Software Engineering COMP1035

Lecture 18

Project Planning

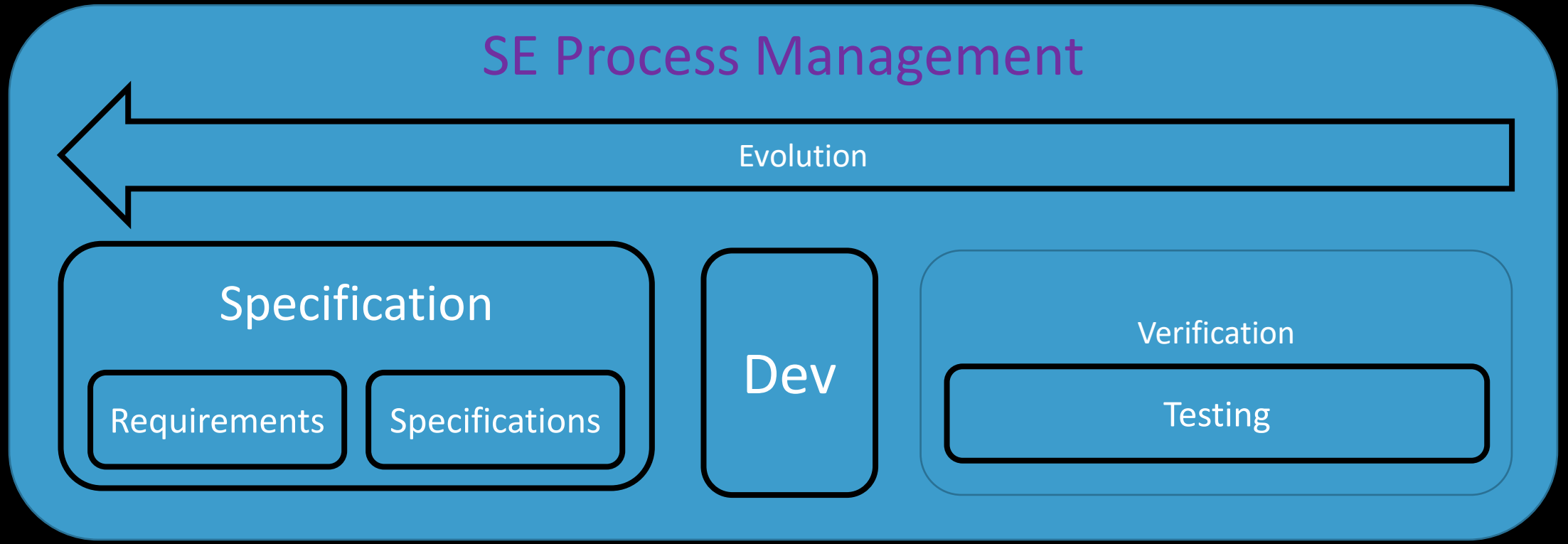
Outline

- Aspects involved in a Project Plan
- Plans and diagrams you can use to organise work
- Estimating time and budget from those

Where We Are In the Process

University of Nottingham
UK | CHINA | MALAYSIA

Keeping Track of SE Module



Process/Project Management

- Deliver good quality product
- Use good reliable methods
- Reducing as much risk as possible
- **Plan / budget / manage a project based on these**

Project Planning Documents

1. Introduction
2. Project Organisation
3. Risk Analysis
4. Hardware & Software
5. Work Breakdown
6. Project schedule
7. Monitoring and Revision Plans

Managing a Project (workflow)

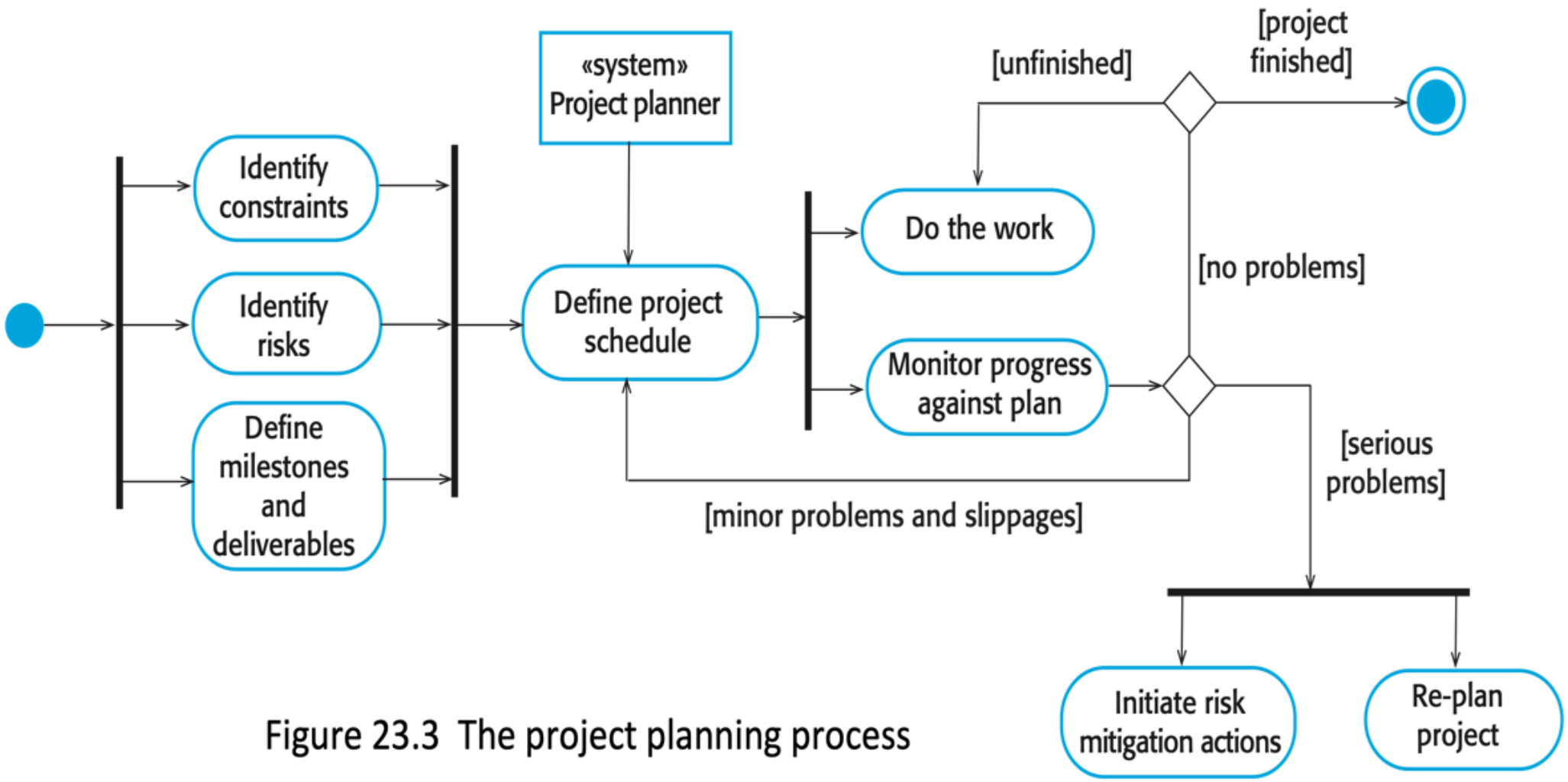
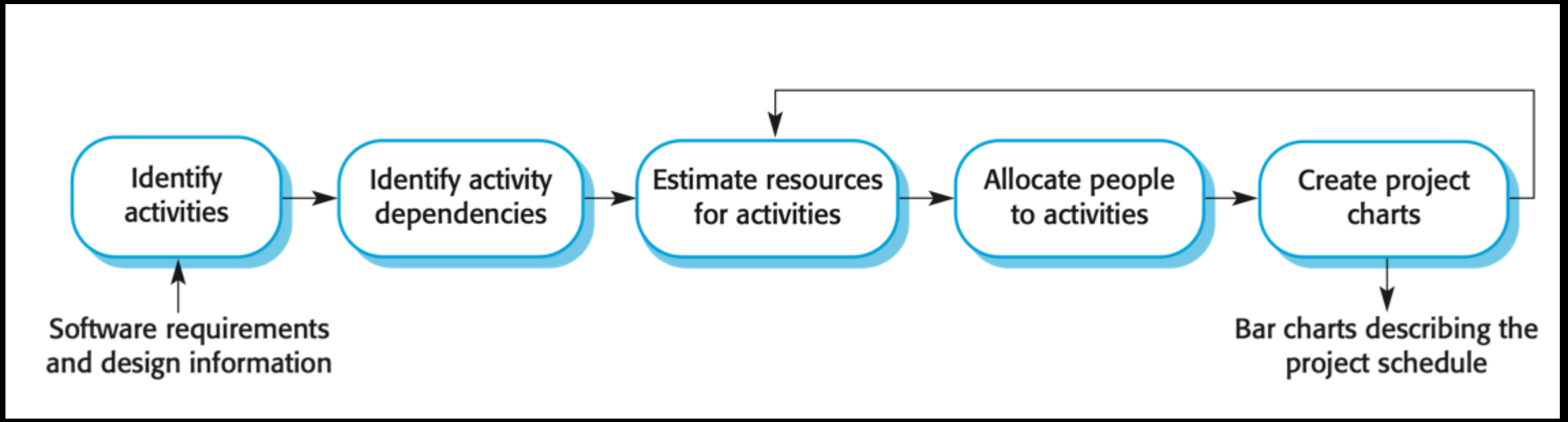


