

Software Project Management

Bachelor's Degree in Computer Engineering

University of Valladolid

2024-25 Academic Term

 Jesús Vegas (jvegas@uva.es)

Based in Software Project Management, Bob Hughes and Mike Cotterell, 5th ed. Most of the figures with schematics and diagrams are taken from this edition. The rest of the figures included are free stock images.



Project and Project Management

Subject Zero



Project

A **planned** and **temporary** initiative aimed at achieving a specific **goal** by organizing **resources** and **activities** within predefined **constraints**



Project Management

- Critical discipline in today's business and organizational environments
- To turn ideas into reality
 - structured and executable project
- Specific methodologies and tools



Major Projects

1. Apollo 11 Moon Landing
2. Burj Khalifa (Dubai, UAE)
3. Panama Canal Expansion (Third Set of Locks Project)
4. Apple iPhone Development
5. Tesla Gigafactory (Nevada, USA)
6. COVID-19 Vaccine Development (Pfizer-BioNTech and Moderna)



Apollo 11 Moon Landing

- **Type:** Aerospace and Science
- **Objective:** Land humans on the moon and return them safely to Earth.
- **Key Figures:**
 - Nearly a decade, from the early 1960s to the successful moon landing in 1969
 - Total program cost was approximately \$25.4B
 - Over 400,000 engineers, scientists, and technicians
- **Tools:** Complex simulations, risk management, and groundbreaking technology



Burj Khalifa (Dubai, UAE)

- **Type:** Construction and Engineering
- **Objective:** Build the tallest skyscraper in the world
- **Key Figures:**
 - Construction took 6 years, from 2004 to 2010
 - Cost estimated at \$1.5B
 - More than 12,000 workers were on-site during the peak of construction
- **Tools:** Project scheduling software, risk management tools, and advanced architectural modeling



Panama Canal Expansion (Third Set of Locks Project)

- **Type:** Infrastructure and Transportation
- **Objective:** Expand the capacity of the Panama Canal to accommodate larger ships
- **Key Figures:**
 - Took 9 years, from 2007 to 2016
 - Cost estimated at \$5.25B
 - More than 30,000 workers
- **Tools:** Advanced project planning software, environmental impact assessments, and logistical management tools



Apple iPhone Development

- **Type:** Technology and Product Development
- **Objective:** Develop and launch the first smartphone with a touch-based interface
- **Key Figures:**
 - Took roughly 2.5 years from initial concept to launch in 2007
 - Estimated cost was between \$150M and \$200M
- **Tools:** Prototyping, design thinking, and agile project management methodologies



Tesla Gigafactory (Nevada, USA)

- **Type:** Manufacturing and Renewable Energy
- **Objective:** Build a factory to produce lithium-ion batteries for electric vehicles at an unprecedented scale
- **Key Figures:**
 - To be the largest building in the world by footprint
 - Started in 2014, and the factory began partial operations in 2016
 - Estimated cost over \$5B
- **Tools:** Lean manufacturing, supply chain, resource planning



COVID-19 Vaccine Development (Pfizer-BioNTech and Moderna)

- **Type:** Healthcare and Pharmaceuticals
- **Objective:** Develop an effective vaccine for COVID-19 in record time
- **Key Figures:**
 - Development took less than a year, with emergency use authorizations granted in December 2020
 - Cost around \$2B to develop
- **Tools:** Advanced research techniques, rapid clinical trials, global distribution



Certifications

- Provide individuals with formal recognition of their project management skills, knowledge, and competencies
- Offered by various organizations and cover different aspects of project management, methodologies, and industries
- Valuable for enhancing career opportunities, demonstrating expertise, and ensuring success in diverse project environments



Entry Level Certifications

Certification	Provider	Focus Area
CAPM Certified Associate in Project Management	Project Management Institute (PMI)	Fundamental project management concepts
PRINCE2 Foundation	AXELOS	Entry-Level



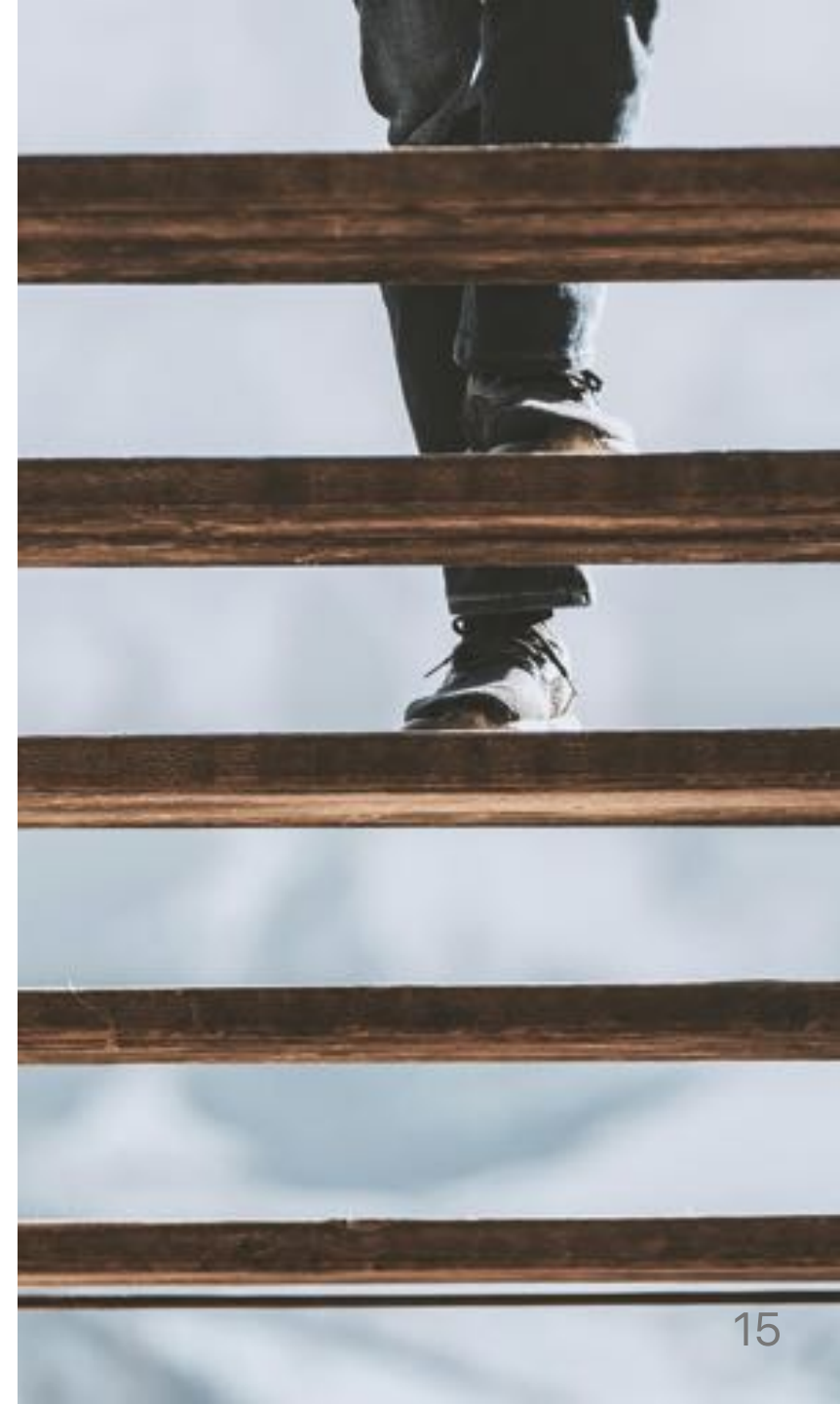
Mid-level Certifications

Certification	Provider	Focus Area
PMP Project Management Professional	PMI	Comprehensive project management skills
PRINCE2 Practitioner	AXELOS	Advanced PRINCE2 application
CSM Certified ScrumMaster	Scrum Alliance	Agile project management (Scrum)



Advanced Level Certifications

Certification	Provider	Focus Area
PgMP Program Management Professional	PMI	Program management
PfMP Portfolio Management Professional	PMI	Portfolio management
PMI-ACP Agile Certified Practitioner	PMI	Agile project management expertise



Specialized Certifications

Certification	Provider	Focus Area
PMI-RMP PMI Risk Management Professional	PMI	Project risk management
PMI-SP PMI Scheduling Professional	PMI	Project scheduling expertise
DA Disciplined Agile	PMI	Enterprise Agile project management



Leadership and Strategic Certifications

Certification	Provider	Focus Area
CPD Certified Project Director	Global Association for Quality Management (GAQM)	High-level project leadership
CSPO Certified Scrum Product Owner	Scrum Alliance	Product Owner role in Scrum



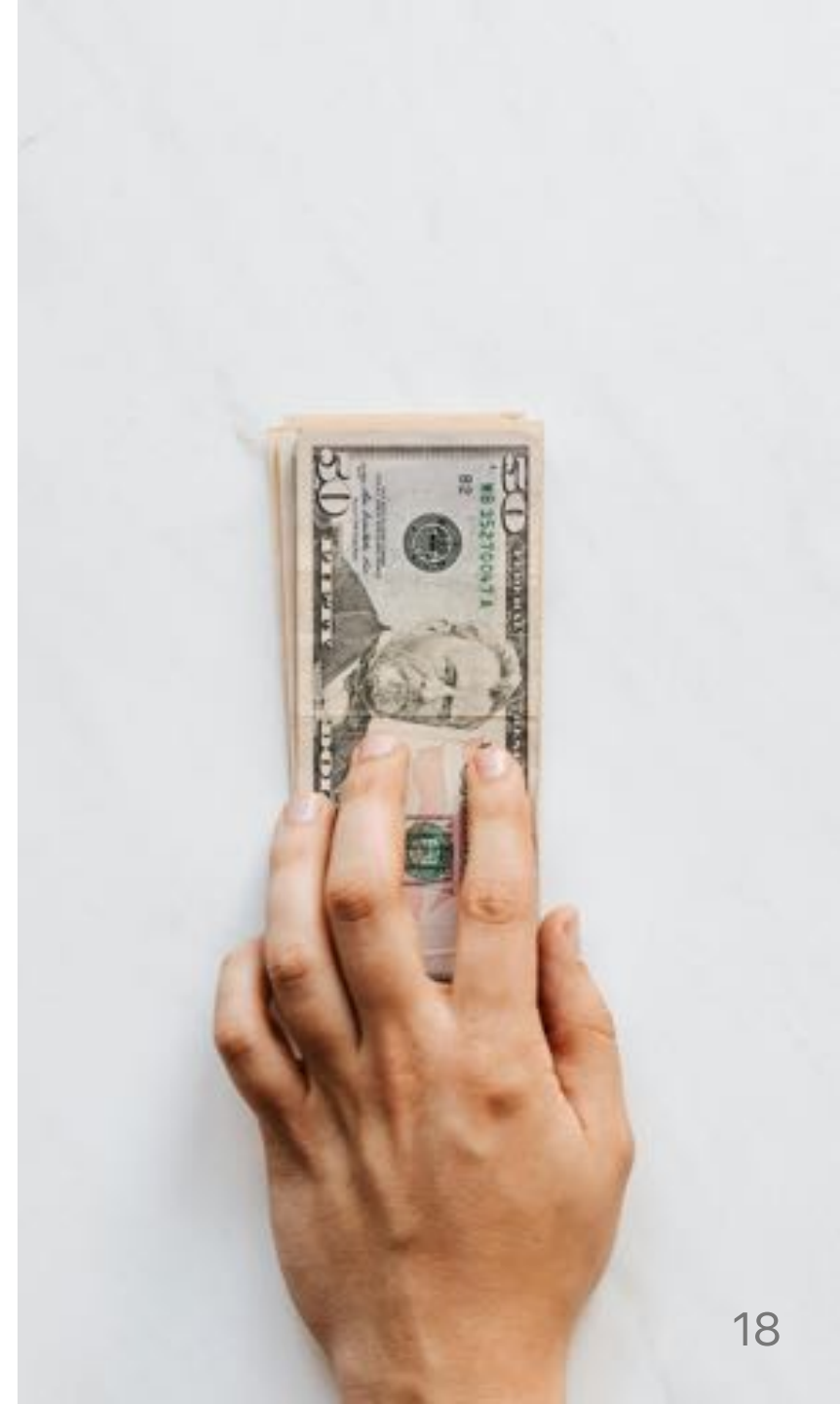
Salary of PMs

- Varies significantly depending on the industry, location, experience level, and certifications

