U08784 Software Project Management

Semester 1 2013

Muhammad Younas (ML) + Chris Cox

Material
https://tech.brookes.ac.uk/modules/U08784

Reading:
Bob Hughes and Mike Cotterell, Software Project Management 4th Ed., Mc Graw Hill

Roger Pressman Software Engineering: A practitioners approach, Mc Graw Hill

Week	Date	Description	Lecturer	Practical
1	23-Sep	Project Planning (ref chap 1+2)	CC	Exercises: Network Analysis
2	30-Sep	Project approaches and project evaluation (ref chap 3+4)	MY	Exercises: Project approaches/evaluation
3	7-Oct	Software Quality	CC	Exercises: Quality Control
4	14-Oct	Consolidation: Help with Assignment 1	CC/MY	Consolidation: Help with Assignment 1
5	21-Oct	Software Configuration Management	CC	Exercise: Software management
6	28-Oct	Risk Management (ref chap 7)	CC	Exercises: Risk Analysis
7	4-Nov	Consolidation: Help with Assignment 2	CC/MY	Consolidation: Help with Assignment 2
8	11-Nov	Estimation 1: COCOMO (ref chap 5)	MY	Assignment 2 Presentations
9	18-Nov	Estimation 2: Function Point Analysis (ref chap 5)	MY	Exercises: COCOMO, FPA
10	25-Nov	Consolidation: Help with Assignment 3	CC/MY	Exercises: COCOMO, FPA
11	2-Dec	Project Control and Managing People and Teams (ref chap 9+11)	MY	Exercises: Project Controls
12	9-Dec			

	Assignment 1	Assignment 2	Assignment 3
Issue date	30-Sep-2013 (Week 2)	21-Oct-2013 (Week 5)	11-Nov-2013 (Week 8)
Due date	21-Oct-2013 (Week 5)	11-Nov-2013 (Week 8)	2-Dec-2013 (Week 11)
Type	Group work report	Group work report + presentation	Individual
Weight	30%	30%	40%

U08784 Software Project Management

Lecture 1a Project Management

Lecture 1b Project Planning

Reading:

Bob Hughes and Mike Cotterell, Software Project Management 4th Ed., Mc Graw Hill

Roger Pressman Software Engineering: A practitioners approach, Mc Graw Hill

http://en.wikipedia.org/wiki/Project_management

C Cox

What is Project Management

C Cox

Involves:

- Organising and managing resources (people, equipment, materials, time, money, space, energy, carbon etc.)

Primary objective:

- Ensuring that projects are delivered within the defined resource constraints

Secondary objective:

- To optimise and minimise the use of available resources

Motivation for Software Project Management

Software Crisis:

- Growth in computer power
- Advances in technology
- Growth in application size and complexity
- Less rapid development (and adoption) of design methods resulted in many high profile software project failures.

Led to developments in:

- Software engineering design methods (OO, UML, testing, etc)
- Software Quality Assurance (SQA)
- Software testing
- Project management

as a means of improving the likelihood of project success, but no guarantee

Features and functions used in a typical system
(Standish group, 2002)

Always 7%
Often 13%
} 20% useful features

Sometimes 16%

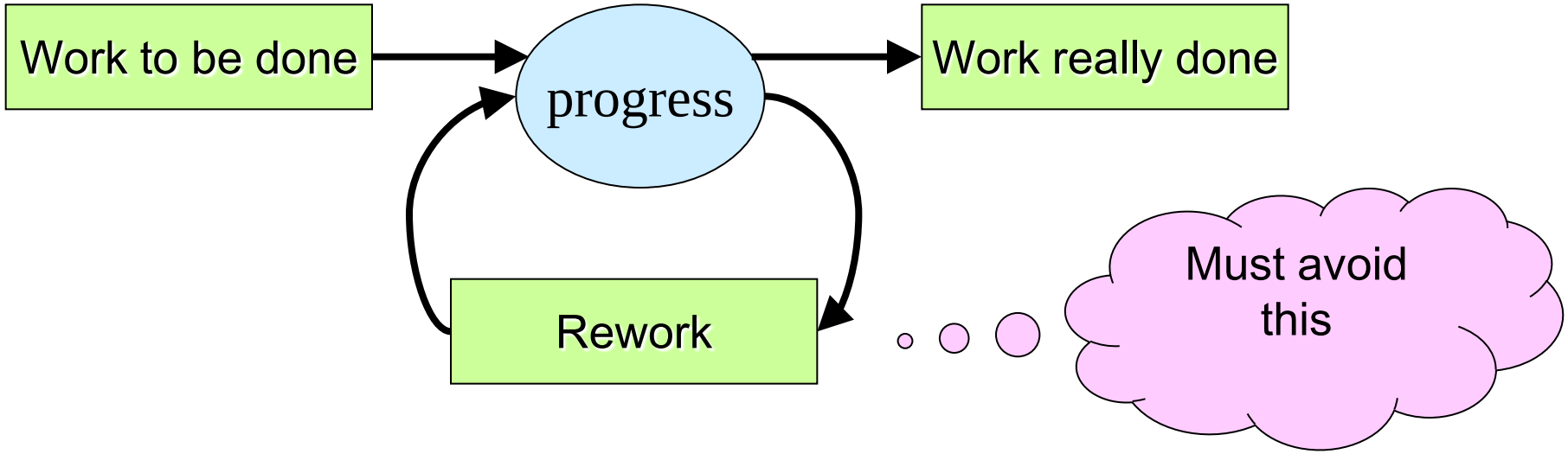
Rarely 19%
Never 45%
} 64% mostly wasted effort

Project Success
(Standish group, 2009)

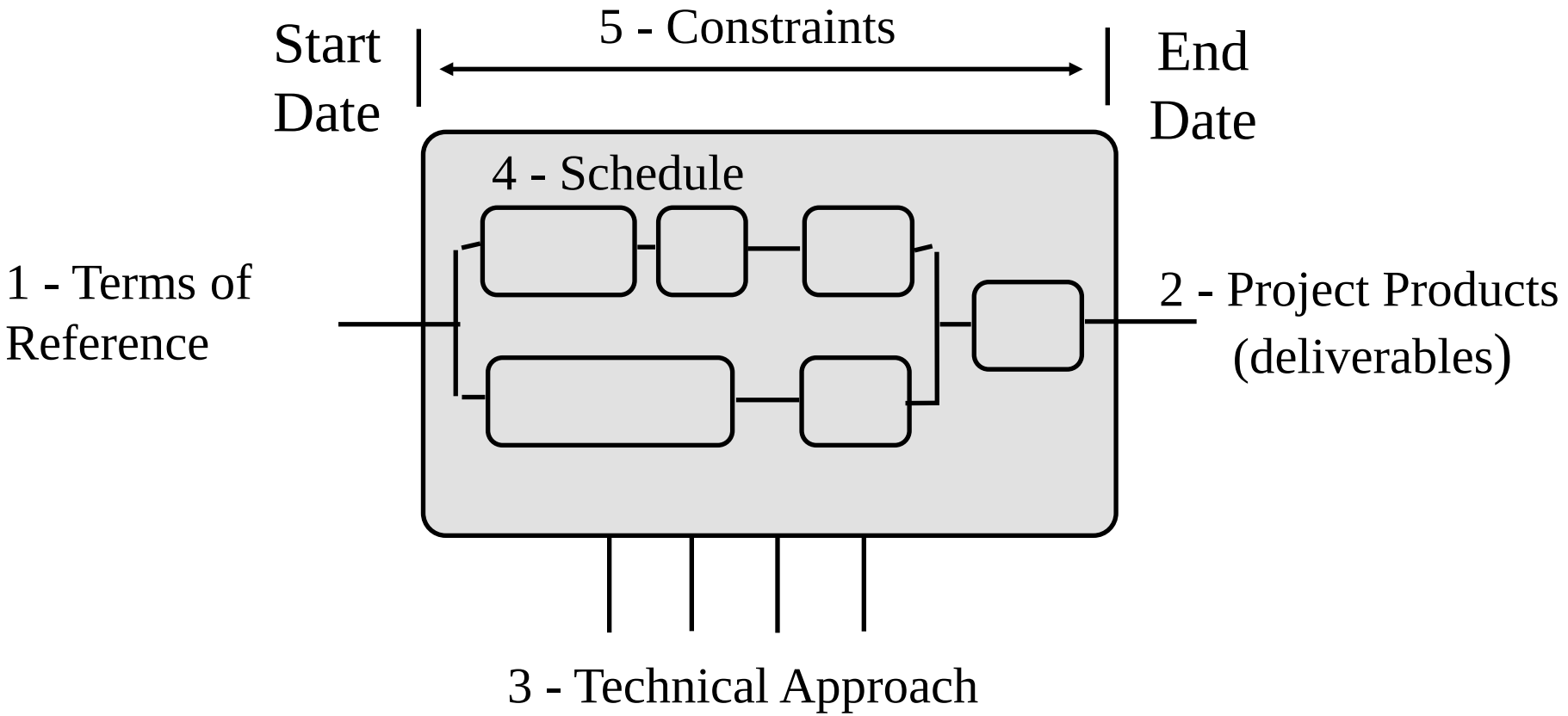
(Project outcome)	1994	1996	1998	2000	2002	2004	2009
Succeeded	16%	27%	26%	28%	34%	29%	32%
Challenged	53%	33%	46%	49%	51%	53%	44%
Failed	31%	40%	28%	23%	15%	18%	24%

Why is software project management hard

- Poorly defined project objectives
- Shifting system specification
- Mis-match between objectives and resources
- Work actually being done not exactly as planned



Key Features of a Project



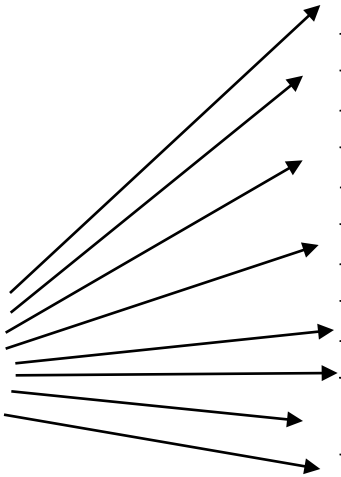
Why do Project Fail?

Lack of Project Definition

Poor Project Management

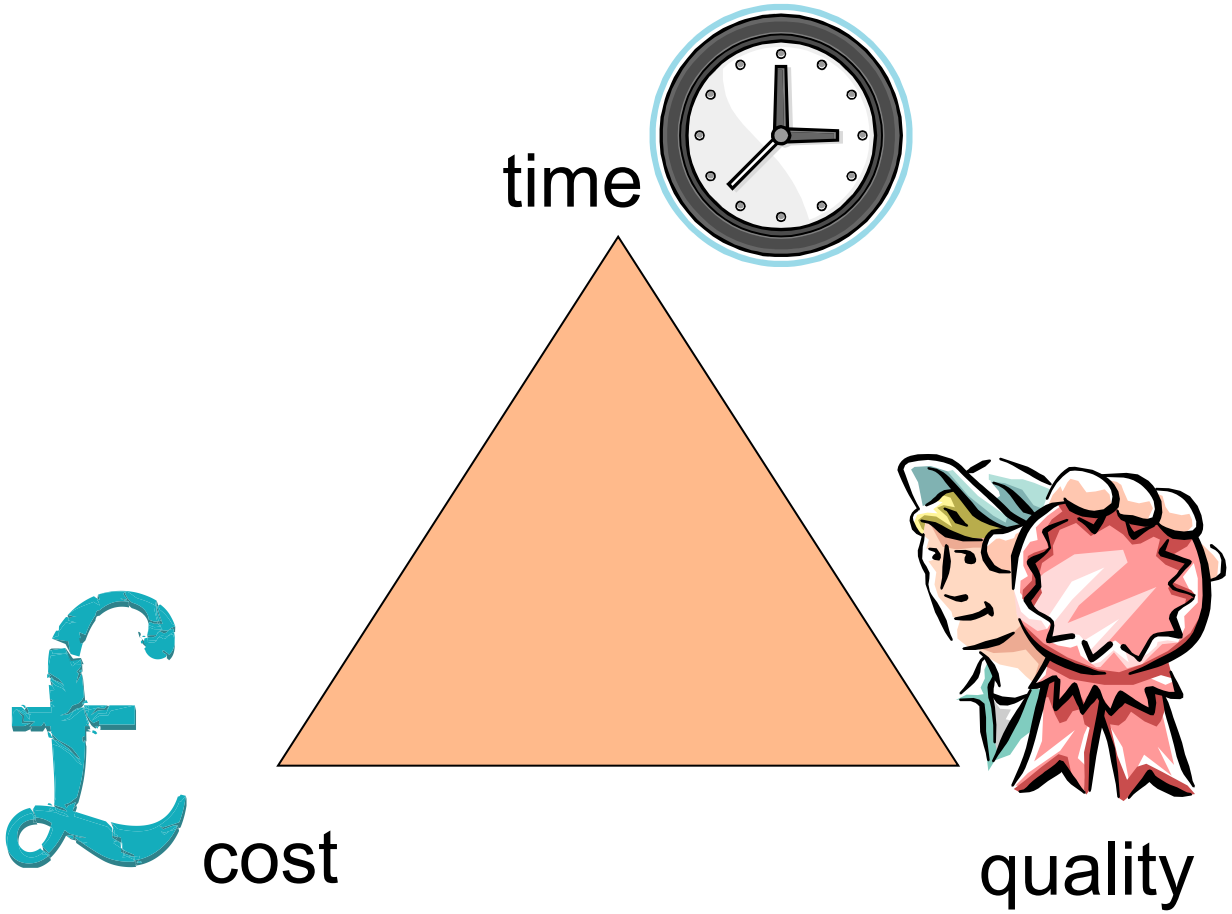
Poor Quality Control

Lack of Client Involvement



- Late Delivery
- Excessive Cost
- Rushed Implementation
- Not Relevant to User Needs
- Hard to Use
- Unreliable
- Difficult to Modify

Linked project constraints

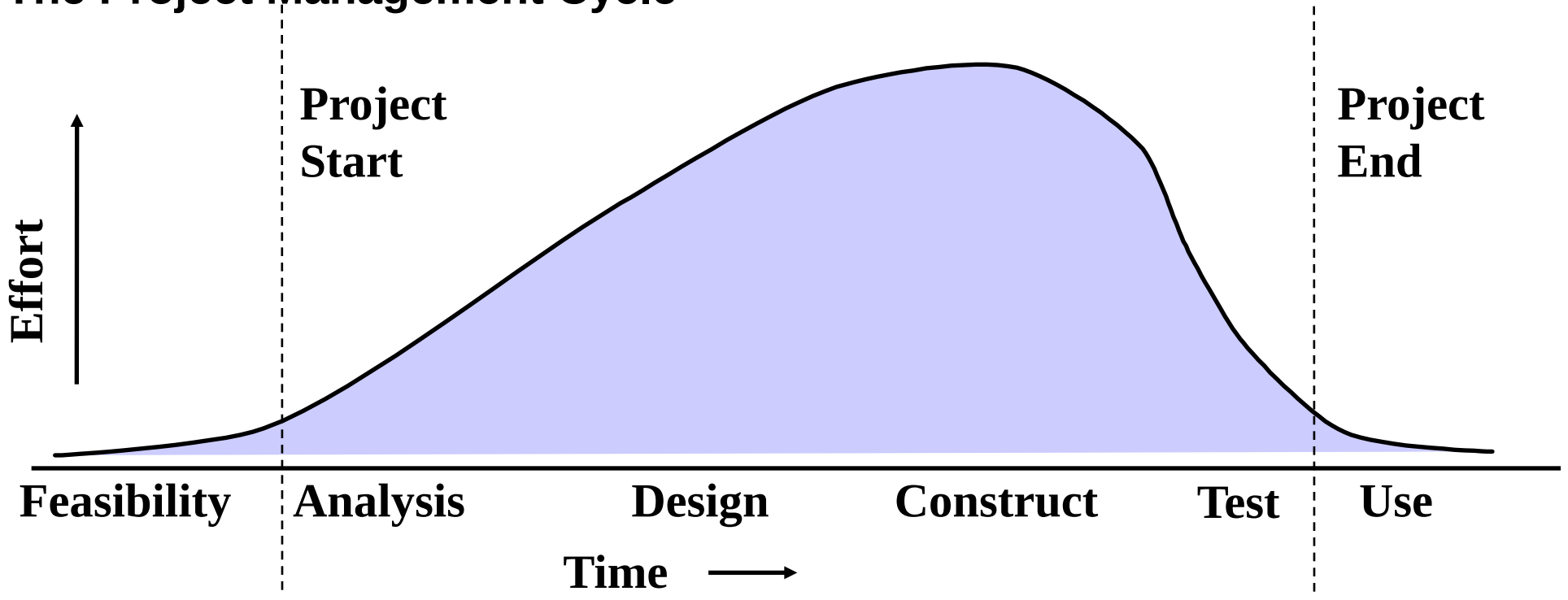


A generic software development design process

Stage	Activity	Deliverable	SQA
Requirements (specify What)	Tender Feasibility Systems analysis	Tender Feasibility Proposal Requirement spec.	Project set-up Requirements review
Design (specify how)	System design	Test Plan Design spec.	Design review
Implementation (build)	Program development Testing	Executables User guide Release notes Software reference	Release procedure Acceptance testing
Operation (live use)	Training Support maintenance	Training request Enhancement request Bug report	Change request