Figure 23.3 The project planning process

Project Scheduling

- Activities should produce some **measurable outcomes**
  - so progress can be assessed
- A **milestone** is the end point of an activity
- Deliverables are results delivered to customers



Identifying the tasks

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

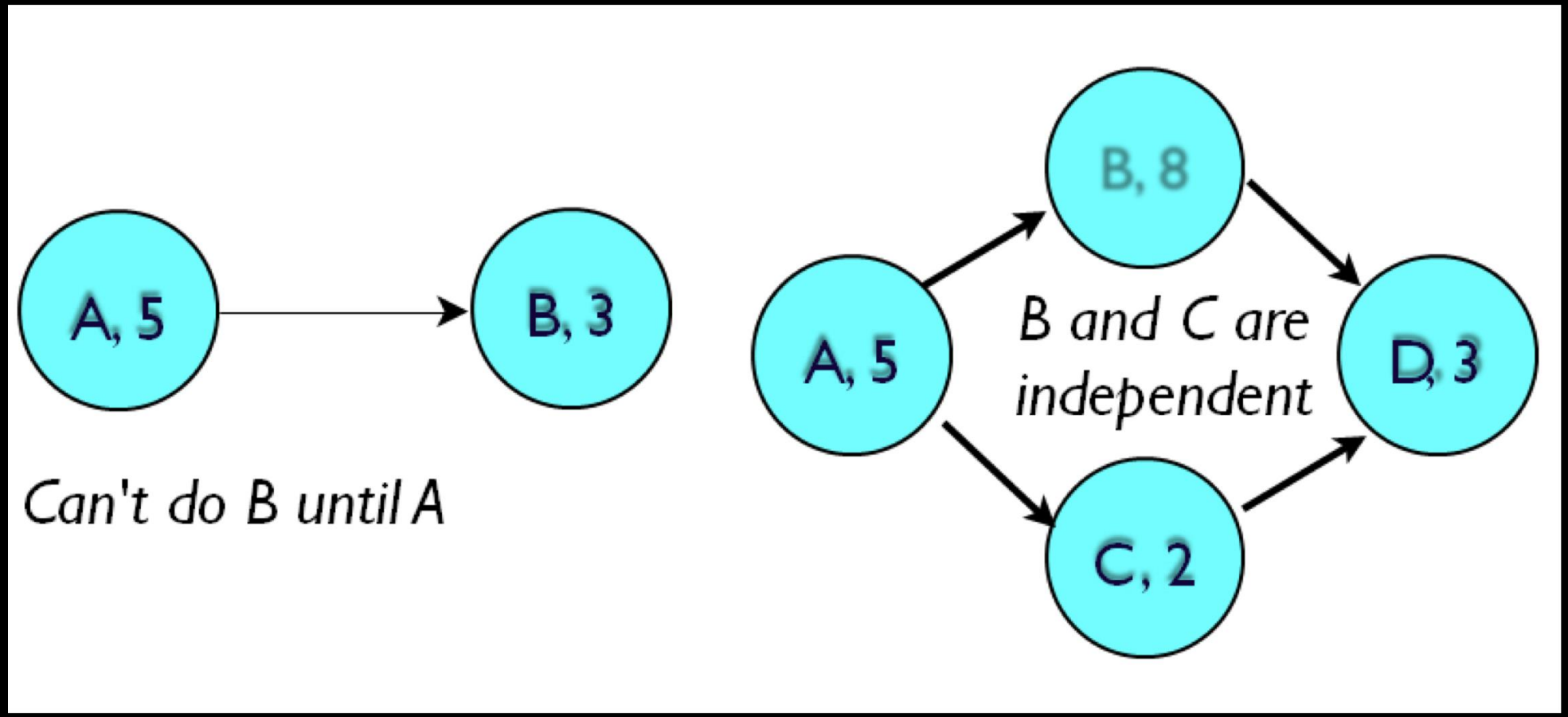
Milestones and Outputs

- Every activity/task should produce a **tangible** output
  - a req doc, spec doc, design, code frame, database, etc
- Tasks should be a few weeks in size
  - at least 1 week, at most 8-10 weeks
  - if longer it should be broken down into smaller tasks
  - note that weeks is different to person-days of effort
- The project needs certain milestones
  - which produce project deliverables
  - these might be times when a few activities/tasks all finish

Planning Diagrams

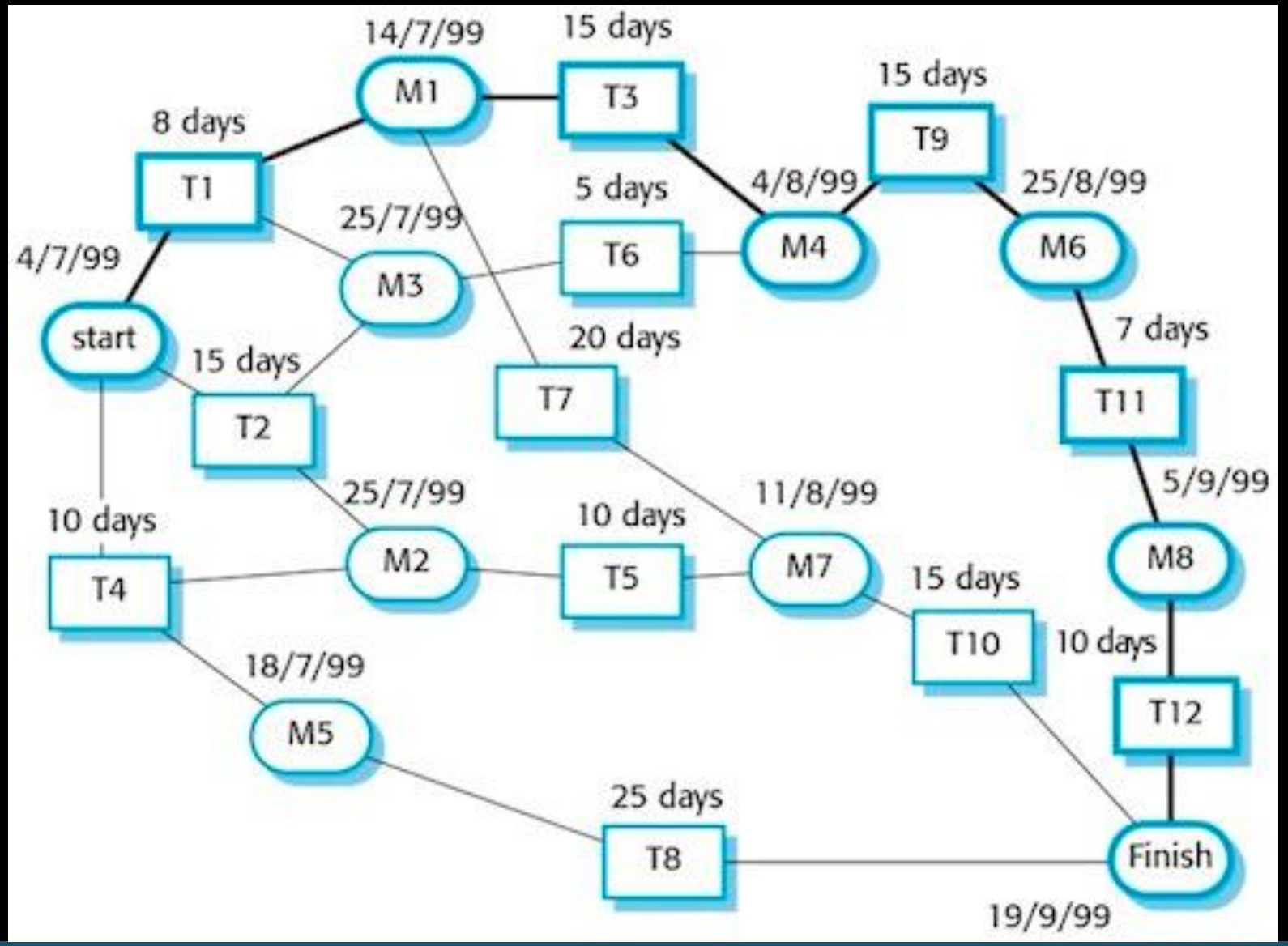
- PERT (1958) - tasks and dependencies
- Critical Path Method (1960) - risk analysis
- Gantt Chart (1910) - adding time to the tasks/dependencies
- Staff Allocation charts - who can do the tasks/dependencies

