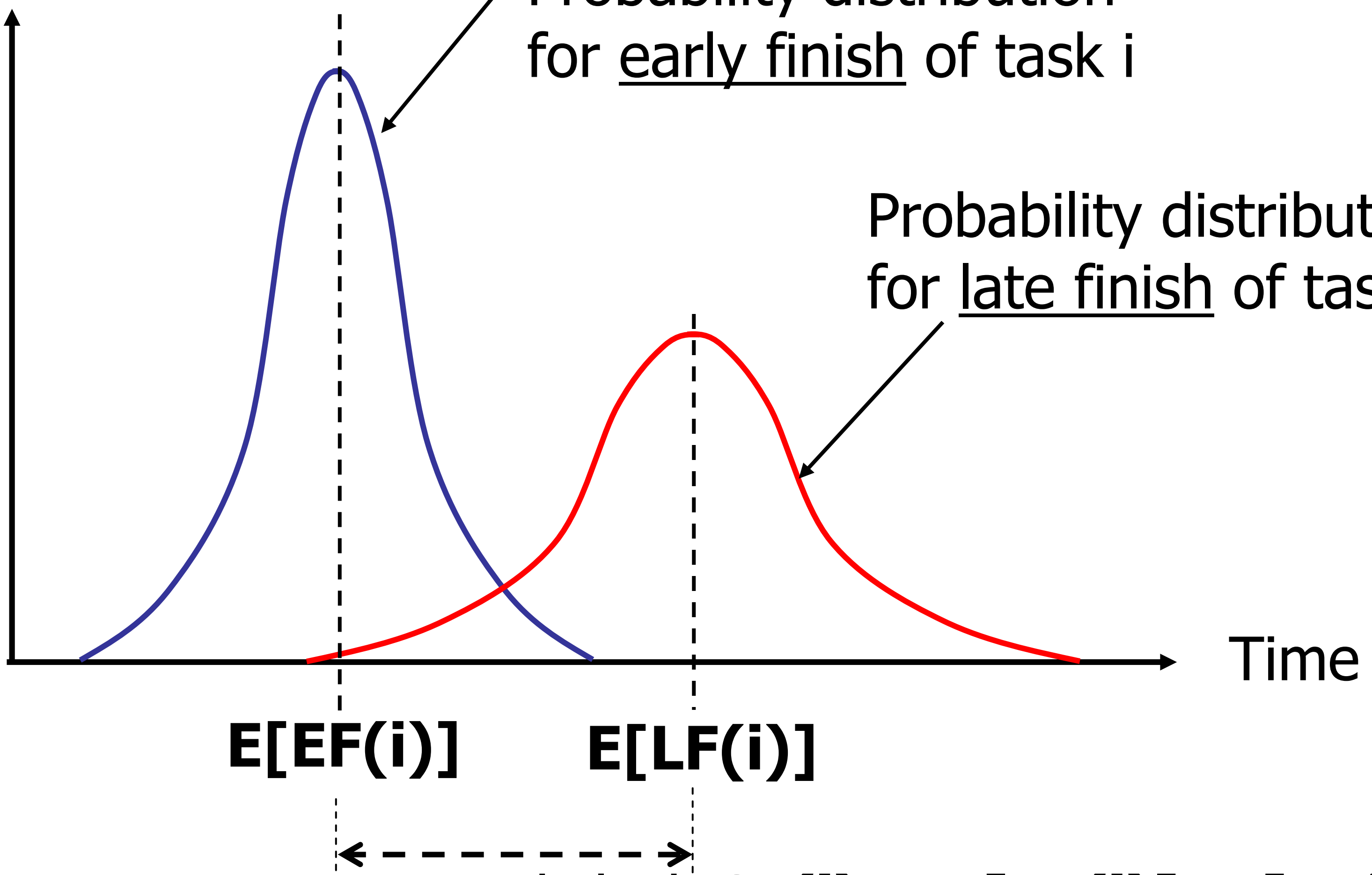
beta function:

$$B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)} = \frac{(p-1)!(q-1)!}{(p+q-1)!}.$$