Based on: Plessey Research (Caswell) Ltd. CAD Software Standards

The Project Management Cycle



Define	Plan	Monitor & Control	Review

Project Management

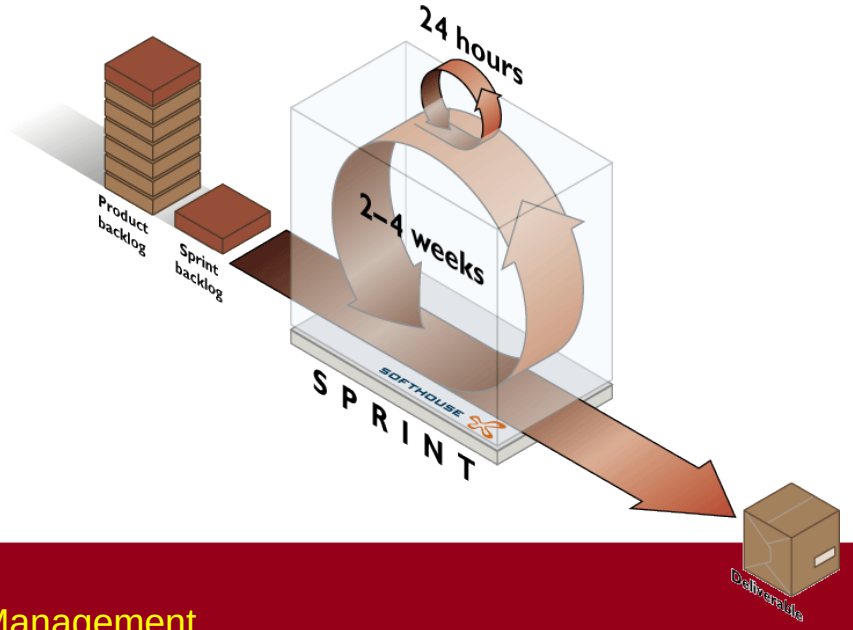
C Cox

Scrum – An Agile software development method

- Manage people, product and development
- Iterative and incremental development
- Ken Schwaber & Mike Beedle (2002) Agile software development with Scrum, Prentice Hall
- Core Roles: product owner(customer voice), development team(3-9 people), Scrum master (enforce rules and keep development focus but not PM)
- Ancillary roles: stakeholder(customers,vendors), managers(control staff and environment)

Scrum – Sprints

- The basic unit of development
- 1 week – 1 month activity
- Preceded by planning meeting (resource estimation)
- Followed by review of progress, and to inform next sprint
- Each spring creates a portion of a product deliverable
- Sprint features taken from list of requirements
- Recognises that customers may alter requirements



C Cox

Scrum – Meetings

Daily Scrum – to review project status

- Members come prepared with updates
- Punctual start, same location & time each day
- Max 15 mins
- Core roles speak
- All team welcome to attend
- Discuss progress, plans & problems (since yesterday)
- Documented by scrum manager
- Detailed discussions is outside scope of daily scrum meeting

Sprint review

- Review of sprint work
- Present/demo to stakeholders
- 4 hour max

Conclusion: characteristics of a software project

- Has Start and End Date
- Has Specific Objectives (e.g. functional+non functional requirements)
- Developed by a Team for a Client
- Uses Limited Resources and Incurs Costs
- Involves a Number of Simultaneous and Dependant Activities
- Has Associated Risks & Uncertainties
- Needs to satisfy Clients need
- Needs to satisfy functional criteria
- Needs to satisfy non functional criteria
- Needs to satisfy Performance criteria
- Needs to satisfy Quality Criteria
- Needs to satisfy legal and standards requirements (data protection, H&S, etc)

C Cox

. . . Project examples

http://www.bbc.co.uk/news/uk-14990549

http://www.bbc.co.uk/news/uk-14986690

http://www.bbc.co.uk/news/uk-15014288

http://www.bbc.co.uk/news/health-13430375

http://www.bbc.co.uk/news/uk-politics-24130684

U08784 Software Project Management

Lecture 1a Project Management

Lecture 1b Project Planning

Aims: - Appreciate the need for project plans
- Understand the use of project planning
- Use appropriate technique to plan projects

Reading:

• Chapter 6 – Activity Planning in Bob Hughes and Mike Cotterell (1999) Software Project Management 2nd Ed., Mc Graw Hill.

• Chapter 4 – Project Planning and Risk Management in Christian W. Dawson (2005) Projects in Computing and Information Systems: A Students Guide, Addison Wesley.

• Roger Pressman Software Engineering: A practitioners approach, Mc Graw Hill.

C Cox

1 Stages in project planning
2 Work breakdown structure
3 Estimating task duration
4 Network or PERT chart
5 Gantt – task vs. time bar chart
6 Risk analysis
7 Monitoring progress
8 Builds
9 Teamwork

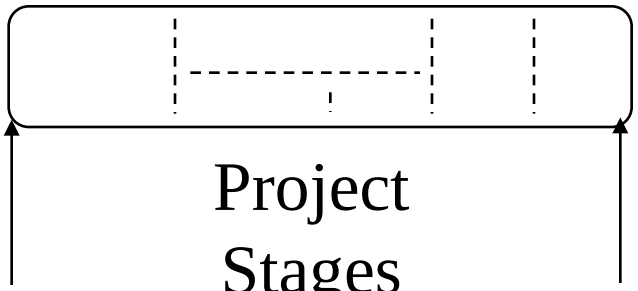
1 Stages in Project Planning

- Project scope – set out tasks and methods to be used
- Estimate task durations
- Work breakdown structure to show hierarchy and grouping of tasks
- Task dependencies identified, from which a network analysis (PERT chart) is used to determine project duration and critical path
- Task versus time planning using Gantt chart
- Resource costs estimated for each task

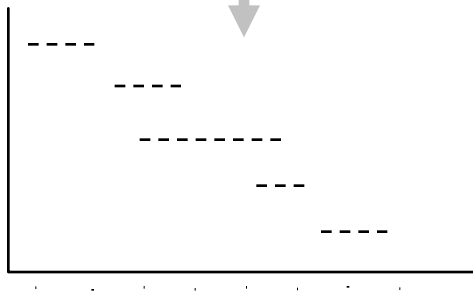
Project Planning - Overview

Client Requirements

(may be one *phase* of a larger development)



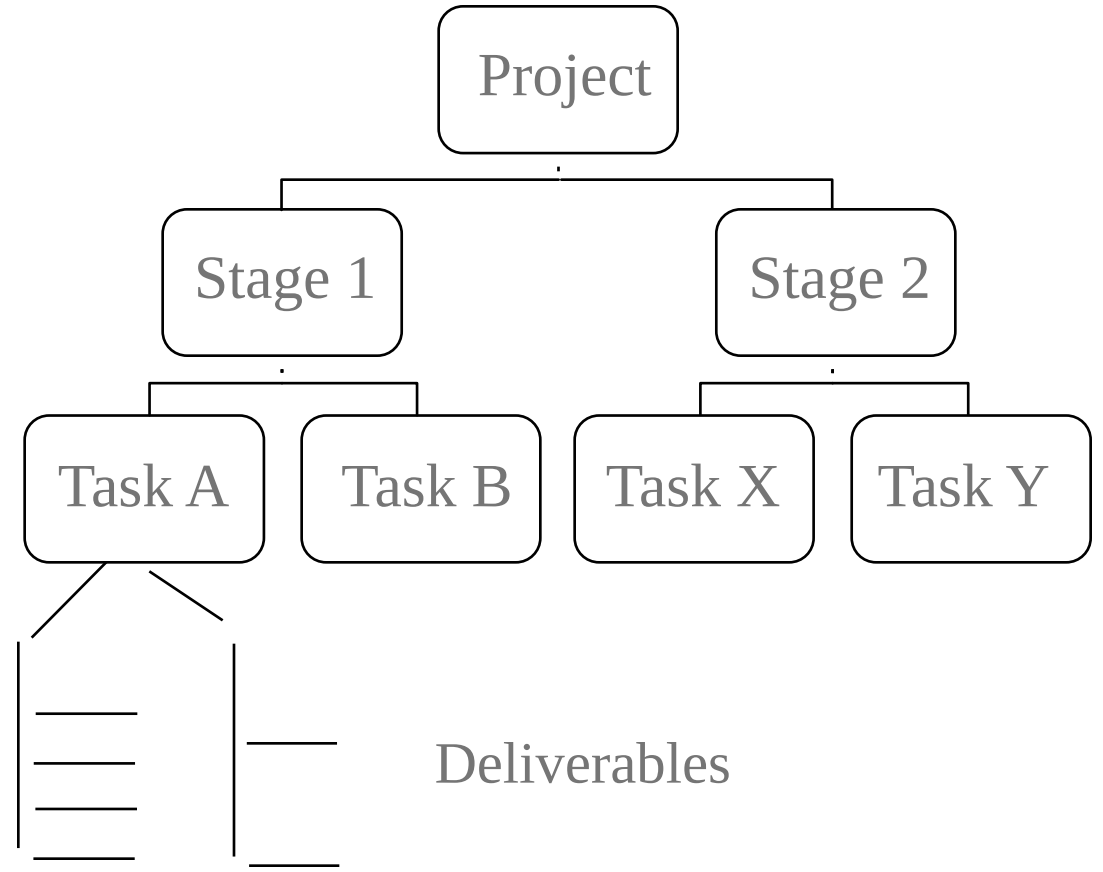
Outline Project Plan



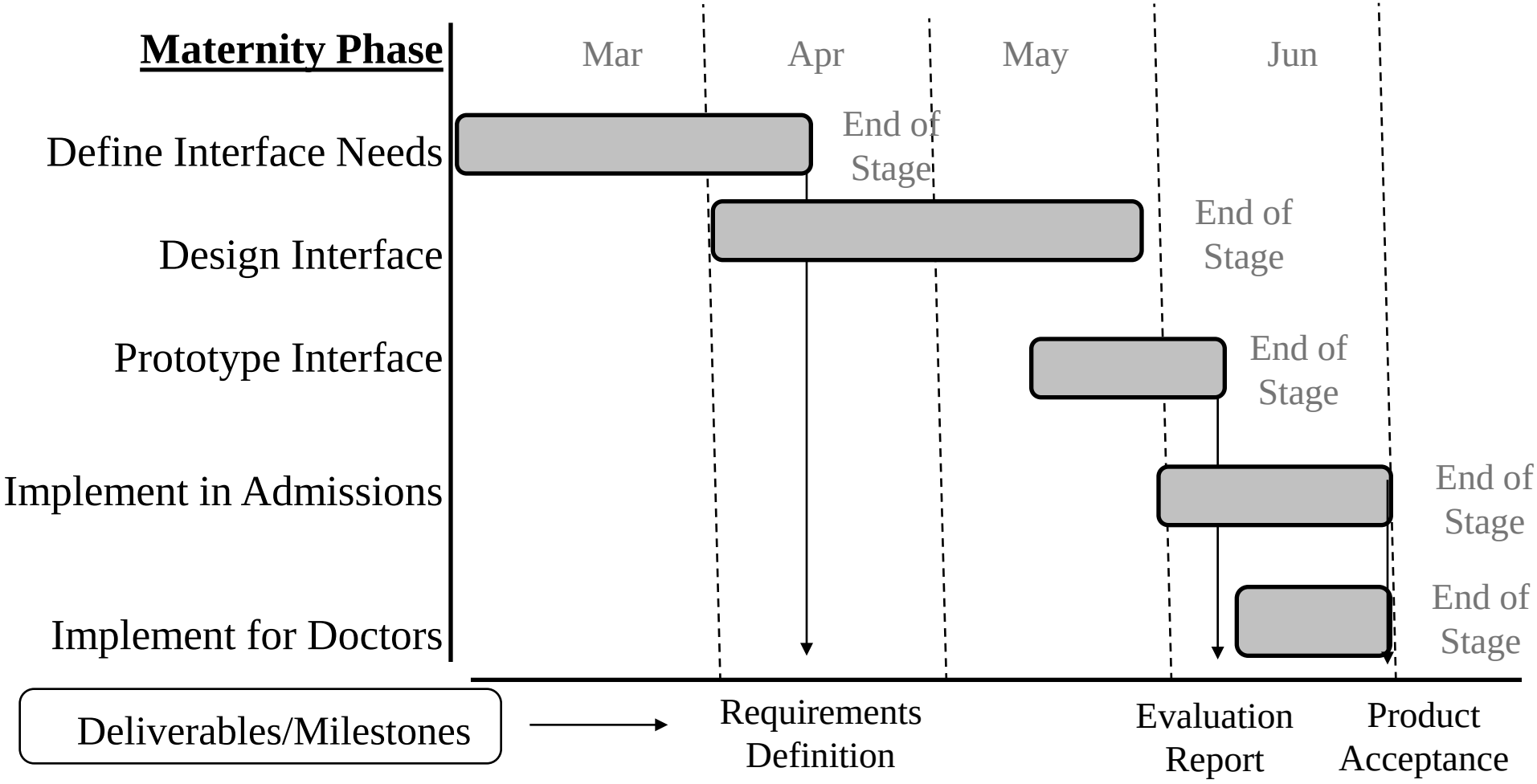
Phase Plan

Levels of Breakdown

Task/Activity/Deliverable checklists should be defined



Gantt Chart – map tasks onto time



Break down project task into:

Activity
– unit of work to be completed within a defined time window
– may be dependant on concurrent activities
– typically assigned to an individual

Deliverable
– an output from a project that can be assessed, e.g. document, code
– are indicators of progress

Milestone
– are a point in time at which progress on a project can be assessed, e.g. delivery of document or completion of alpha testing

2 Work Breakdown Structure

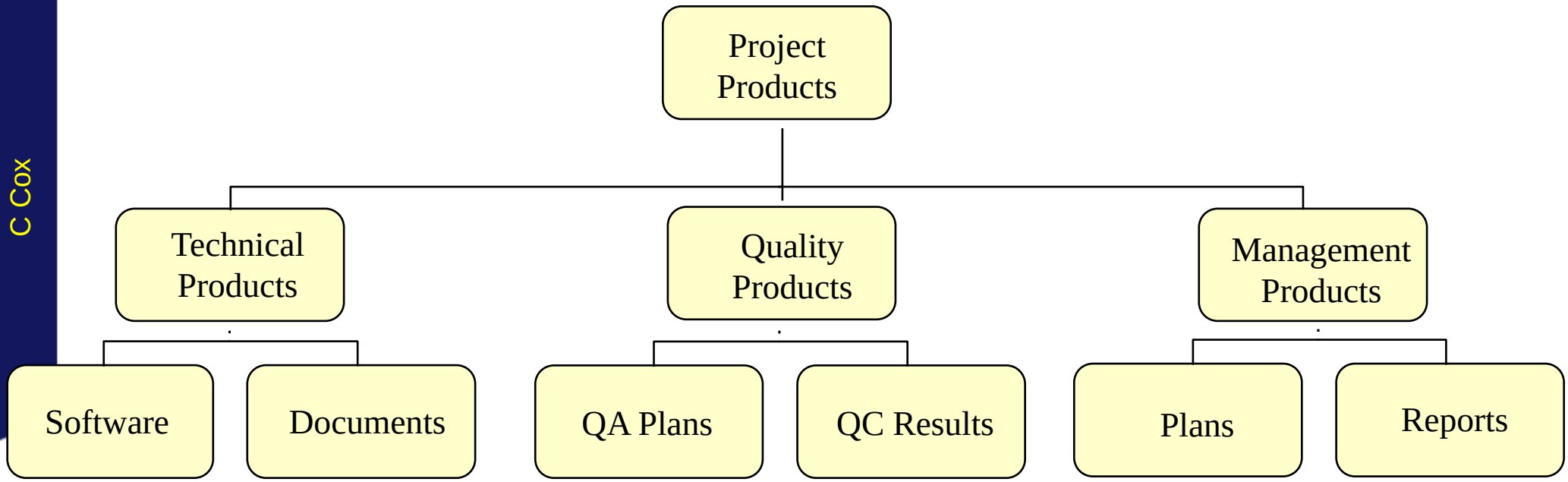
Work breakdown structure (WBS): Shows **hierarchy** of project tasks or work products. It forms the framework for a project plan specifying what tasks are needed, but not when the activities will occur. Its essentially a structured project To-Do list.