PERT Charts



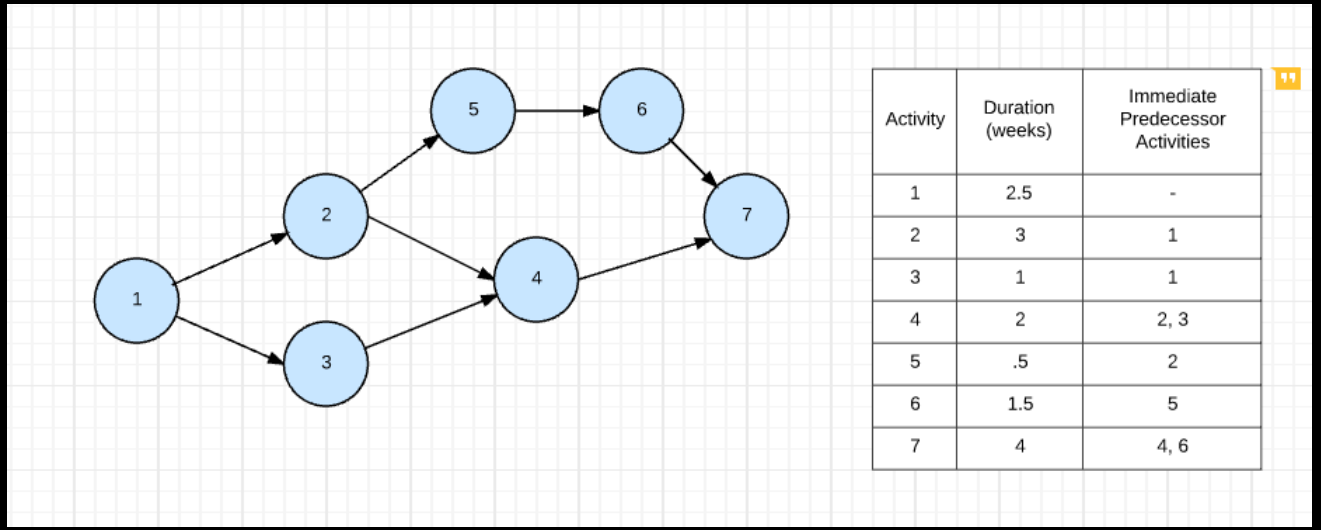
- Particularly for dependencies and parallel tasks etc

PERT Charts


14/7/99
M1
15 days
T3
15 days
T9
8 days
T1
5 days
4/8/99
M4
25/8/99
M6
25/7/99
M3
T6
4/7/99
start
15 days
T2
20 days
T7
7 days
T11
5/9/99
25/7/99
11/8/99
10 days
10 days
M2
T5
M7
M8
T4
15 days
T10
10 days
18/7/99
M5
T12
25 days
T8
Finish
19/9/99

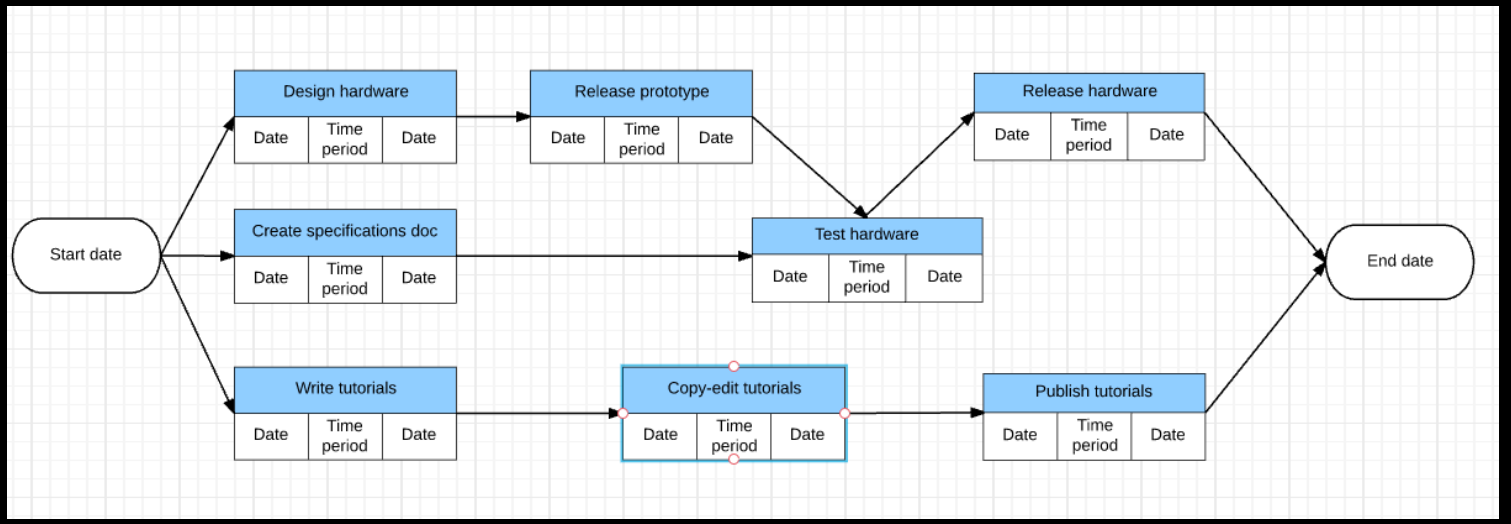
Can have different details

- Simple



Activity	Duration (weeks)	Immediate Predecessor Activities
1	2.5	-
2	3	1
3	1	1
4	2	2, 3
5	.5	2
6	1.5	5
7	4	4, 6

