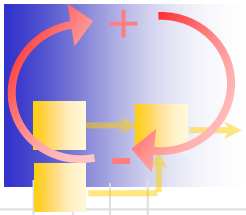
Project Scheduling Management

Lecture 2



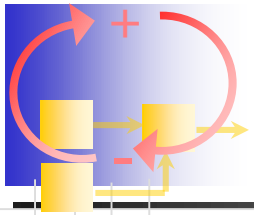
Critical Path Method (CPM) & Critical Chain

Instructor

Dr. Huang Dan

Sep 22, 2018

上海交通大學
Shanghai Jiao Tong University



Today's Agenda

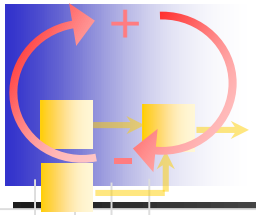
- Overview of PM methods and tools
- CPM 101
- Critical Paths, Slack
- Critical Chain Method
- Conclusions
- Introduce HW1

History of Project Management

- Big Projects since antiquity
  - Pyramids (Egypt), Great Wall (China)
  - Enormous workforce, but little documented evidence of formal project management
- Formal Project Management
  - Henry Gantt (1861-1919) → bar chart 1910
  - 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

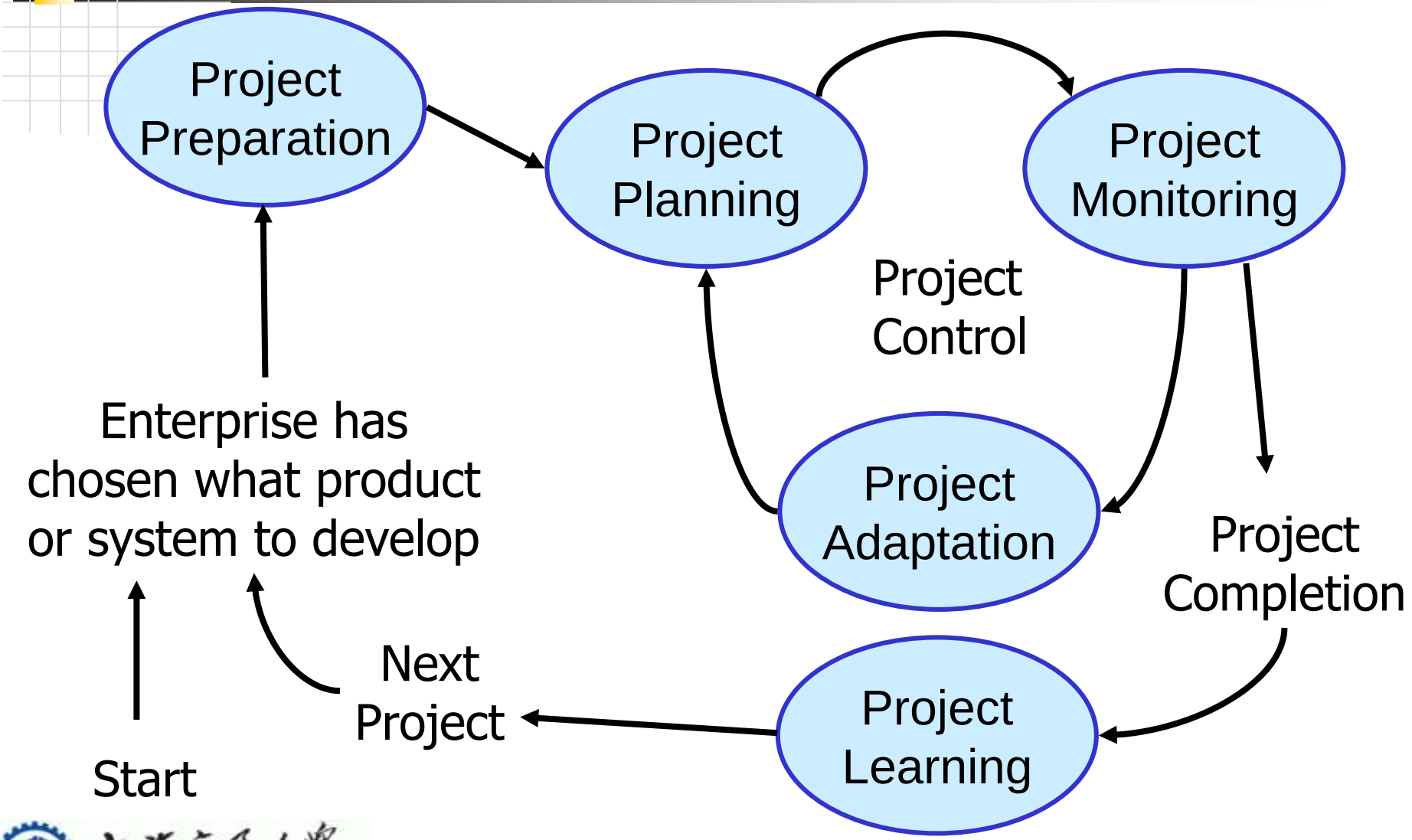
- Project decomposition necessary due to complexity
- Resource allocation and workload smoothing
- Schedule urgency .."before the decade is out" *JFK*
- Circumstances
  - Complex Relations between Government and Contractors
  - "Shielded" from Society, Competition, Regulations
  - Cold War Pressures 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"



Functional View of Project Management

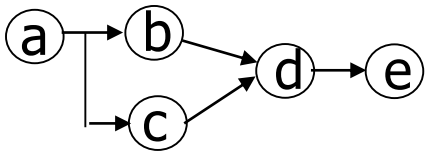
- Preparing
- Planning
- Monitoring
- Adaption
- Learning

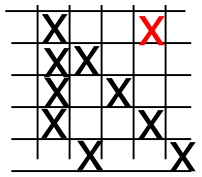
System Project Management Framework

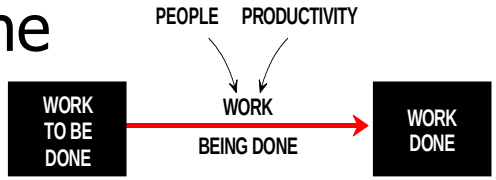


Fundamental Approaches

- 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
- 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"





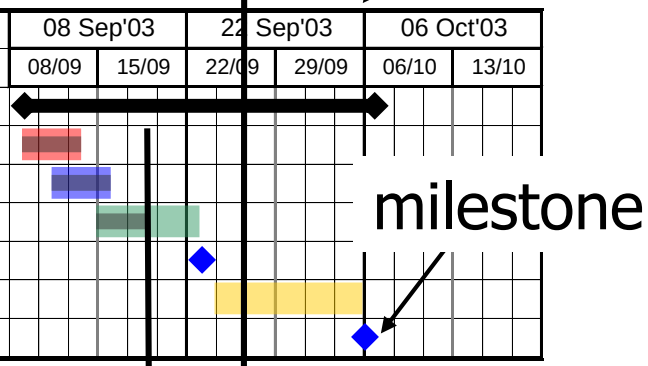


Gantt Charts

- 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

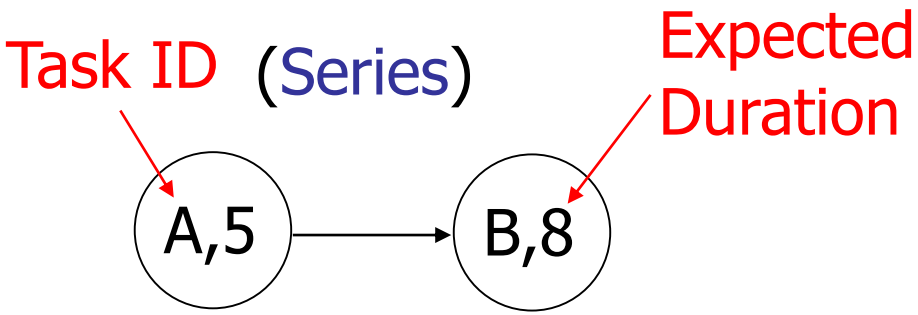
Bus Unit	Project	%	Start	Finish
	Project "XYZ"	**30**	**9-Sep-03**	**6-Oct-03**
Mkt	Customer Clinic	100	09-Sep-2003	12-Sep-2003
Sys	Requirements Definition	100	11-Sep-2003	15-Sep-2003
Eng	Parts Design	50	15-Sep-2003	22-Sep-2003
Sys	Design Review	0	23-Sep-2003	23-Sep-2003
Mfg	Manufacturing	0	24-Sep-2003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



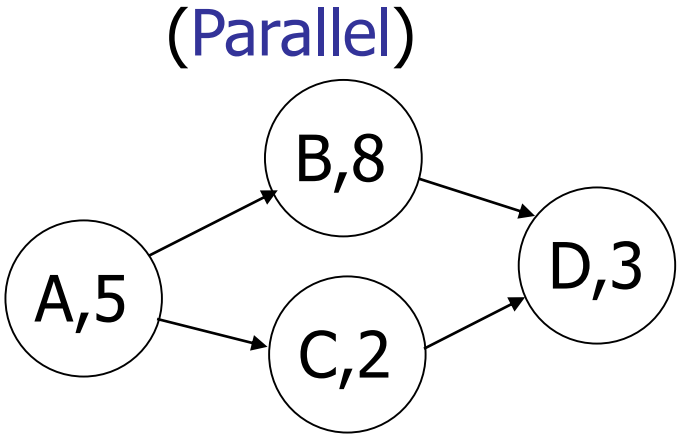
completion
today
calendar
milestone
tasks
actual
Δ
planned

CPM 101

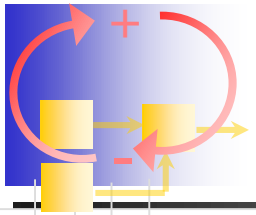
- Represent a project (set of tasks) as a network using graph theory
  - Capture task durations
  - Capture task logic (dependencies)



"B can only start after A is completed"



"B and C do not depend on each other"

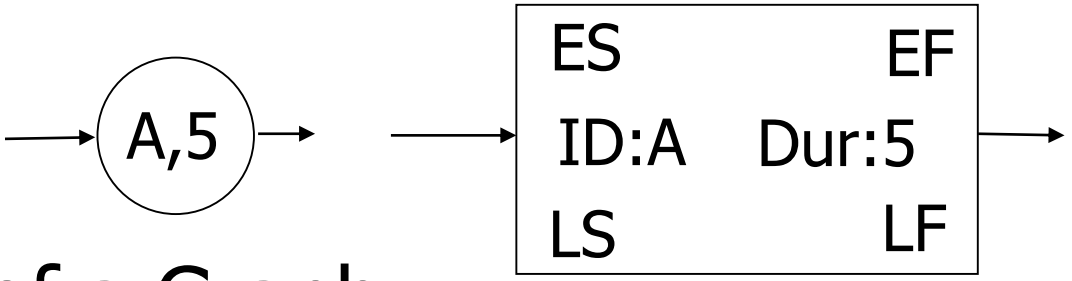


CPM Assumptions

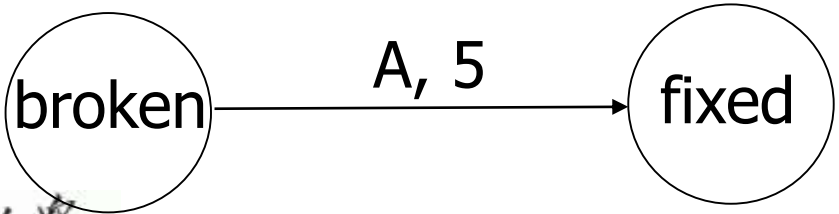
- Project consists of a collection of well defined tasks (jobs)
- Project ends when all jobs completed
- Jobs may be started and stopped independently of each other within a given sequence (no "continuous-flow" processes)
- Jobs are ordered → "technological sequence"