[How Much Do Project Managers Make? Complete 2024 Salary Guide](#)

[Project Manager Salary Guide: Average By Country & Role 2024](#)

[Project Manager Salary: 2023 Guide to Knowing Your Worth](#)



Depending on Industry

- **IT:** Project managers in information technology generally earn higher salaries. For example, in the U.S., they earn an average of \$95,904 annually
- **Construction:** Salaries are slightly lower, with PMs earning around \$91,283 per year
- **Finance:** Similar to IT, finance-related PM roles pay around \$92,065 annually
- **Government:** Government PMs tend to earn less, with an average salary of \$69,237 per year



Depending on Location

- In the U.S., salaries fluctuate by region. For instance, PMs in San Francisco earn \$103,872 annually, while in Florida, they earn about \$69,665
- In Canada, the average salary is between CAD 50,000 to CAD 180,000, depending on experience and industry. PMs in healthcare earn about CAD 94,533
- In the UK, salaries for PMs range from £33,000 to £72,350 depending on experience



Depending on Experience

- Entry-level PMs earn significantly less than senior project managers
- In Canada, PMs with less than one year of experience earn CAD 50,000, while those with more than 10 years of experience earn CAD 97,000



Depending on Certifications

- PMs with certifications such as PMP (Project Management Professional) tend to earn significantly more
- In the U.S., a PMP-certified PM earns approximately \$120,000 annually, a 22% increase compared to non-certified PMs



Salaries for PMs in Spain

- **Junior Project Managers:** €30,000 to €40,000 per year
- **Mid-Level Project Managers:** €40,000 to €60,000 per year
- **Senior Project Managers:** €60,000 to €90,000 per year
- **Project Managers in high-demand industries (like IT or pharmaceuticals):** €70,000 to €100,000 or more per year



General Industry Data and Salary Surveys

- **Glassdoor**: Offers salary insights based on employee-reported data
- **Payscale**: Provides salary information and compensation trends
- **Michael Page** and **Hays**: Recruitment agencies that often publish salary guides and market reports



Key Points in Lecture

- Projects as means to turn ideas into reality as planned and temporary initiatives
- Project management is a critical discipline
- The nature and size of the projects varies greatly
- There are specific certification programmes in project management
- The experience and certification of the PM determines the salary

SUMMARY

Introduction to Software Project Management

Subject One



Outline of Talk

In this introduction the main questions to be addressed will be:

- **What is software project management?** Is it really different from 'ordinary' project management?
- **How do you know when a project has been successful?** For example, do the expectations of the customer/client match those of the developers?

Why is Project Management Important?

- Large amounts of money are spent on ICT (*Information and Communications Technology*)
- Projects often fail
- Poor project management a major factor in these failures



What is a Project?

Some dictionary definitions:

- "A specific plan or design"
- "A planned undertaking"
- "A large undertaking e.g. a public works scheme"
(Longmans dictionary)

Key points above are *planning* and *size of task*

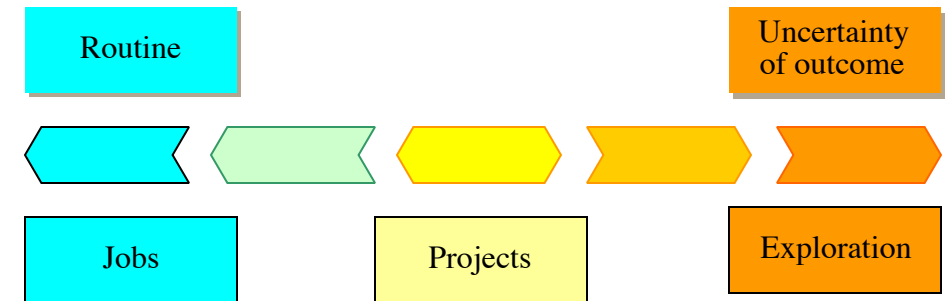


BSO ISO 10006 (1997)

'Unique process, consisting of a set of **coordinated** and **controlled activities** with start and finish **dates**, undertaken to achieve an **objective** conforming to specific **requirements**, including constraints of **time, cost** and **resources**'

Jobs vs Projects

- **Jobs** - repetition of very *well-defined* and well understood tasks with very little uncertainty
- **Exploration** - the outcome is very *uncertain*
- **Projects** - in the middle!



A task is more 'project-like' if it is:

- Non-routine
- Planned
- Aiming at a specific target
- Carried out for a customer
- Carried out by a temporary work group
- Involving several specialisms
- Made up of several different phases
- Constrained by time and resources
- Large and/or complex



Exercise: ranking the projects

Order	Description
	Producing an edition of a newspaper
	Installing a new version of a word processing package in an organization
	Putting a robot vehicle on Mars to search for signs of life
	Amending a financial computer system to deal with a common European currency

Are Software Projects really Different from other Projects?

Not really ... but

- Invisibility
- Complexity
- Conformity
- Flexibility

make software more problematic to build than other engineered artefacts



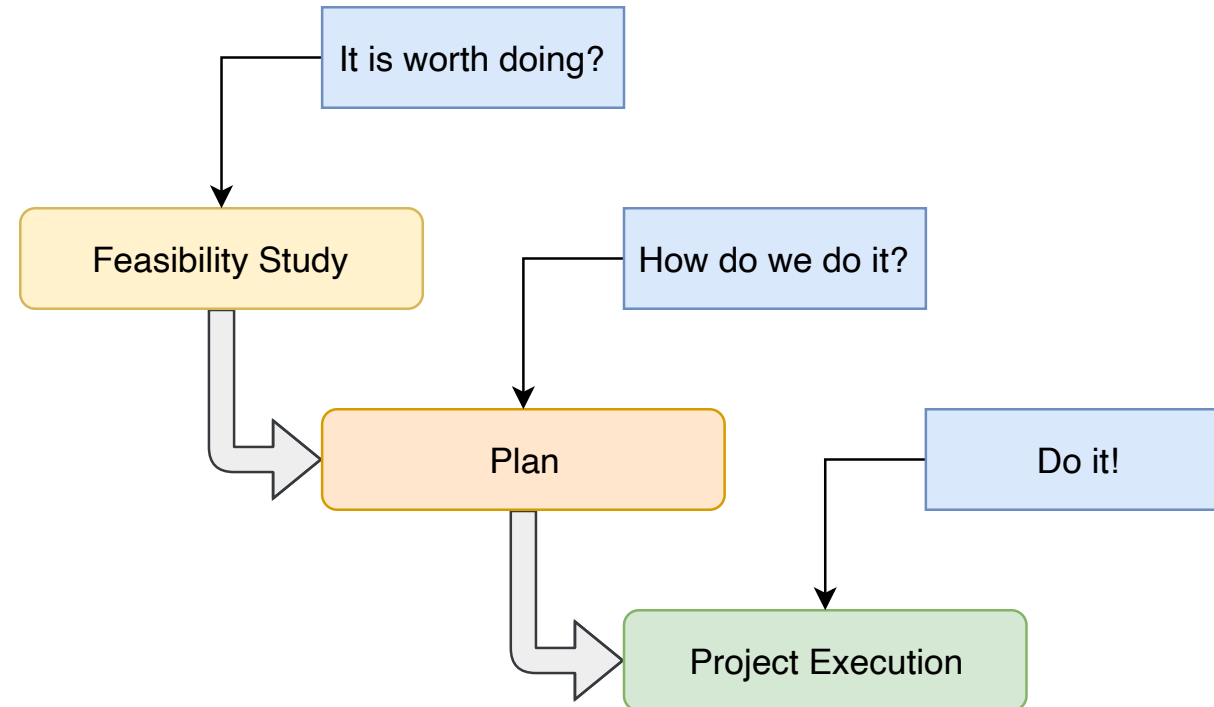
Contract vs Technical Project Management

- Projects can be:
 - **In-house**
 - **Out-sourced**
- Project manager could be:
 - *a contract manager* in the client organization
 - *a technical project manager* in the supplier/services organization

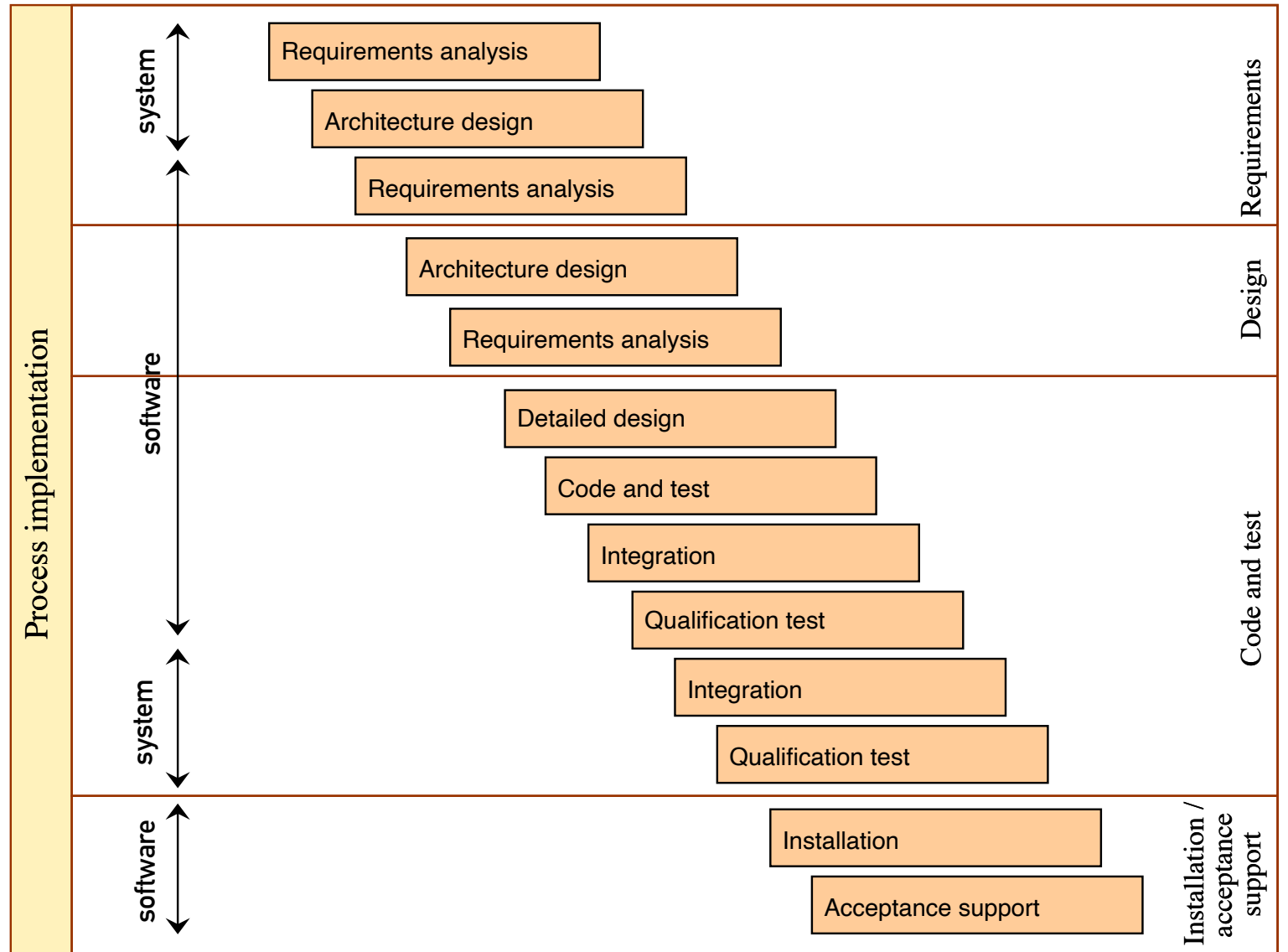


Activities

1. **Feasibility study:** Technically feasible and worthwhile from a business point of view?
2. **Planning:** Only done if project is feasible
3. **Execution:** Implement plan, may be changed as it goes along