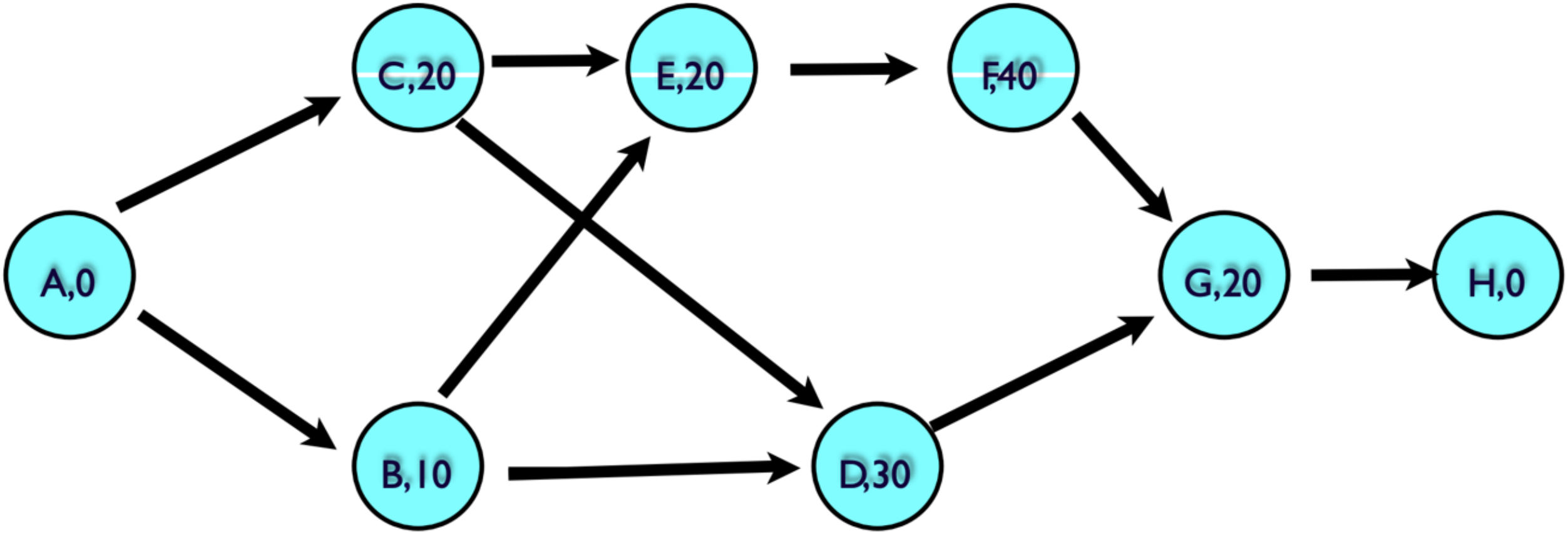
- Detailed
  - optimistic time
  - pessimistic time
  - most likely?



PERT - Critical Path

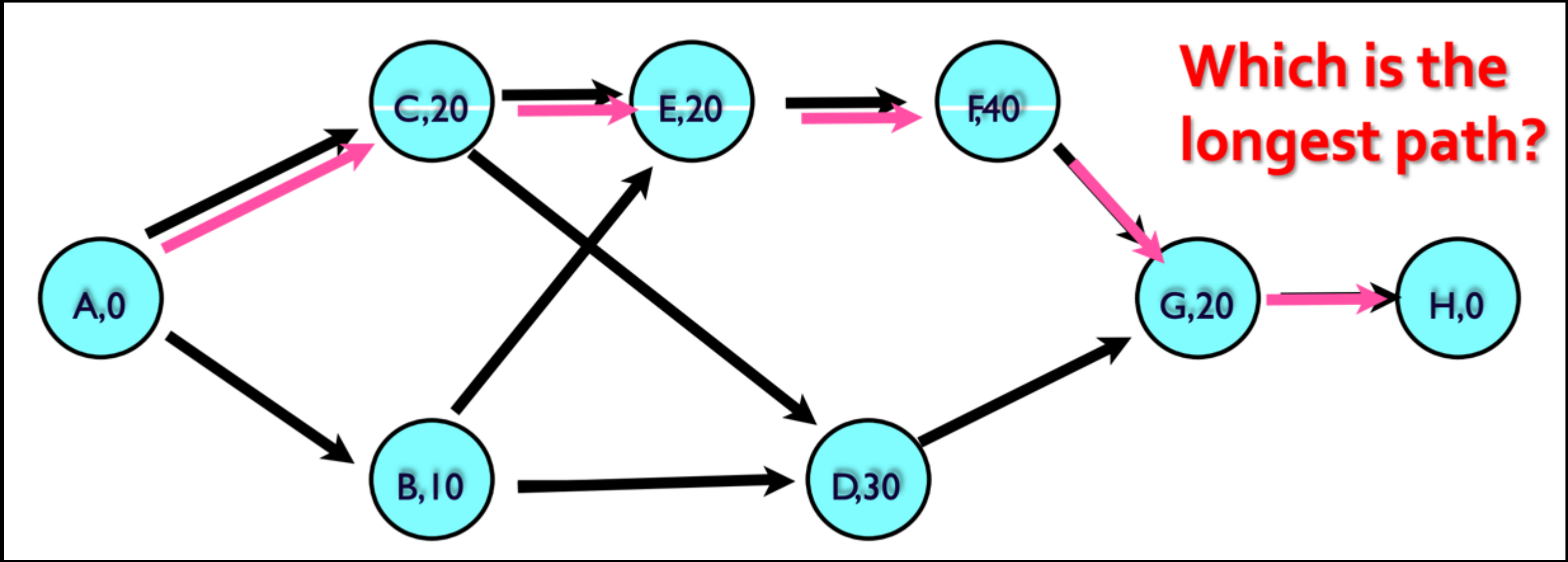
- Take a PERT chart
- Identify all the paths through the PERT chart
- Identify the length of time taken for each path
- The **longest path** (worst case) is the critical path
  - which helps you estimate the effort for the worst case
  - rather than the best case

PERT - Critical Path (example)



- How many unique paths are here?
- Which is the longest path?

PERT - Critical Path (example)

