Queen Mary
University of London

ECS505U SOFTWARE ENGINEERING

MUSTAFA BOZKURT

LECTURER IN SOFTWARE ENGINEERING

WEEK 10

PROJECT MANAGEMENT

OBJECTIVES

- Understand why you need project management
- Techniques for estimating the required amount of work to develop a system.
- The basics of organizing software engineering teams.
- Techniques for planning, scheduling and tracking the work.
- Use graphical schedule representations
- Gain a better understanding of risks

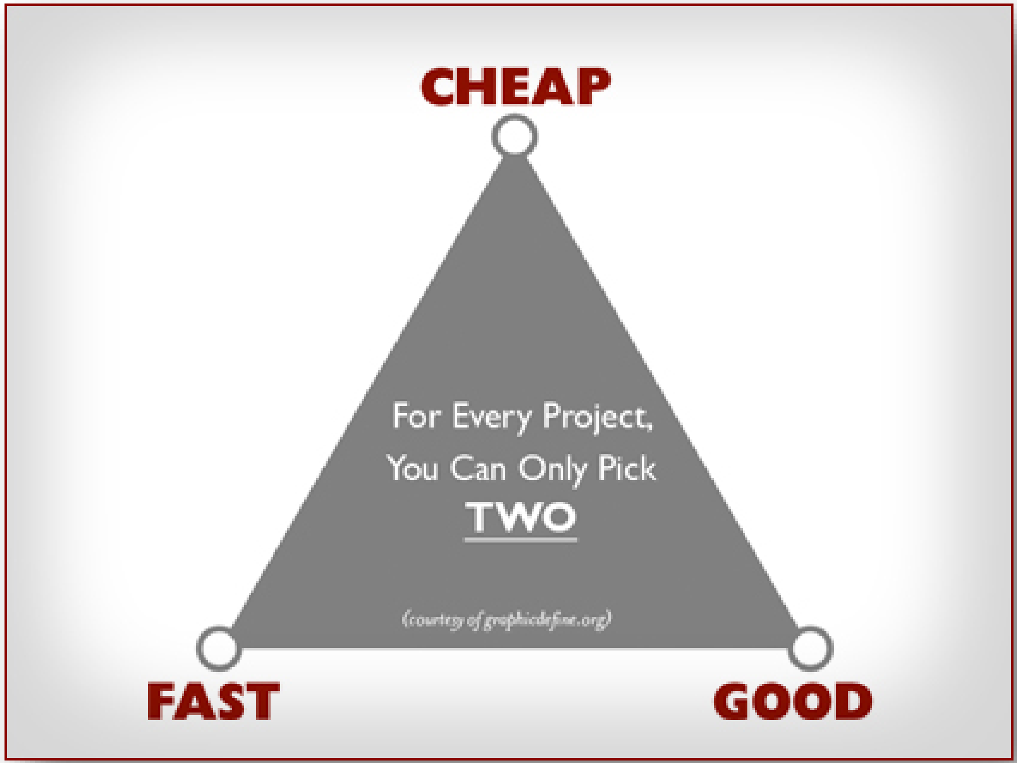
WHAT IS A PROJECT?

A project is temporary in that it has a **defined beginning and end in time**, and therefore **defined scope and resources**.

A project is unique in that it is not a routine operation, but a **specific set of operations designed to accomplish a singular goal**.

PROJECT CONSTRAINTS

- **Time**
- **Cost**
- **Scope/Quality**


CHEAP
For Every Project, You Can Only Pick TWO
(courtesy of graphicdefine.org)
FAST
GOOD

http://www.mopdog.com/wp-content/uploads/2009/09/revised-triangle.jpg

WHAT IS PROJECT MANAGEMENT?

Project management is the process and activity of planning, organizing, motivating, and controlling resources, procedures and protocols to achieve specific goals in scientific or daily problems.
[Wikipedia]

Project management is the application of processes, methods, knowledge, skills and experience to achieve the project objectives. [Association for PM]

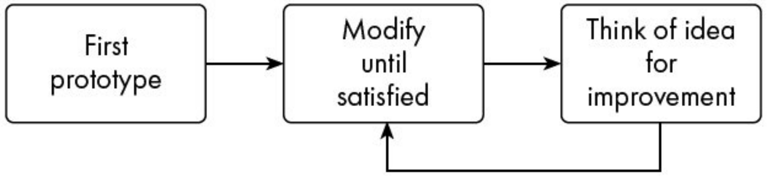
Project management, is the application of knowledge, skills and techniques to execute projects effectively and efficiently.
[Project Management Institute]