Derived from the WBS are:

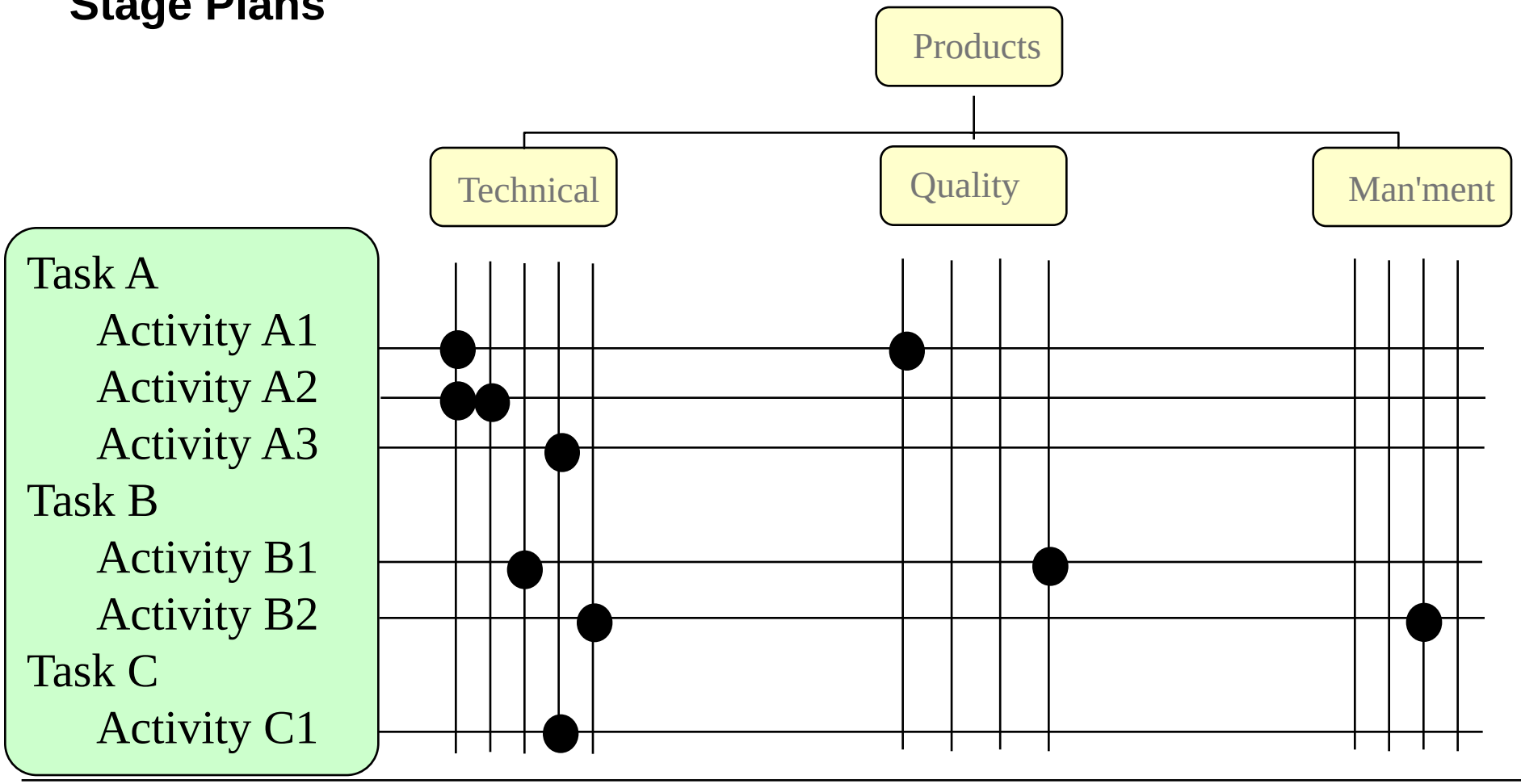
- **PERT chart**: Shows the **sequence** in which activities must be done (dependencies)

- **Gantt Chart or Schedule**: Shows scheduling of work products as a function of time

Product Breakdown

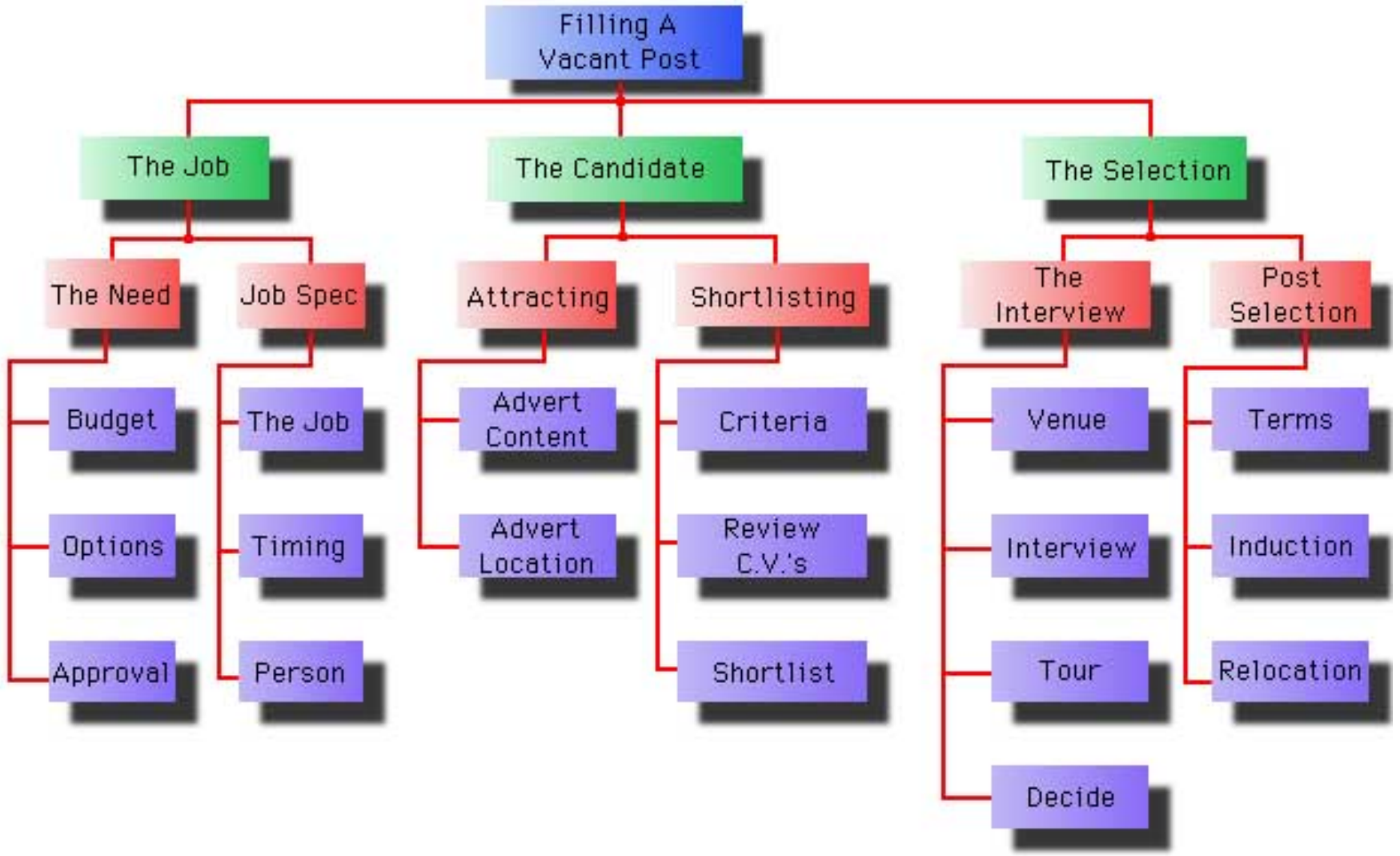


Stage Plans

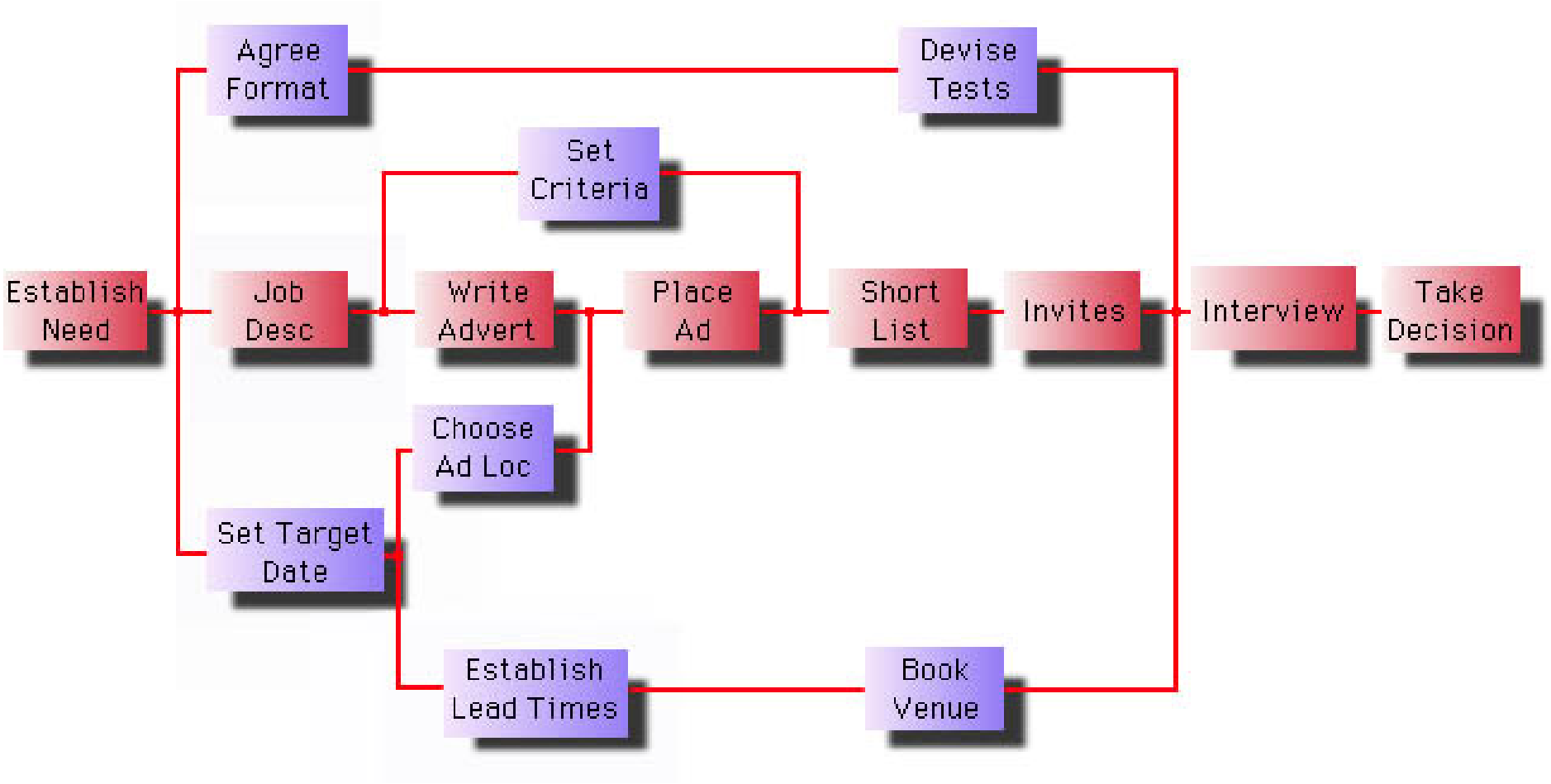


Need to identify which task/activity contributes to which product

Example Work Breakdown Structure (WBS) Diagram (1)

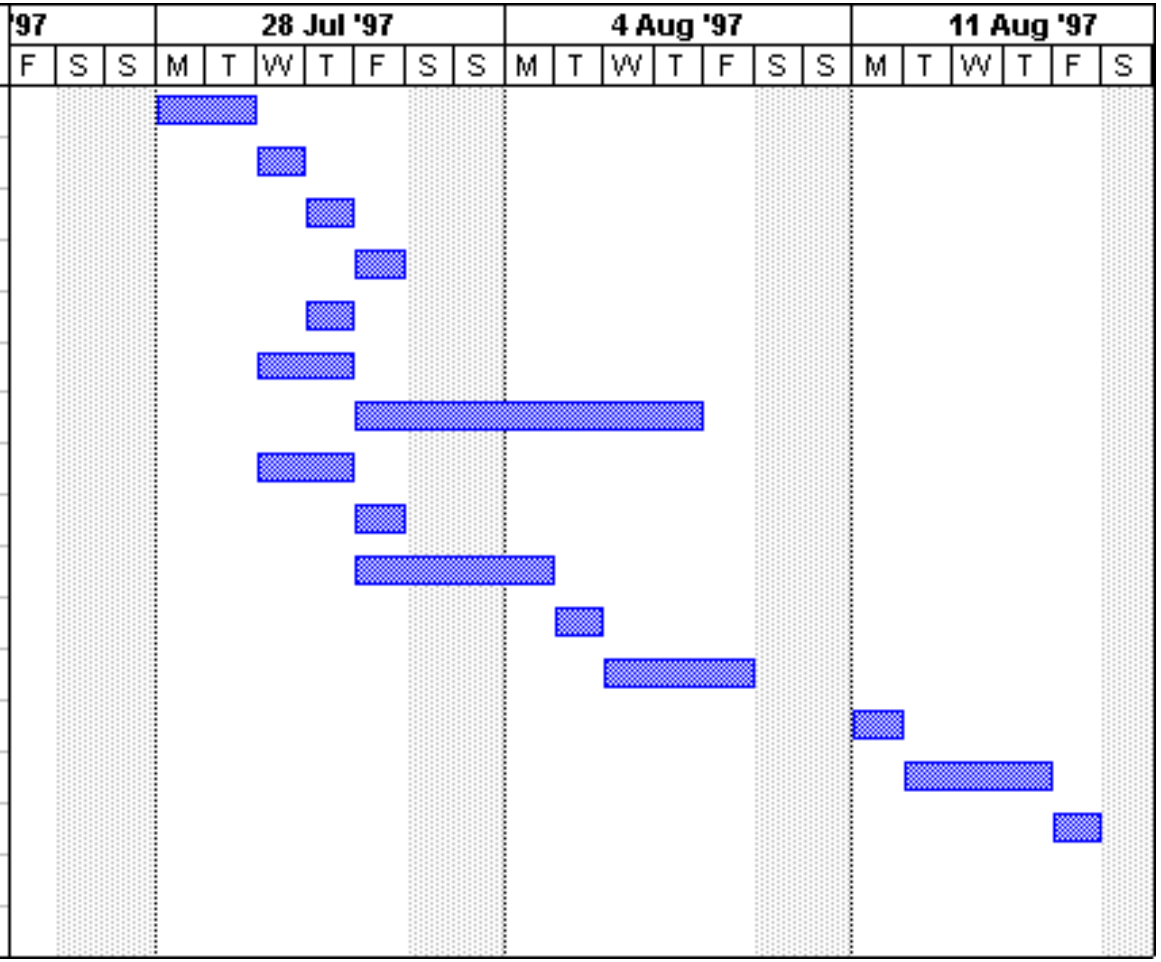


Corresponding PERT chart (or Network diagram)