The Software Development Life-Cycle (ISO 12207)



ISO 12207 Life-Cycle

- **Requirements analysis**
 - Requirements elicitation: what does the client need?
 - Analysis: converting 'customer-facing' requirements into equivalents that developers can understand
 - Requirements will cover
 - Functions
 - Quality
 - Resource constraints

ISO 12207 Life-Cycle (ii)

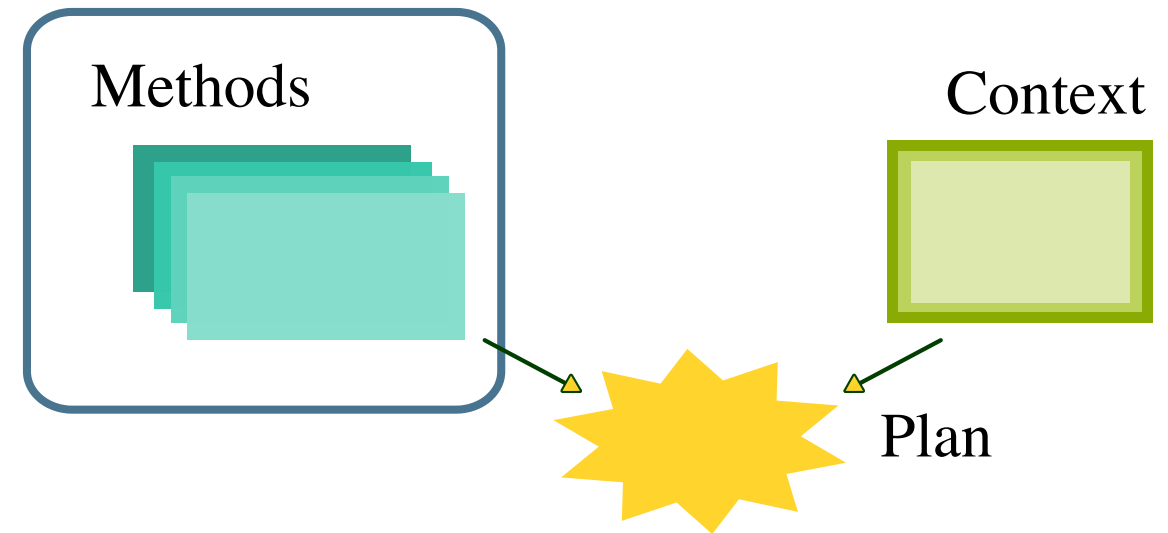
- **Architecture design**
 - Based on *system requirements*
 - Defines components of system: hardware, software, work processes
 - *Software requirements* will come out of this
- **Code and test**
 - Of individual components
- **Integration**
 - Putting the components together

ISO 12207 Life-Cycle (iii)

- **Qualification testing**
 - Testing the *system* (not just the *software*)
- **Installation**
 - The process of making the system operational
 - Includes setting up standing data, setting system parameters, installing on operational hardware platforms, user training, organizational changes
- **Acceptance support**
 - Including maintenance and enhancement

Plans, Methods and Methodologies

- **Methodology:** a set of methods
- **Method:** a way of working
- **Plan:** methods + dates + resources



Categorizing Projects

Distinguishing different types of project is important as different types of task need different project approaches e.g.

- **Voluntary systems** (such as computer games) versus **compulsory systems** e.g. the order processing system in an organization
- **Information systems** versus **embedded systems**
- **Objective-based** versus **product-based**



Compulsory vs Voluntary Users

- In workplaces, the users **have to** use a system if they want to do something
- In the case of computer games, it's use is increasingly **voluntary**
- More difficult to elicit precise requirements from potential users than from compulsory users
 - Market surveys, focus groups and prototype evaluation



Information Systems vs Embedded Systems

- **Information systems** enable staff to carry out office processes
 - **Embedded systems** control machines
- 🤔 Would an operating system on a computer be an information system or an embedded system?



Objectives vs Products

- Projects can be distinguished by whether their aim is to produce a **product** or to meet certain **objectives**
 - If to produce a product, it will be strictly defined by clients
 - More freedom if to meet an objective or to address a problem

👉 An objective-driven project might identify the need of a product, developed by an product-driven project



Stakeholders

Who has interest in the project?

- Could be *users/clients* or *developers/implementers*
 - Within the project team 🙌 👤
 - Outside the project team, but within the same organization 🙌 👩
 - Outside both the project team and the organization 🙌 👨
- With different interests ➡ ensure that the project benefits everyone



Project Authority

🤔 *'What do we have to do to success?'*

- Sets the project scope
- Allocates/approves costs
- Could be one person or a group
 - Project Board
 - Project Management Board
 - Steering committee



Objectives

- *Informally*, the objective of a project can be defined by completing the statement:
 - *The project will be regarded as a success if ...*
- Rather like *post-conditions* for the project
- Focus on **what** will be put in place, rather than **how** activities will be carried out



Objectives should be SMART

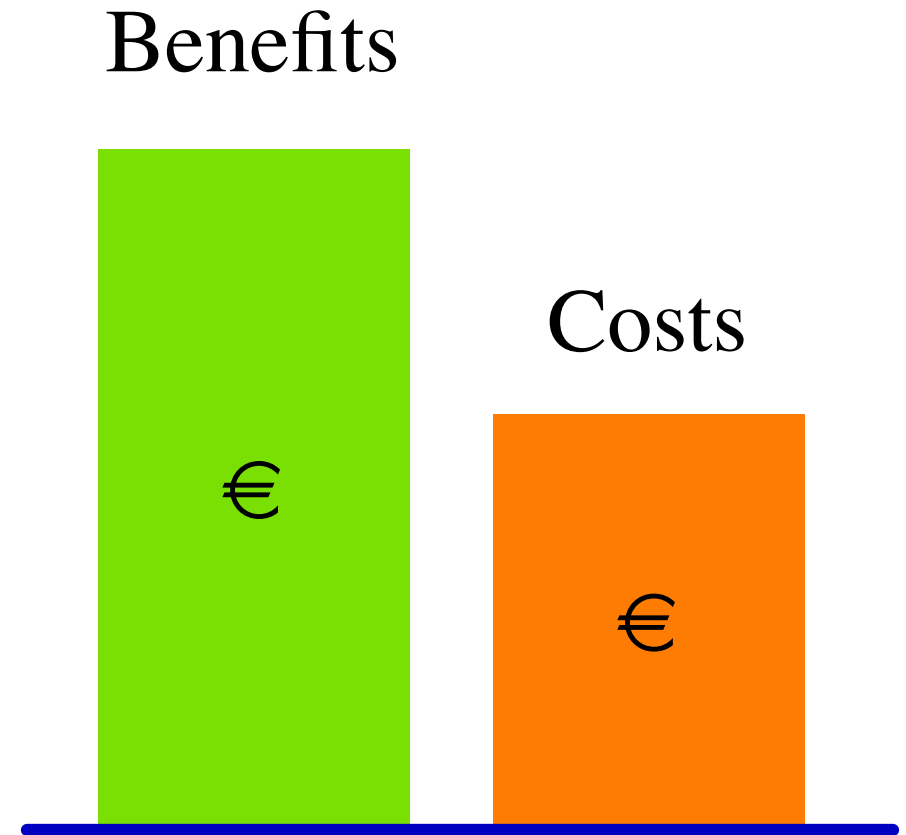
- *Specific* → concrete and well-defined
- *Measurable* → satisfaction of the objective can be objectively judged
- *Achievable* → it is within the power of the individual or group concerned to meet the target
- *Relevant* → the objective must be relevant to the true purpose of the project and resource constrained
- *Time constrained* → there is a defined point in time by which the objective should be achieved



Exercise: Appropriateness of each Objective Description?

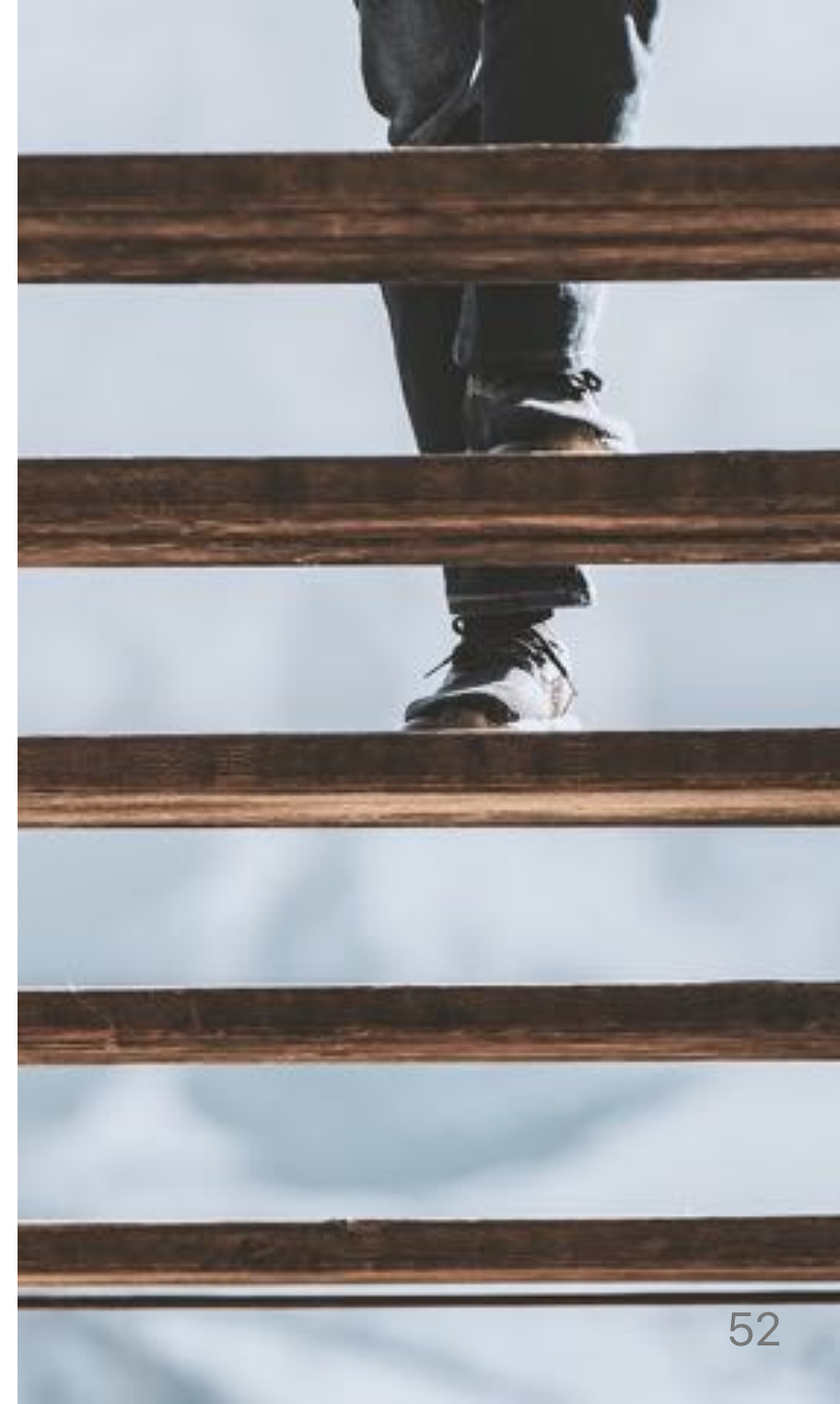
#	Objective definition
1	To implement the new application on time and within budget
2	To implement the new software application with the fewest possible software errors that might lead to operational failures
3	To design a system that is user-friendly
4	To produce full documentation for the new system

The Business Case



The Business Case (ii)

- Benefits of delivered project must outweigh costs
- Costs include:
 - Development
 - Operation
- Benefits
 - Quantifiable
 - Non-quantifiable