Task Representations

- Tasks as Nodes of a Graph
  - Circles
  - Boxes

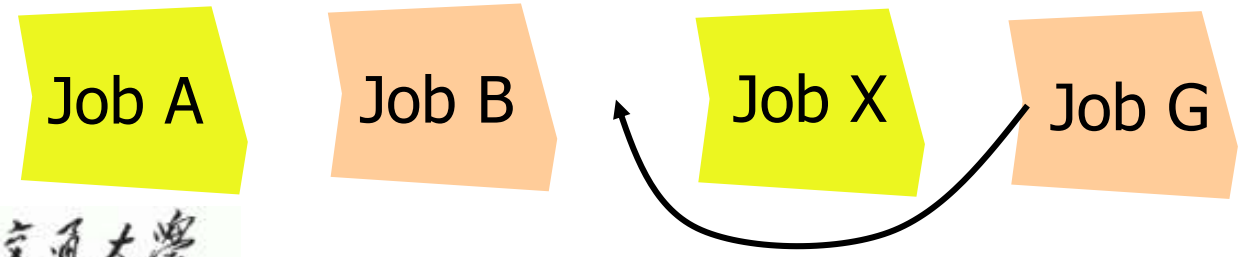


- Tasks as Arcs of a Graph
  - Tasks are uni-directional arrows
  - Nodes now represent "states" of a project
  - Kelley-Walker form



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

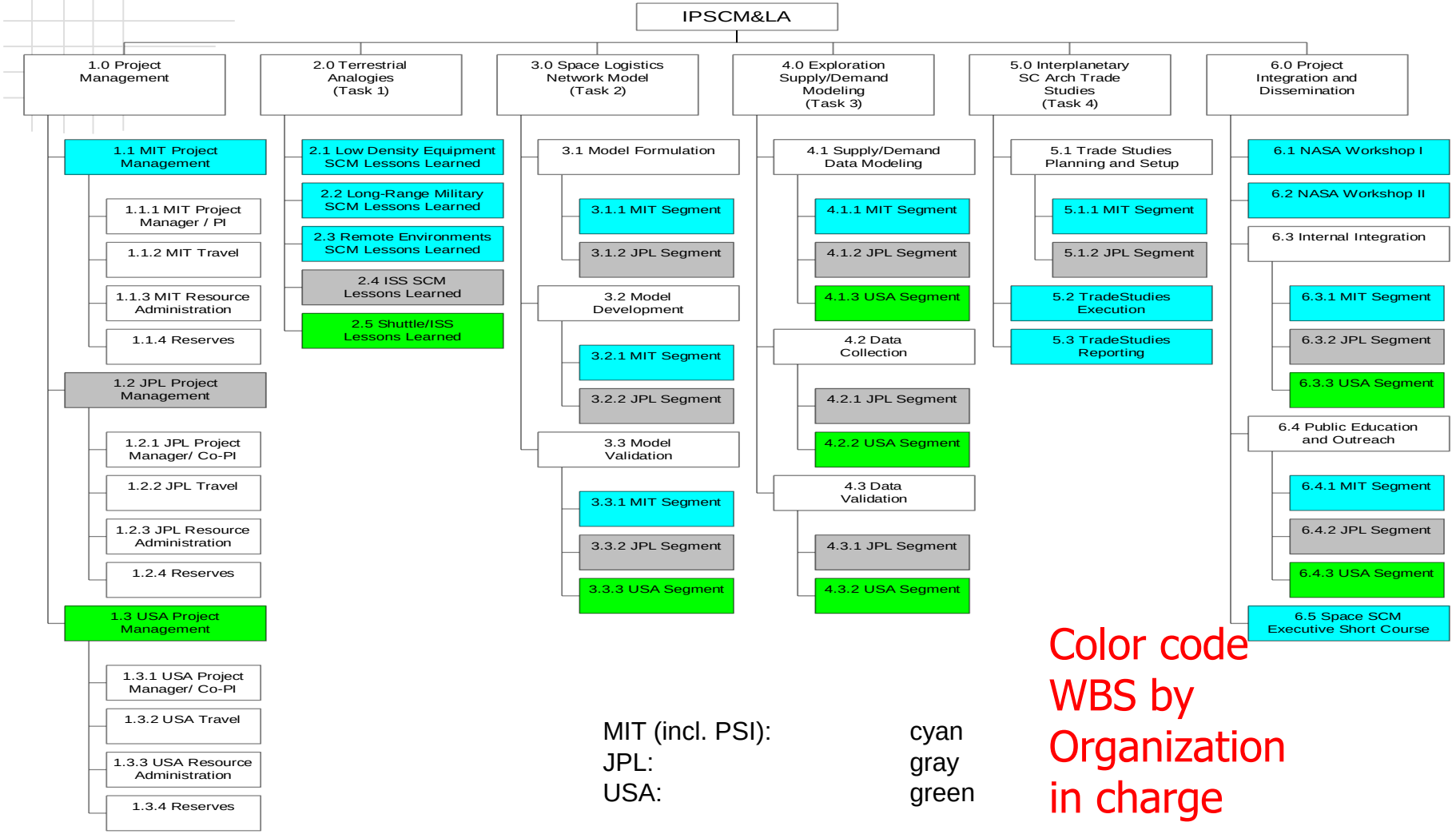


WBS – Painting a Room

- 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

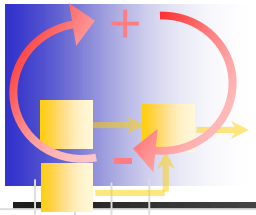
Source: http://www.wikipedia.org

WBS of MIT/NASA Space Logistics Project



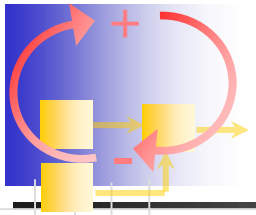
MIT (incl. PSI): cyan
JPL: gray
USA: green

Color code WBS by Organization in charge



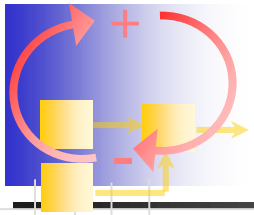
Discussion Point

- Why is it difficult to come up with a good WBS (task list, task structure) in a complex project?



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



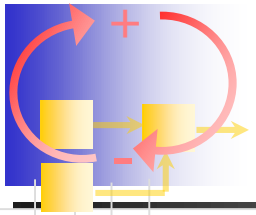
Task List

- List all tasks in a table with
  - Identifying symbol (tag, ID number)
  - 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 iterations 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

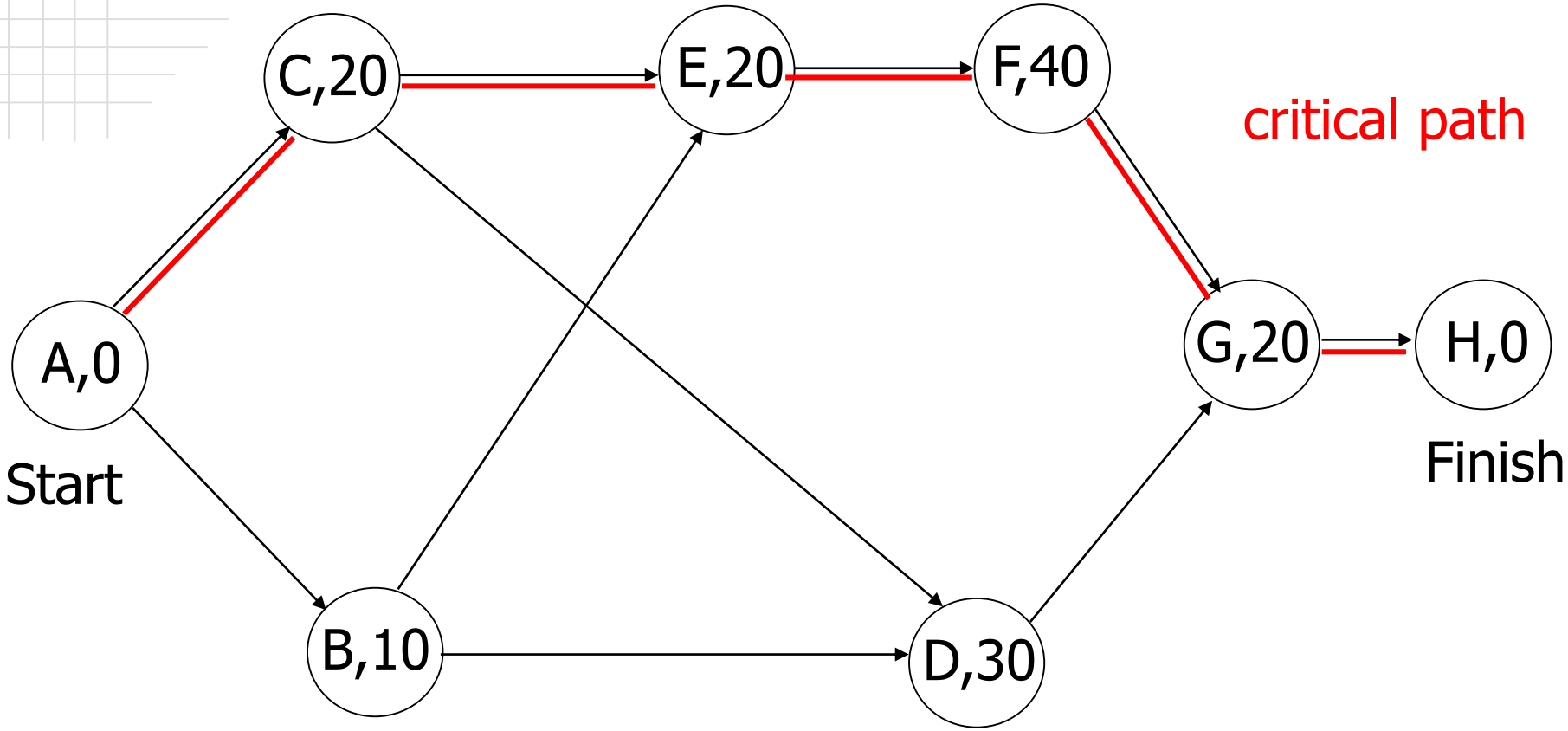


Project Graph

- 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

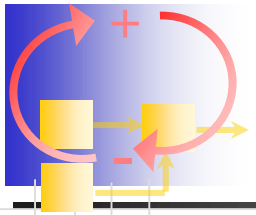
* or other symbol, see before

Project Graph



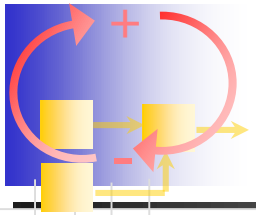
4 unique paths: A,C,E,F,G,H; A,C,D,G,H; A,B,D,G,H; A,B,E,F,G,H

100 70 60 90



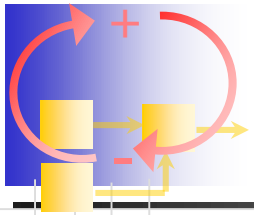
Critical Path

- 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).