Corresponding Gantt chart

ID	Task Name	Duration
1	Establish the need	2d
2	Set target date	1d
3	Establish leadtimes	1d
4	Book venue	1d
5	Choose advert location	1d
6	Agree format	2d
7	Devise tests	5d
8	Job description	2d
9	Set criteria	1d
10	Write advert	2d
11	Place advert	1d
12	Shortlist	3d
13	Invite candidates	1d
14	Conduct interviews	3d
15	Take decision	1d
16		
17		



3 Estimating Task Durations - qualitative

- Start with the design
- Break project into *tasks*
  - Indivisible units of work for one person
  - Rules of thumb (for software development):
    - Nothing less than a day is a task
    - Anything more than a week is at least two tasks
    - Longer tasks harder to estimate
    - Need to think about how to break into logical pieces

- Assign tasks to resources and individuals
- Individuals estimate time for their own tasks
  - They know their own abilities best
  - Genuine commitment to their own promises (sense of responsibility)
- Apply fudge factor (people generally underestimate what is required)
- Underestimating the resources required is a generally accepted risk

Duration time estimates – quantitative

Each task requires a duration (time required for the work to complete). Using PERT (programme evaluation and review techniques), three estimates are required for each activity:

Optimistic time - i.e. the best time possible for completing the activity
Pessimistic time - i.e. worst possible time
The most likely time

These three times are used to give a weighted (best estimate) mean using:

$$\text{Time} = \frac{\text{Optimistic time} + 4*\text{Most likely time} + \text{Pessimistic time}}{6}$$

C Cox

Fudge Factor

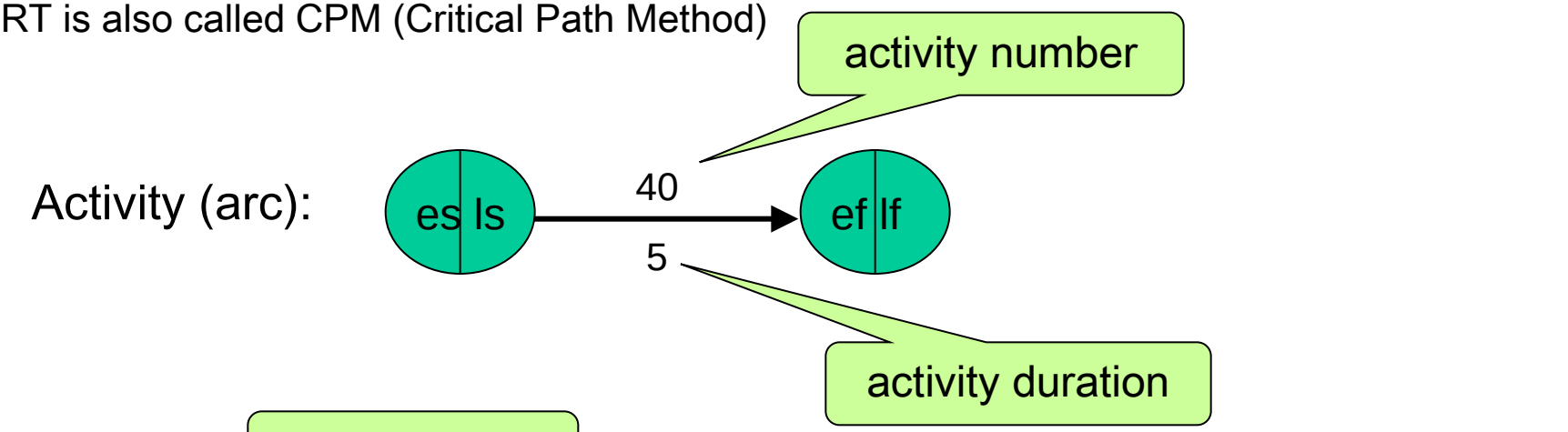
- Scale estimated time by a fudge factor
  - Allows for the inevitable unexpected problems
  - Individuals generally underestimate what is required

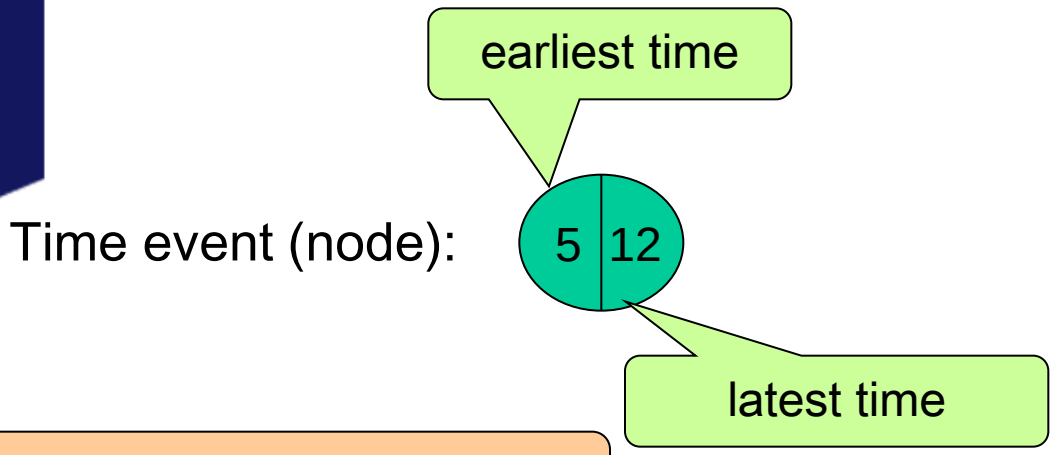
E.g. Take an individuals estimated time and double it.

4 Network or PERT Chart - Notation

PERT "Program Evaluation and Review Technique" U.S. Navy, 1957

PERT is also called CPM (Critical Path Method)





dummy activity – used to indicate a task dependency

Node=event or milestone

Activity on Node vs. Activity on Arc

- **We will use activity on arc (or arrow)**

Activity on Arc

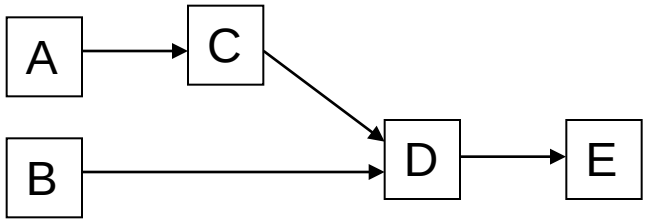
- **Nodes represent time points**
- **Arcs represent activities**
- **Have 1 start and 1 end node**

Activity on Node

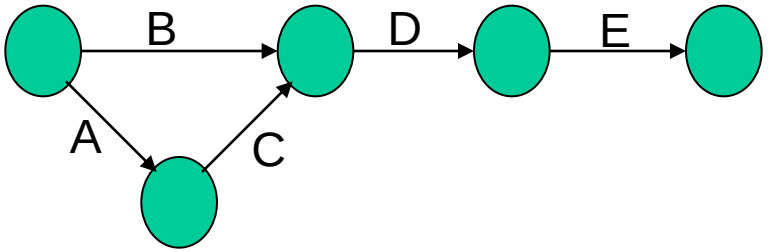
- Node represents activity
- Arc represents dependency
- Could have more than 1 start and more than 1 end node

Activity	Dependency
A	-
B	-
C	A
D	B, C
E	D

Activity on Node



Activity on Arc



Creating a PERT chart

- Break project into *tasks*
  - Requires a good design with good interfaces
  - Allows tasks to be correctly enumerated
  - Allows parallel development to be identified
- Realism in estimating task length
- Observable progress
  - Tasks are clearly done or not done
- Prioritization

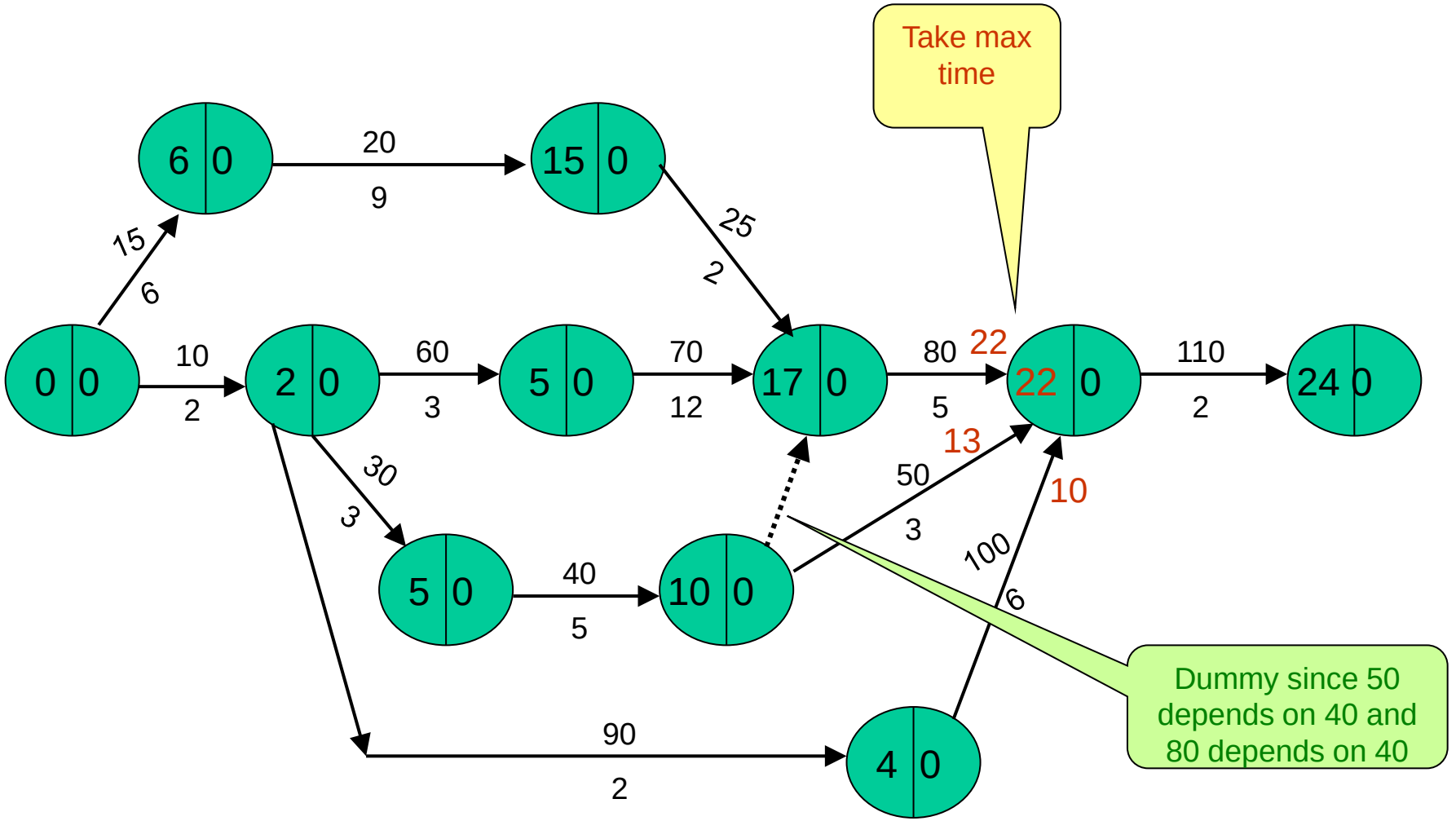
Steps in Constructing A PERT Chart

- Step
- 1 Prepare a work breakdown structure
- 2 Estimate the duration of activities
- 3 Milestone identification
- 4 Establish logical sequence between activities
  – determine dependencies between tasks
- 5 Draw the basic network in rough
- 6 Forward pass to determine earliest start, earliest finish
- 7 Backward pass to determine latest start, latest finish
- 8 Calculate the total project time and float (spare) time
- 9 Identify activities on the critical path
- 10 Allocate resources
- 11 Smooth out the network
- 12 Check the network
- 13 Discuss and refine until it is appropriate

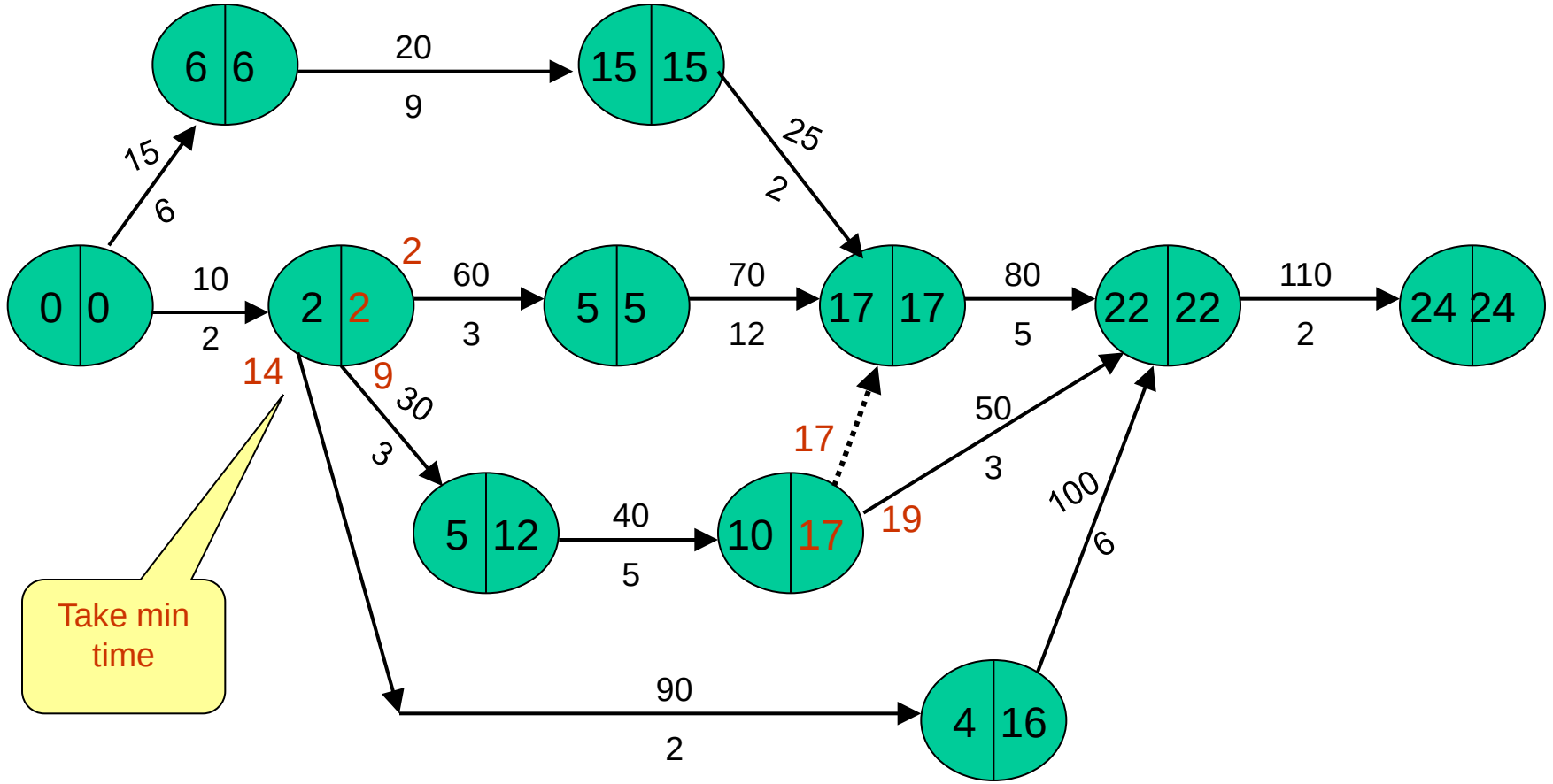
Example Work Breakdown

ACTIVITY		DURATION (Weeks)	DEPENDS ON
10	Confirm Design	2	
15	Prepare Room	6	
20	Install Network	9	15
25	Test Hardware	2	20
30	Data entry software	3	10
40	Create Data files	5	30
50	Do Stock update	3	40
60	Write Program Specs.	3	10
70	Construct Programs	12	60
80	Test System	5	25,70,40
90	Familiarise users	2	10
100	Train Staff	6	90
110	Handover	2	100,80, 50