Project Success/Failure

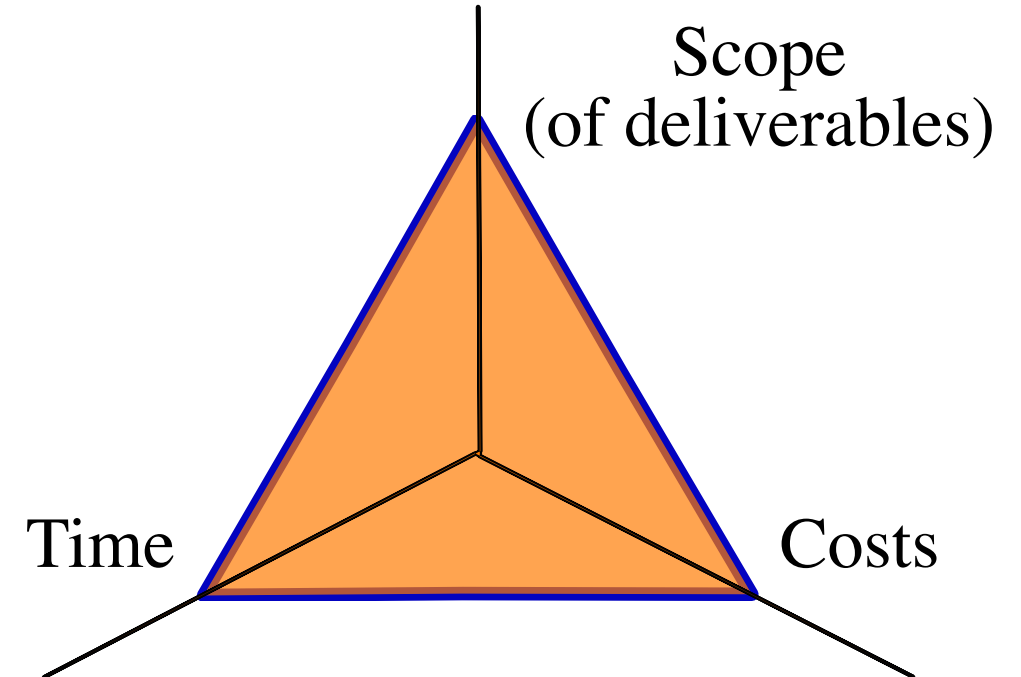
- Objectives are met
 - Time
 - Cost
 - Scope (of deliverables)
- In general if, for example, project is running out of time, this can be recovered for by reducing scope or increasing costs
- **Project objectives and Business objectives**



Project Success/Failure (ii)

- Adjust corners of the **project triangle**

👉 Project management triangle



Measures of Effectiveness

- How do we know that the goal or objective has been achieved?
- By a practical test, that can be objectively assessed.
e.g. for user satisfaction with software product:
 - Repeat business – they buy further products from us
 - Number of complaints – if low etc



Definition of Done

- In Scrum, agreed between the **Product Owner** and the **Development Team** at the beginning of the project, it can be improved during its development
- When a **product backlog** item is described as completed, all team members must understand what this means
 - thus allowing us to know where we are in the project



Other Success Criteria

- These can relate to longer term, less directly tangible assets
 - Improved skill and knowledge
 - Creation of assets that can be used on future projects e.g. software libraries
 - Improved customer relationships that lead to repeat business



What is Management?

- **Planning** - deciding what is to be done
- **Organizing** - making arrangements
- **Staffing** - selecting the right people for the job
- **Directing** - giving instructions

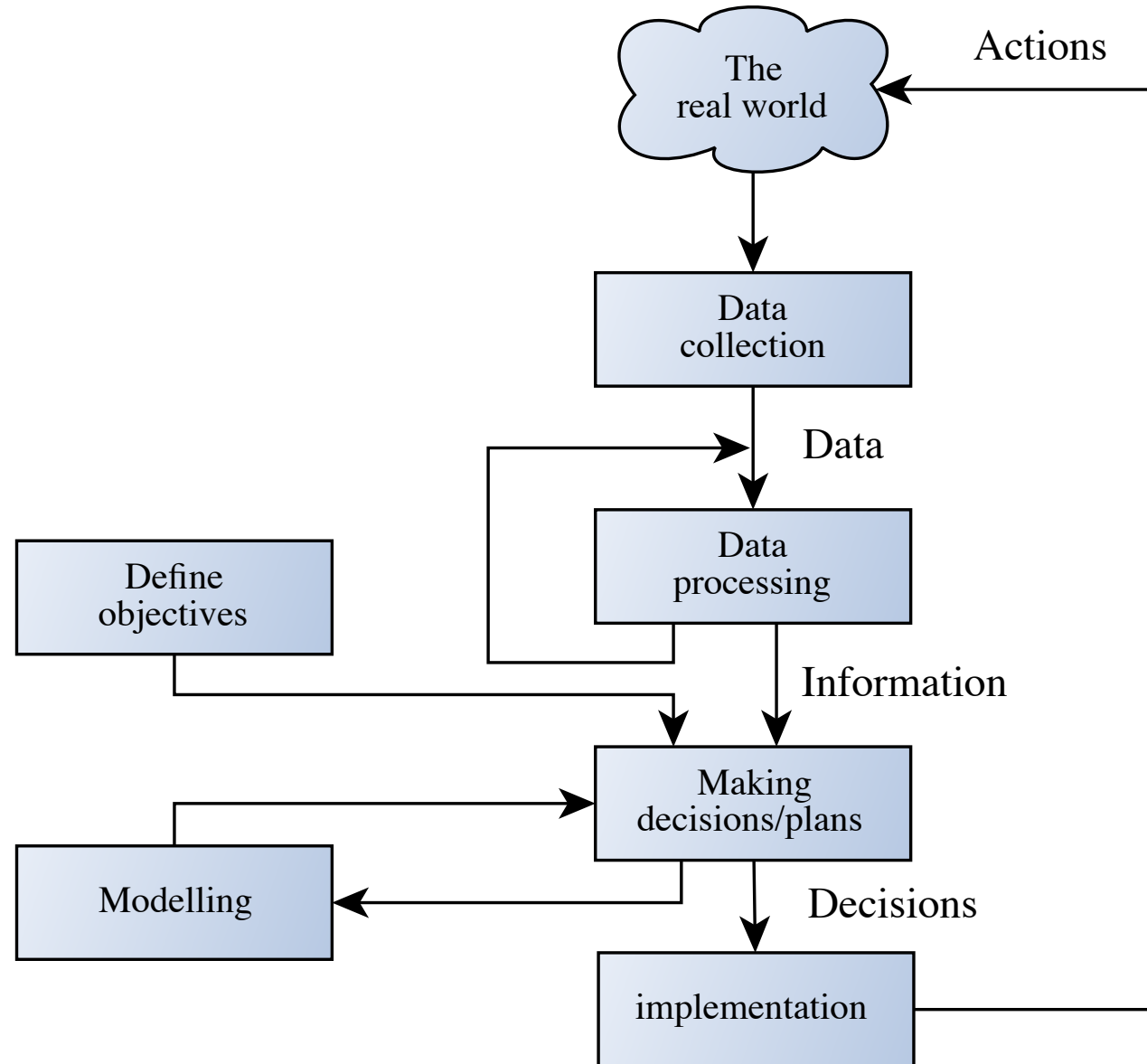


What is Management? (ii)

- **Monitoring**, checking on progress
- **Controlling**, taking action to remedy hold-ups
- **Innovating**, coming up with solutions when problems emerge
- **Representing**, liaising with clients, users, developers and other stakeholders



Management Control



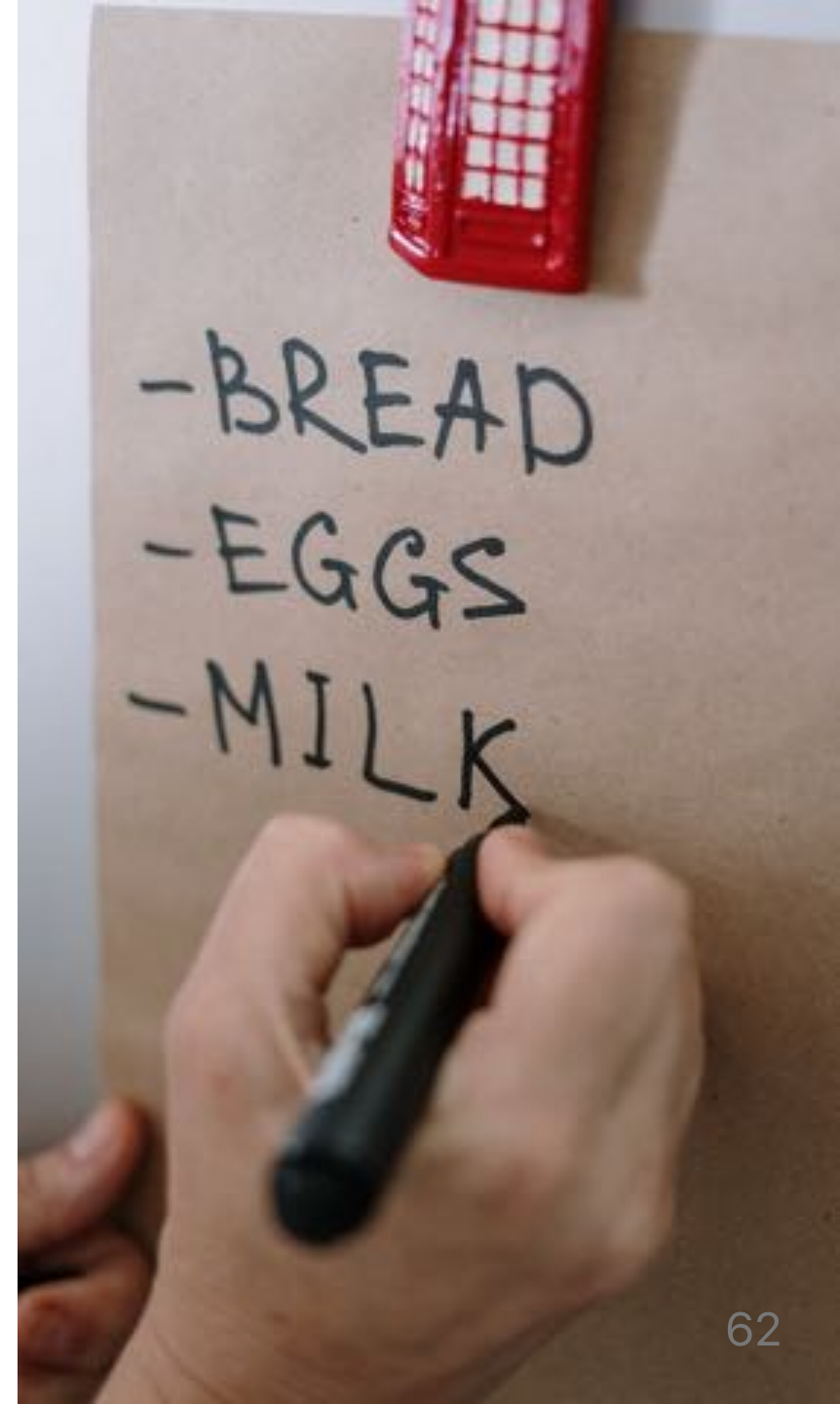
Key Points in Lecture

- Projects are non-routine - thus uncertain
- The particular problems of projects e.g. lack of visibility
- Clear objectives which can be objectively assessed are essential
- Stuff happens. Not usually possible to keep precisely plan – need for control
- Communicate, communicate, communicate!