Expected Time & Variance

- Mean expected Time (**TE**) $$TE = \frac{A + 4M + B}{6}$$
- Time Variance (**TV**) $$TV = \sigma_t^2 = \left(\frac{B - A}{6}\right)^2$$
- Early Finish (**EF**) and Late Finish (**LF**) computed as for CPM with **TE**
- Set **T**=**F** for the end of the project
- Assume that times are **Gaussian distributed** due to Central Limit Theorem
- Example: A=3 weeks, B=7 weeks, M=5 weeks --> then **TE**=5 weeks

Probabilistic times



Probability distribution for <u>early finish</u> of task i

Probability distribution for <u>late finish</u> of task i

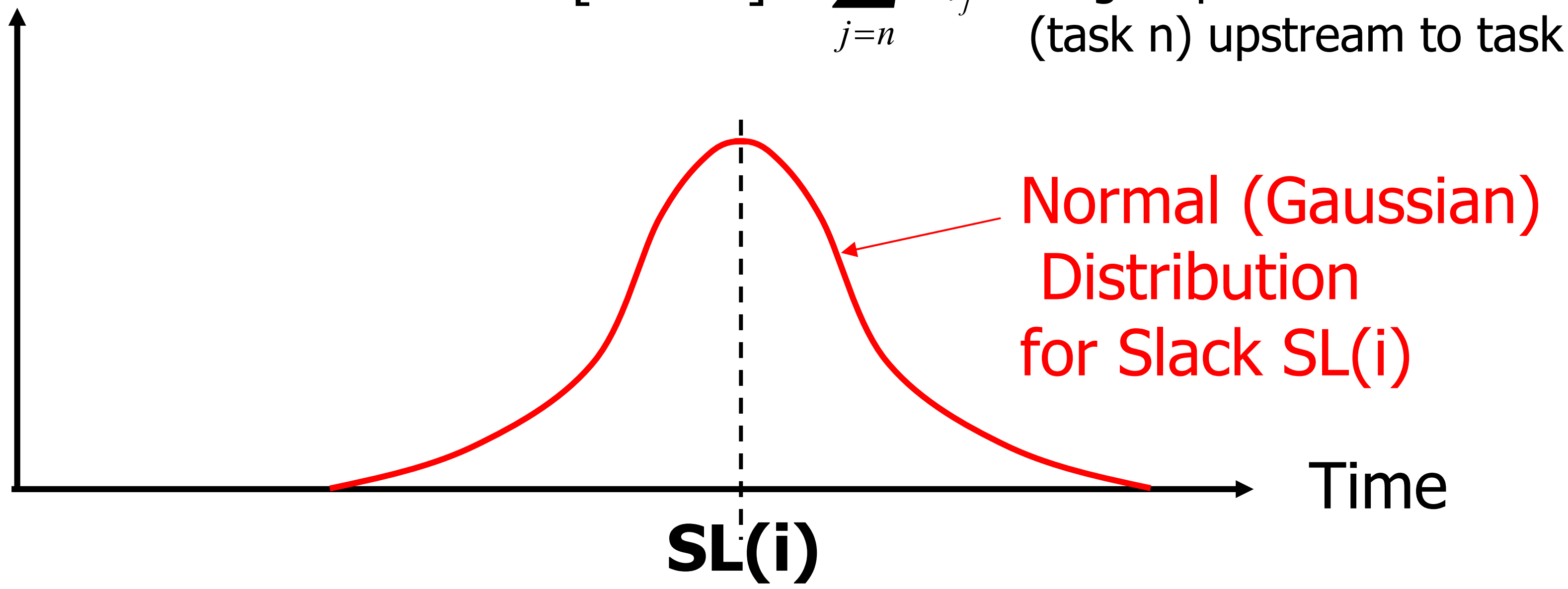
expected slack **SL(i) = E[EF(i)]- E[LF(i)]**