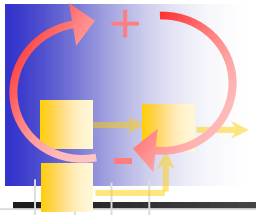
Discussion Point

- What is the usefulness of knowing the CP in a project?
  - Tells which task to shorten to finish project earlier.
  - Others …?



Critical Path Algorithm

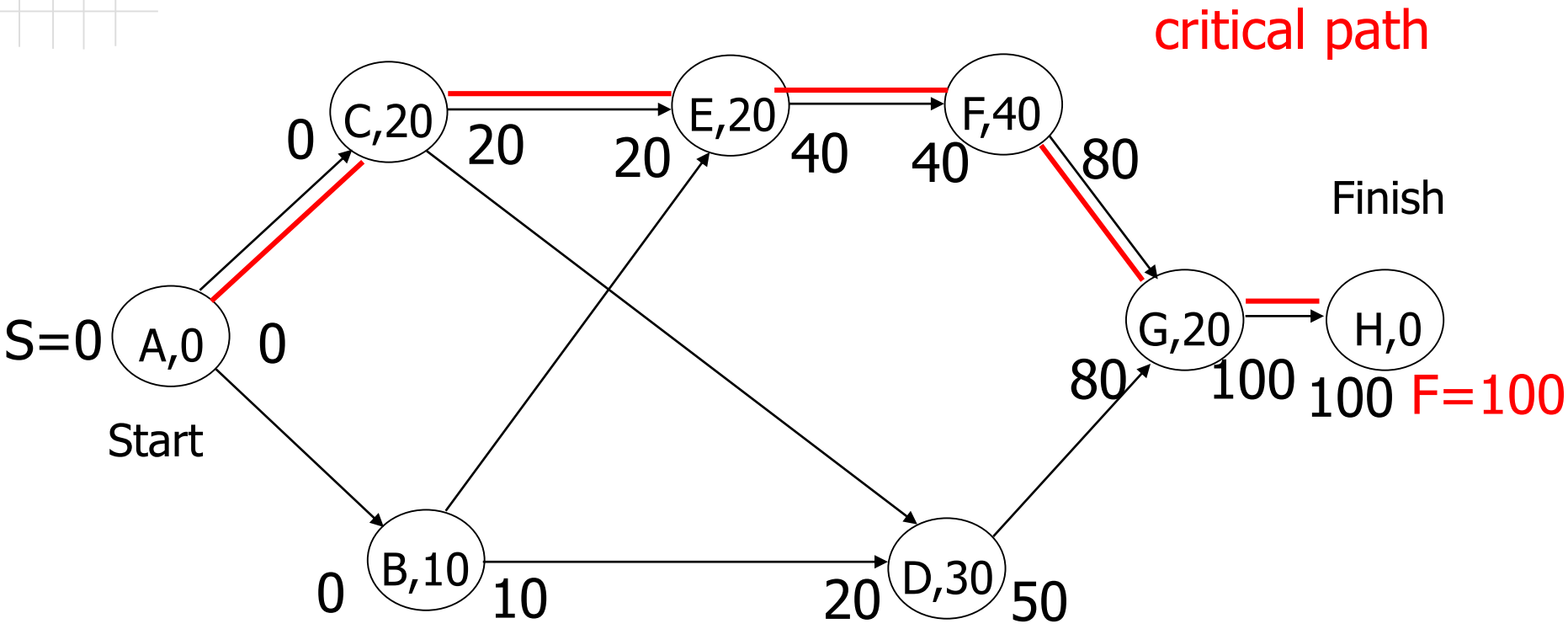
- For large projects there are many paths
- Need a algorithm to identify the CP efficiently
- Develop information about each task in context of the overall project
- Times
  - Start time (S)
  - For each job: Earliest Start (ES)
    - Earliest start time of a job if all its predecessors start at ES
  - 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



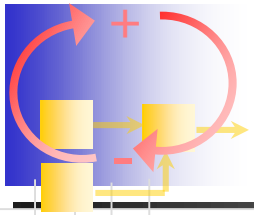
Concept Question 1

A project starts with (A,5). Task (B,10) can start after A is completed. This is also true for task (E,5). Task (C,8) depends only on (B,10), while task (F,10) depends on both (B,10) and (E,5). Task (D,5) is the last task in the project and it can start once (C,8) and (F,10) have been finished.

The Earliest Finish (EF) time for the whole project is:

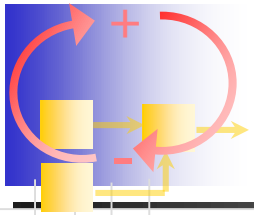
http://www.diaochapai.com/survey1668702

- 25
- 27
- 30
- 35
- 40



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**

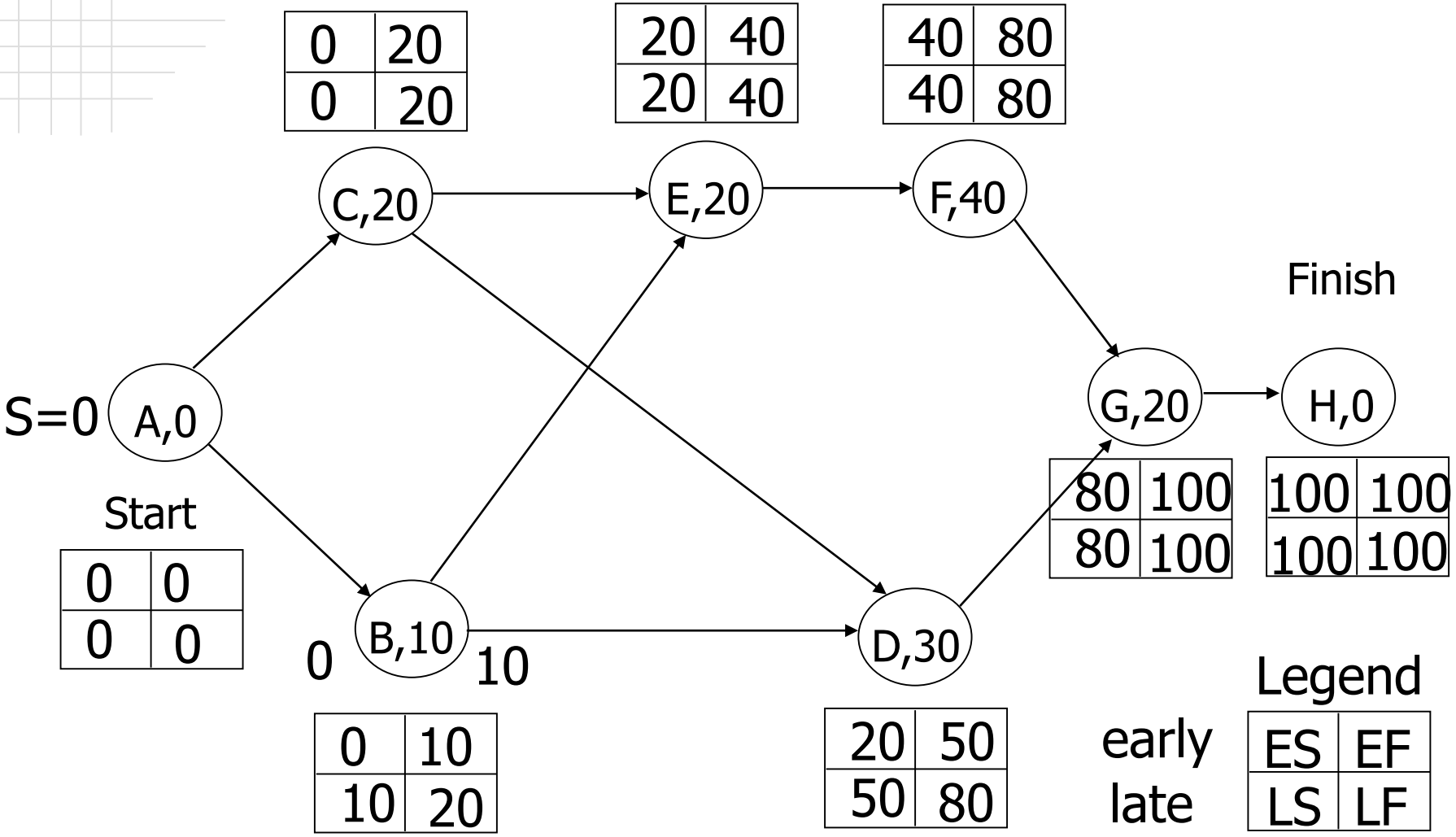


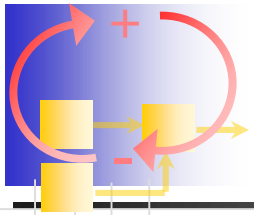
Determine LF and LS

- 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

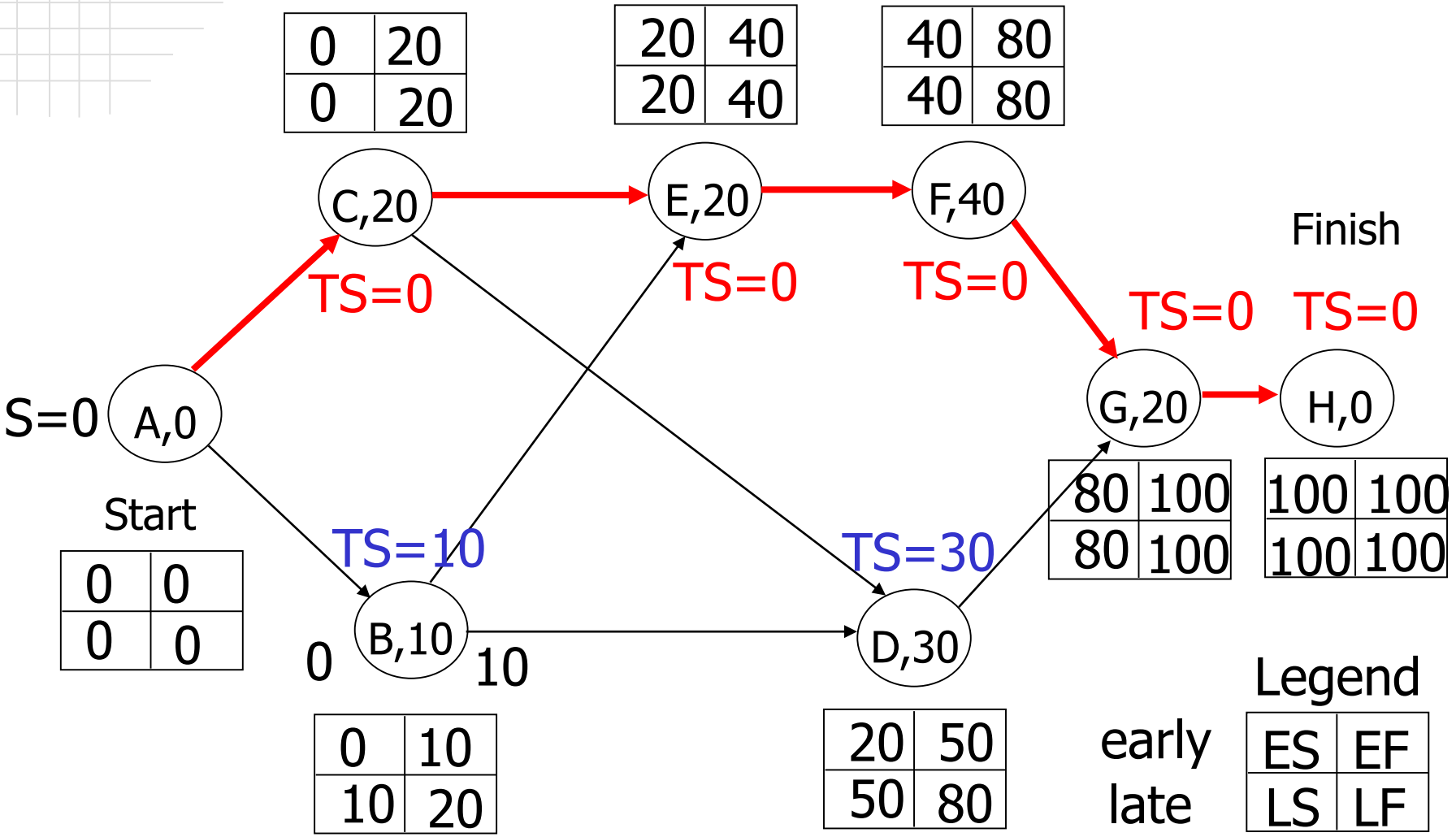




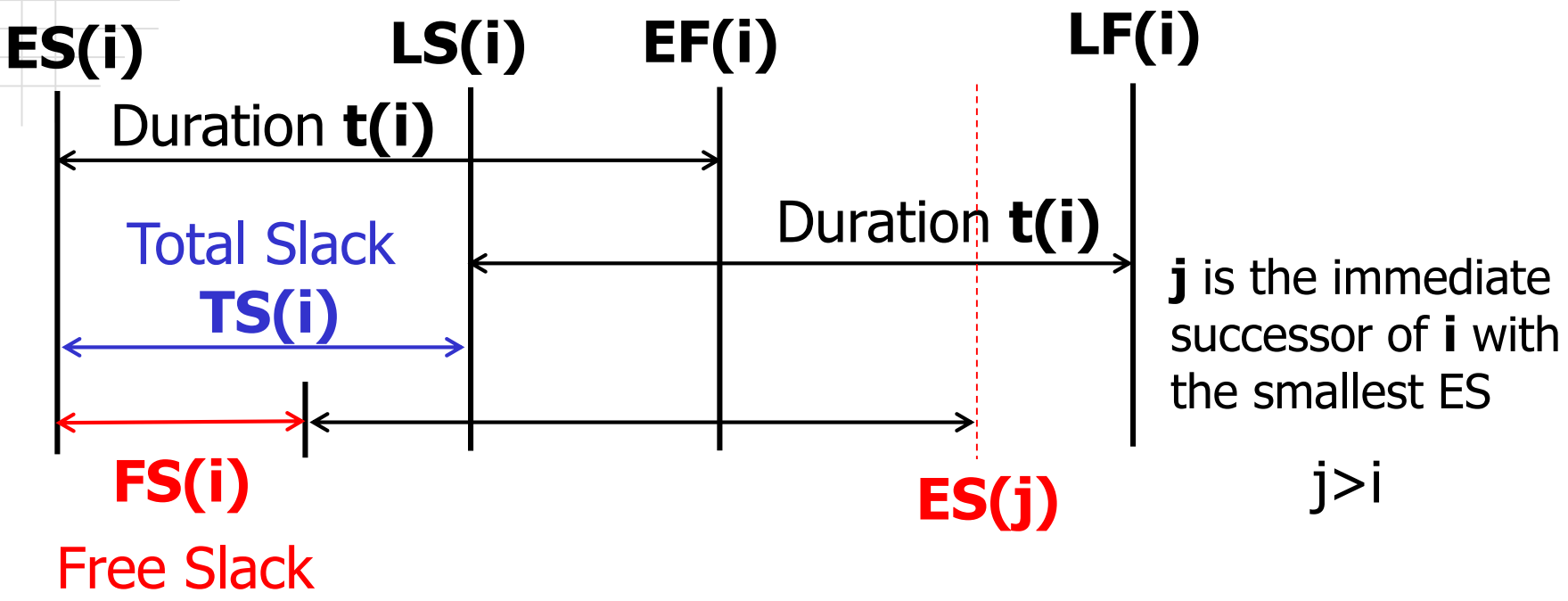
Slack

- 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

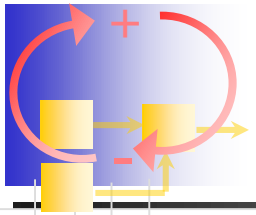


Task Times Detail - Task i



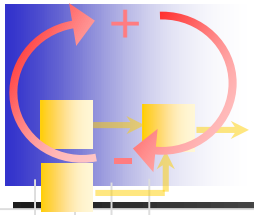
- Free Slack (**FS**) is the amount a job can be delayed without delaying the Early Start (ES) of any other job.

FS<=TS always



Main CPM Errors

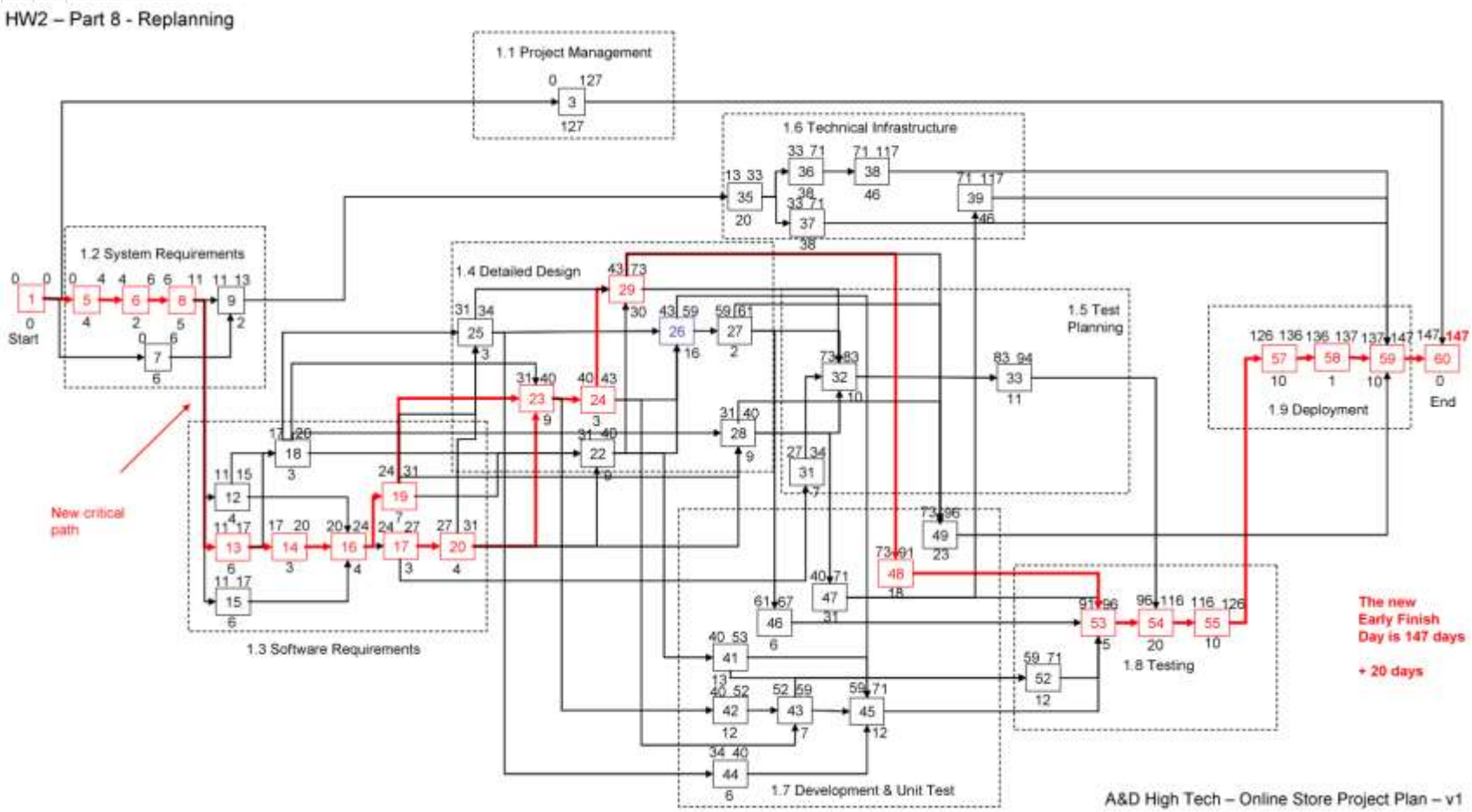
- Estimated job times are wrong
- Predecessor relationships may contain cycles → "cycle error"
- List of prerequisites contains more than the immediate predecessors, e.g. **a→b**, **b→c** and **a,b→c**
- Overlooked some predecessor relationships
- Some predecessor relationships may be listed that are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



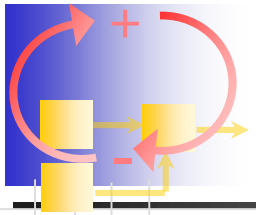
Gradual Refinement of CPM

- 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!

Sample CPM

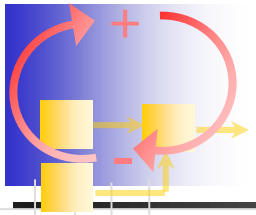


A&D High Tech Case, Online Store Project, 60 tasks



CPM Judgment

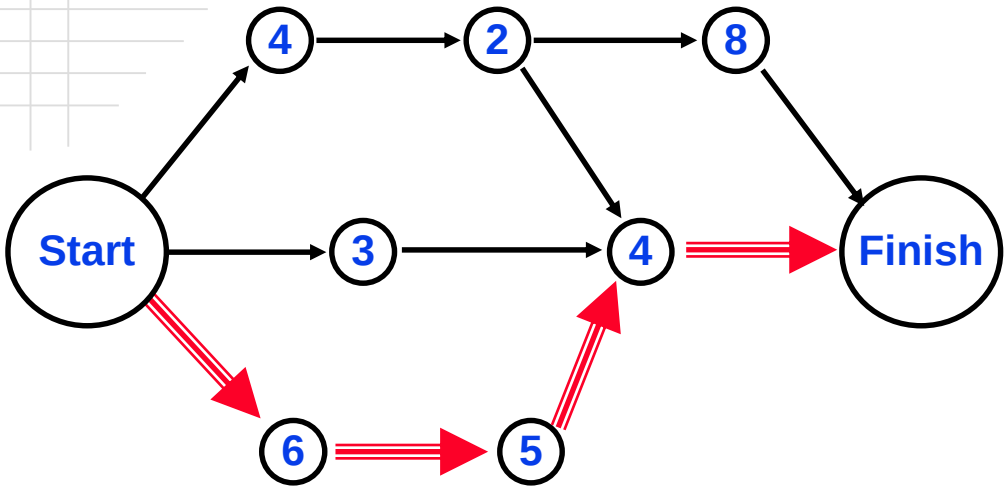
- +
  - Focuses attention on a subset of critical tasks
  - Determine effect of shortening/lengthening tasks
  - Evaluate costs of a ”crash” program
- -
  - Doesn’t capture task iterations, in fact …
  - Prohibits iterations = “cycle error”
  - Treats task durations as deterministic