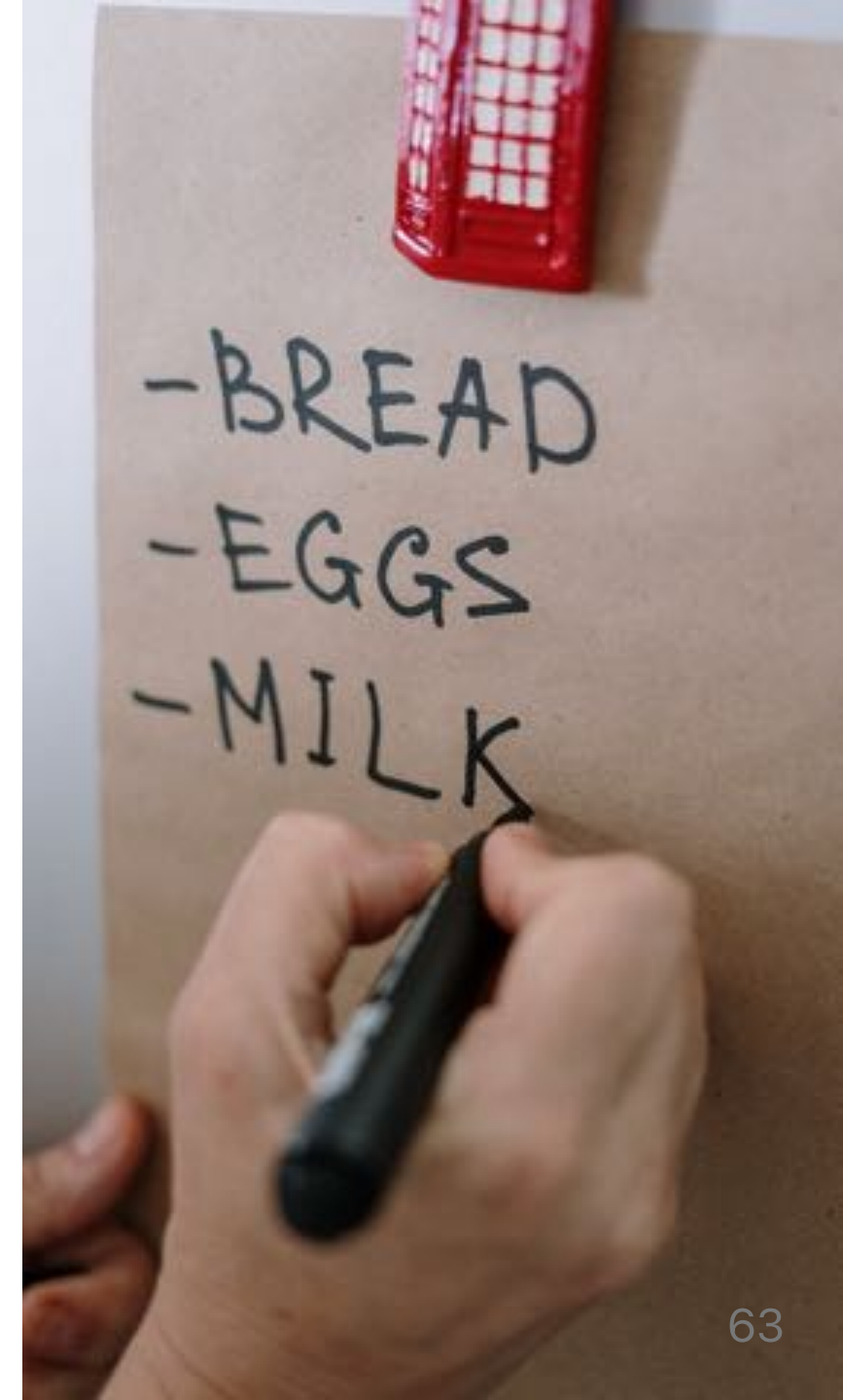
Annex 1: Content List for a Project Plan

1. Introduction
2. Background: including reference to the business case
3. Project objectives
4. Constraints (could be included with project objectives)
5. Methods
6. Project products: deliverable and intermediate products



Annex 1: Content List for a Project Plan

7. Activities to be carried out
8. Resources to be used
9. Risks to the project
10. Management of the project, including
 - Organizational responsibilities
 - Management of quality
 - Configuration management



Project Evaluation and Programme Management

Subject Two

Main Topics to be Covered

- The business case for a project
- Project portfolios
- Project evaluation
 - Cost benefit analysis
 - Cash flow forecasting
- Programme management
- Benefits management

The Business Case

- **Feasibility studies** can also act as a 'business case'
- Provides a justification for starting the project
- Should show that the benefits of the project will exceed development, implementation and operational costs
- Needs to take account of business risks



Contents of a Business Case

1. Introduction and background
2. The proposed project
3. The market
4. Organizational and operational infrastructure
5. The benefits
6. Outline implementation plan
7. Costs
8. The financial case
9. Risks
10. Management plan



Content of the Business Case

- **Introduction/background**

Describes a problem to be solved or an opportunity to be exploited

- **The proposed project**

A brief outline of the project scope

- **The market**

The project could be to develop a *new product* or a *new service capability*. The likely demand for the product or service would need to be assessed



Content of the Business Case - (ii)

- **Organizational and operational infrastructure**
How the organization would need to change. This would be important where a new information system application was being introduced
- **Benefits**
These should be express in financial terms where possible. In the end it is up to the client to assess these – as they are going to pay for the project



Content of the Business Case - (iii)

- **Outline implementation plan**
How the project is going to be implemented. Which activities can be outsourced, which are best kept in-house. Key decision points, or *Milestones*, where check on the state of implementation
- **Costs**
The implementation plan will supply information to establish these. A schedule of expected costs



Content of the Business Case - (iv)

- **Financial case**

Combines costs and benefit data to establish value of project

- **Risk**

Distinguish between *project risk* - threats to successful project execution - from *business risk* - factors threatening the benefits of the delivered project. In business case the main focus is on business risk



Project Portfolio Management

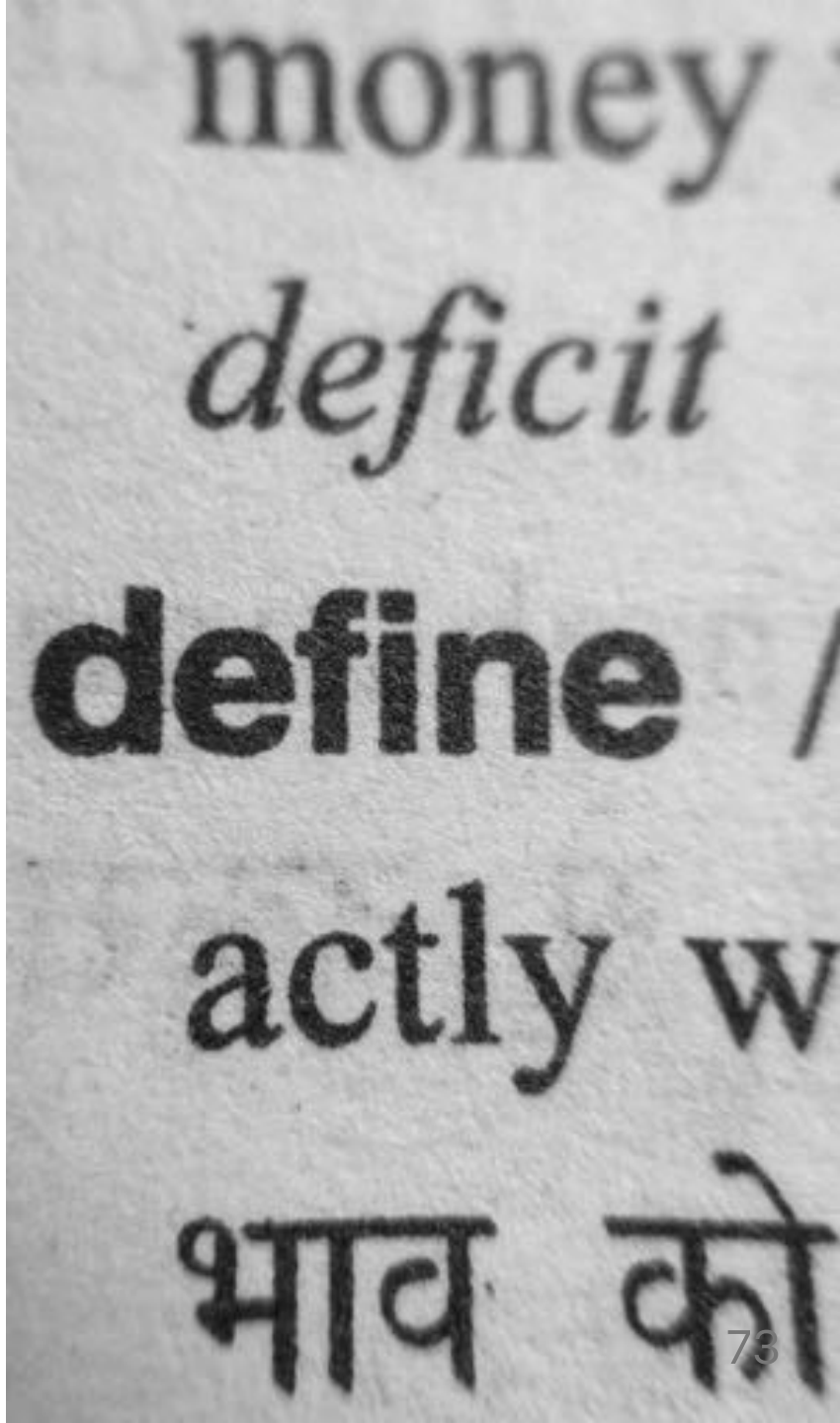
- Evaluating proposals for projects
- Assessing the risk involved with projects
- Deciding how to share resources between projects
- Taking account of dependencies between projects
- Removing duplication between projects
- Checking for gaps



Project Portfolio Management - (ii)

There are three elements to PPM:

- **Project portfolio definition**
 - Create a central record of all projects within an organization
 - Must decide whether to have ALL projects in the repository or, say, only ICT projects
 - Note difference between new product development (NPD) projects and renewal projects e.g. for process improvement



Project Portfolio Management - (iii)

- **Project portfolio management**

Actual costing and performance of projects can be recorded and assessed

- **Project portfolio optimization**

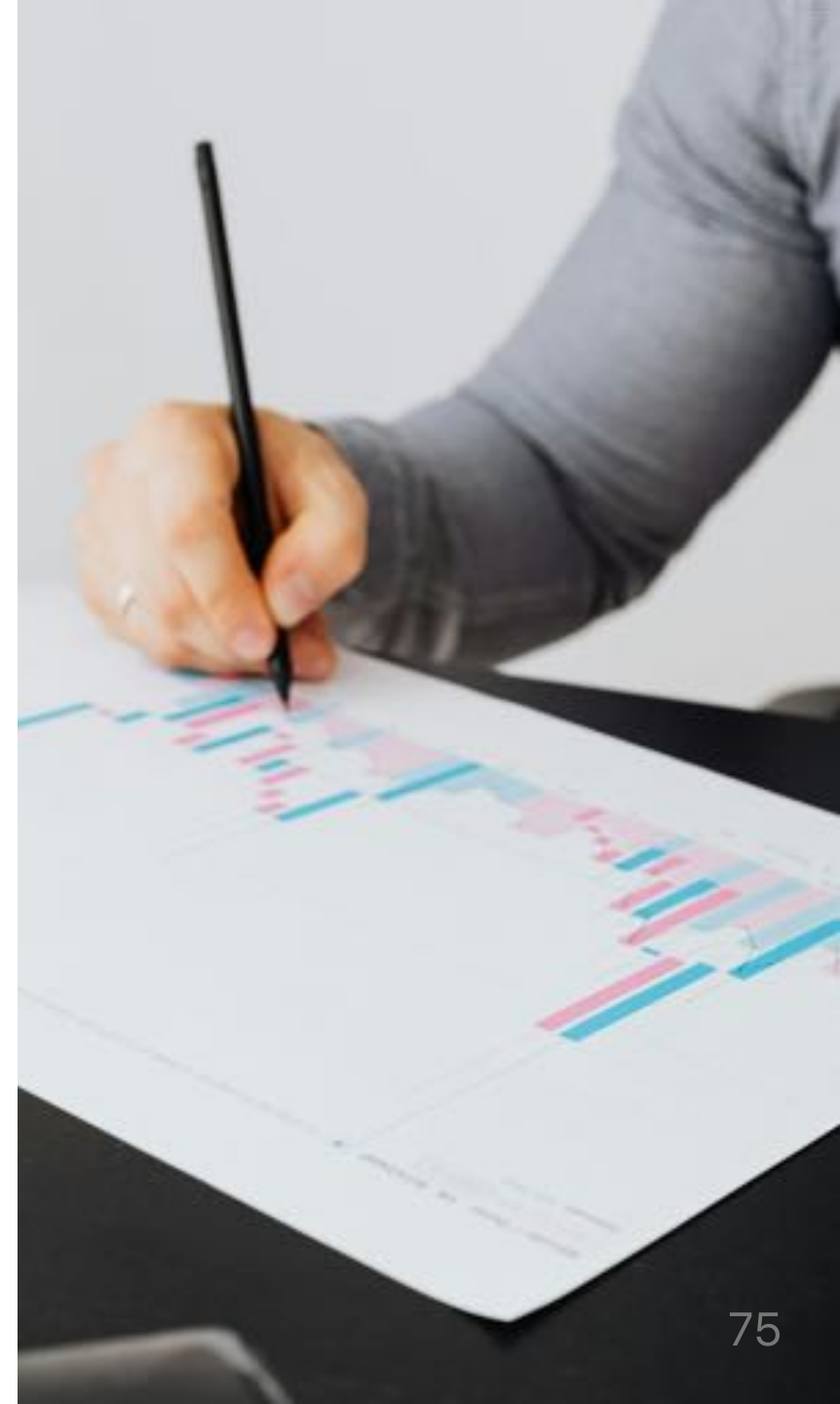
Information gathered above can be used achieve better balance of projects e.g. some that are risky but potentially very valuable balanced by less risky but less valuable projects