Probabilistic Slack

- Variance of task times:
  - Start times: $\sigma^2\left[ES(i)\right]=\sum_{j=0}^{i}\sigma_{t_j}^2$
  - Finish times: $\sigma^2\left[EF(i)\right]=\sum_{j=n}^{i}\sigma_{t_j}^2$

Sum all variances of the longest path from start (task 0) to task i

Sum all variances of the longest path from finish (task n) upstream to task i



Variance of Slack: $\sigma^2\left[SL(i)\right]=\sigma^2\left[EF(i)\right]+\sigma^2\left[LF(i)\right]$

Probability of no slack?

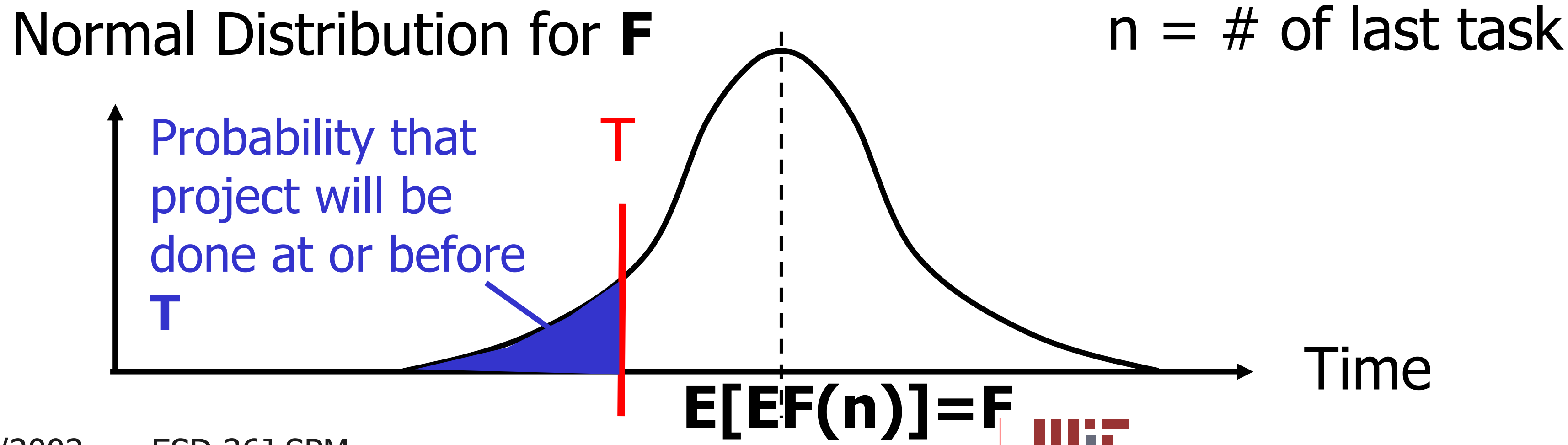
- Target date is not met when **SL(i)<0,** i.e. negative slack occurs
- Convert slack to a normalized random variable **z**:

$$z = \frac{EF - LF}{\sqrt{\sigma^2(EF) + \sigma^2(LF)}} = \frac{-SL}{\sigma\left(SL\right)}$$

- For each value of **z** one can look up the probability that SL=0 from a table of the normal (Gaussian) distribution function
- For tasks on the critical path, the probability that SL=0 is always **50%**

Probability of meeting target ?