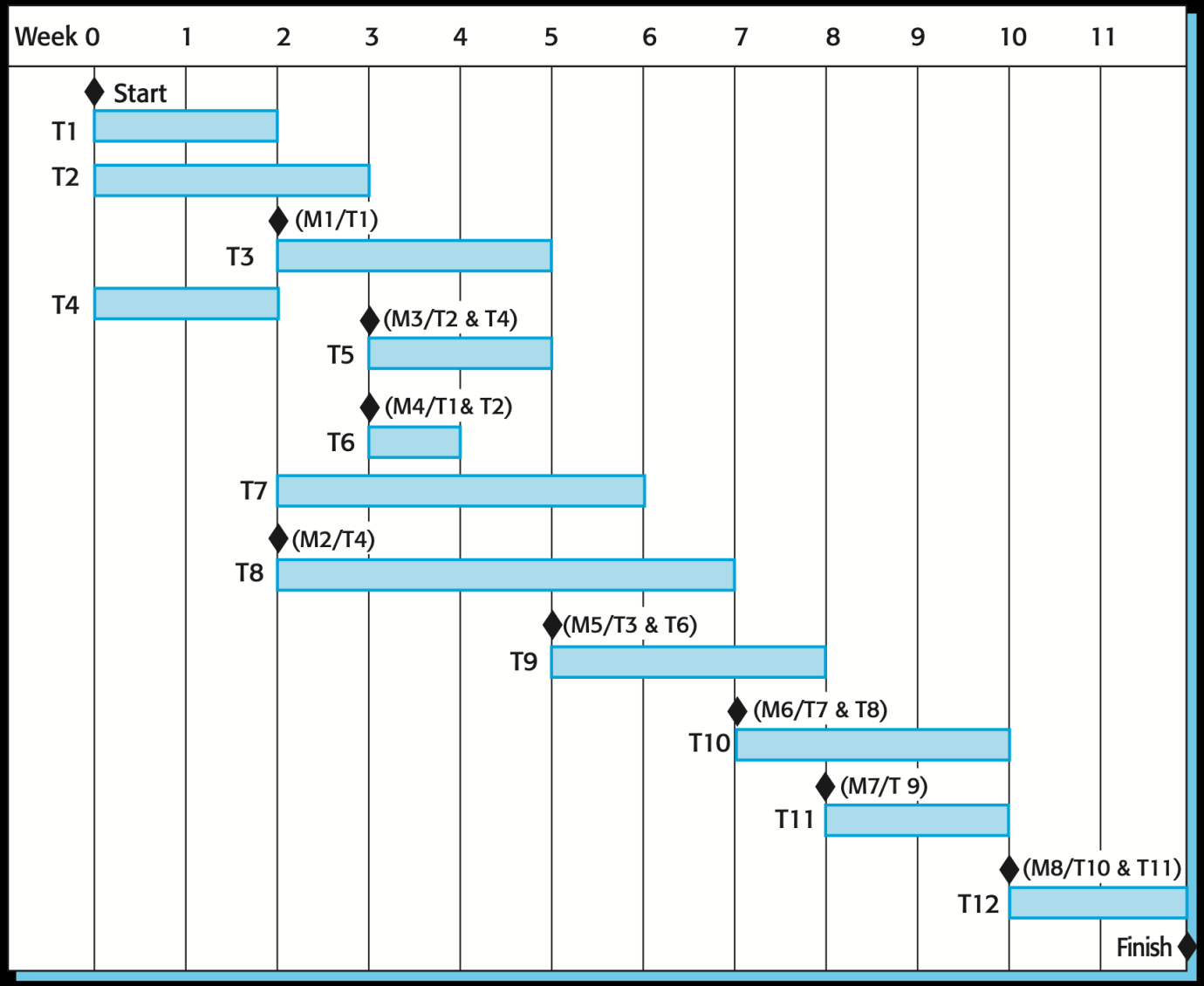


- **ACEFGH**, ACDGH, ABDGH, **ABEFGH**

PERT - Critical Path

- The Critical Path is the **Bottleneck Route**
- Modification of a task along the critical path will affect the project duration
- Can become very complicated in large projects
  - so we often use software to derive the CP
  - e.g. Open Project, Project Libre, MS Project, …

Gantt Charts



Gantt Chart - Good example

Task overviews

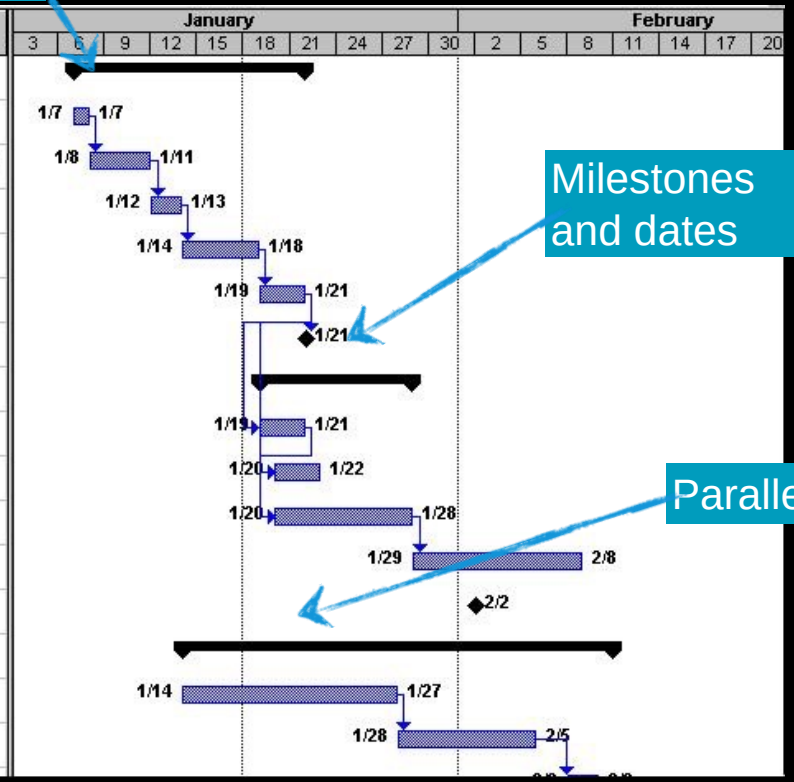
Categorised Work

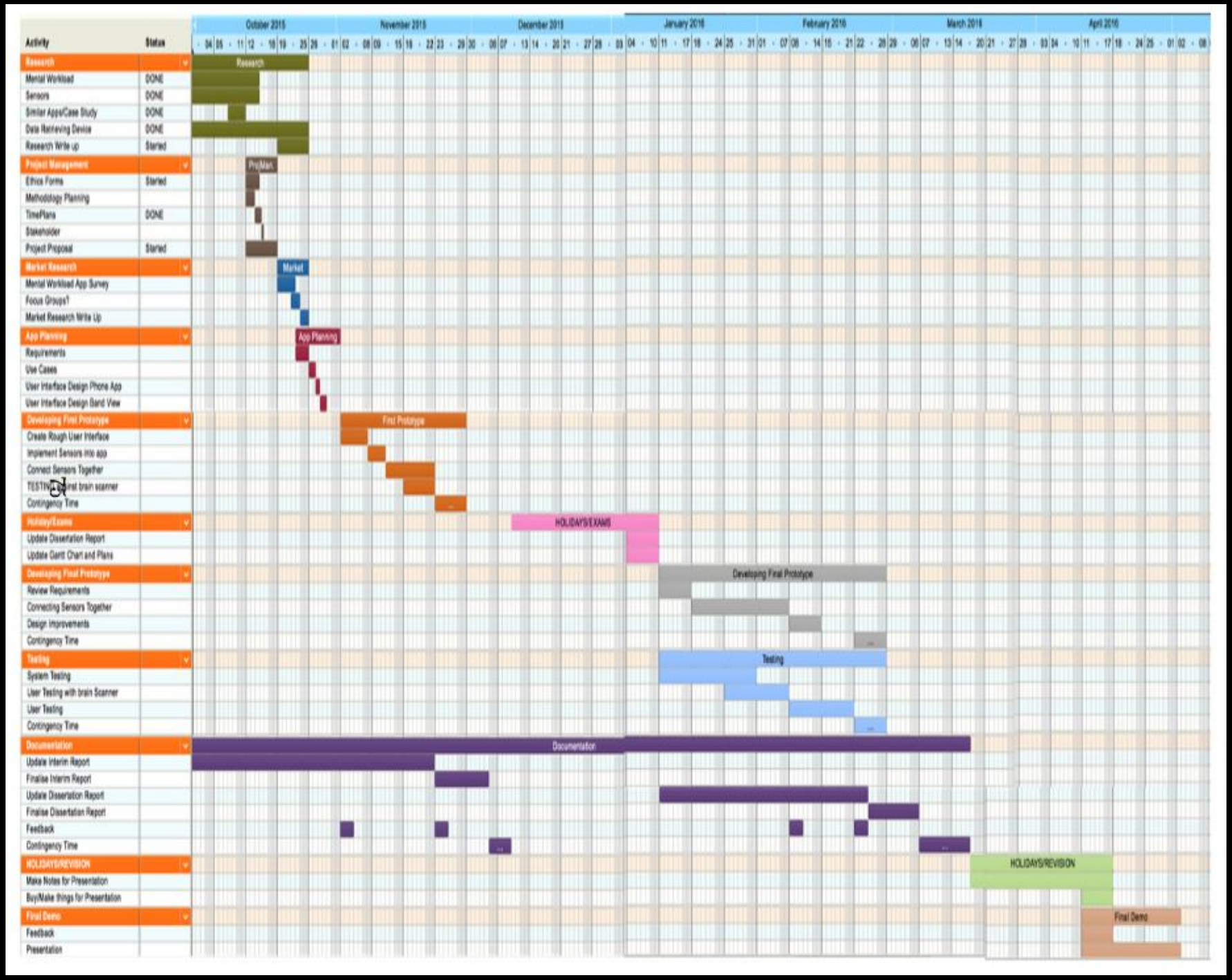
Task IDs

Milestones and dates

Parallel work

WBS	Task Name
1	**Define specifications**
1.1	Identify customers
1.2	Interview 10 customers
1.3	Interpret requirements
1.4	Benchmark products
1.5	Define target PDS
1.6	Target PDS Released
2	**Generate concepts**
2.1	Review comp products
2.2	Search patents
2.3	Brainstorm concepts
3	Select top 2 concepts
4	MQ Presented
5	**Profile motor power**
5.1	Design test stand
5.2	Build test stand