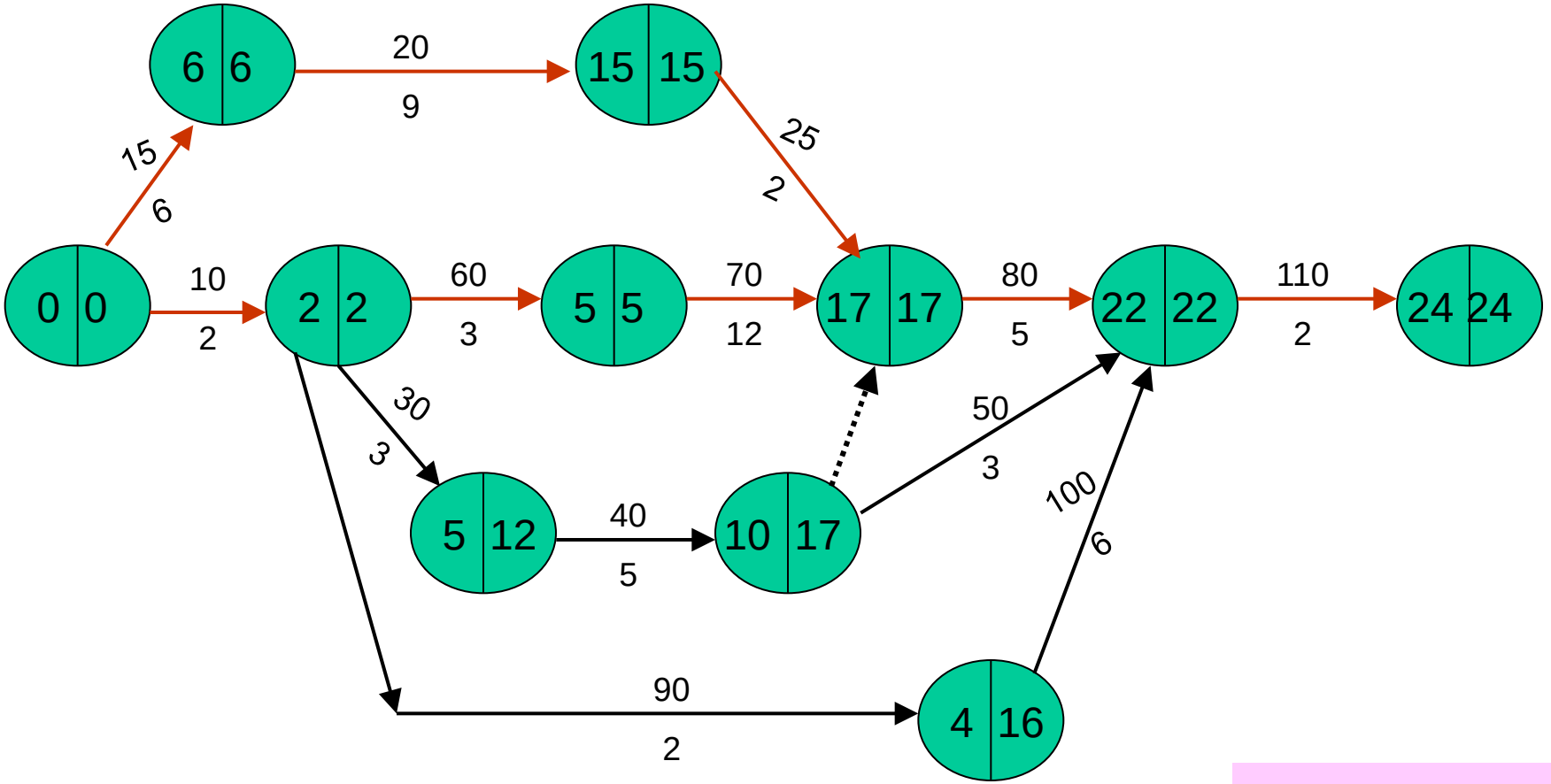
Forward pass (left to right) to determine task earliest start and finish times



Backward pass (right to left) to determine task latest start and finish times

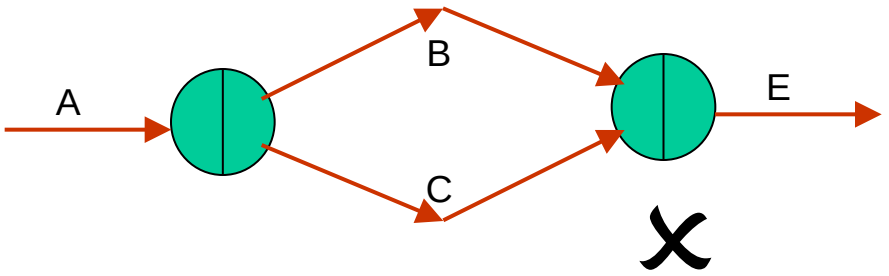


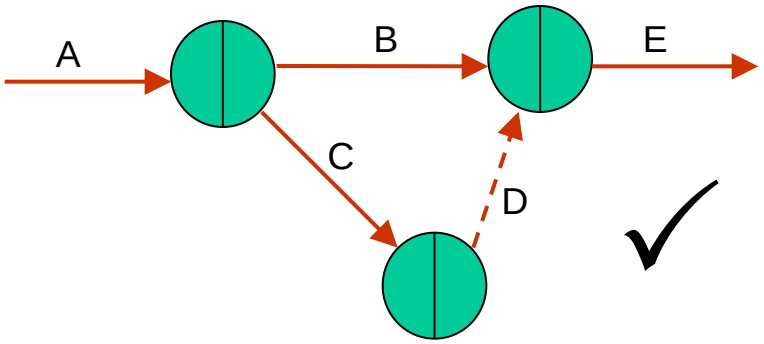
Identify Critical Path/s (path where float=0)



. . . Half way . . .

Dummy Activity – example 1





B depends on A

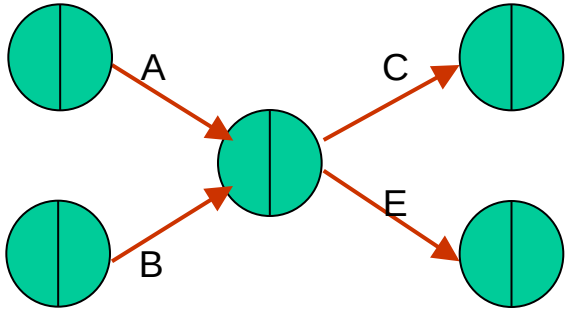
C depends on A

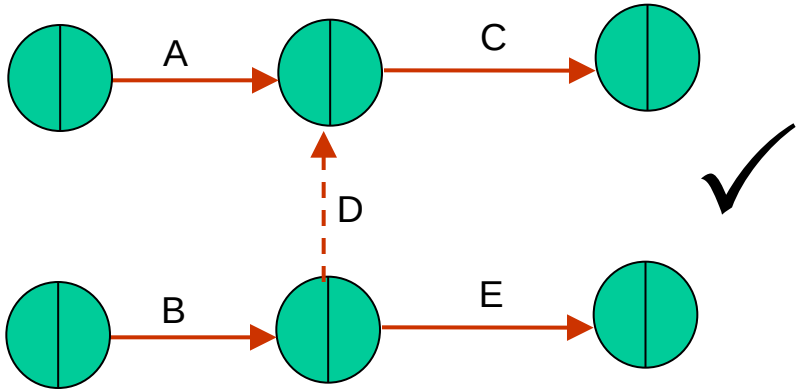
E depends on B and C

But can't have multiple paths between same end points

Solution: use dummy D

Dummy Activity – example 2





C depends on A and B

E depends on B alone

Dependency here is wrong on diagram

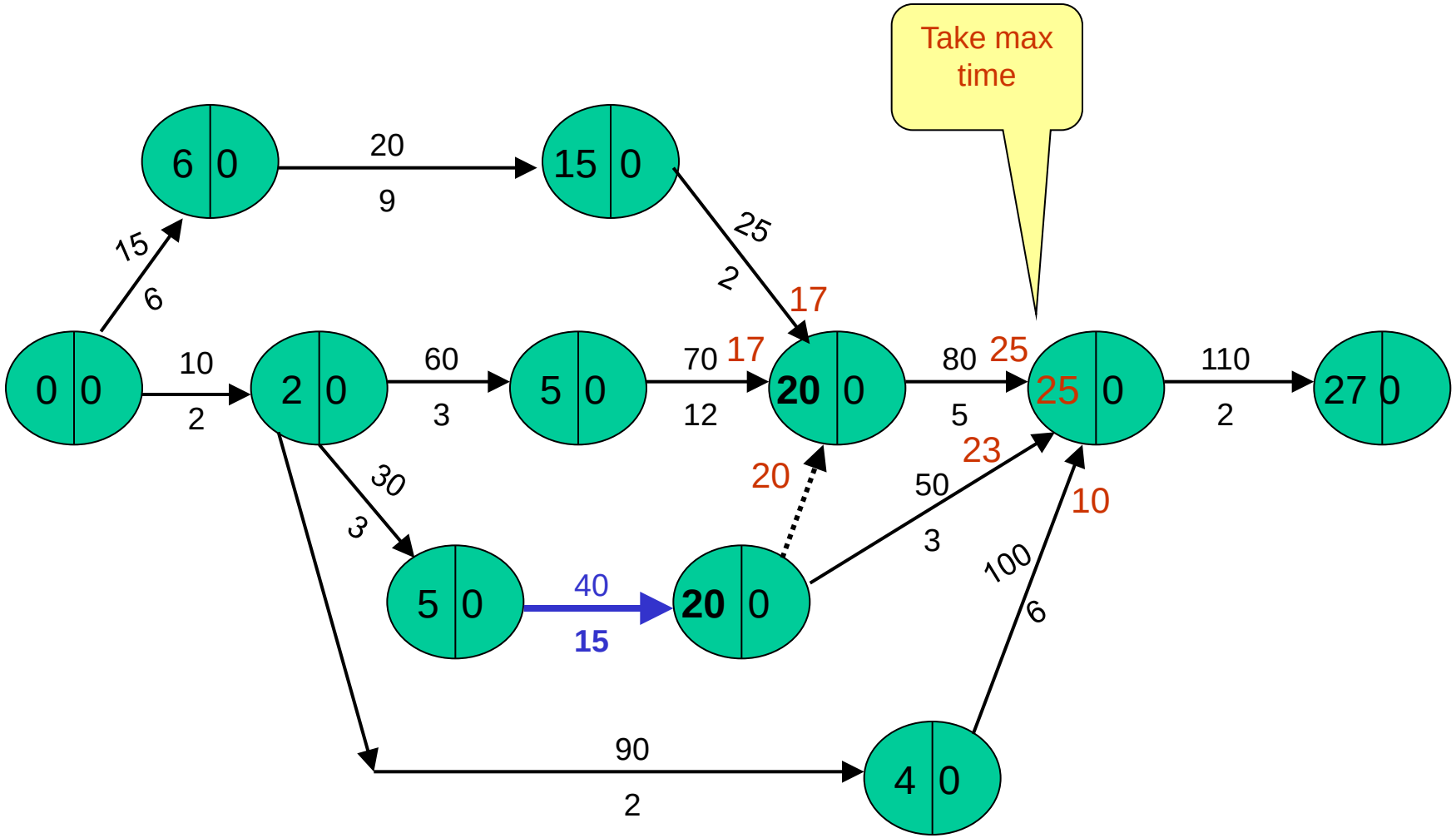
Solution: use dummy D

Changes to the PERT Chart

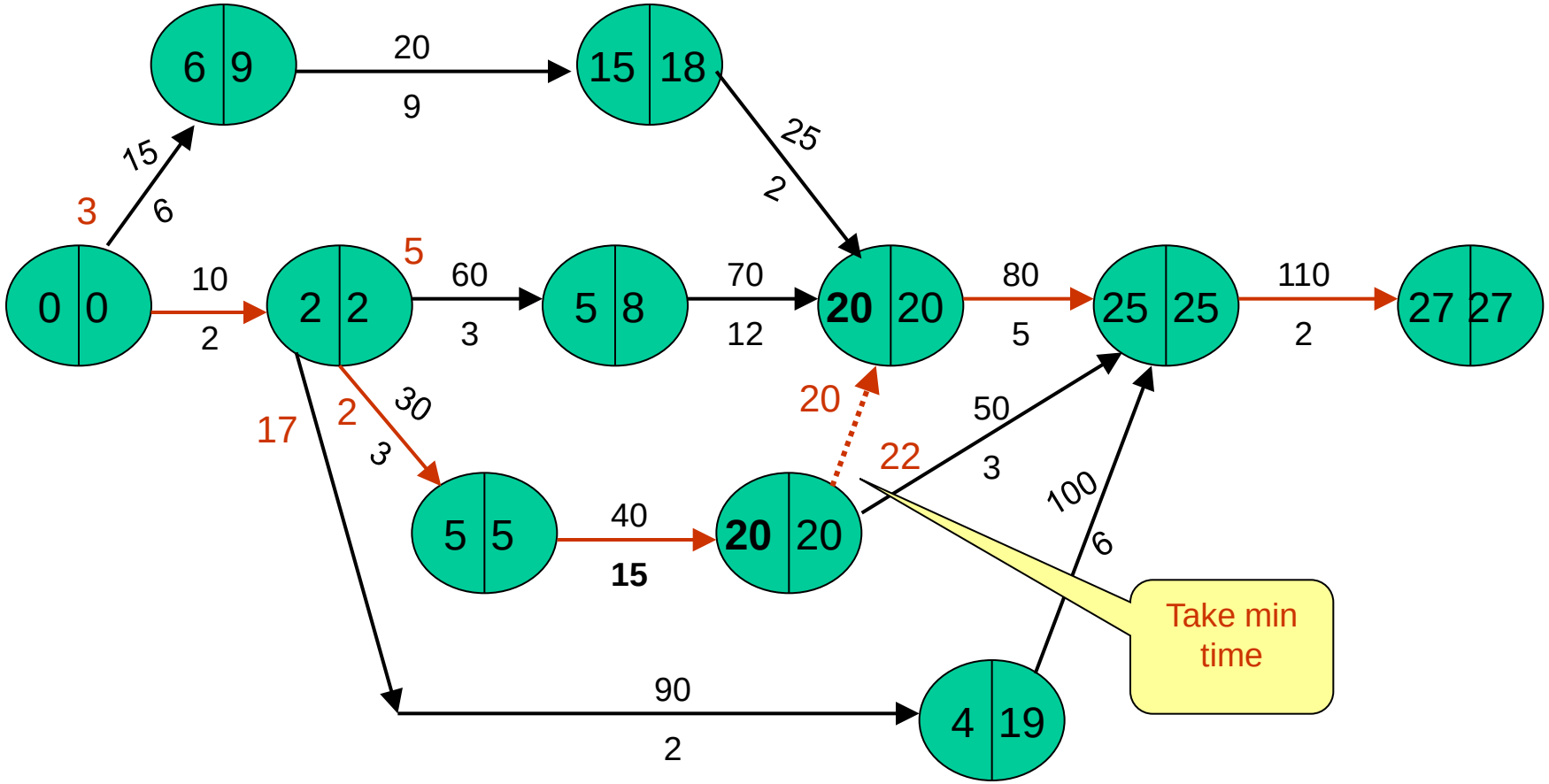
When the activity list was drawn up and the durations calculated, assumptions were made about staff availability. If it now transpires that only two instead of four people will be available to do activity 40 - Create data Files. This means that activity will now take 15 instead of 5 weeks.