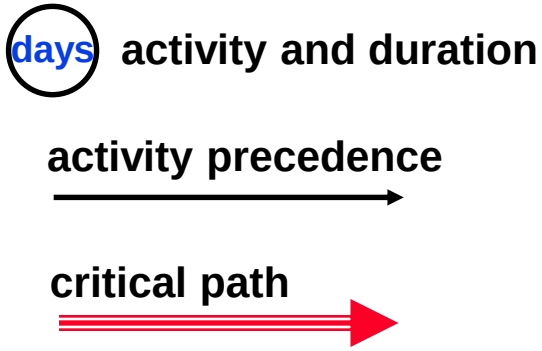


Summary of CPM

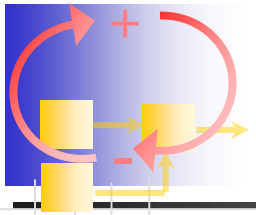
- CPM is useful, despite criticism, to identify the critical path - focus on a subset of the project
- Slack (TS and FS) is precious
  - apply flexibility to smooth resource/schedules
- PERT treats task times as probabilistic
  - Next lecture
- Selective "crashing" of critical tasks can reduce (or increase) total project cost
- CPM does not allow for task iterations

Traditional Project Control





- Critical Path Scheduling
  - sequential/parallel task network
- Due dates and slack times for each task
- Focus attention on critical path
- Monitor task completion dates
- **Later tasks crash to compensate for early delays**

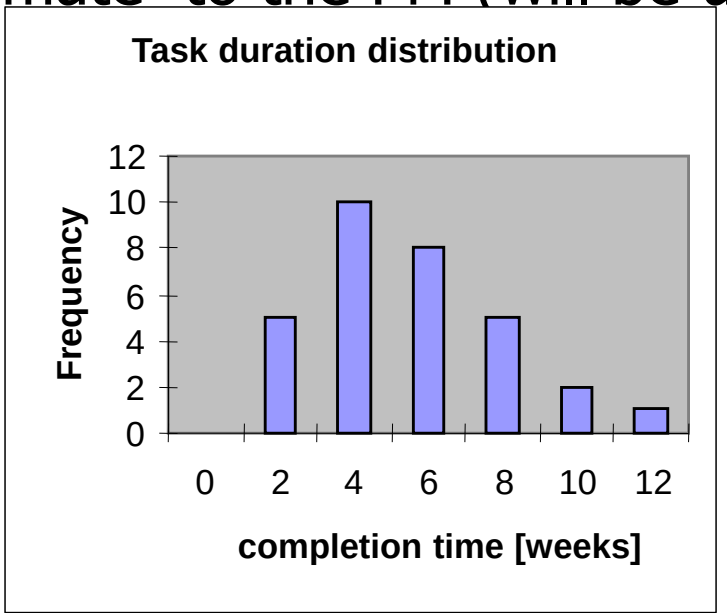


Typical Result of Project Crashing

- Schedule Slippage
- Budget Overrun
- Scope Reduction

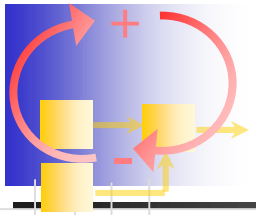
Concept Question 2

- You are a team leader starting a new project and your Project Manager asks for a task duration estimate for your task to build the overall plan. Below is the historical distribution of actual durations for your task. What do you provide back as a duration estimate to the PM (will be used in the CPM plan) and why?



- Answers:
  - **2**
  - **4**
  - **6**
  - **8**
  - **10**
  - **12**
  - **>12**

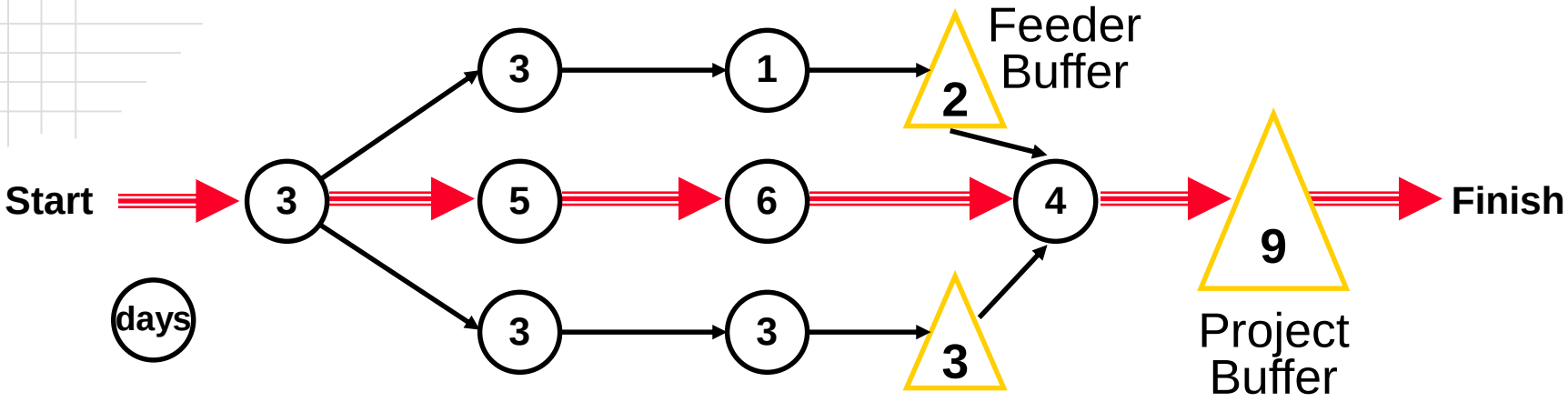
http://www.diaochapai.com/survey1668704



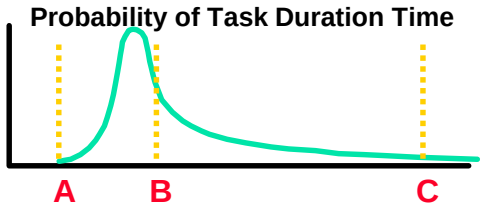
Critical Chain Method

- The goal of the critical chain (CC) method is **to avoid crashing by planning for the variance inherent in each task duration.**
- **CLAIM**: This helps projects finish on time, within budget, and without cutting scope.
- Main points:
  - Don't waste safety time.
  - Aggregate safety time into project and feeder buffers.
  - Monitor the buffers to know when to control.
  - Concentrate on the constraint of the project: the longest chain of dependent tasks or resources. (Theory of Constraints TOC)
  - Need a cultural change in how to manage projects and evaluate team members.
- Reference:
  - E.M. Goldratt, *Critical Chain*, North River Press, 1997.

Critical Chain Method

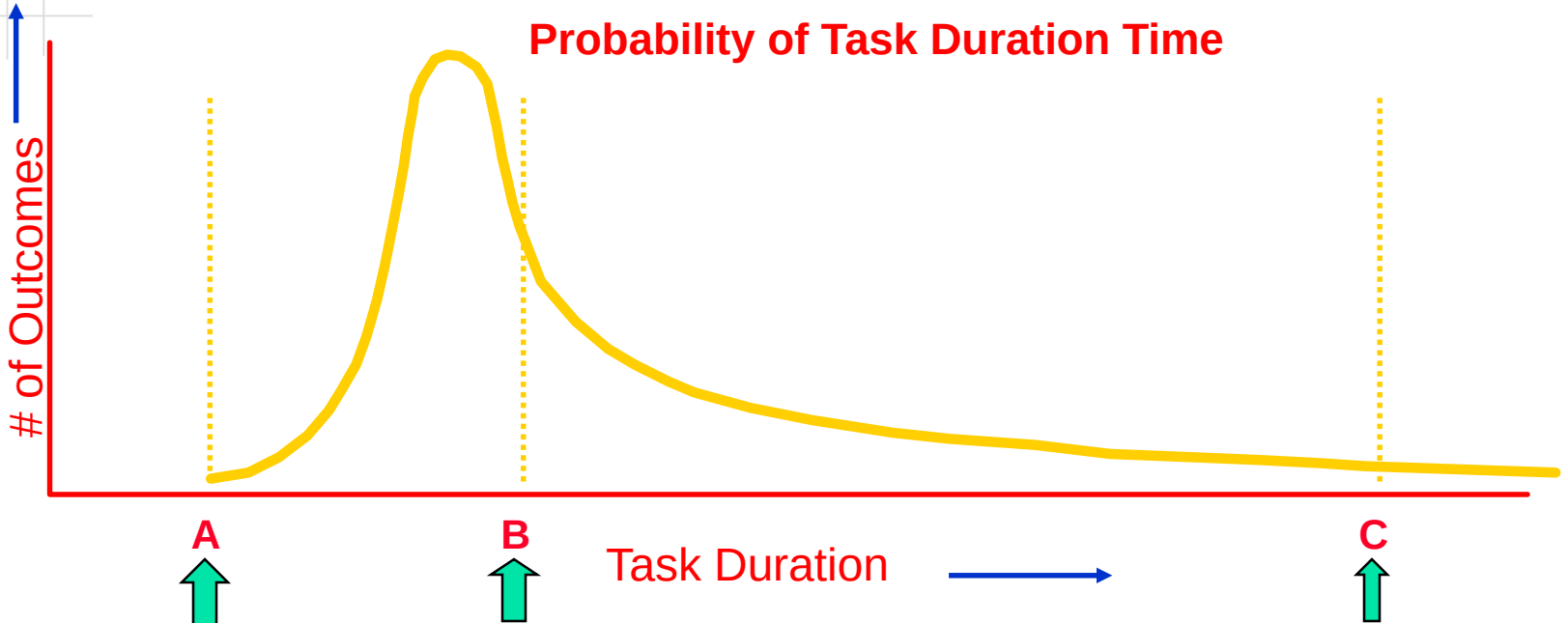


- Start with a sequential/parallel network.
- Use 50/50 (median) task duration estimates.
- Compute the critical path, noting resources.
- Insert feeder and project buffers as safety.
- Ideal buffers are 50% of path duration.
- Monitor buffer status.
- Reduce buffers when tasks overrun.



Ref: E.M. Goldratt, *Critical Chain*, North River Press, 1997.

Where's All the Safety Time? Which Time Are You Likely to Promise?