Activity
Status
October 2015
November 2015
December 2015
January 2016
February 2016
March 2016
April 2016
Research
Mental Workload
DONE
Sensors
DONE
Similar Apps/Case Study
DONE
Data Retrieving Device
DONE
Research Write up
Started
Project Management
ProjMan
Ethics Forms
Started
Methodology Planning
TimePlans
DONE
Stakeholder
Project Proposal
Started
Market Research
Market
Mental Workload App Survey
Focus Groups?
Market Research Write Up
App Planning
App Planning
Requirements
Use Cases
User Interface Design Phone App
User Interface Design Band View
Developing First Prototype
First Prototype
Create Rough User Interface
Implement Sensors into app
Connect Sensors Together
Contingency Time
Holiday/Exams
HOLIDAYS/EXAMS
Update Dissertation Report
Update Gantt Chart and Plans
Developing Final Prototype
Developing Final Prototype
Review Requirements
Connecting Sensors Together
Design Improvements
Contingency Time
Testing
Testing
System Testing
User Testing with brain Scanner
User Testing
Contingency Time
Documentation
Documentation
Update Interim Report
Finalise Interim Report
Update Dissertation Report
Finalise Dissertation Report
Feedback
Contingency Time
HOLIDAYS/REVISION
HOLIDAYS/REVISION
Make Notes for Presentation
Buy/Make things for Presentation
Final Demo
Final Demo
Feedback
Presentation

Staff-Allocation Charts

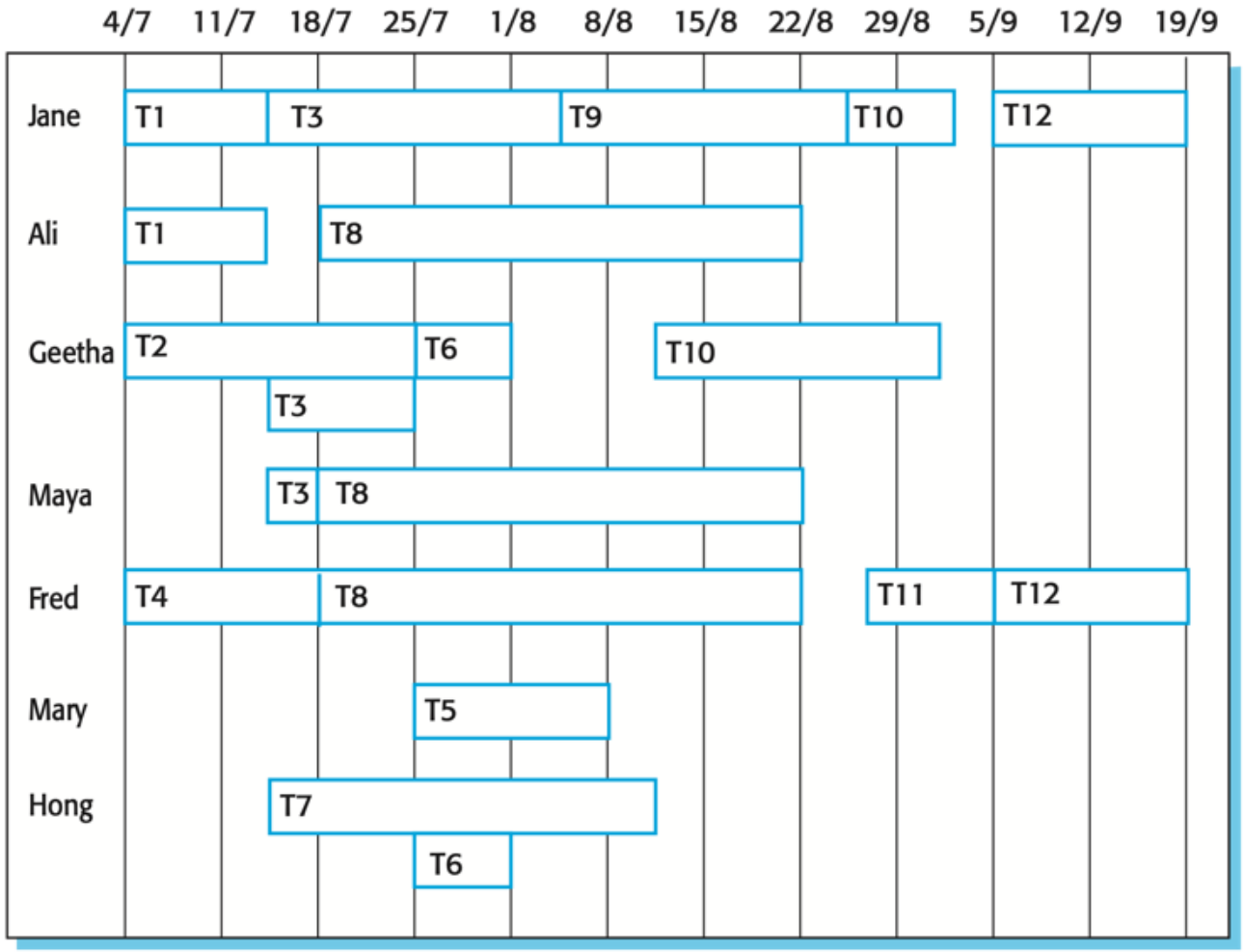
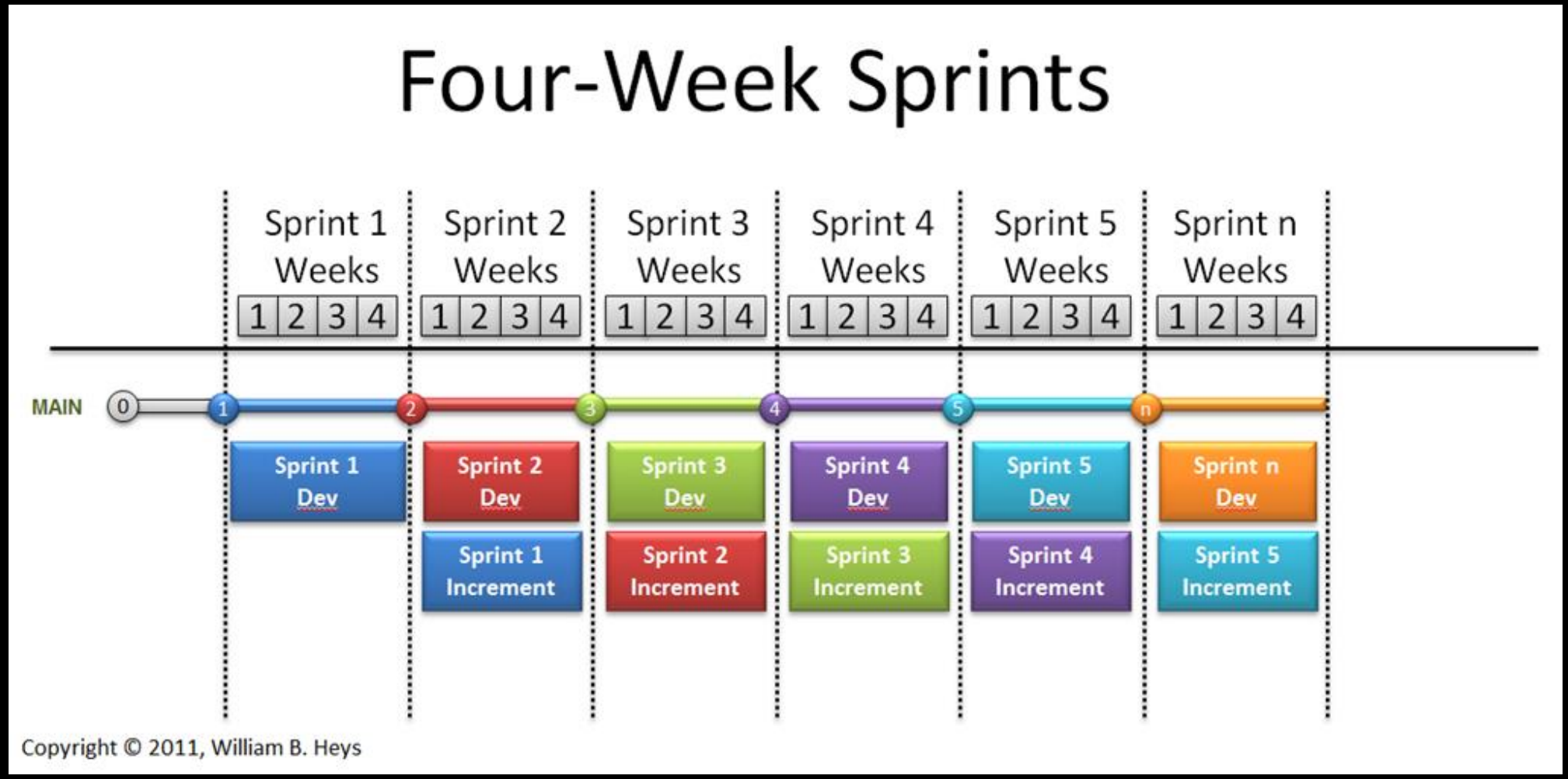