- Many Projects have target completion dates, **T**
  - Interplanetary mission launch windows 3-4 days
  - Contractual delivery dates involving financial incentives or penalties
  - Timed product releases (e.g. Holiday season)
  - Finish construction projects before winter starts
- Analyze expected Finish **F** relative to **T**



Normal random variable z

- Compute $$z = \frac{T - E[F]}{\sigma(F)}$$
- Look up probability in a standard normal probability table
- http://www.math2.org/math/stat/distributions/z-dist.htm

Example: Company wants to have prototype at car show on **T**=May 3rd. Expected Early Finish **F** for project from PERT plan is: May 12 with a standard deviation s=20. z= (5/2-5/12)/20=-8/20=-0.4 * there are 8 working days between 5/2 and 5/12. For z=-0.4 we obtain a Probability of **35%** that we can meet the target date.

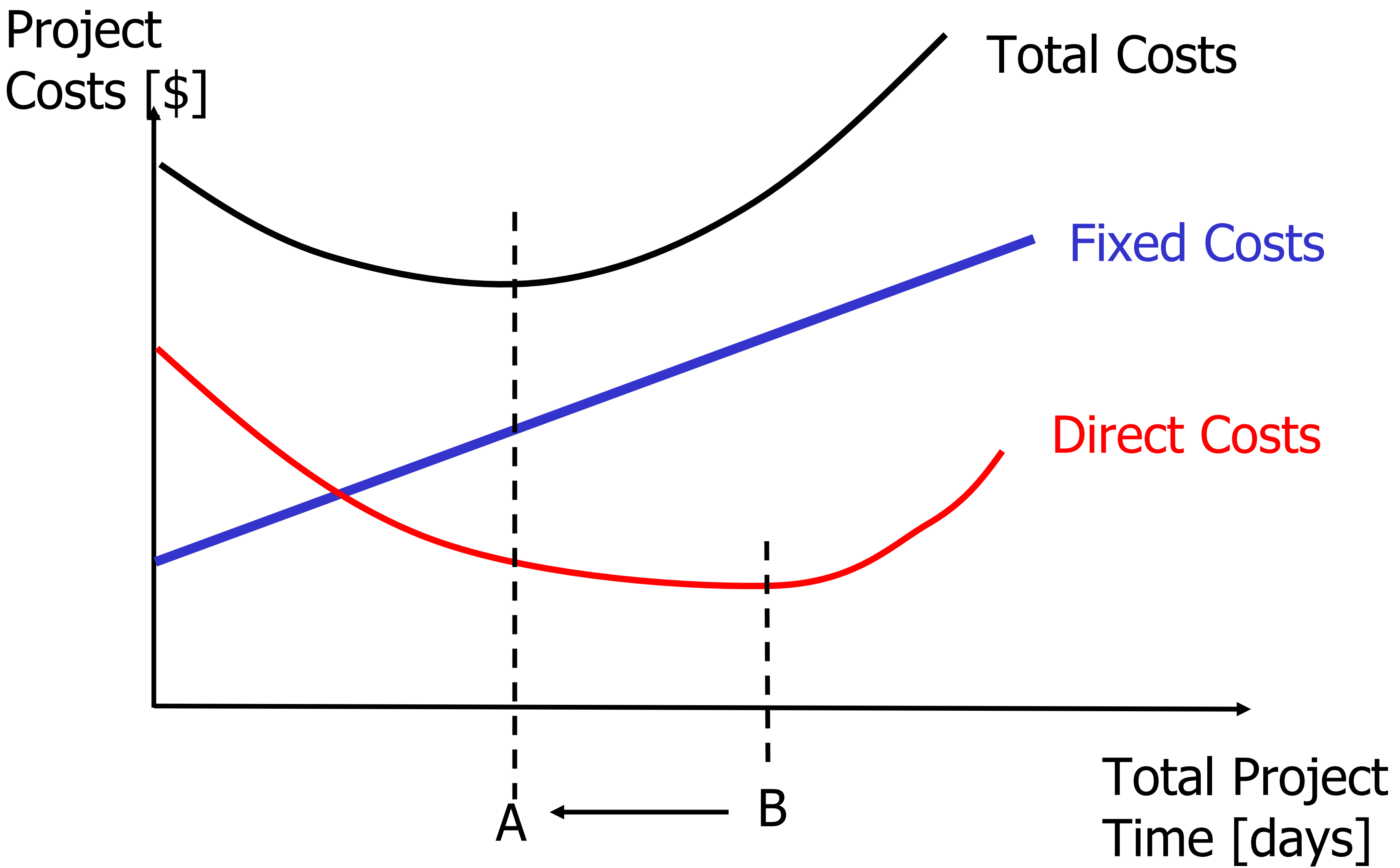
Crashing Tasks

- If we theoretically were building 100 prototypes, ~35 of them would make it on time
- Important decision basis for management
  - How could we speed up the project?
- Cost of speedup?
- Is there a net savings resulting from reduction in overall project time?

Cost Calculations

- Can compute project costs if cost of each job is included in the task data
- (Potentially) shorten crew jobs by adding personnel
- Speedup carries price tag: “normal time”, “crash time”
- Assign some critical jobs to their “crash time”
- Direct costs will increase as we “crash” critical tasks
- Indirect (fixed, overhead) costs will decrease as the overall project duration decreases – “standing army phenomenon”
- Minimize the sum of fixed and direct costs