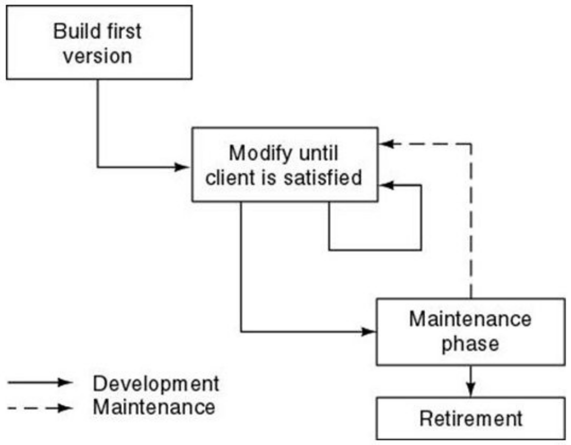
PROJECT MANAGEMENT

- Deciding what needs to be done
- Estimating costs
- Ensuring there are suitable people to undertake the project
- Defining responsibilities
- Scheduling
- Making arrangements for the work
- Directing
- Being a technical leader
- Reviewing and approving decisions made by others
- Building morale and supporting staff
- Monitoring and controlling
- Coordinating the work with managers of other projects
- Reporting
- Continually striving to improve the process

SOFTWARE PROCESS MODELS

- Opportunistic approach (build & fix)
- Waterfall
- Formal transformation
- Rapid prototyping
- Spiral
- Unified process
- Agile approaches
  - Extreme programming
  - Scrum





COST ESTIMATION

- One of the biggest challenges in software engineering
  - It’s **very very very** hard!
- Calculation is based on several metrics
  - Elapsed time
  - Development effort (person-months/days)
  - Project size and complexity

Project cost = *weighted average cost* x *effort*

Queen Mary University of London

COST ESTIMATION

Algorithmic cost modelling	A model is developed using historical cost information that relates some software metric (usually its size) to the project cost. An estimate is made of that metric and the model predicts the effort required.
Expert judgement	Several experts on the proposed software development techniques and the application domain are consulted. They each estimate the project cost. These estimates are compared and discussed. The estimation process iterates until an agreed estimate is reached.
Estimation by analogy	This technique is applicable when other projects in the same application domain have been completed. The cost of a new project is estimated by analogy with these completed projects.
Parkinson's Law	Parkinson's Law states that work expands to fill the time available. The cost is determined by available resources rather than by objective assessment. If the software has to be delivered in 12 months and 5 people are available, the effort required is estimated to be 60 person-months.
Pricing to win	The software cost is estimated to be whatever the customer has available to spend on the project. The estimated effort depends on the customer's budget and not on the software functionality

EFFORT ESTIMATION

Project managers base their estimates on factors such as

- The number of use cases.
- The number of distinct requirements.
- The number of classes in the domain model.
- The number of widgets in the prototype user interface.
- An estimate of the number of lines of code.

ALGORITHMIC EFFORT ESTIMATION

The COnstructive COst MOdel (COCOMO) model is an empirical model derived from the data of earlier software projects.

The reason for discussing COCOMO

- It is well documented, available in the public domain and supported by public domain and commercial tools.
- It has been widely used and evaluated in a range of organizations.

COCOMO MODEL

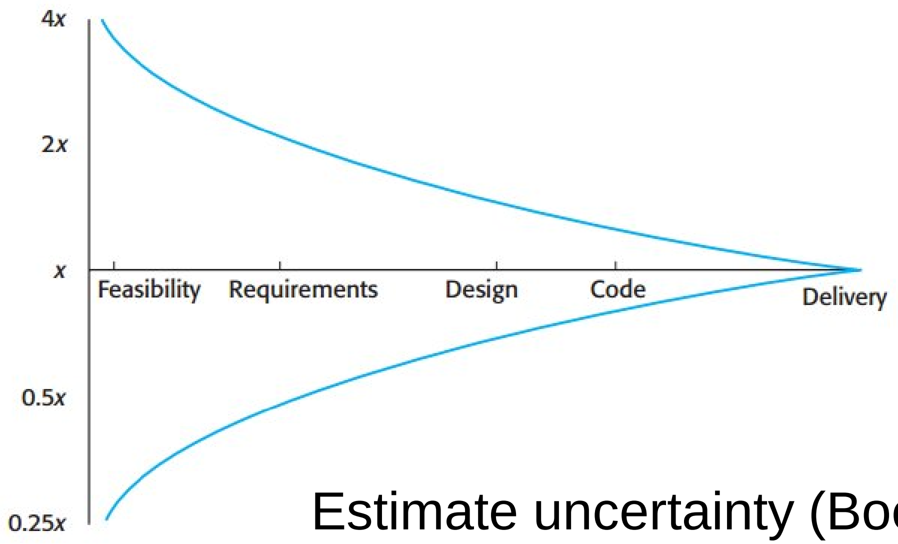
History of COCOMO

- Instantiation in 1981 (Boehm, 1981)
- Refinement to Ada software (Boehm and Royce, 1989)
- Latest ver. COCOMO II in 2000 (Boehm, et al., 2000)

COCOMO MODEL

Difficulties:

- Difficult to estimate size at an early stages.
  - Function-point and object-point easier to produce but are often still inaccurate.
- Estimating the factors contributing to *b* and *c* are subjective.