Figure 23.7 Staff allocation chart

Reviewing Progress

- Milestones are times to review progress and the plan
- A plan was an **estimate**, and so you need to check for slippage
  - hopefully, a conservative plan includes time for slippage
  - If slippage is small, it might be reduced later in the project
- If serious - you check into your risk strategies
  - and re-plan the project
- If the new plan is more expensive
  - you need to change the tasks, or negotiate for more money
  - or decide if the project should continue! or be cancelled.

Agile Planning



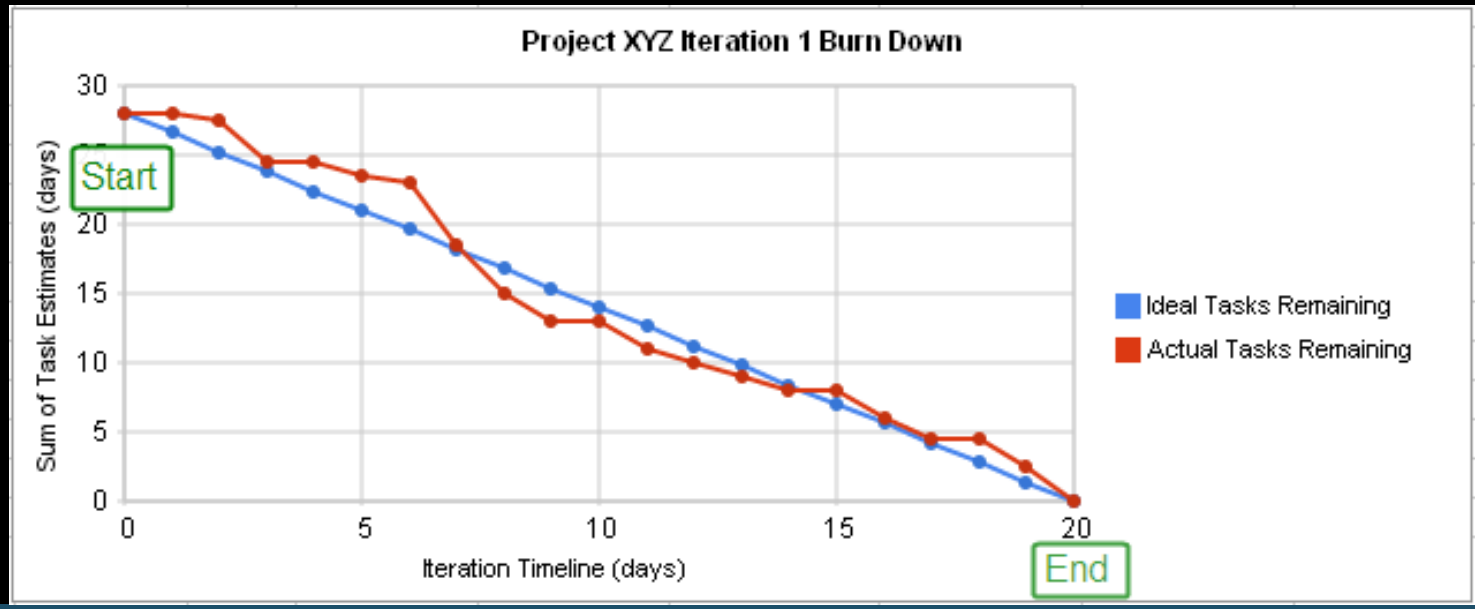
Agile Planning

- Suggested Activity 1:
  1) make a sprint-based version of the tasks in task table
  2) how many people needed in your team?

- Suggested Activity 2:
  1) create a list of requirements (features, or user stories etc)
  2) break the development into a series of sprints
  3) create an alternative ‘plan-based’ PERT chart for same project

Agile - Reviewing Progress

- Project Reviewing is built right into e.g. Scrum
  - » sprint reviews
  - » next sprint planning
- The aim of the start of every sprint: plan ahead
  - - look at the overall backlog
  - - set targets for a sprint
  - - reflect also on overall progress
- Burndown Charts help do this



Estimating Cost & Budget: Two approaches

- By work that needs doing (then estimating developer time)

- By developer time (and defining how much work to do)
  - » one reason scrum is popular
    - you agree an number of sprints for a team
    - cost revolves around this
    - expanding a project is ‘buy more sprints’

Based on expected plan / charts

- You've analysed the main risks and activities
- First - you estimate the min/max time the project will take
- One option add a contingency estimate (e.g. 30%-50% extra)
  - to cover what you didn't include in the plan
  - or risks that do happen
  - book calls this 'a good rule of thumb'
- Or - by using formulas etc

Cost estimation

- Experience-based techniques
  - based on experience of prior similar projects
  - based on managers familiarity with work involved
- Algorithmic Cost modelling
  - mathematical calculations based on
    - estimated amount of effort
    - experience-based constants
    - e.g. COCOMO II
    - although often as much as 25%~400% wrong

Algorithmic Cost Modelling

$$Effort = A \times Size^{B} \times M$$

- A - a constant based on organisation overheads
- Size - A Size estimate on code, or amount of functionality
- B - size of project, likelihood of overrun
  - B is bigger for larger projects ($1<B<1.5$)
- M - a multiplier for product/process/people attributes
  - i.e., a project needing more rigorous process for safety

Algorithmic Cost Modelling

- Are often complex and make people nervous of using them
- Are typically inaccurate,
  - because e.g. lines of code is hard to estimate at the start
- Recommend that you produce:
  - optimal cost model
  - worse case cost model
  - likely cost model
- Then use the experience of project management staff

COCOMO

- An empirical independent model based on project experience
- Initial version was 1981
  - COCOMO II has lots of possible multipliers
- It has a number of weights for the project
  - multiplier for remote work
  - multiplier for product complexity
  - multipliers for previous types of experience etc
- try it: http://softwarecost.org/tools/COCOMO/

COCOMO II

Software Size Sizing Method: Source Lines of Code

	SLOC	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	50000						
Reused	20000	0	0	20	5		
Modified	10000		10	20	4	25	0.3

Software Scale Drivers

Precedentedness	Low	Architecture / Risk Resolution	Nominal	Process Maturity	Low
Development Flexibility	Low	Team Cohesion	High		

Software Cost Drivers

Product		Personnel		Platform	
Required Software Reliability	Very High	Analyst Capability	Nominal	Time Constraint	Nominal
Data Base Size	Nominal	Programmer Capability	Nominal	Storage Constraint	Nominal
Product Complexity	Nominal	Personnel Continuity	Nominal	Platform Volatility	Nominal
Developed for Reusability	Nominal	Application Experience	Nominal	**Project**	
Documentation Match to Lifecycle Needs	Nominal	Platform Experience	Nominal	Use of Software Tools	Nominal
		Language and Toolset Experience	Nominal	Multisite Development	High
				Required Development Schedule	Nominal