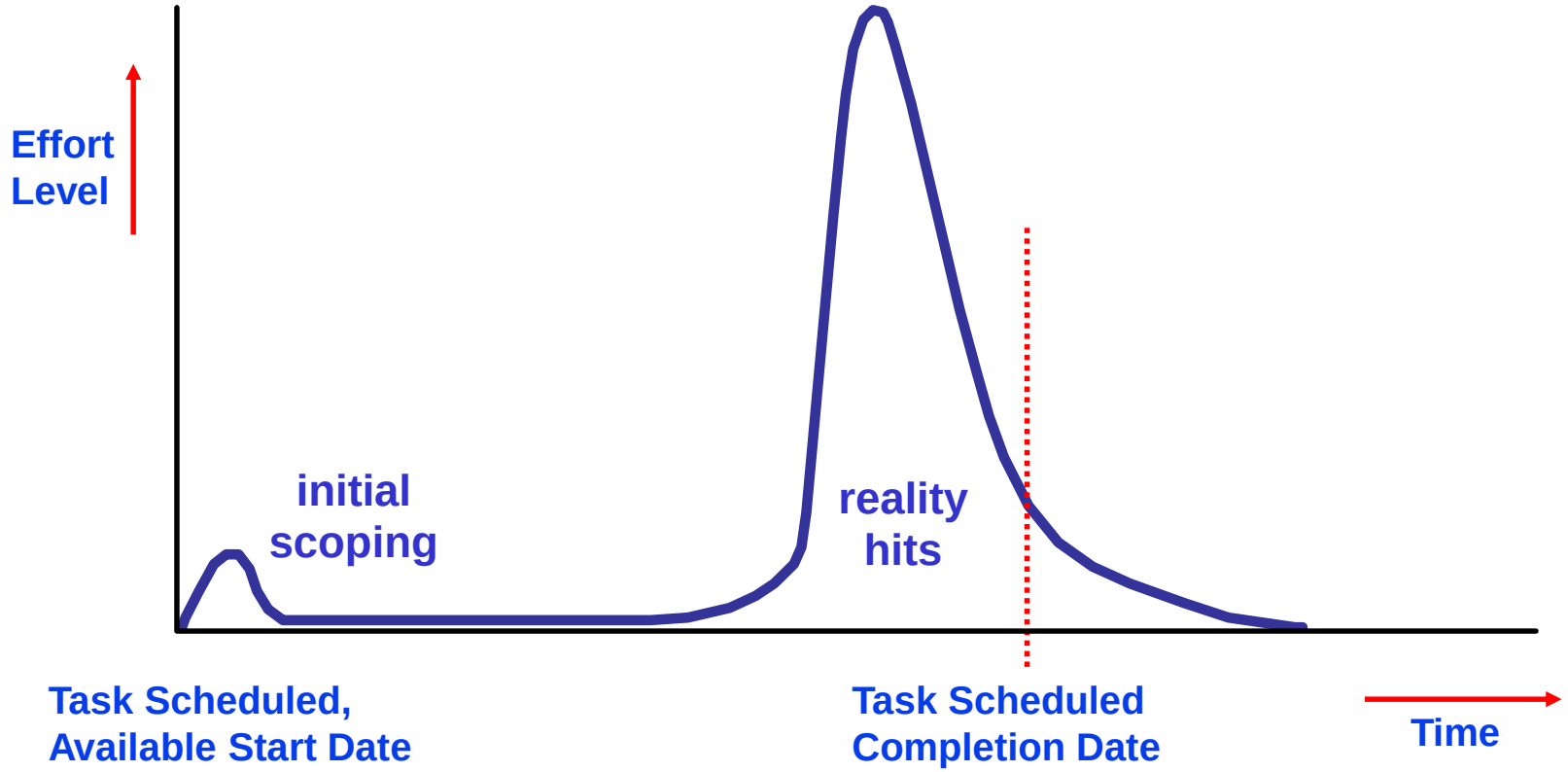


"Rare Success" —Don't count on it.

Possibly one can meet this time, even with some surprises. 50% of the outcomes are less than this, 50% are greater. This is the estimate CC builds into the schedule.

Even with a major disaster this time is highly achievable, and is commonly used.

How Do We "Waste" Safety Time?
1) Procrastination: the "Student Syndrome"



Source: Avraham Y. Goldratt Institute

How Do We "Waste" Safety Time?
2) The Effect of Multi-Tasking

Task A
Project A
One Week

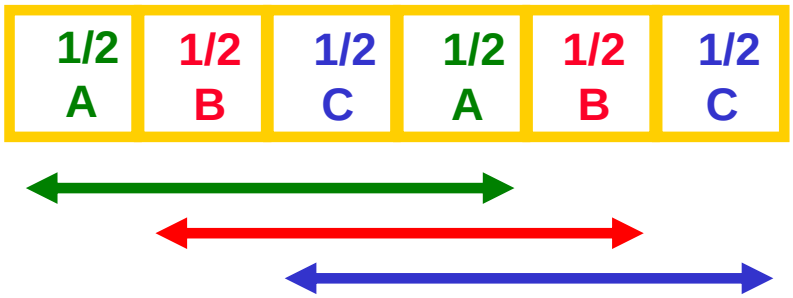
Task B
Project B
One Week

Task C
Project C
One Week

The resource does half of task A, then half of task B, then half of task C, then finishes task A, then B, then C.

How long does each task take to complete?

What happened to the safety time?

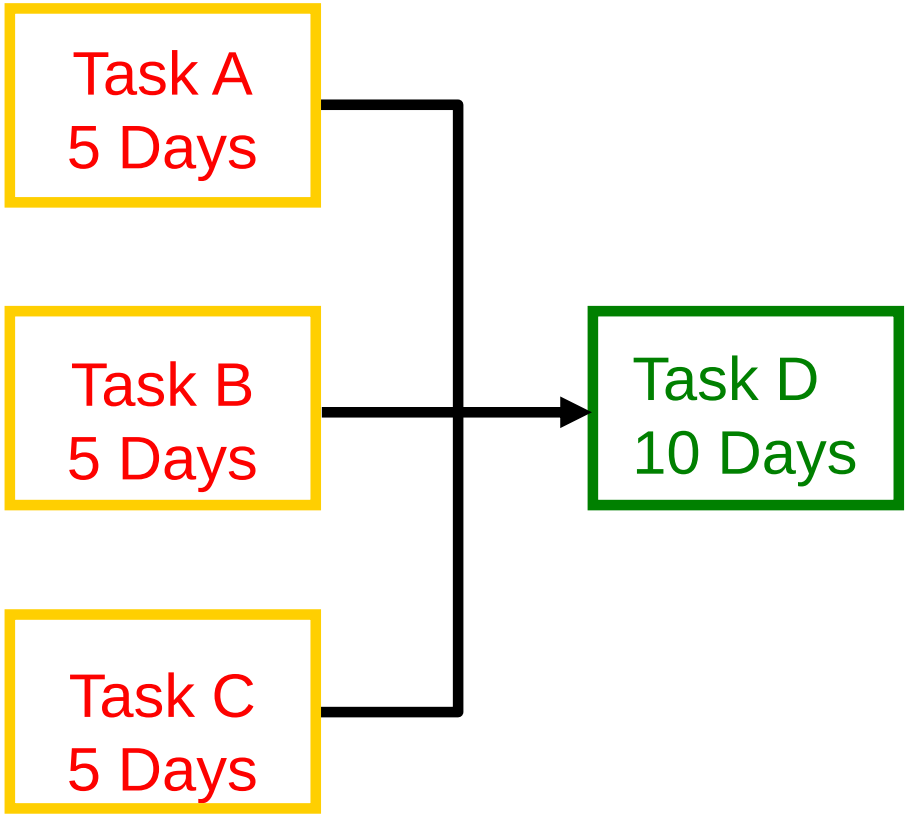


Source: Avraham Y. Goldratt Institute

How Do We "Waste" Safety Time?
3) Delays Are Passed On – Gains Are Not

Merging paths don't allow us to benefit from tasks completed early.

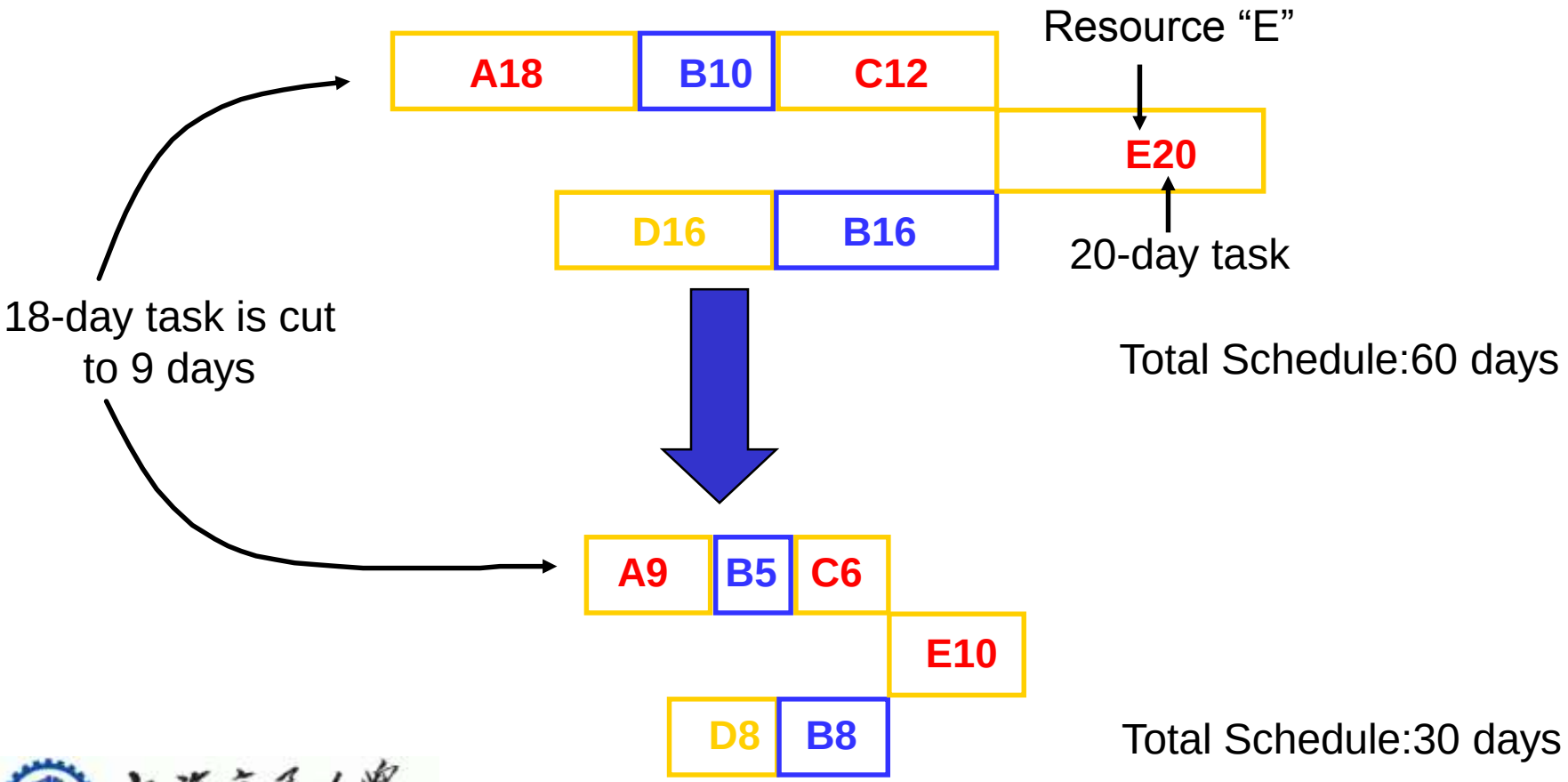
- What is the impact on the total project if Task A is done in only 3 days?
- What if Task C takes 8 days?
- What if Tasks A, B, and C all get done in 3 days?
  Will Task D be ready to start 2 days early?