Estimate uncertainty (Boehm, et al., 1995)

BASIC COCOMO MODEL

Simplistic COCOMO model for effort estimation

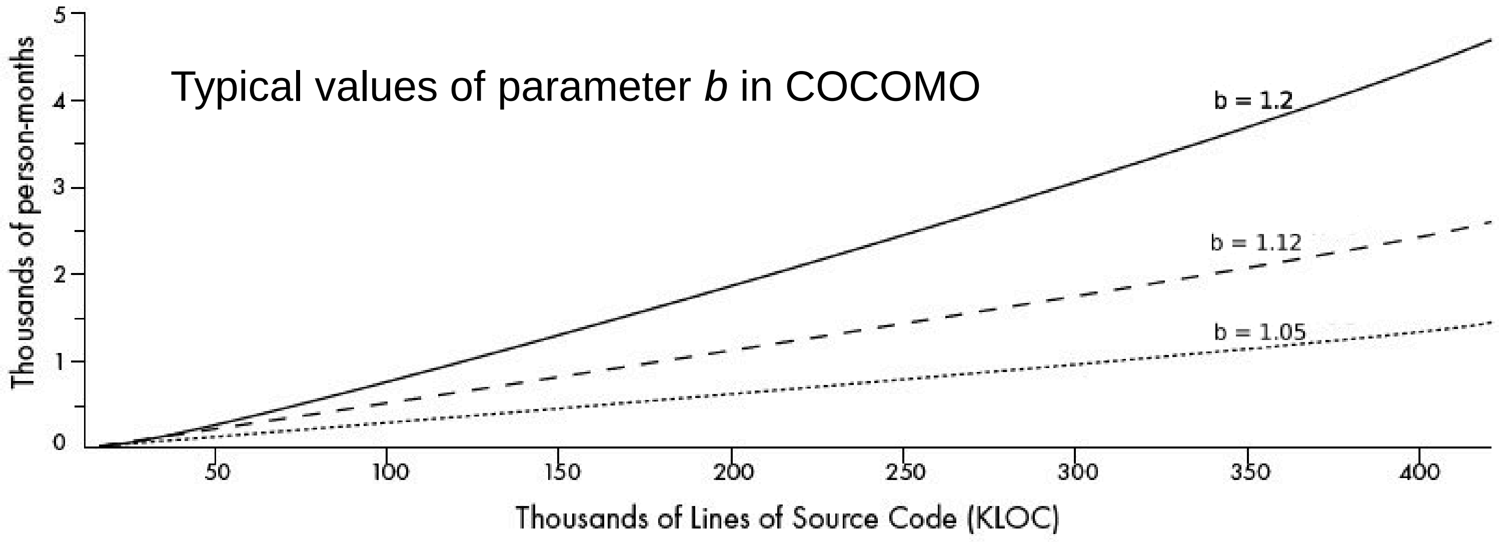
Effort (person-month)	$E = a \ x \ N^b$
Development Time (month)	$D = 2.5 \ x \ E^c$
People required	$P = \frac{E}{D}$

where

- N is the size measurement (KDSI, KLOC)
- a is a constant factor that depends on local organizational practices and the type of software that is developed
- b is a constant representing relation between effort and project size

BASIC COCOMO MODEL

$$E = a \ x \ N^{b}$$



BASIC COCOMO MODEL

Software project	a	b	c	
Organic	2.4	1.05	0.38	Well-understood applications developed by small teams
Semi-detached	3	1.12	0.35	More complex projects where team members may have limited experience of related systems
Embedded	3.6	1.2	0.32	Complex projects where the software is part of a strongly coupled complex of hardware

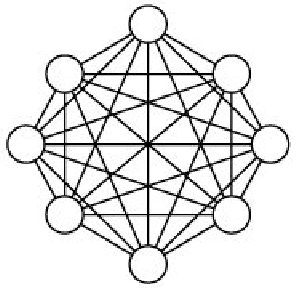
COCOMO II VS COCOMO

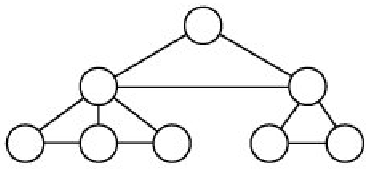
- COCOMO requires software size in KDSI as an input, but COCOMO II is based on KSLOC (logical code).
- COCOMO II addresses the following three phases of the spiral life cycle: applications development, early design and post architecture
- COCOMO provides point estimates of effort and schedule, but COCOMO II provides likely ranges of estimates that represent one standard deviation around the most likely estimate.
- The estimation equation exponent (*b*) is determined by five scale factors (instead of the three development modes)
- Changes in cost drivers are:
  - Added cost drivers (7): DOCU, RUSE, PVOL, PLEX, LTEX, PCON, SITE
  - Deleted cost drivers (5): VIRT, TURN, VEXP, LEXP, MODP
  - Alter the retained ratings to reflect more up-do-date software practices
- Data points in COCOMO I: 63 and COCOMO II: 161
- COCOMO II adjusts for software reuse and reengineering where automated tools are used for translation of existing software, but COCOMO'81 made little accommodation for these factors
- COCOMO II accounts for requirements volatility in its estimates

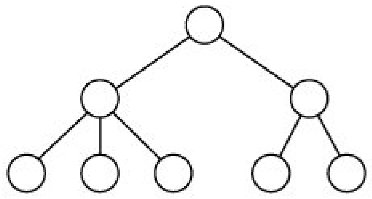
BUILDING SOFTWARE ENGINEERING TEAMS

Team organization:

- Egoless approach: Decisions made by consensus
- Strict hierarchy approach: Everyone do their own work
- Chief programmer approach: Decisions made by specialists and chief programmer.