Maintenance Off

Software Labor Rates

Cost per Person-Month (Dollars) 50

Calculate

COCOMO II

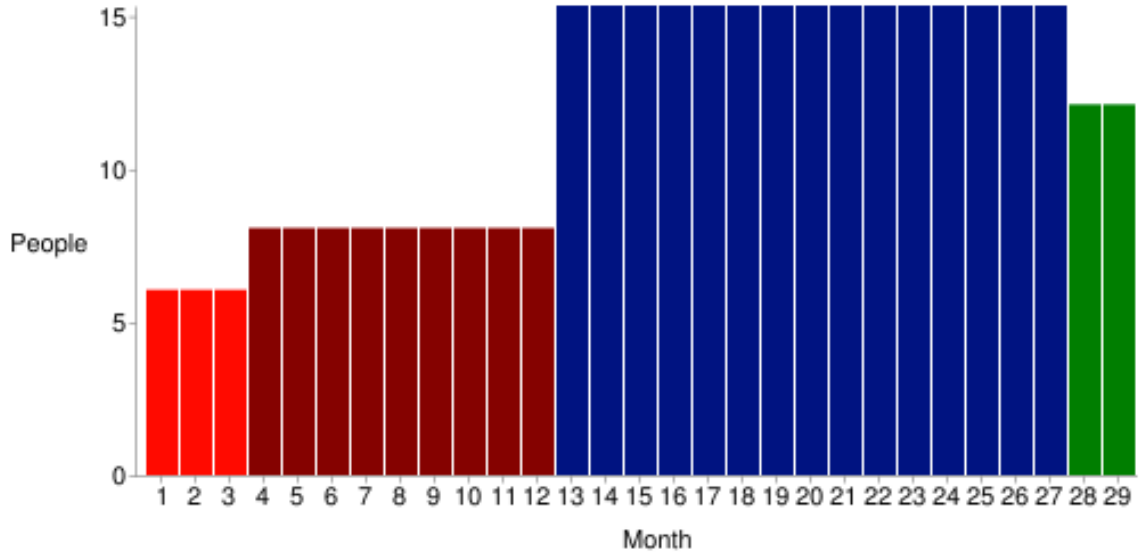
Results

Software Development (Elaboration and Construction)

Effort = 306.2 Person-months
Schedule = 24.3 Months
Cost = $15307

Total Equivalent Size = 53635 SLOC

Staffing Profile



Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	18.4	3.0	6.1	$918
Elaboration	73.5	9.1	8.1	$3674
Construction	232.7	15.2	15.3	$11634
Transition	36.7	3.0	12.1	$1837

Software Effort Distribution for RUP/MBASE (Person-Months)

Phase/Activity	Inception	Elaboration	Construction	Transition
Management	2.6	8.8	23.3	5.1
Environment/CM	1.8	5.9	11.6	1.8
Requirements	7.0	13.2	18.6	1.5
Design	3.5	26.5	37.2	1.5
Implementation	1.5	9.6	79.1	7.0
Assessment	1.5	7.3	55.8	8.8
Deployment	0.6	2.2	7.0	11.0

Your output file is http://csse.usc.edu/tools/data/COCOMO_April_30_2015_03_17_04_341136.txt

Actually Pricing the Project

- Is very hard - and relies on lots of experience
- You can estimate how much work a project will take
  - but that estimate is only as good as the spec of the project
- You may add time/costs for projects risks
- You may reduce cost for competitiveness (if bidding)

Project Pricing Factors

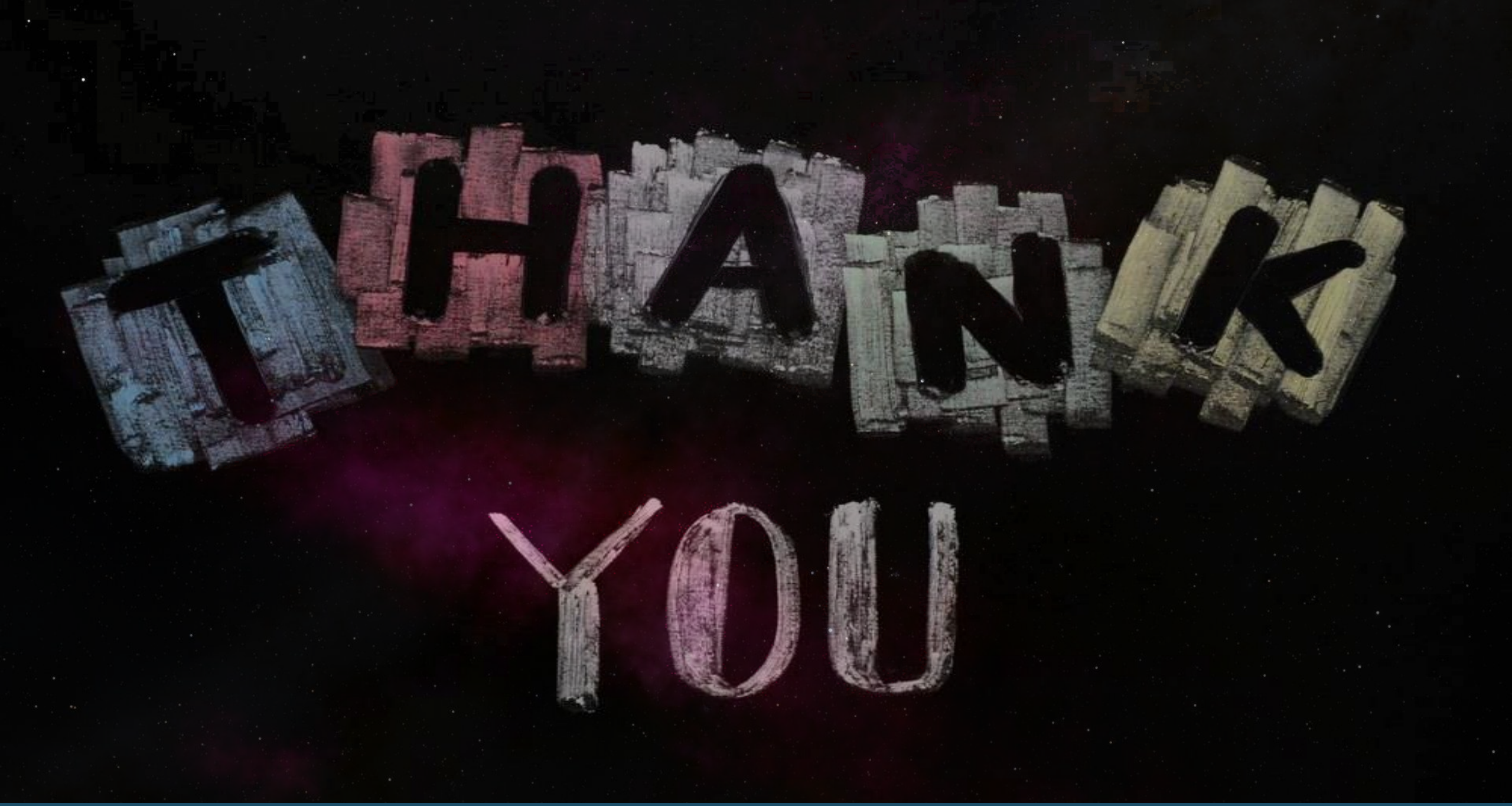
Factor	Description
Market opportunity	A development organization may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the organization the opportunity to make a greater profit later. The experience gained may also help it develop new products.
Cost estimate uncertainty	If an organization is unsure of its cost estimate, it may increase its price by a contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organization may lower its price to win a contract. After the contract is awarded, high prices can be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a smaller than normal profit or break even than to go out of business. Cash flow is more important than profit in difficult economic times.

Project Pricing - For Now

- You've been told how to estimate the cost
  - break down work - and plan order of work/deliverables/etc
  - planning/estimate effort and time for the project
  - account for risks etc
- The cost will be **at least** this
  - Then sales people figure out profit, demand, etc.
- This guides the cost of building the software
  - experience and managers will decide what they charge for it

Summary

- There are several techniques to plan a project
- After you break down the tasks, its a lot easier to:
  - estimate how many people you need
  - estimate how long it will take
  - estimate how much it will cost
- Given that experience tells you how to break down tasks


THANK
YOU