Source: Avraham Y. Goldratt Institute

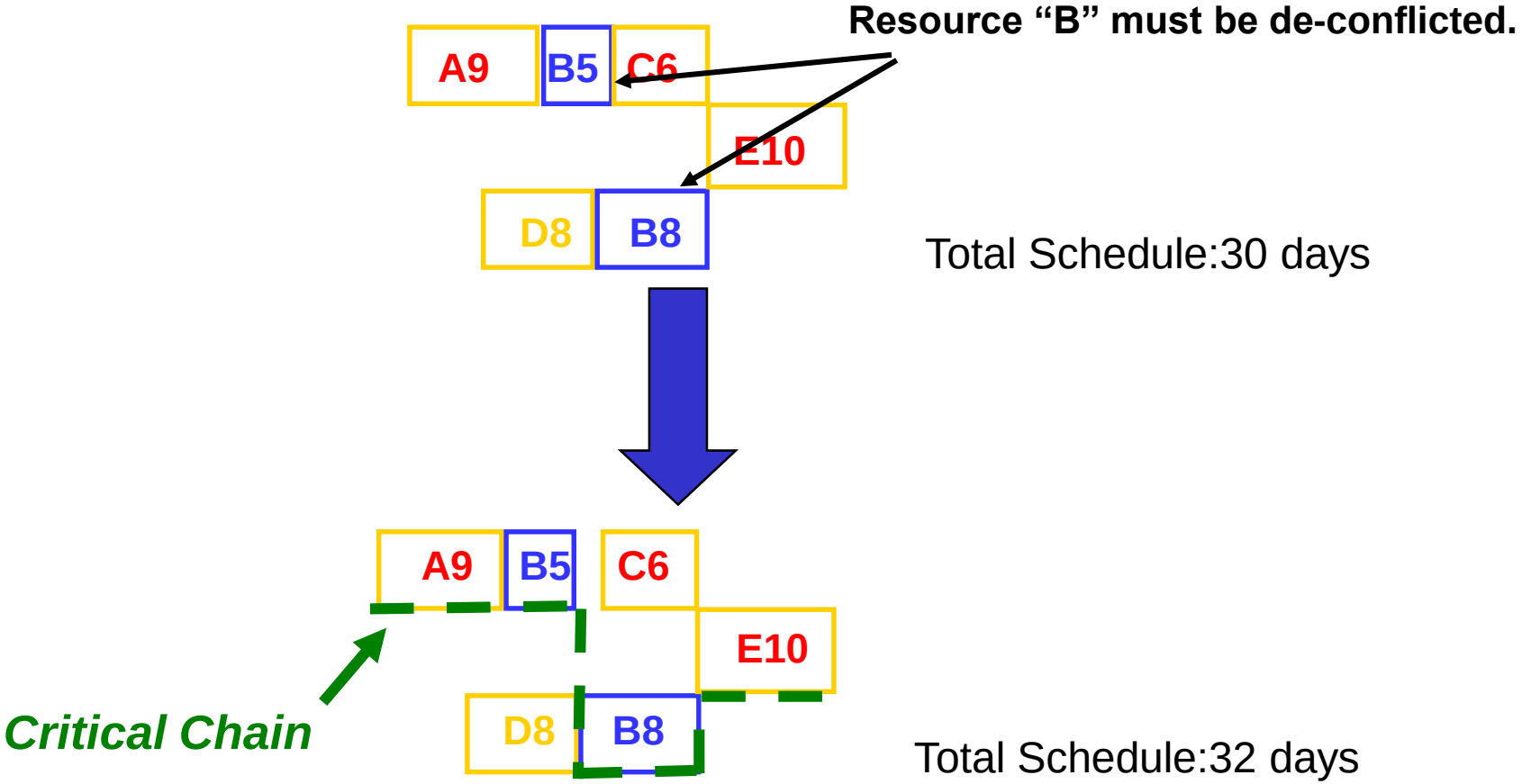
Critical Chain Scheduling (1)

- Create a sequential/parallel project network.
- Remove safety time to shorten task durations.



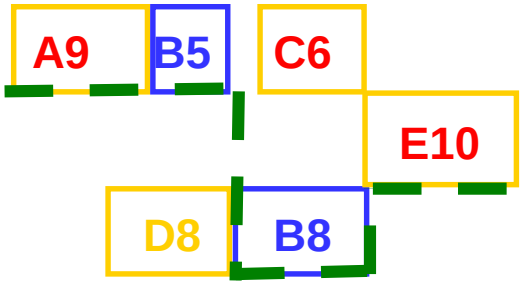
Critical Chain Scheduling (2)

- De-conflict resources.
- Identify the critical chain.



Critical Chain Scheduling (3)

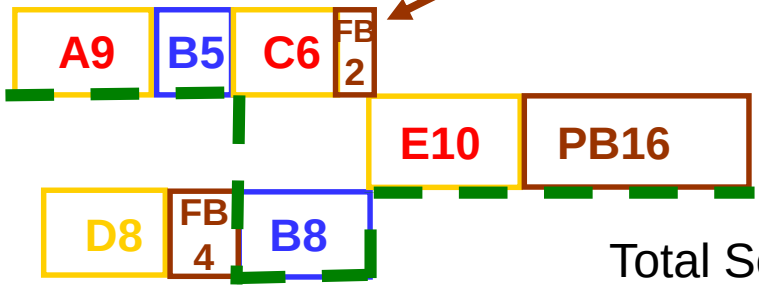
- Create 50% project buffer and feeder buffers.



Total Schedule:32 days

Because of aggregation, the effective variance is lower, and less buffer protection is necessary

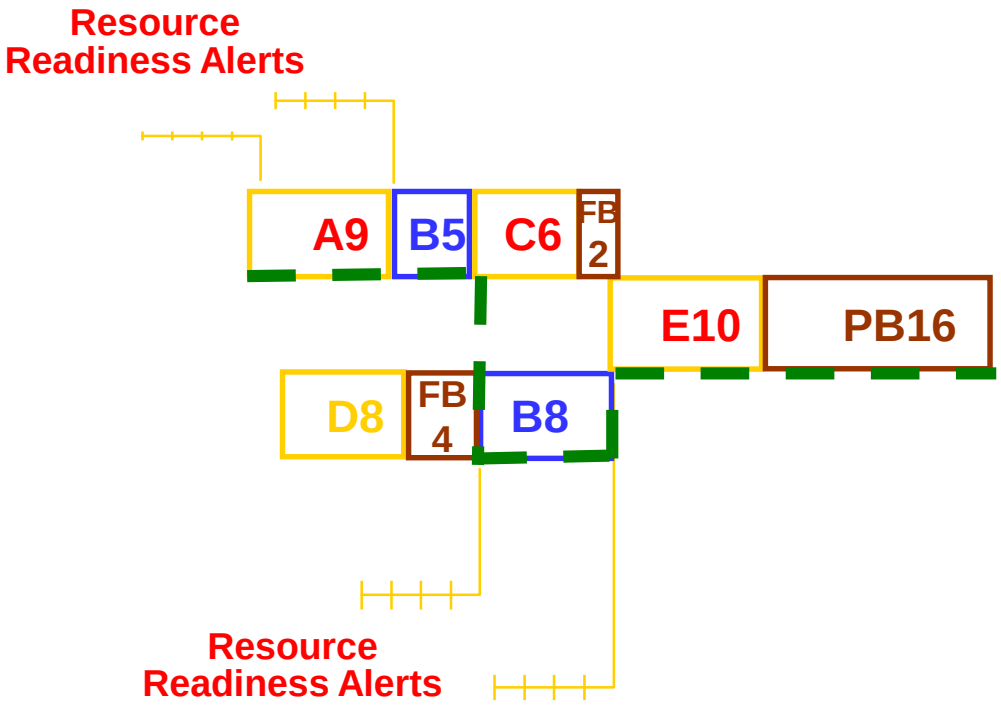
FB2 is 2/3 of ideal (50%) buffer size

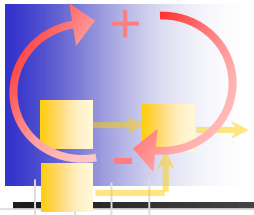


Total Schedule:48 days

Critical Chain Scheduling (4)

- Add resource readiness alerts along the critical chain.
- Implement using red and yellow chains.



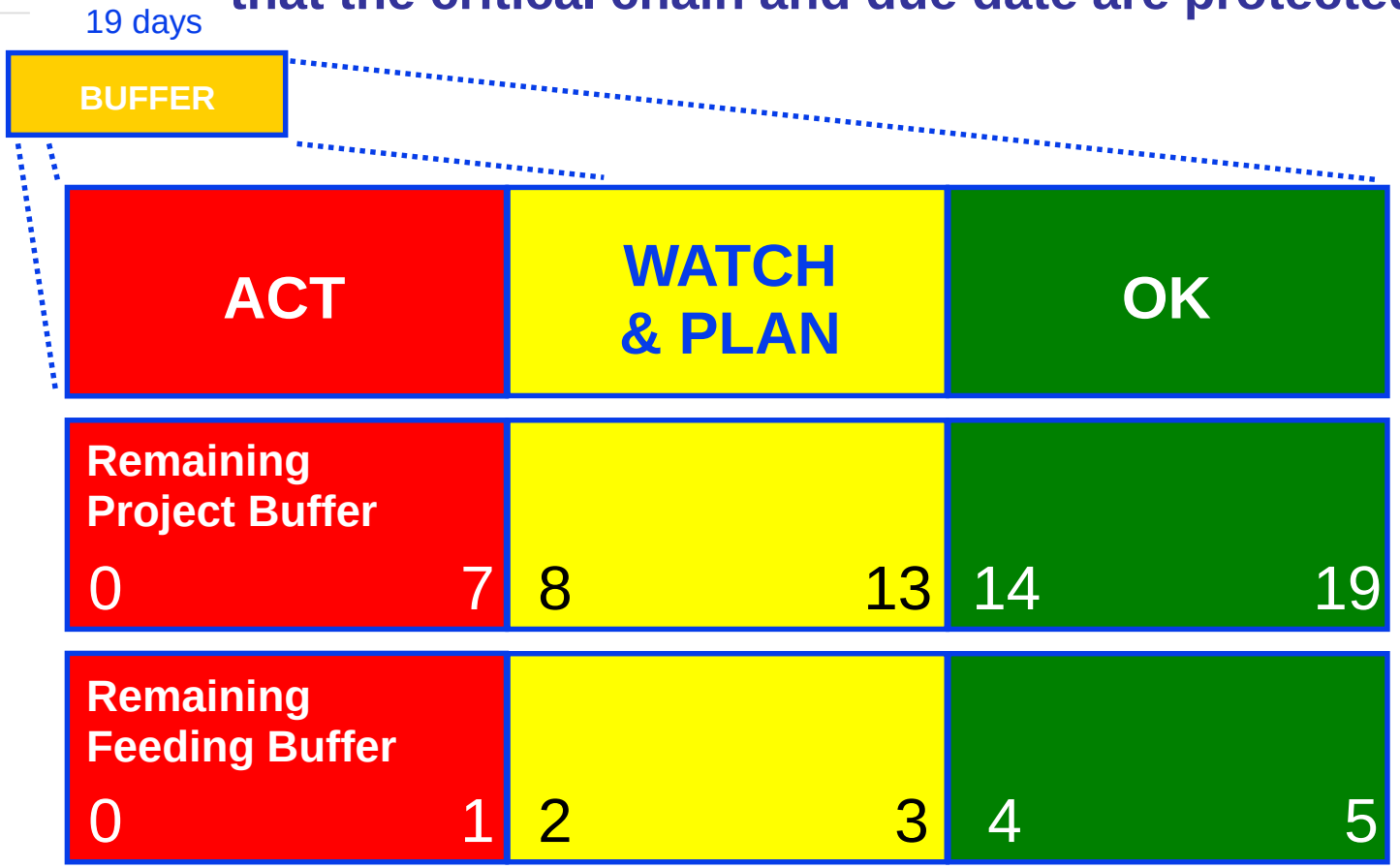


Critical Chain Definition

The critical chain is the set of activities for which there is no slack (reserve) after resource de-confliction.