(a) Egoless


(b) Chief programmer


(c) Strict hierachy

BUILDING SOFTWARE ENGINEERING TEAMS

Team size:

- Relation between number of staff, total effort and development time is *not linear*.
- Adding more people to a late project makes it later.

PROJECT ROLES

- Analyst/Architect
- Designer
- Programmer
- Tester
- Maintainer
- Trainer
- Liaison
- Minute Taker
- Project Manager
- Document Editor
- Configuration Manager
- Researcher

COMMUNICATION IS IMPORTANT

Queen Mary
University of London

Working in a large project

- More time in communicating than coding
- A software engineer needs to learn the so-called soft skills
  - Technical writing
  - Communication
  - Collaboration
  - Etc…
- The project manager has to ensure the right level and types of communication take place

TECHNICAL WRITING

- A crucial skill
- Key component of software engineering
- It’s how you are assessed
- You can learn to do it better!

TECHNICAL WRITING: MAIN POINTS

- Ensure all reports conform to the basic structure
- Keep things as simple as possible
- Avoid long sentences
- Do not turn verbs into nouns
- Use active rather than passive
- Use personal rather than impersonal style

PROJECT SCHEDULING

Two important aspects of project management

- Scheduling
  - Set of activities performed
  - Start and complete
- Tracking
  - Cost
  - Schedule

Queen Mary University of London

EXAMPLES OF ACTIVITIES

Major Activities:

- Planning
- Requirements Elicitation
- Requirements Analysis
- System Design
- Object Design
- Implementation
- System Testing
- Delivery

Activities during requirements analysis:

- Refine scenarios
- Define Use Case model
- Create use cases
- Define acceptance tests

EXAMPLES OF TASKS

- Unit test class “Foo”
- Test subsystem “Bla”
- Write user manual
- Write meeting minutes and post them
- Write a memo on Mac OS vs Windows
- Schedule the code review
- Develop the project plan

Queen Mary
University of London

PROJECT SCHEDULING

Two diagrams

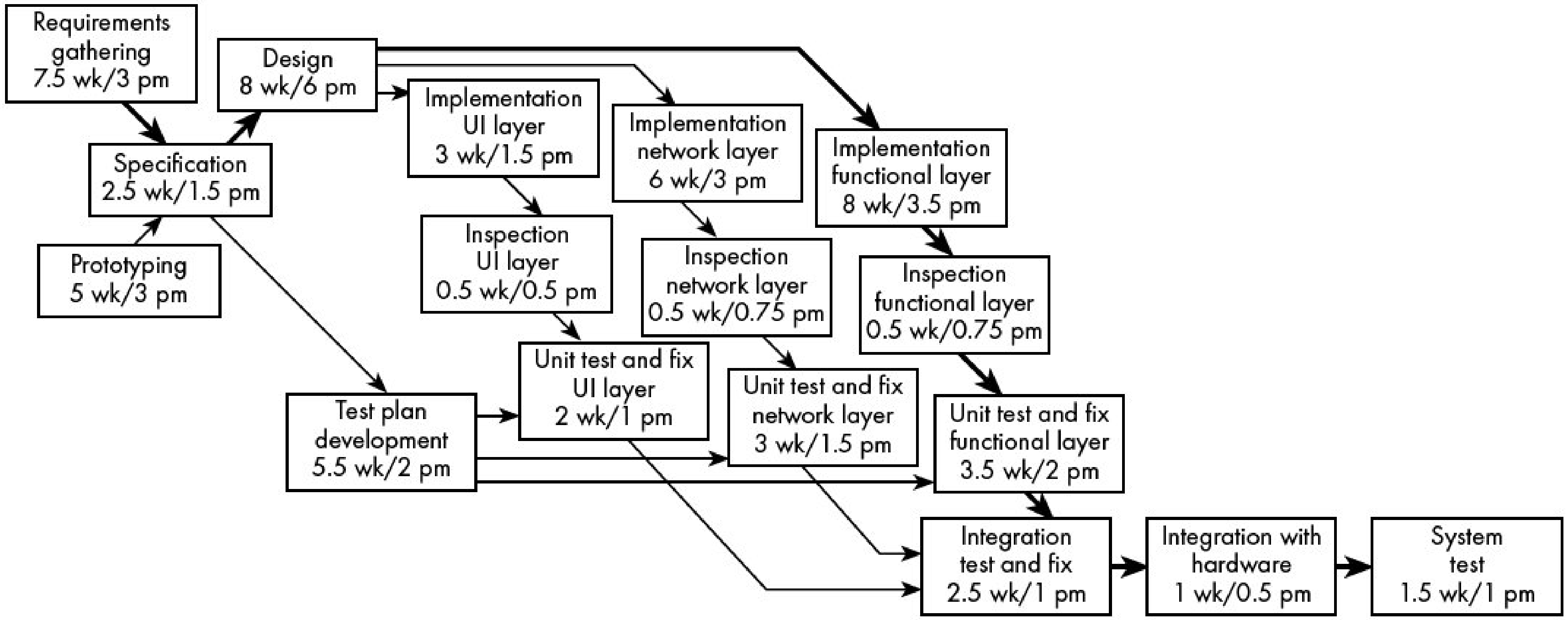
- Program Evaluation Review Technique (PERT) charts
- Gantt chart