☞ You may want to allow some work to be done outside the portfolio e.g. quick fixes



Evaluation of Individual Projects

- Technical assessment
- Cost-benefit analysis
- Cash flow forecasting



Technical Assessment

- Evaluating whether the required functionality can be achieved with current affordable technologies
- The cost of the technology adopted must be taken into account in the cost-benefit analysis



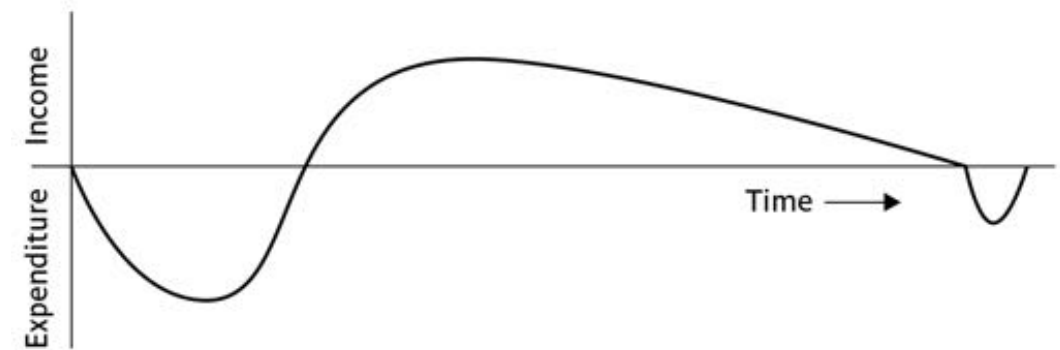
Costs-Benefit Analysis

- Identify all the costs which could be:
 - Development costs
 - Set-up
 - Operational costs
- Identify the value of benefits
- Express costs and benefits in common units - money
- Check benefits are greater than costs



Product/System Life Cycle Cash Flows

- The timing of costs and income for a product or system needs to be estimated
- The development of the project will incur costs
- When the system or product is released it will generate income that gradually pays off costs
- Some costs may relate to decommissioning – think of demolishing a nuclear power station



Cost-Benefit Evaluation Techniques

- Net profit
- Payback period
- Return of investment
- Net present value
- Internal rate return



Net Profit

Is the difference between the total costs and the total income over the live of the project

Year	Cash-flow	Year	Cash-flow
0	- 100,000	3	10,000
1	10,000	4	20,000
2	10,000	5	100,000
		Net profit	50,000



Net Profit

- '*Year 0*' represents all the costs before system is operation
- '*Cash-flow*' is value of income less outgoing
- **Net profit** value of all the cash-flows for the lifetime of the application



Four Project Cash Flow Projections

Year	Project 1	Project 2	Project 3	Project 4
0	- 100K	-1,000K	-100K	-120K
1	10K	200K	30K	30K
2	10K	200K	30K	30K
3	10K	200K	30K	30K
4	20K	200K	30K	30K
5	100K	300K	30K	75K
Net Profit	50K	100K	50K	75K



Pay Back Period

- Time it takes to start generating of income over outgoings

Year	Cash-flow	Accumulated
0	- 100,000	- 100,000
1	10,000	- 90,000
2	10,000	- 80,000
3	10,000	- 70,000
4	20,000	- 50,000
5	100,000	50,000



Pay Back Period

- Last year in which the accumulated cash flow was negative + (absolute accumulated cash flow at the end of that year / cash-flow for the next year)

👉 In the previous example:

$$4 + \frac{|-50,000|}{100,000} = 4.5 \text{ years}$$

Exercise

Consider the four projects cash flow given and calculate the payback period for each of them



Return On Investment (ROI)

- Provides a way of comparing the net profitability to the investment required

$$ROI = \frac{\textit{Average annual profit}}{\textit{Total investment}} \times 100$$

👉 In the previous example

- *Average annual profit* =
 $50,000/5 = 10,000$
- $ROI = 10,000/100,000 \times 100 = 10\%$



Exercise

Consider the four projects cash flow given and calculate the return on investment for each of them



Net Present Value

- Would you rather I gave you 100€ today or in 12 months time? 🤔
 - If I gave you 100€ now you *could* put it in savings account and get interest on it
 - If the interest rate was 10% how much would I have to invest now to get 100€ in a year's time?
 - This figure is the **net present value** of 100€ in one year's time

FUTURE

Discount Factor

- Discount factor = $1 / (1 + r)^t$
 - r the interest rate (e.g. 10% is 0.10)
 - t is the number of years

👉 In the case of 10% rate and one year

$$\text{Discount factor} = 1 / (1 + 0.10)^1 = 0.9091$$

👉 In the case of 10% rate and two years

$$\text{Discount factor} = 1 / (1 + 0.10)^2 = 0.8294$$



Applying Discount Factors

Year	Cash-flow	Discount Factor	Discounted cash-flow
0	-100,000	1.0000	-100,000
1	10,000	0.9091	9,091
2	10,000	0.8264	8,264
3	10,000	0.7513	7,513
4	20,000	0.6830	13,660
5	100,000	0.6209	62,090
		NPV	618

Internal Rate of Return

- Internal rate of return (IRR) is the discount rate that would produce an NPV of 0 for the project
- Can be used to compare different investment opportunities



Dealing with Uncertainty: Risk Evaluation

- Project A might appear to give a better return than B but could be riskier
- Could draw up draw a project risk matrix for each project to assess risks – see next overhead
- For riskier projects could use higher discount rates
 - 🖱️ Add 2% for a reasonably safe project 🌤️
 - 🖱️ Add 5% for a fairly risky one ⚡



Programme Management

One definition ¹

A group of projects that are managed in a coordinated way to gain benefits that would not be possible were the projects to be managed independently

¹Ferns. International Journal of Project Management.
August 1991



Programmes May Be

- Strategic
- Business cycle programmes
- Infrastructure programmes
- Research and development programmes
- Innovative partnerships



Programme Managers vs Project Managers

Programme manager

- Many simultaneous projects
- Personal relationship with skilled resources
- Optimization of resource use
- Projects tend to be seen as similar

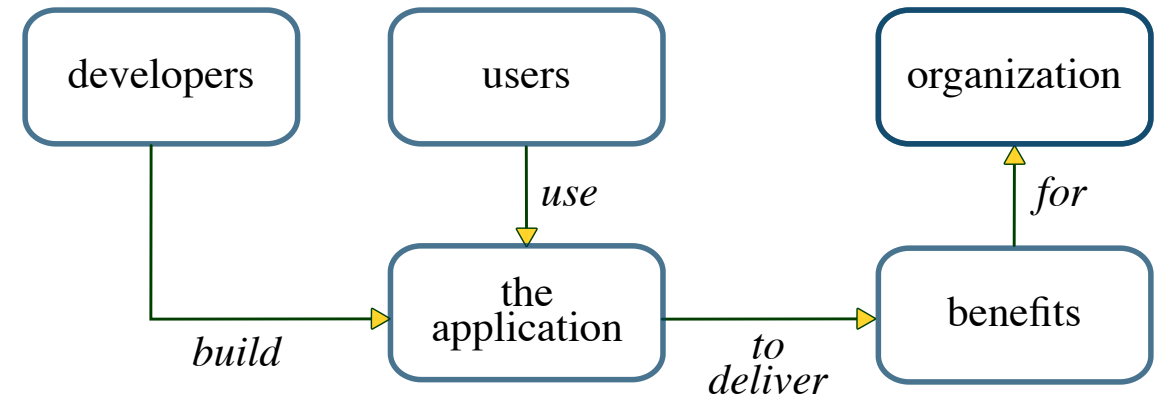
Project manager

- One project at a time
- Impersonal relationship with resources
- Minimization of demand for resources
- Projects tend to be seen as unique



Benefits Management

- Providing an organization with a capability does not guarantee that this will provide benefits envisaged – need for *benefits management*
- This has to be outside the project – project will have been completed
- Therefore done at *programme level*



Benefits Management (ii)

To carry this out, you must:

- Define expected benefits
- Analyse balance between costs and benefits
- Plan how benefits will be achieved
- Allocate responsibilities for their achievement
- Monitor achievement of benefits



Benefits

These might include:

- Mandatory requirement
- Improved quality of service
- Increased productivity
- More motivated workforce
- Internal management benefits



Benefits - (ii)

- Risk reduction
- Economies
- Revenue enhancement/acceleration
- Strategic fit



Quantifying Benefits

Benefits can be:

- Quantified and valued e.g. a reduction of x staff saving y €
- Quantified but not valued e.g. a decrease in customer complaints by x %
- Identified but not easily quantified – e.g. public approval for a organization in the locality where it is based



Remember!

- A project may fail not through poor management but because it should never have been started
- A project may make a profit, but it may be possible to do something else that makes even more profit
- A real problem is that it is often not possible to express benefits in accurate financial terms
- Projects with the highest potential returns are often the most risky