ACTIVITY		DURATION (Weeks)	DEPENDS ON
10	Confirm Design	2	
15	Prepare Room	6	
20	Install Network	9	15
25	Test Hardware	2	20
30	Data entry software	3	10
40	**Create Data files**	**15**	**30**
50	Do Stock update	3	40
60	Write Program Specs.	3	10
70	Construct Programs	12	60
80	Test System	5	25,70,40
90	Familiarise users	2	10
100	Train Staff	6	90
110	Handover	2	100,80, 50

Forward pass (left to right) to determine task earliest start and finish times



Backward pass (right to left) to determine revised finish time and critical path

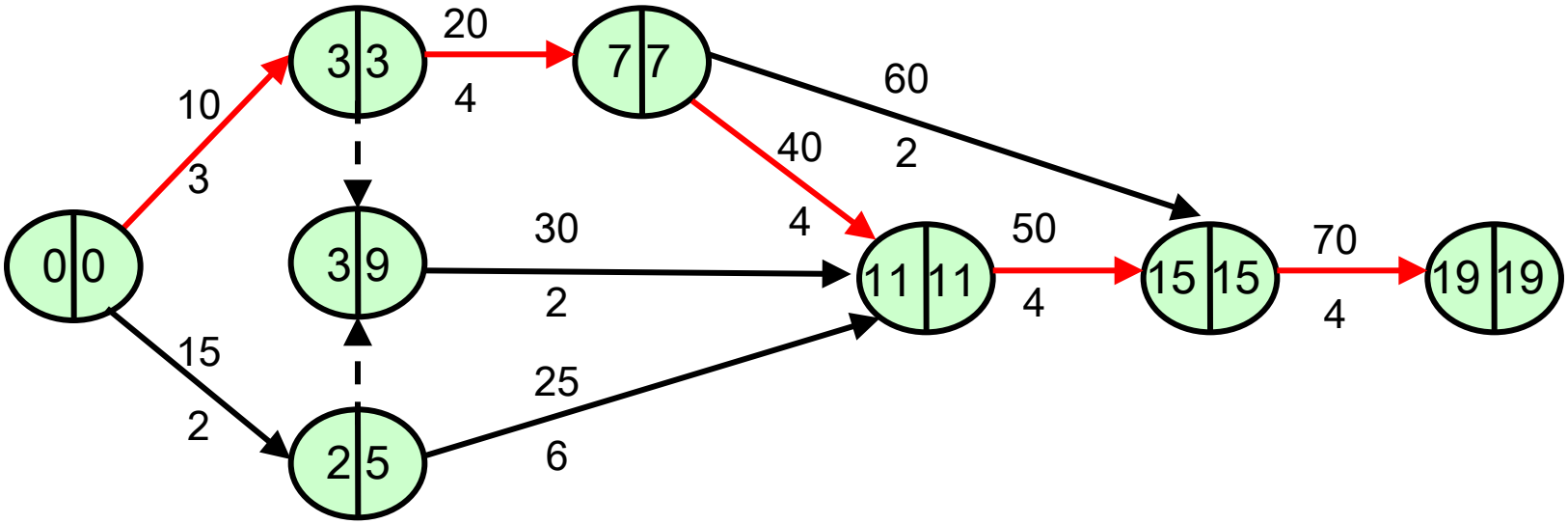


PERT Example 2:
specify tasks and dependencies

C Cox

	ACTIVITY	DURATION (Months)	DEPENDS ON
10	Requirements Analysis	3	
15	Develop test plan	2	
20	Systems Design	4	10
25	Write test drivers	6	15
30	Prepare test data	2	15, 10
40	Code system	4	20
50	System test	4	40, 30, 25
60	Document system	2	20
70	Install	4	50, 60

	ACTIVITY	DURATION (Months)	DEPENDS ON
10	Requirements Analysis	3	
15	Develop test plan	2	
20	Systems Design	4	10
25	Write test drivers	6	15
30	Prepare test data	2	10, 15
40	Code system	4	20
50	System test	4	40, 30, 25
60	Document system	2	20
70	Install	4	50, 60

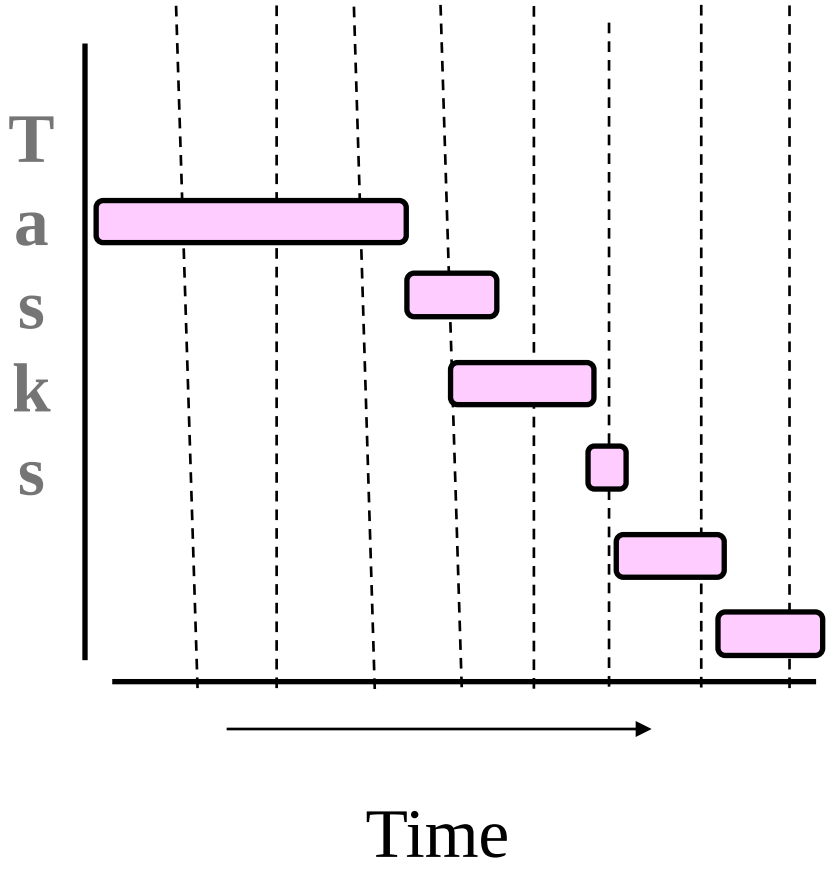


5 Gantt – Task vs. Time Bar Chart
Henry L Gantt, 1917

- Represent project phases or tasks from a work breakdown structure
- Useful for planning, scheduling and monitoring a project
- Important that task sequence dependencies are maintained consistent with the PERT chart
- Project planning tools (e.g. MS Project) automatically maintain consistency between PERT and Gantt chart views

Bar Charts (Gantt Charts)

- Easy to understand
- Drawn to scale
- Can show resources against tasks
- Can show milestones
- Can show current status (today vertical)
- Can show holidays/regular meetings
- Can plot 'actuals' against predictions
- Reported progress can be shown by shaded activity lines

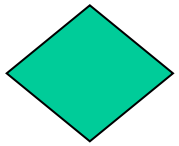


C Cox

Gantt chart - Notation

Activity timeline

Activity with float time (dark)



milestone

PERT -> Gantt

- As we saw, PERT enforces **task dependency** constraints.

- Other kinds of constraints:
  - **Staff assignment and staff loading** and loading constrains number of activities that can occur concurrently.
  - **Resource availability** further constrains time at which various activities can happen.
    Note resources include specific skills, money, hardware or other equipment
- The **Gantt chart** (or schedule) shows a particular scheduling of the tasks in time, subject to all of the constraints.

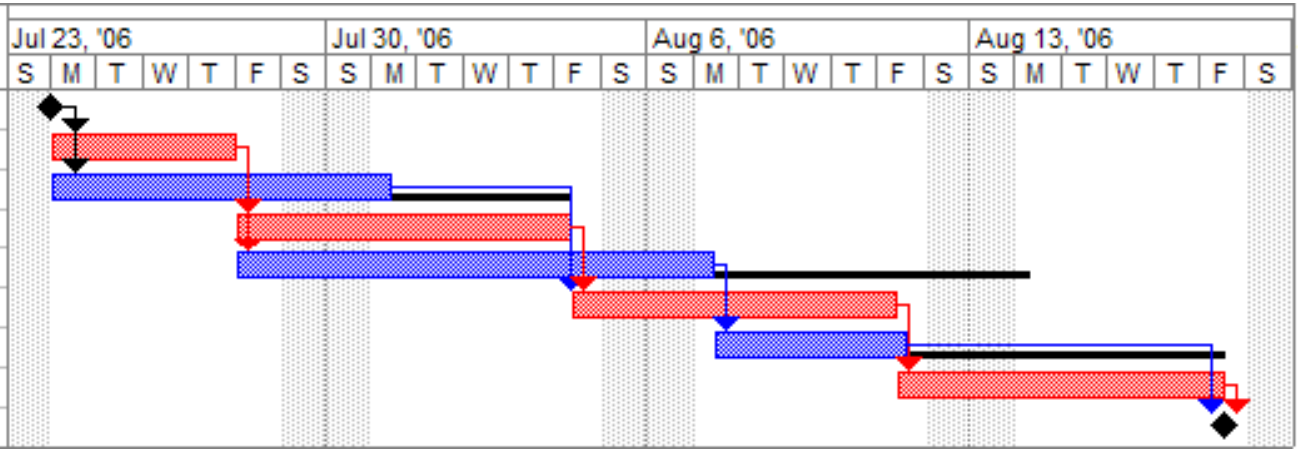
- So Gantt displays task activity against time. Arrows show dependency between tasks.

Task List to Gantt Chart

Task	Predecessor	Duration
a	-	4.0
b	-	5.33
c	a	5.17
d	a	6.33
e	b, c	5.17
f	d	4.5
g	e	5.17