PERT DIAGRAM

Two diagrams

- Shows the sequence of tasks
- A graph with nodes as tasks and arcs dependencies
- Nodes show time and effort
- Optimistic and pessimistic estimates can be shown
- Determines critical path

PERT DIAGRAM



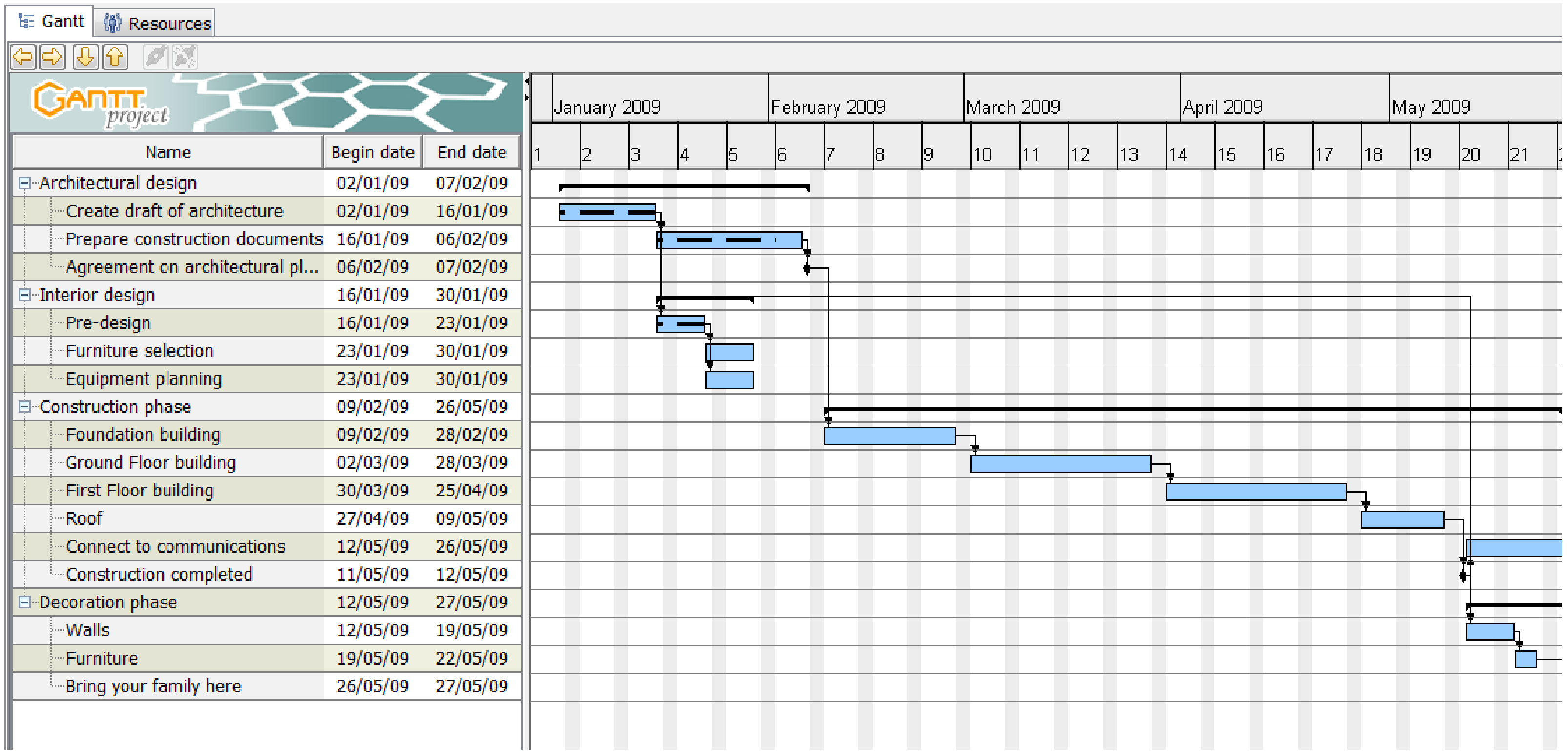
Requirements gathering 7.5 wk/3 pm
Design 8 wk/6 pm
Specification 2.5 wk/1.5 pm
Prototyping 5 wk/3 pm
Implementation UI layer 3 wk/1.5 pm
Implementation network layer 6 wk/3 pm
Implementation functional layer 8 wk/3.5 pm
Inspection UI layer 0.5 wk/0.5 pm
Inspection network layer 0.5 wk/0.75 pm
Inspection functional layer 0.5 wk/0.75 pm
Test plan development 5.5 wk/2 pm
Unit test and fix UI layer 2 wk/1 pm
Unit test and fix network layer 3 wk/1.5 pm
Unit test and fix functional layer 3.5 wk/2 pm
Integration test and fix 2.5 wk/1 pm
Integration with hardware 1 wk/0.5 pm
System test 1.5 wk/1 pm