Typical Cost Pattern



CPM Judgment

- +
  - Focuses attention on a subset of critical tasks
  - Determine effect of shortening/lengthening tasks
  - Evaluate costs of a ”crash” program
- -
  - Doesn’t capture task iterations, in fact …
  - Prohibits iterations = “cycle error”
  - Treats task durations as deterministic

Summary

- CPM is useful, despite criticism, to identify the critical path - focus on a subset of the project
- Slack (TS and FS) is precious
  - apply flexibility to smooth resource/schedules
- PERT treats task times as probabilistic
  - Individual task durations are β-distributed
  - Sums of multiple tasks are normal z-distributed
- Selective "crashing" of critical tasks can reduce total project cost
- CPM and PERT do not allow task iterations

Class Frustrations

- Poor examples set by project managers
- Perception of PM as bureaucratic "box-checking"

Why?

- Traditional Project Management …
  - Doesn't acknowledge the existence of iterations
  - Is inflexible, "changing the plan" considered a failure
  - Does not think of projects in a probabilistic sense
  - "Hostage" to existing project management software
  - In a reactive mode – no "early warning" systems
  - Based on pure reductionism

HW1 Introduction – out 9/9

- You are Project Manager for a UAV Development Project
- Plan the project
  - Task list, project graph
  - Critical path
  - Slack times
  - Replanning after change
  - Challenge Question → probability of completion at target time **T?**
  - "managerial"-type questions

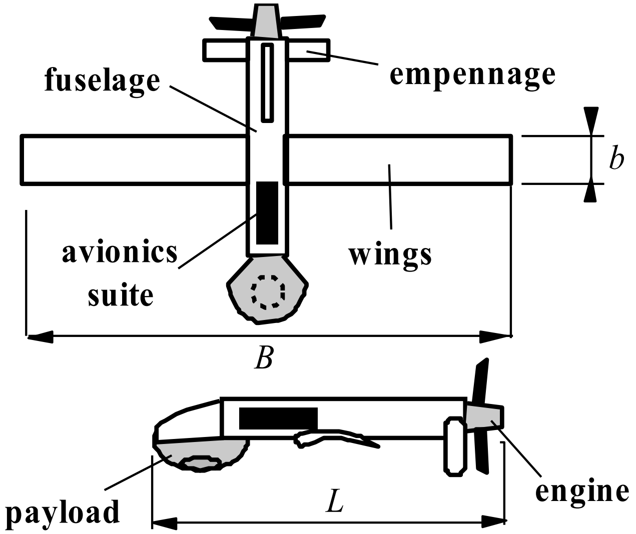


Fig 1. UAV concept, Specifications: L=2000 mm, B=3500 mm, b=500 mm

Due 9/18