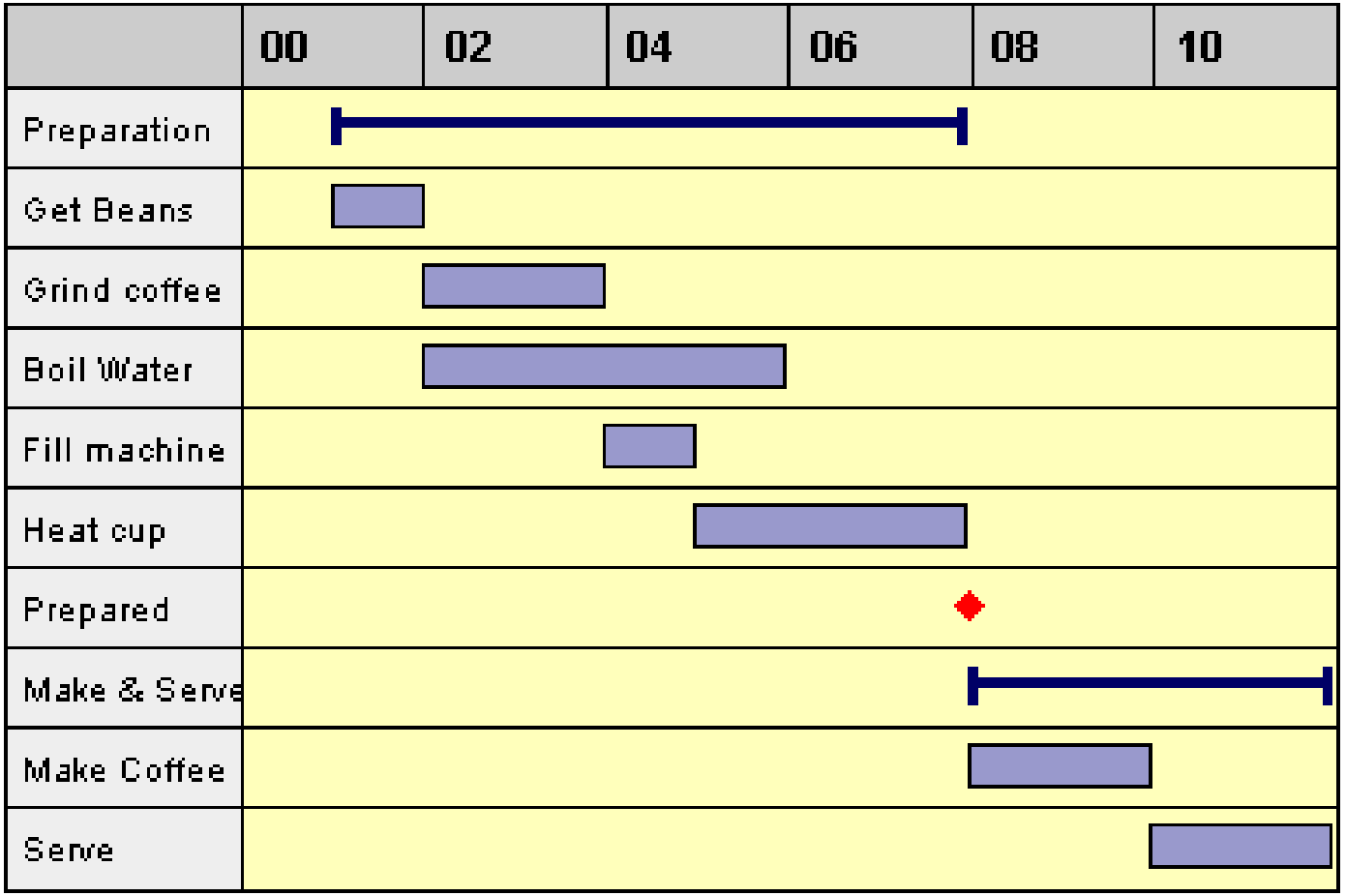
Using Microsoft project

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days

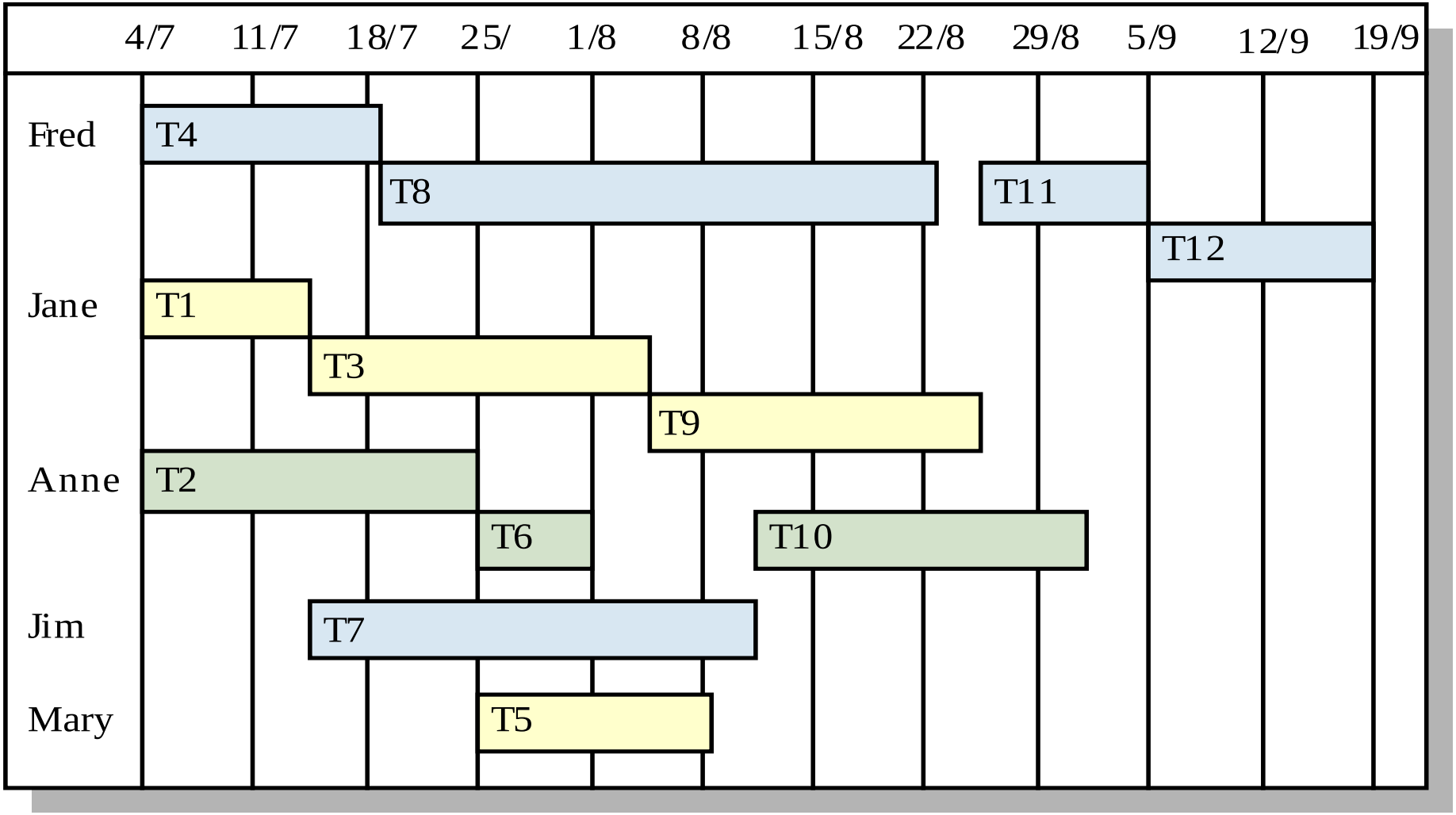


C Cox

GANTT chart with Summary and Milestone



Staff loading

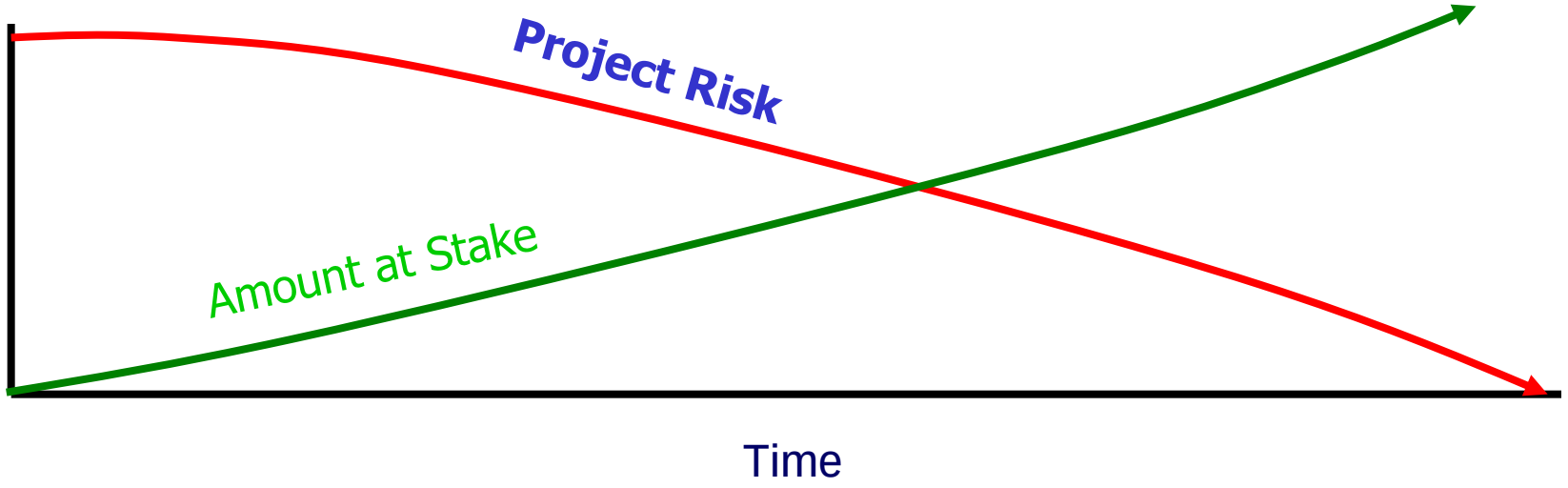


Staff Loading Issues

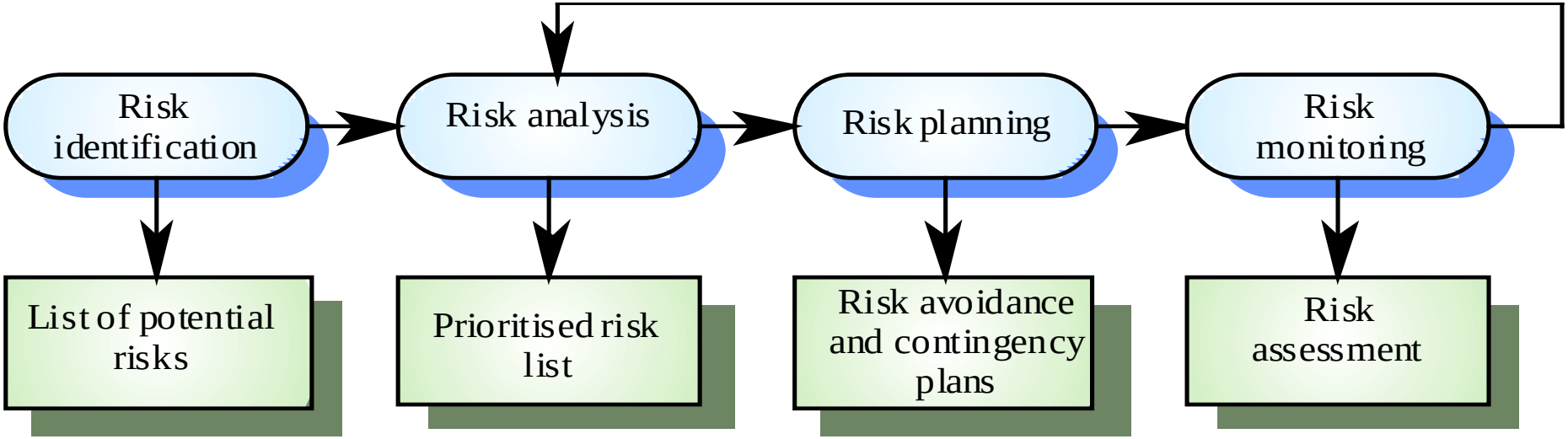
- Some tasks can be assigned only to certain staff members, based on specialized skills.
- A given staff member can only do so much at a time.
- Staff members may differ in their productivity on a given task.
- Staff loading and resource constraints are two aspects of scheduling not represented directly on PERT charts
- Generally they have the effect of providing added sequencing, and therefore lengthening overall project time

6 Risk Analysis

- Concerned with
  - identifying project risk factors
  - defining strategies to minimise effects on a project
- Risk – A measure of the probability and impact of not achieving a project objective: risk exposure = likelihood x impact



The risk management process

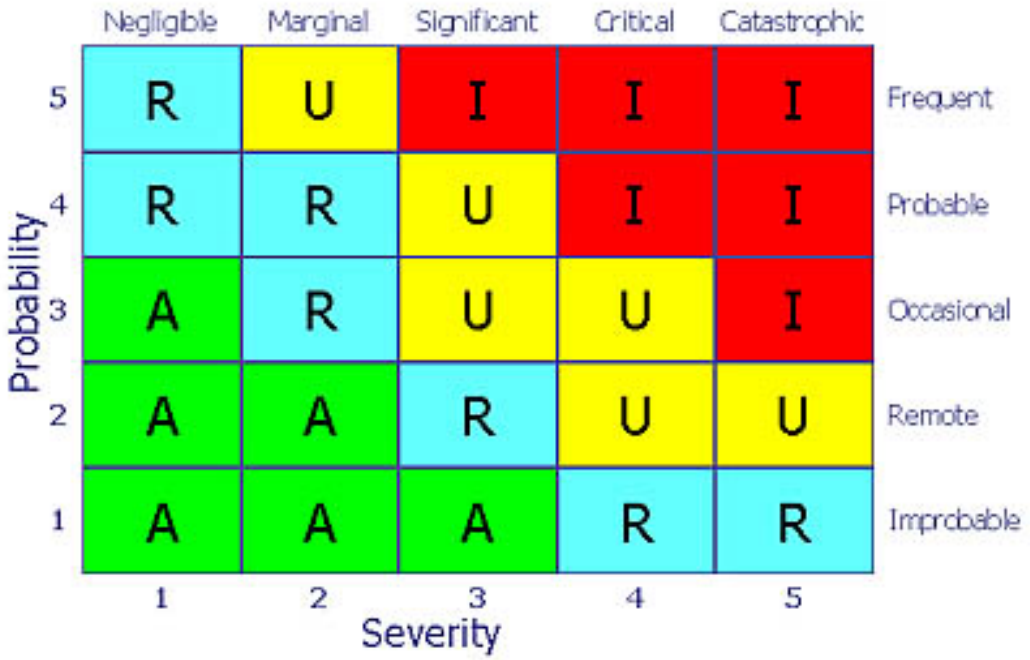


Risk Factors

- Staff change
- Management change
- Hardware unavailability
- Requirements change
- Specification delay
- Size underestimate
- Technology change
- Product competition

Risk Matrix

Used in risk assessment to quantify risk
Matrix of likelihood (probability) vs. consequence (severity)



A – acceptable

R – tolerable

U – undesirable

I - intolerable

More risk in week 6

7 Monitoring Progress

- A monitoring model:

Measure	- progress made
Compare	- with work planned
Evaluate	- what action needed to realign to plan
Predict	- consequences of no action or corrective action
Act	- deal with problems as they occur

- Real progress only if task completion is observable
  - Bad
    - Task 1: 10% of feature, task 2: 20% of feature
    - What does 10% mean?!
  - Good
    - Task 1: All menus implemented and system respond correctly to mouse clicks.

Monitoring Information required regularly from team members

-Tasks involved in

- Effort spent on each task

- Estimate of effort needed to complete each task

- Any problems encountered

<u>Progress Reports</u>

Aim: Inform management of progress achieved in relation to schedule

Content:

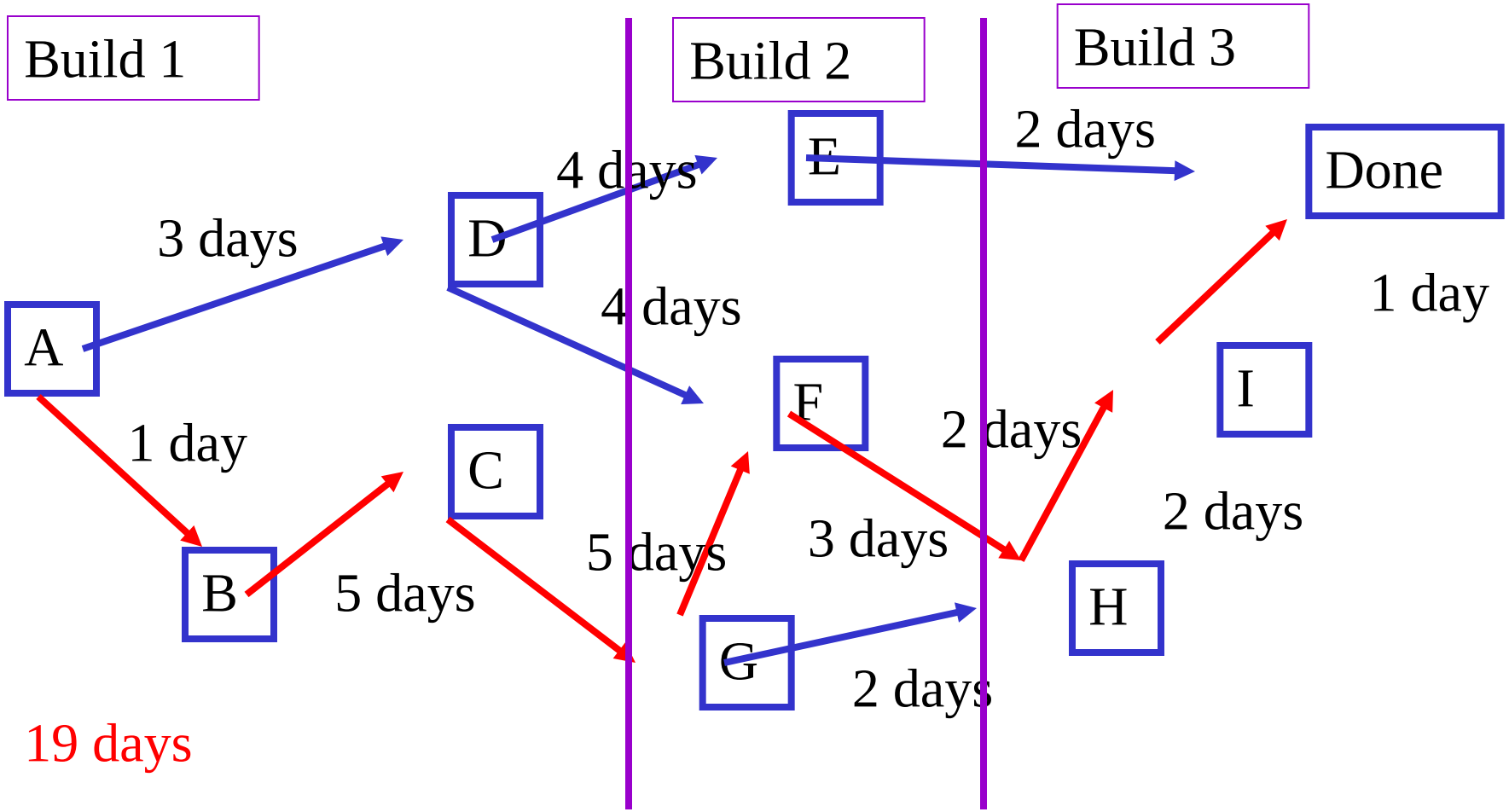
current status – in relation to schedule
progress – milestones and goals accomplished
problems and issues – how were/will problems be solved
recommendations and requests
plans for the next project phase

8 Builds

Make the (software development) plan a sequence of (software) builds

- Get the first build up as soon as possible
- After that, always maintain a working system
- System grows as tasks are checked off
- Move from build to build: key principle – move from working system to working system
- Always available to go back a build version
- Build history is an important SQA deliverable

Builds as staged development shown on Gantt chart



Why Builds?