GANTT CHART

Developed by Henry Gantt in the 1910s

- A bar chart that illustrates a project schedule.
- Present start and end dates of each task
- Does not show allocation of resources

GANTT CHART



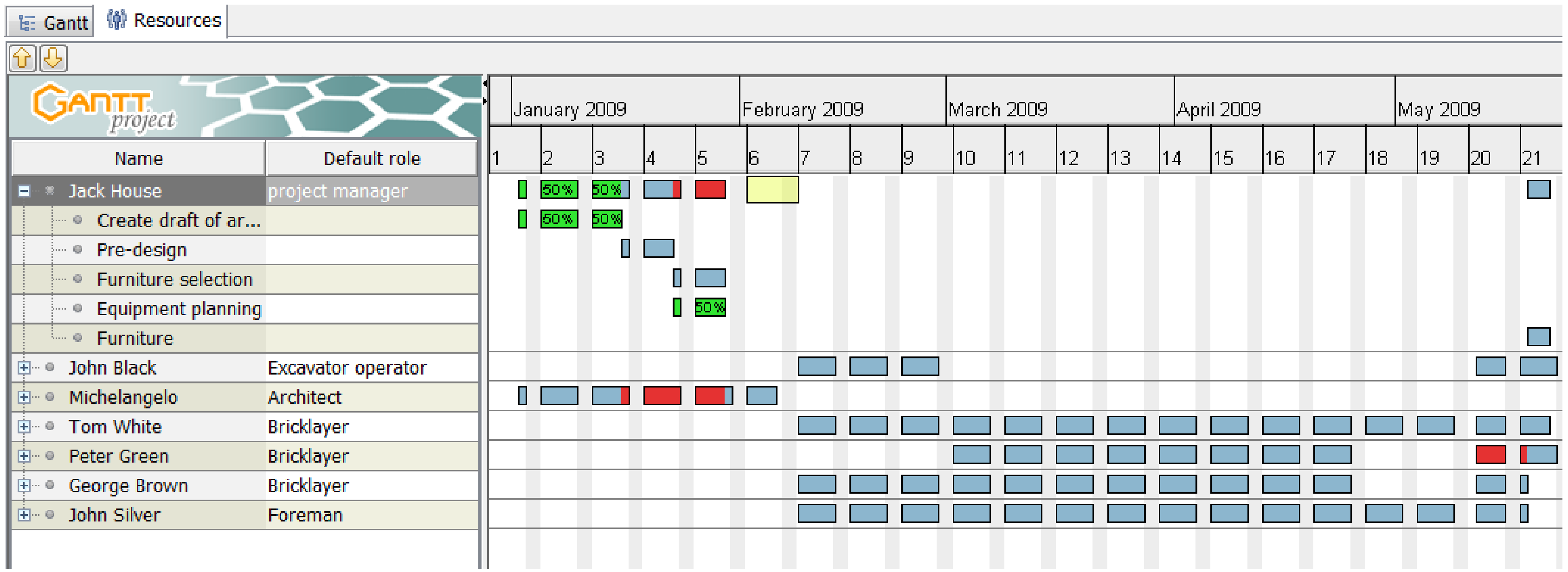
Name	Begin date	End date
Architectural design	02/01/09	07/02/09
Create draft of architecture	02/01/09	16/01/09
Prepare construction documents	16/01/09	06/02/09
Agreement on architectural pl...	06/02/09	07/02/09
Interior design	16/01/09	30/01/09
Pre-design	16/01/09	23/01/09
Furniture selection	23/01/09	30/01/09
Equipment planning	23/01/09	30/01/09
Construction phase	09/02/09	26/05/09
Foundation building	09/02/09	28/02/09
Ground Floor building	02/03/09	28/03/09
First Floor building	30/03/09	25/04/09
Roof	27/04/09	09/05/09
Connect to communications	12/05/09	26/05/09
Construction completed	11/05/09	12/05/09
Decoration phase	12/05/09	27/05/09
Walls	12/05/09	19/05/09
Furniture	19/05/09	22/05/09
Bring your family here	26/05/09	27/05/09

Gantt charts show schedule against calendar time as well as task dependencies

RESOURCE CHART

- Presents what each member of the team is
  - Currently working on,
  - What they have completed
  - Which tasks will they do
- Provides insight into time spent on each individual task
- Provides percentage of available time for each team member.