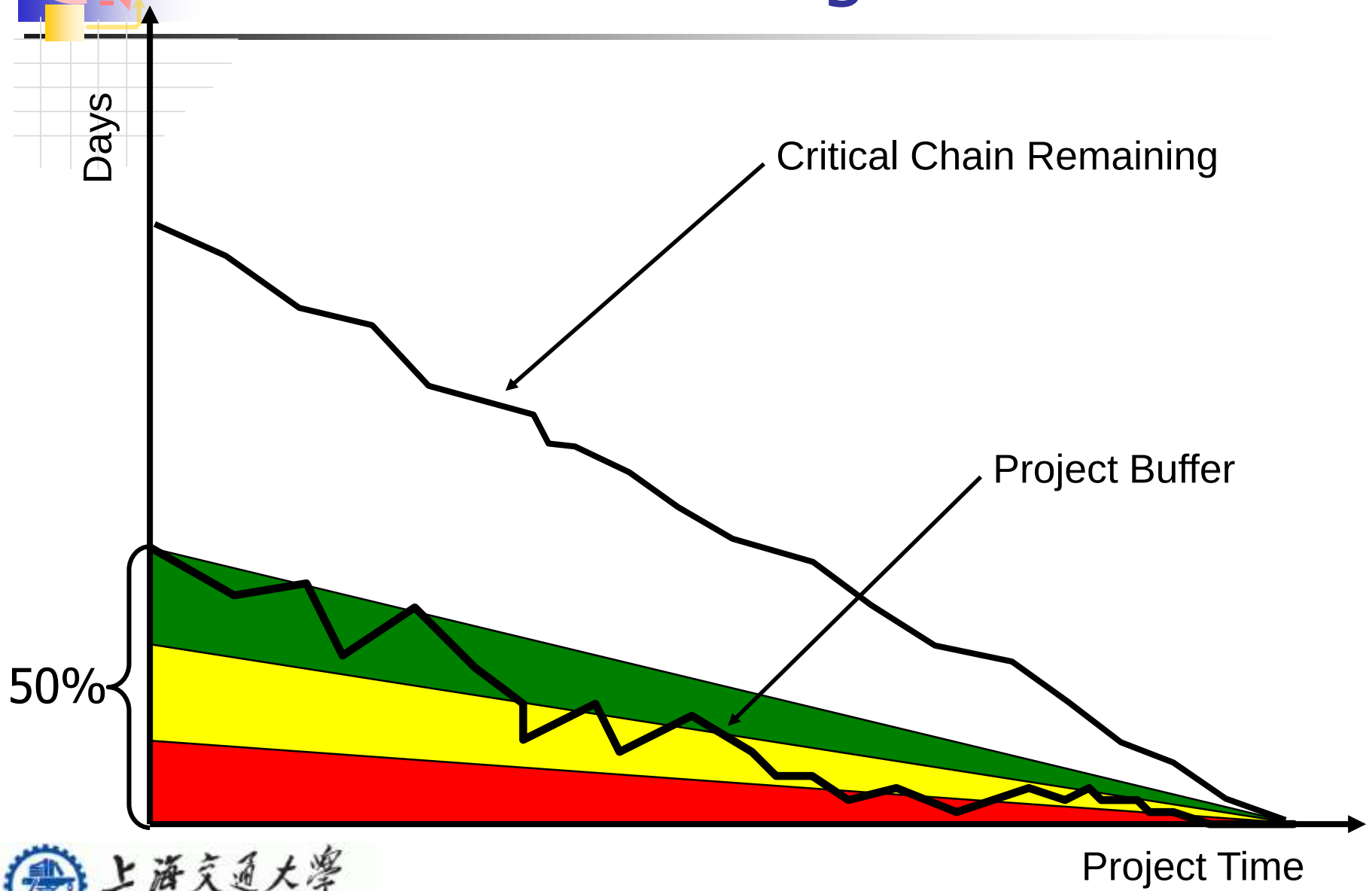
Monitoring the Project using Critical Chain

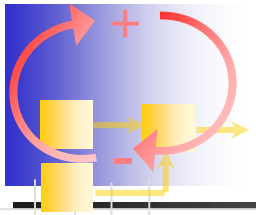
Monitor buffers to provide focus and early warnings to ensure that the critical chain and due date are protected.



Source: Avraham Y. Goldratt Institute

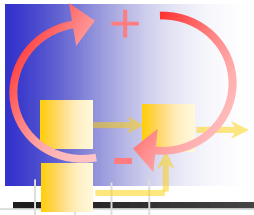
Buffer Monitoring Method





Discussion Point

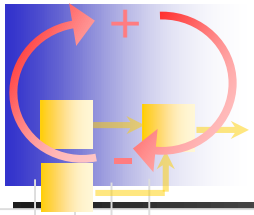
- Resource Conflicts
  - Experiences with multi-tasking
  - Lack of deconfliction of resources (designers, lab facilities,…)
    - Within a project
    - Between projects
  - Consequences of resource conflicts



Concept Question 3

- What are the **key differences** between CPM and Critical Chain?
  - A) None, the two methods are about the same
  - B) CPM focuses on managing individual due dates, whereas Critical Chain focuses on buffers
  - C) Critical Chain looks at resource constraints
  - D) CPM considers variation in task durations
  - E) Both B and C
  - F) Both C and D

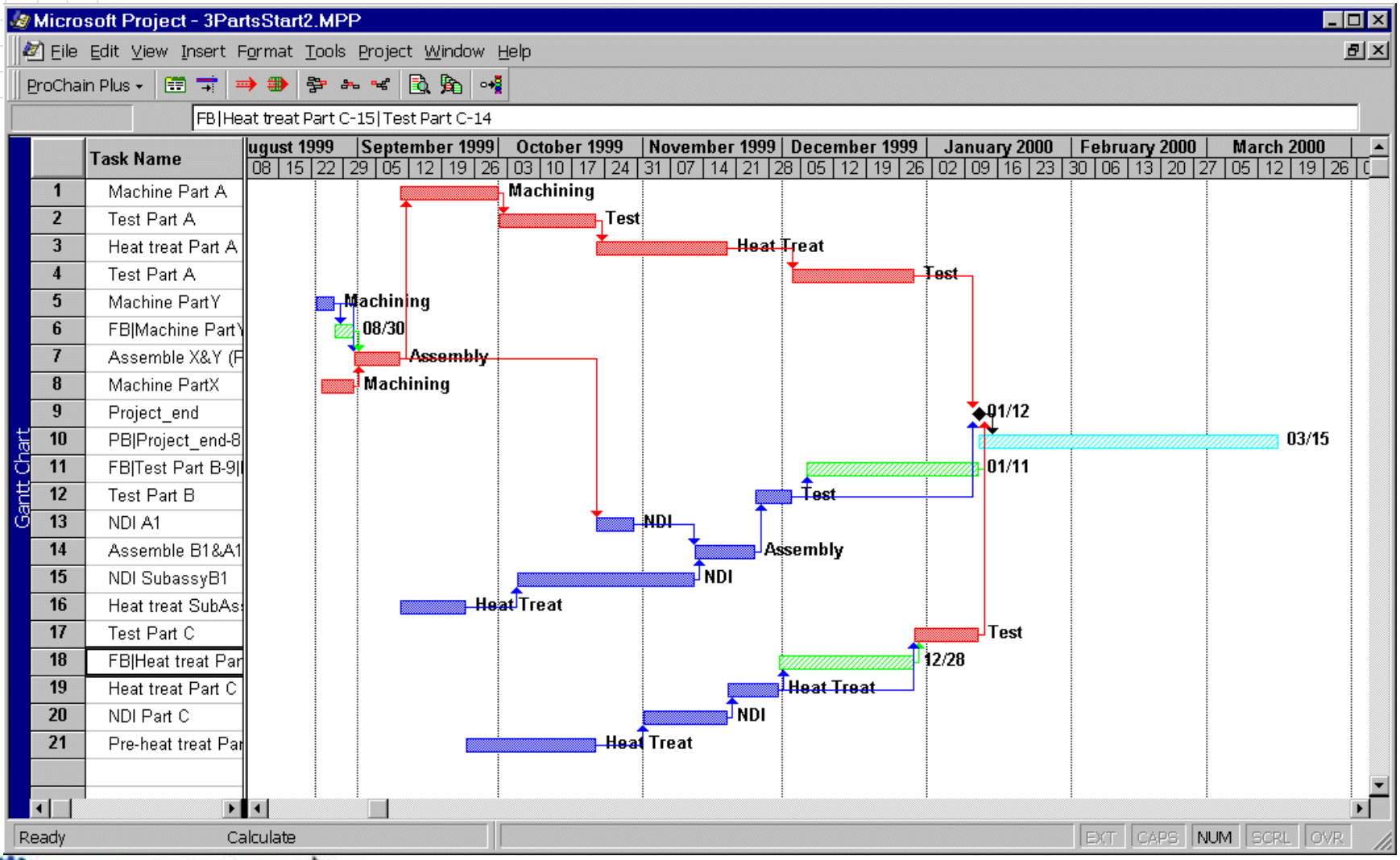
http://www.diaochapai.com/survey1668706



Cultural Keys to CCPM Successful Implementation

- How team members are evaluated:
  - Team is evaluated as a unit on overall project completion success.
  - Individual task completion due dates and milestones must be de-emphasized to avoid sub-optimization.
- Management must hold up their end of the bargain (don't multi-task).
- Need support from the top.
- All key team members must be trained and participate in putting together the schedule.
- Need very clear communication between the schedule keeper and team members.

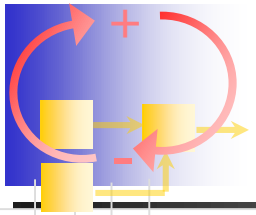
ProChain Software



Courtesy of ProChain Solutions, Inc.
http://www.ProChain.com

Assigning Task Times in CC

- Predict how long the task will take if:
  - 1) You have all necessary resources and inputs
  - 2) You only work on the task non-stop
  - 3) Either give best estimate of 50/50 time or give 85%-90% time and cut this time in half.
- Peer pressure really helps to get honest estimates.
- We found that teams do make accurate estimates.



Summary: Critical Chain

- Critical chain method makes the effect of task variance explicit (visible).
- For longer projects, we implement buffered schedule within each phase or stage.
- We have some critique of minor points, but overall critical chain appears to be a powerful new common sense management tool.
- For long-term success, the cultural paradigm shifts must take place.

HW1 Introduction

- You are Project Manager for the new UAV development project
- Plan the project
  - Task list
  - Create project graph
  - Critical path
  - Slack times
  - Re-planning after change
  - "managerial"-type questions

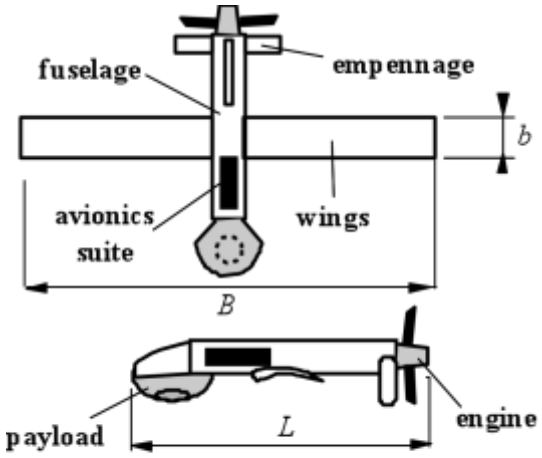


Fig 1. UAV concept, Specifications: L=2000 mm, B=3500 mm, b=500 mm

Readings for Next Class

- Book Chapter: “Design Structure Matrix Method”.
- Steven D. Eppinger, Murthy V. Nukala, and Daniel E. Whitney. "Generalised Models of Design Iteration Using Signal Flow Graphs", Research in Engineering Design. vol. 9, no. 2, pp. 112-123, 1997.
- Steven D. Eppinger, Daniel E. Whitney, Robert P. Smith, and David A. Gebala. "A Model-Based Method for Organizing Tasks in Product Development“, Research in Engineering Design. vol. 6, no. 1, pp. 1-13, 1994