SUMMARY

Step Wise: An Approach to Planning Software Projects

Subject Three

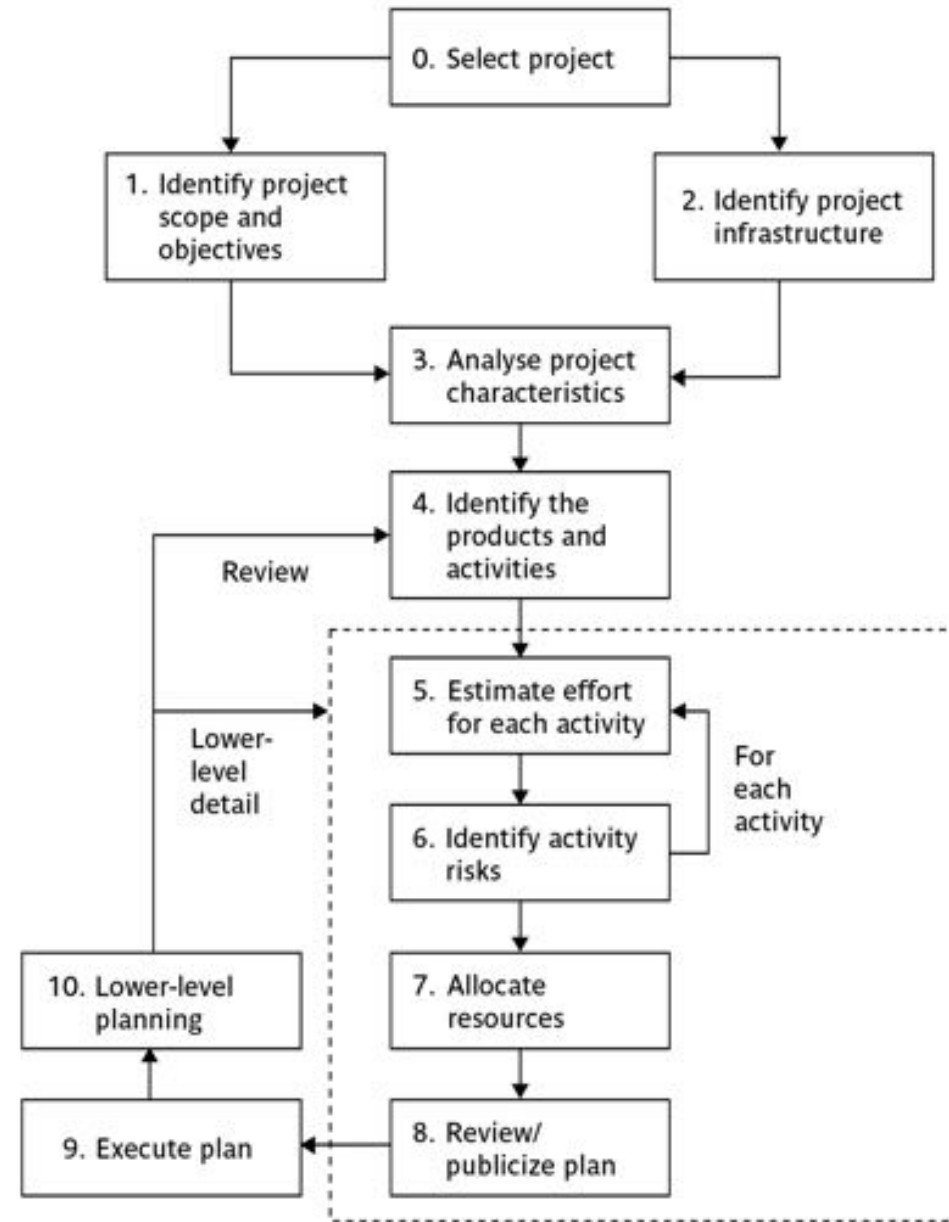


'Step Wise' - Aspirations

- Practicality
 - Tries to answer the question 'what do I do now?'
- Scalability
 - Useful for small project as well as large
- Range of application
- Accepted techniques
 - e.g. borrowed from PRINCE, etc



'Step Wise' - An Overview



Step 1: Establish Project Scope and Objectives

1.1 Identify objectives and measures of effectiveness

How do we know if we have succeeded?

1.2 Establish a project authority

Who is the boss?

1.3 Identify all stakeholders in the project and their interests

Who will be affected/involved in the project?



Step 1: Establish Project Scope and Objectives

1.4 Modify objectives in the light of stakeholder analysis

Do we need to do things to win over stakeholders?

1.5 Establish methods of communication with all parties

How do we keep in contact?



Step 2: Establish Project Infrastructure

2.1 Establish link between project and any strategic plan

Why did they want the project?

2.2 Identify installation standards and procedures

What standards do we have to follow?

2.3. Identify project team organization

Where do I fit in?



Step 3: Analysis of Project Characteristics

3.1 Distinguish the project as either objective or product-based

Is there more than one way of achieving success?

3.2 Analyse other project characteristics (including quality based ones)

What is different about this project?



Step 3: Analysis of Project Characteristics

3.3 Identify high level project risks

What could go wrong?

What can we do to stop it?

3.4 Take into account user requirements concerning implementation

3.5 Select general life cycle approach

Waterfall? Increments? Prototypes?

3.6 Review overall resource estimates

Does all this increase the cost?

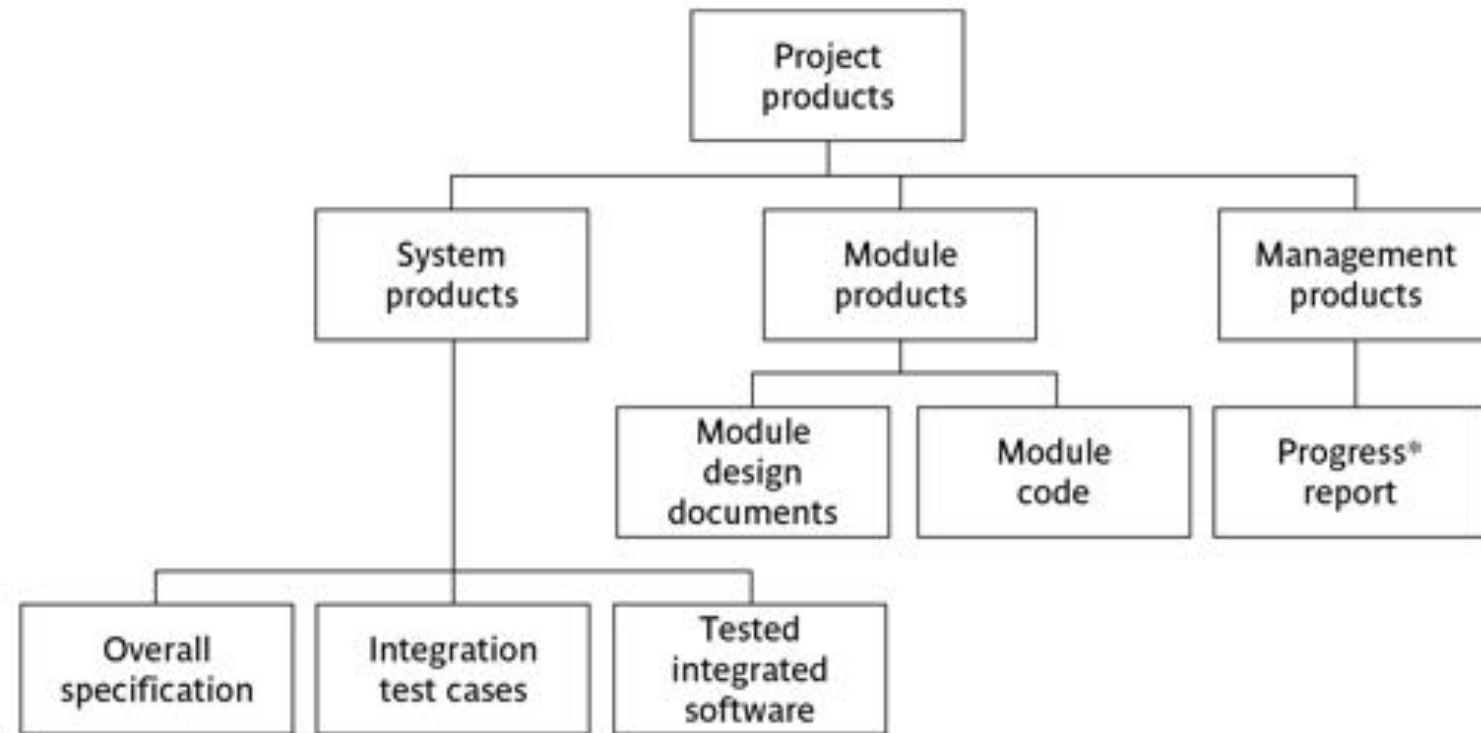


Step 4: Identify Project Products and Activities

4.1 Identify and describe project products

- What do we have to produce?*

👉 *Product Breakdown Structure, PBS*



Products

- The result of an activity
- Could be, among other things:
 - physical thing (*installed pc*)
 - a document (*logical data structure*)
 - a person (*trained user*)
 - a new version of an old product (*updated software*)



Products (ii)

- The following are NOT normally products:
 - activities (e.g. *training*)
 - events (e.g. *interviews completed*)
 - resources and actors (e.g. *software developer*)
 - may be exceptions to this
- Products CAN BE **deliverable** or **intermediate**



Product Description (PD)

- Product identity
- Description - *what is it?*
- Derivation - *what is it based on?*
- Composition - *what does it contain?*
- Form of the product
- Relevant standards
- Quality criteria



Step 4: Identify Project Products and Activities

4.2 Document generic product flows

👉 *Product Flow Diagram, PFD*

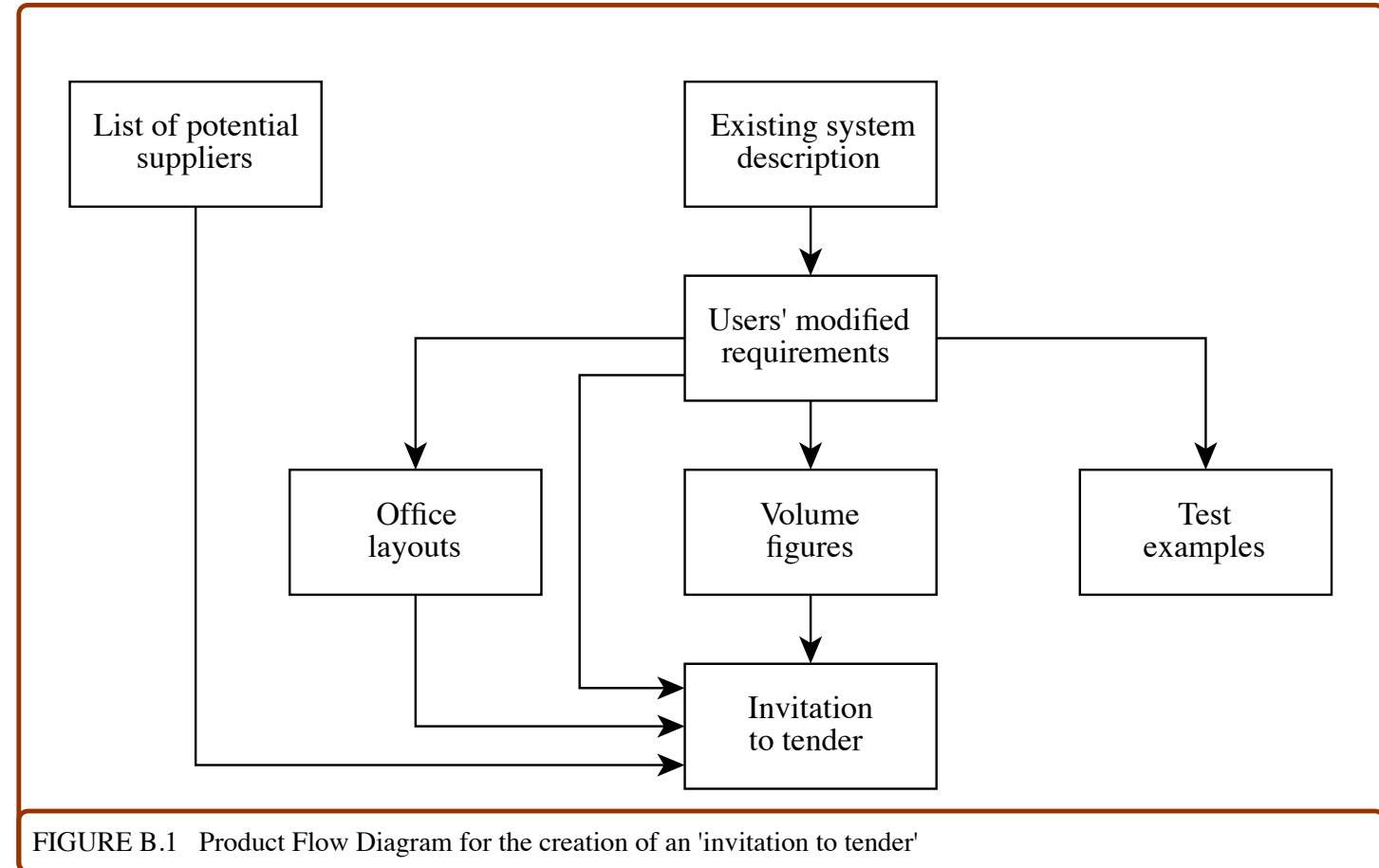


FIGURE B.1 Product Flow Diagram for the creation of an 'invitation to tender'

Step 4.3: Recognize Product Instances

- The PBS and PFD will probably have identified generic products (e.g. *software modules*)
- It might be possible to identify specific instances (e.g. *module A, module B ...*)
- But in many cases this will have to be left to later, more detailed, planning