RESOURCE CHART



Name	Default role
Jack House	project manager
Create draft of ar...	
Pre-design	
Furniture selection	
Equipment planning	
Furniture	
John Black	Excavator operator
Michelangelo	Architect
Tom White	Bricklayer
Peter Green	Bricklayer
George Brown	Bricklayer
John Silver	Foreman

Resource charts show people effort against calendar time and tasks

TRACKING

Gates, checkpoints and milestones

- Milestones are points of completion or achievement on a plan.
  For example completing a document or getting agreement from a customer.
- Checkpoints are calendar points at which the project reports progress.

  For example weekly checkpoint meetings.
- Gates are formal points in the project where resourcing for continued work is agreed.

CHECKPOINT MEETING

Date

Period covered

Present

Apologies

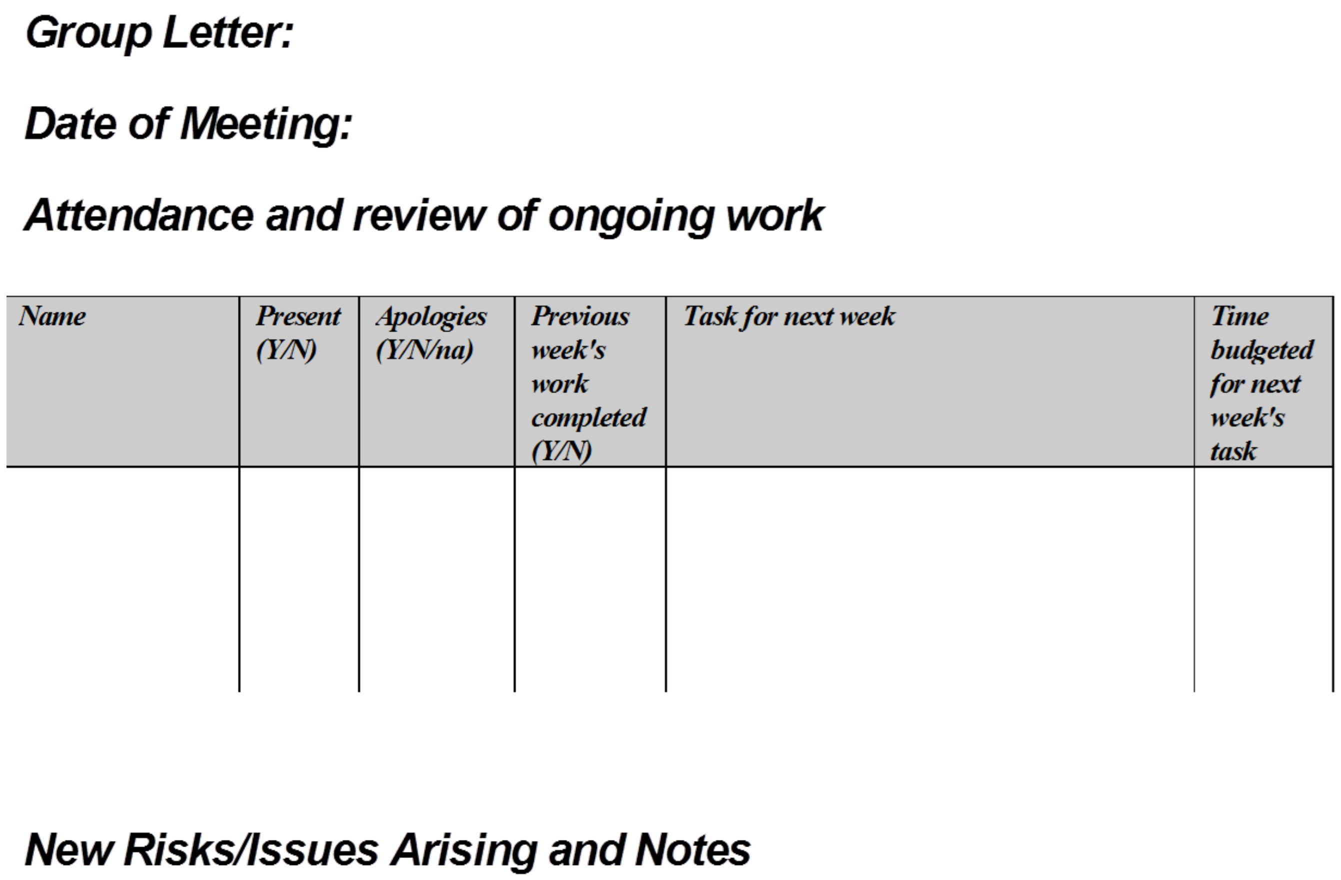
Review of issues arising

Review of tasks agreed at last meeting

Tasks to be worked on between now and next meeting (+ named individual responsible)

Issues

CHECKPOINT TEMPLATE



Group Letter:

Date of Meeting:

Attendance and review of ongoing work

Name	*Present (Y/N)*	*Apologies (Y/N/na)*	*Previous week's work completed (Y/N)*	*Task for next week*	*Time budgeted for next week's task*