Can observe true progress

- If nothing runs, hard to know if we are close to running

Makes changes in plan easier

- Each build provides a natural point for changes

Allows priorities

- Put most critical features in first build
- If schedule slips, just don't get to lower-priority builds later in the schedule

9 Teamwork

Key to effective teams:

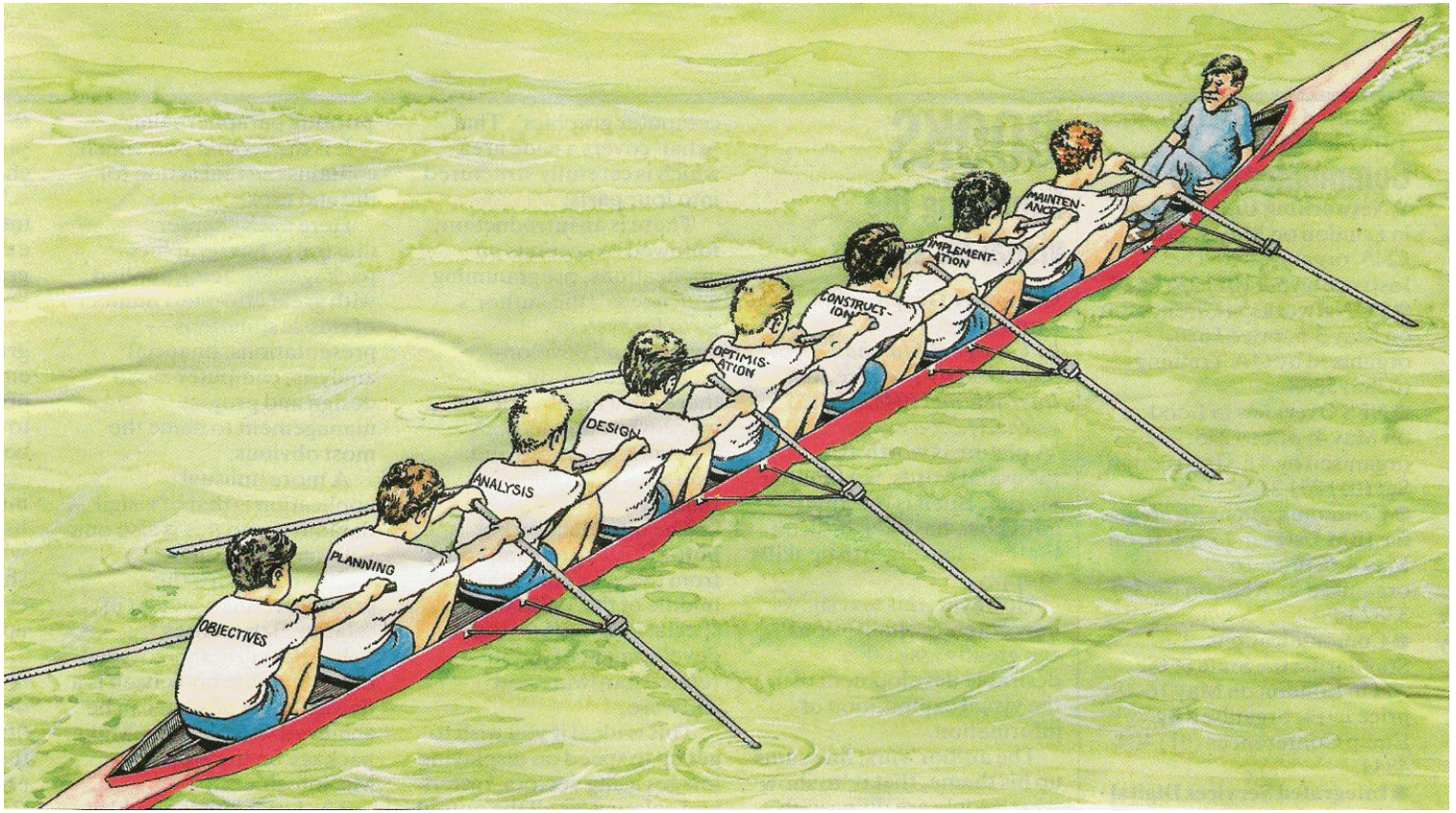
- Clearly defined purpose
- Shared Ownership
- Shared Responsibility
- Everyone informed
- Don’t rely on one individual
- No free-riders

Teamwork – Belbin Roles: (http://www.belbin.com/)

Overall	Belbin roles	Description
Doing / acting	Implementer	Well-organized and predictable. Takes basic ideas and makes them work in practice. Can be slow.
	Shaper	Lots of energy and action, challenging others to move forwards. Can be insensitive.
	Completer/ Finisher	Reliably sees things through to the end, ironing out the wrinkles and ensuring everything works well. Can worry too much and not trust others.
Thinking / problem-solving	Plant	Solves difficult problems with original and creative ideas. Can be poor communicator and may ignore the details.
	Monitor/ Evaluator	Sees the big picture. Thinks carefully and accurately about things. May lack energy or ability to inspire others.
	Specialist	Has expert knowledge/skills in key areas and will solve many problems here. Can be disinterested in all other areas.
People / feelings	Coordinator	Respected leader who helps everyone focus on their task. Can be seen as excessively controlling.
	Team worker	Cares for individuals and the team. Good listener and works to resolve social problems. Can have problems making difficult decisions.
	Resource/ investigator	Explores new ideas and possibilities with energy and with others. Good networker. Can be too optimistic and lose energy after the initial flush.

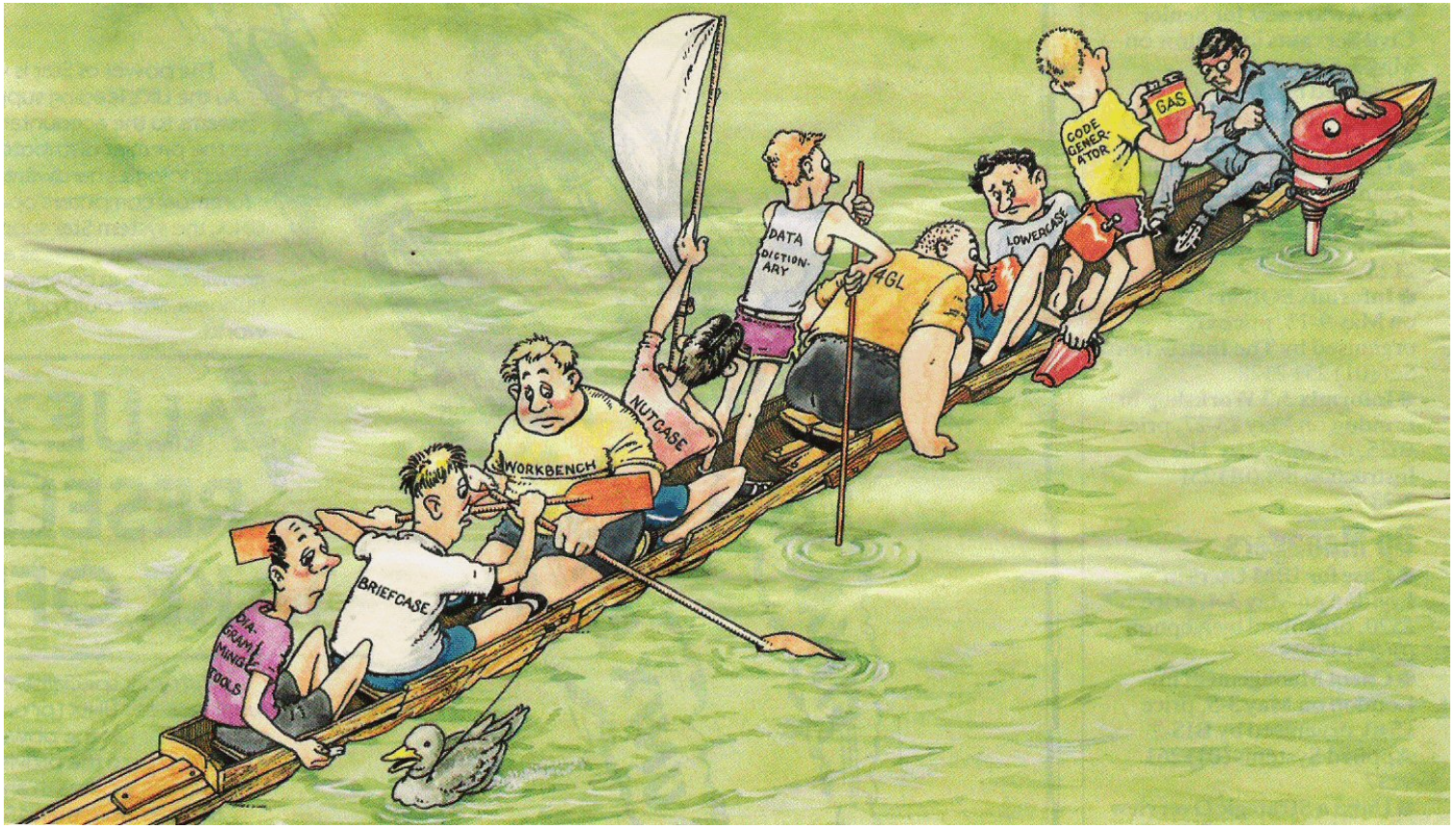
C Cox

. . . aim for synchronised effort …

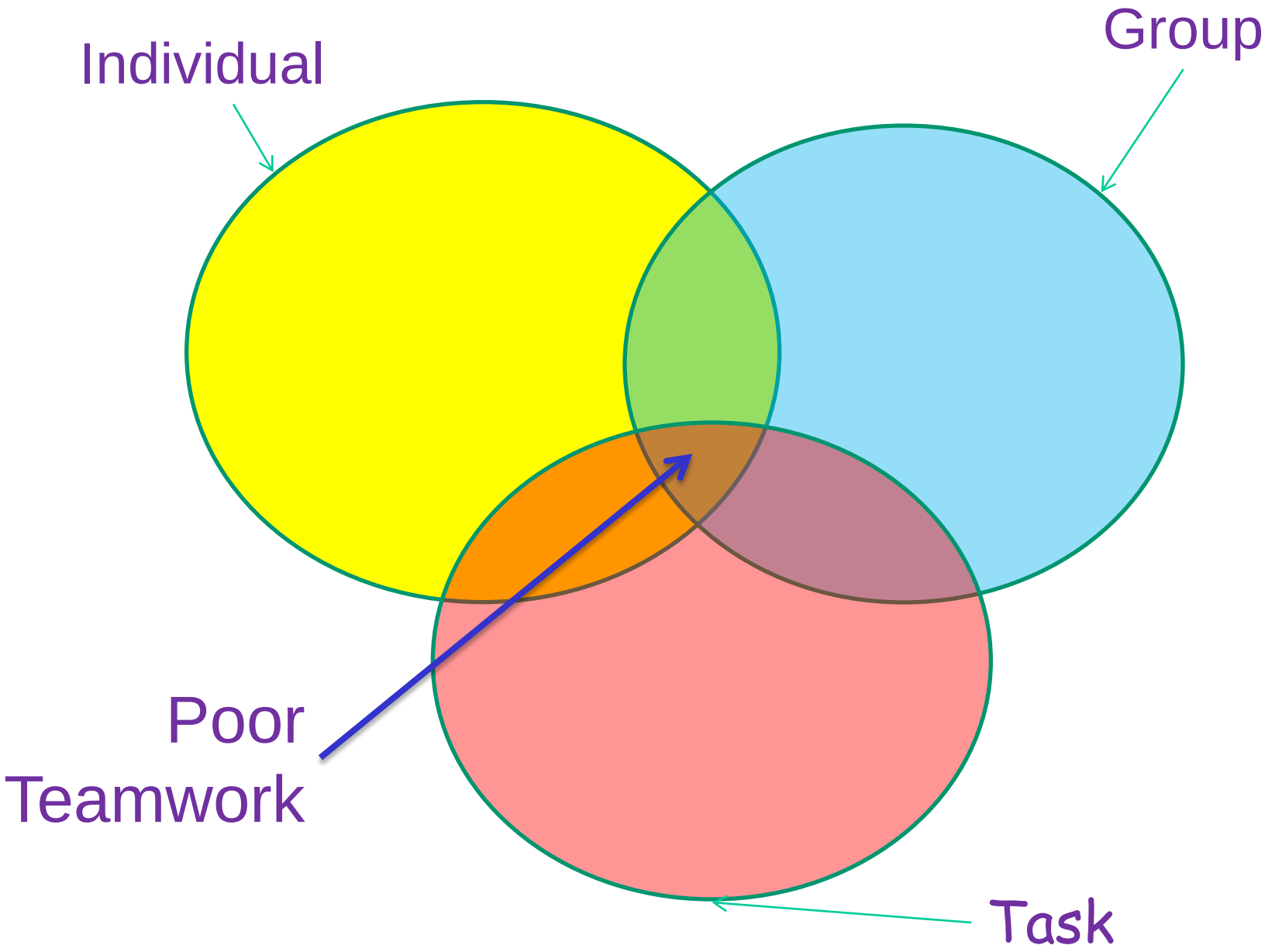


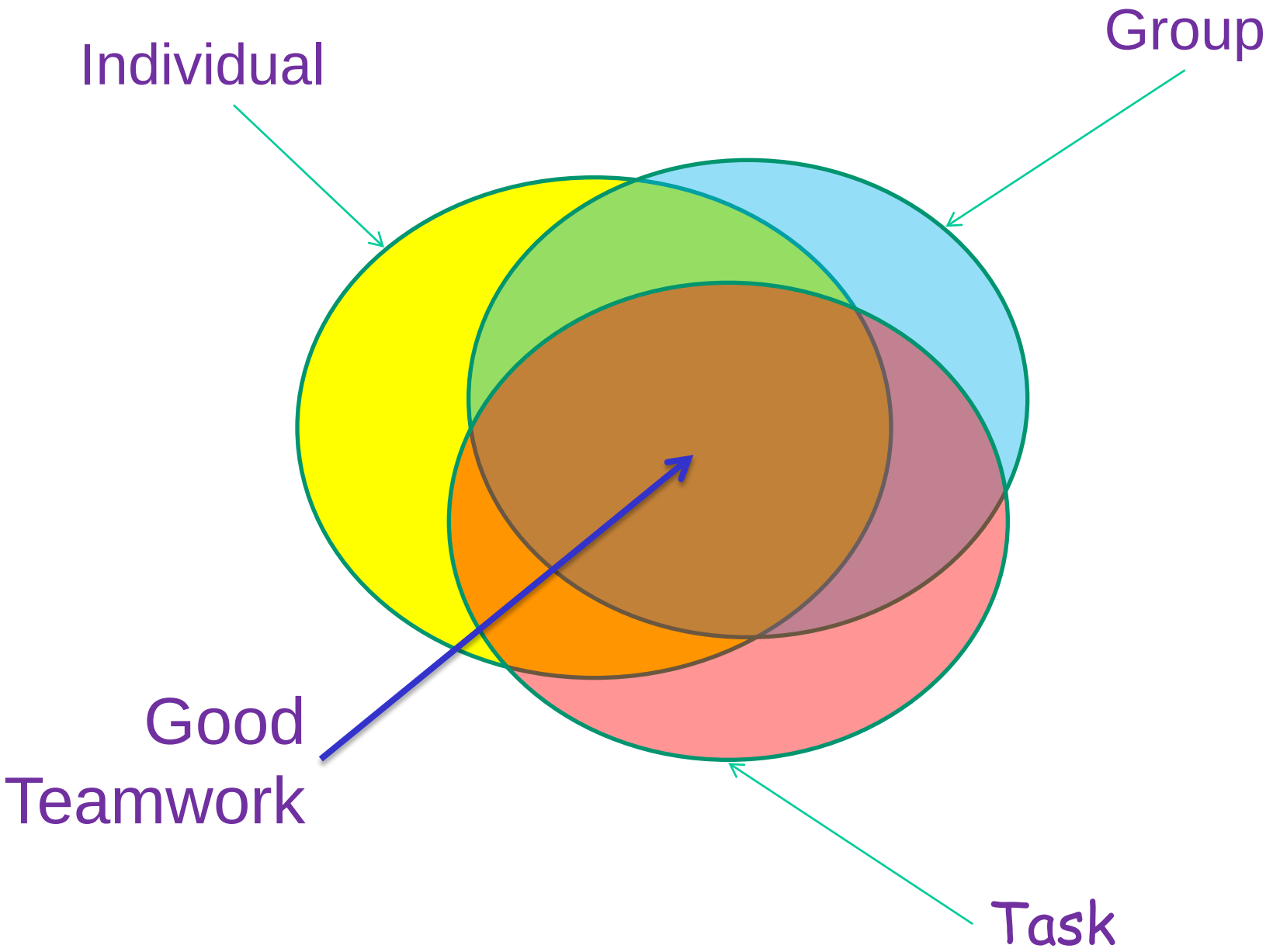
C Cox

. . . not independent attempts . . .



C Cox


Individual
Group
Poor
Teamwork
Task


Individual
Group
Good
Teamwork
Task

Effective team meetings

- Clarify objectives
- Set meeting agenda
- Prepare materials
- Record attendance
- Take minutes of decision, issues and actions
- Keep meetings short
- Circulate minutes

Summary

- Understand software project management issues
- Understand project planning issues
- Construct and use Gantt and PERT charts
- Understand that risks should be considered
- Understand teamwork

C Cox