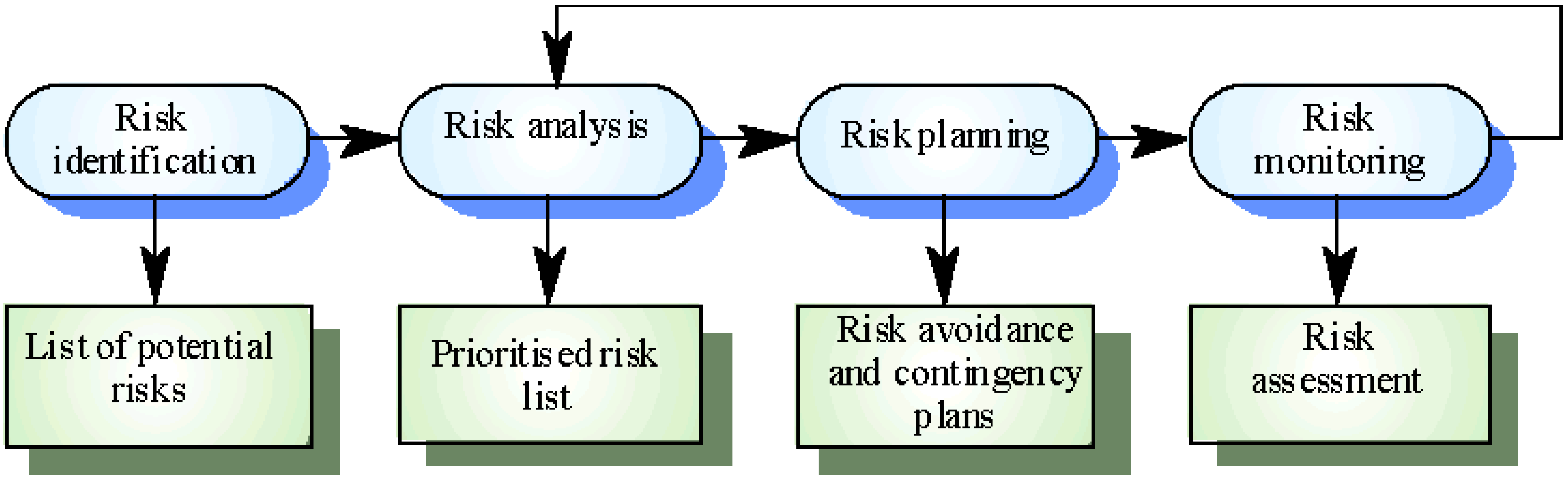
New Risks/Issues Arising and Notes

PROJECT MANAGEMENT PLAN (IDEAL)

Queen Mary
University of London

- Purpose
- Background information
- Processes to be used
- Subsystems and planned releases
- Risks and challenges
- Tasks
- Cost estimates
- Team
- Schedule and milestones

THE RISK MANAGEMENT PROCESS



RISK ANALYSIS

- Assess probability and seriousness of each risk
- Probability may be very low, low, moderate, high or very high
- Risk effects might be catastrophic, serious, tolerable or insignificant

RISK PLANNING

- Consider each risk and develop a strategy to manage that risk
- Avoidance strategies
- Minimisation strategies
- Contingency plans

RISK MONITORING

- Assess each identified risks regularly to decide whether or not it is becoming less or more probable
- Also assess whether the effects of the risk have changed
- Each key risk should be discussed at management progress meetings

PROJECT SCHEDULING: GENERAL ISSUES

- Split project into activities and tasks
- Identify deliverables and/or milestones with each task
- Activities include management, testing as well as coding
- Estimate time and resources required to complete each task
- Minimise task dependencies to avoid delays
  caused by one task waiting for another to complete

SCHEDULING PROBLEMS

- Estimating the difficulty of problems and hence the cost of developing a solution is hard
- Productivity is not proportional to the number of people working on a task
- Adding people to a late project makes it later because of communication overheads
- The unexpected always happens. Always allow contingency in planning

DIFFICULTIES IN PROJECT MANAGEMENT

- Accurately estimating costs is a constant challenge.
- It is very difficult to measure progress and meet deadlines
- It is difficult to deal with lack of human resources or technology that is needed to successfully run a project.
- Communicating effectively in a large project is hard.
- It is hard to obtain agreement and commitment from others.

LAWS OF PROJECT MANAGEMENT

- The 50% rule

  "*If you spend 50% of your time planning, you will get it done twice as fast*" – Old German saying

- The 90% rule

  "*The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time*." - Tom Cargill

- The bad luck rule

  When things are going well, something will go wrong. When things just can't get worse, they will. When things appear to be going better than expected, you have overlooked something.

- The change rule

  If project content is allowed to change freely, the rate of change will exceed the rate of progress.

- The progress reporting rule

  Project teams detest progress reporting because it highlights their lack of progress.

SUMMARY

- Good project management is essential for project success
- Good technical writing is essential – and can be learnt
- Planning and estimating are iterative processes
- Focus detail on next iteration
- Breakdown project into tasks/activities
- Identify and manage risks
- Devise Configuration Management Plan