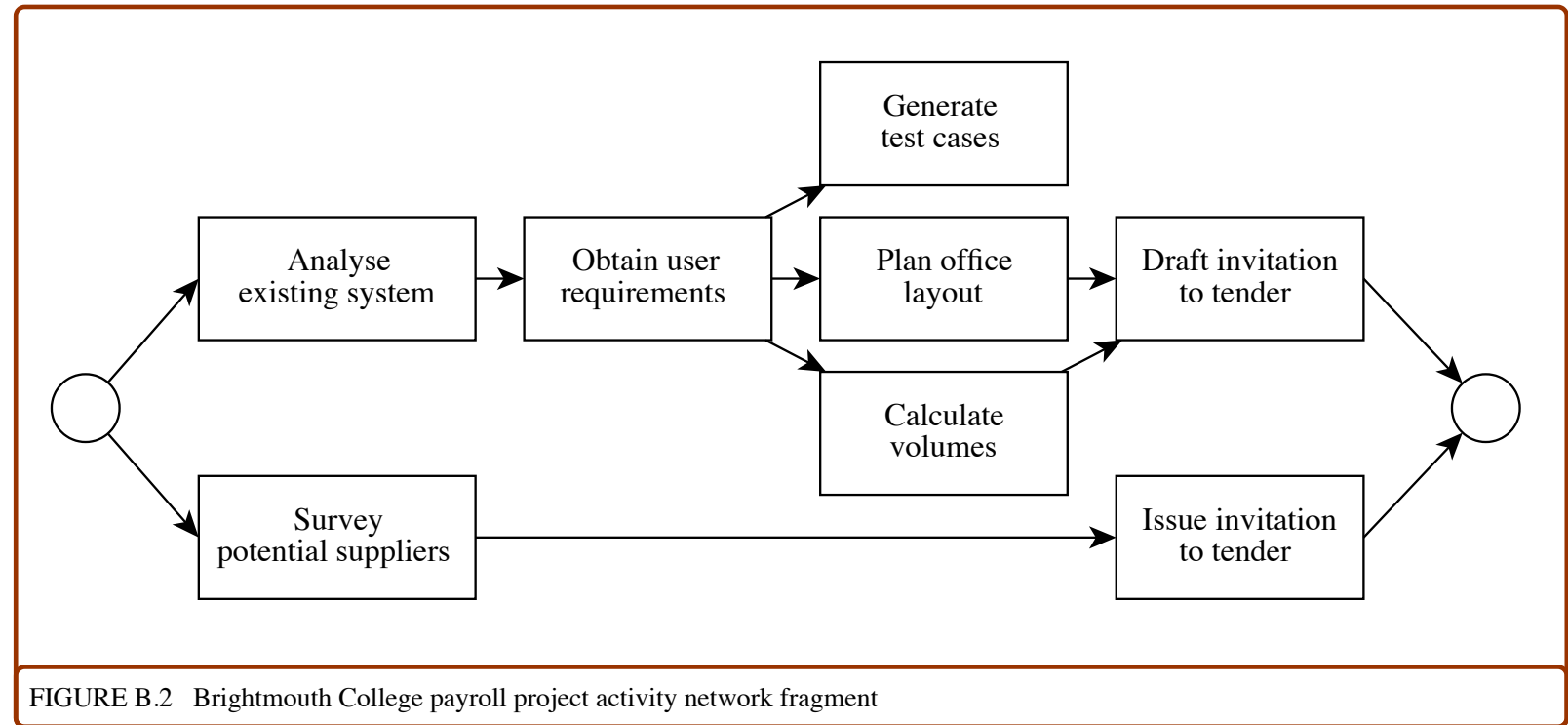


Step 4.4: Produce Ideal Activity Network

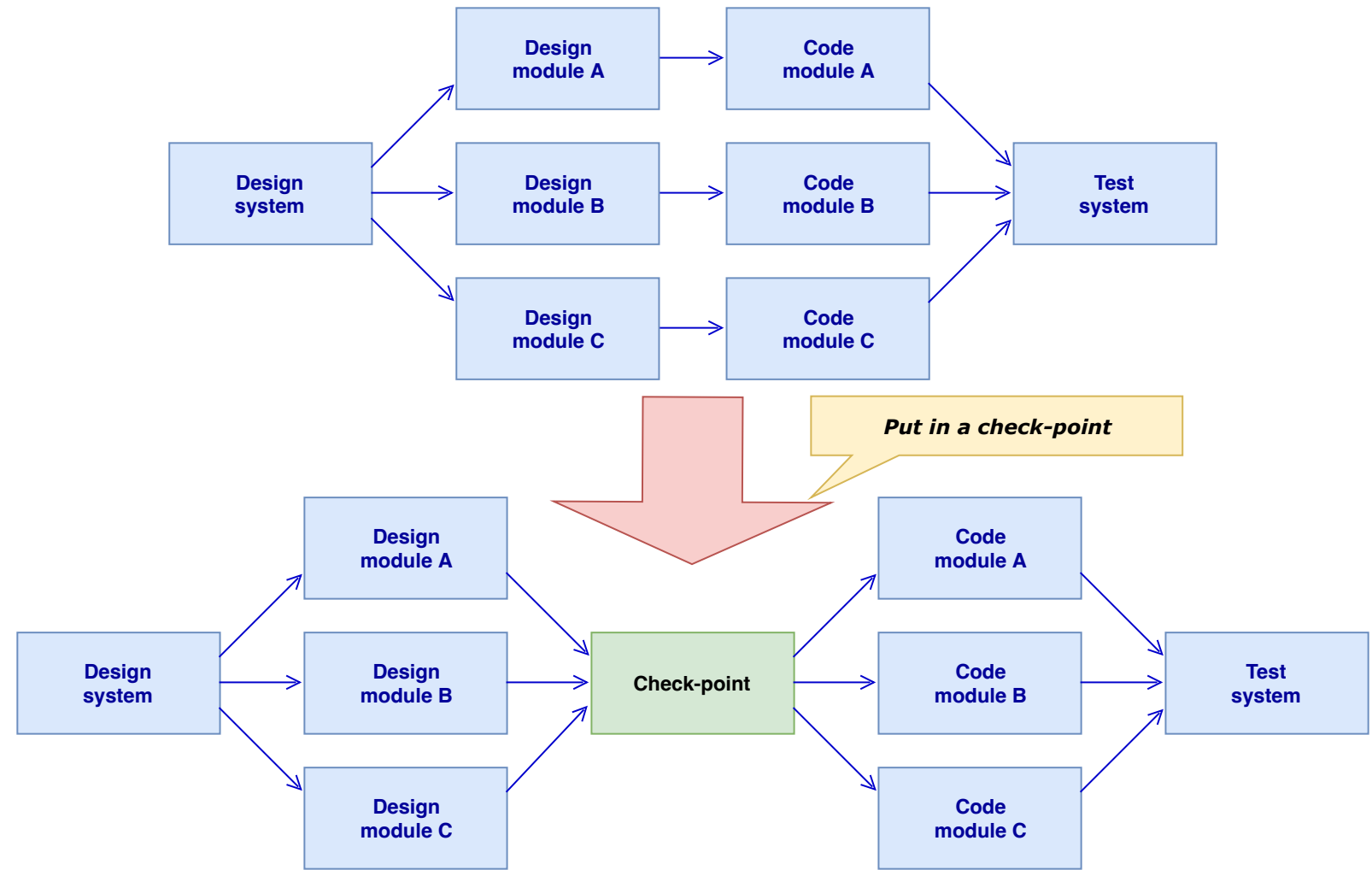
- Identify the activities needed to create each product in the PFD
- More than one activity might be needed to create a single product
- Hint: Identify activities by verb + noun but avoid 'produce...' (too vague)
- Draw up activity network



An 'Ideal' Activity



Step 4.5: Add Check- Points if Needed



Step 5: Estimate Effort for Each Activity

5.1 Carry out bottom-up estimates

- Distinguish carefully between **effort** and **elapsed time**

5.2. Revise plan to create controllable activities

- Break up very long activities into a series of smaller ones
- Bundle up very short activities (create check lists?)



Step 6: Identify Activity Risks

6.1 Identify and quantify risks for activities

- Damage if risk occurs (measure in time lost or money)
- Likelihood if risk occurring

6.2 Plan risk reduction and contingency measures

- Risk reduction: activity to stop risk occurring
- Contingency: action if risk does occur



Step 6: Identify Activity Risks

6.3 Adjust overall plans and estimates to take account of risks

- Adding new activities which reduce risks associated with other activities e.g. training, pilot trials, information gathering



Step 7: Allocate Resources

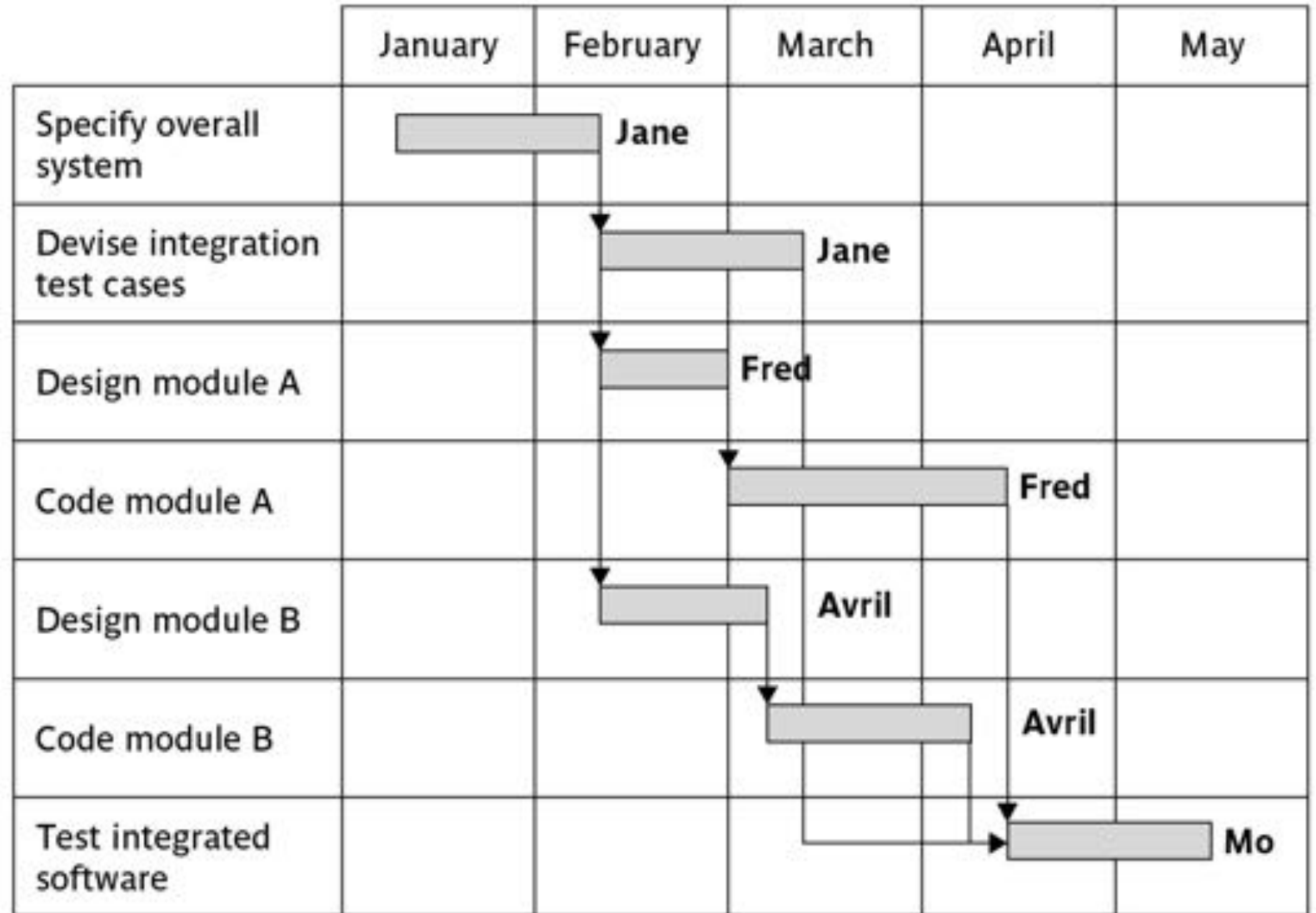
7.1 Identify and allocate resources to activities

7.2 Revise plans and estimates to take into account resource constraints

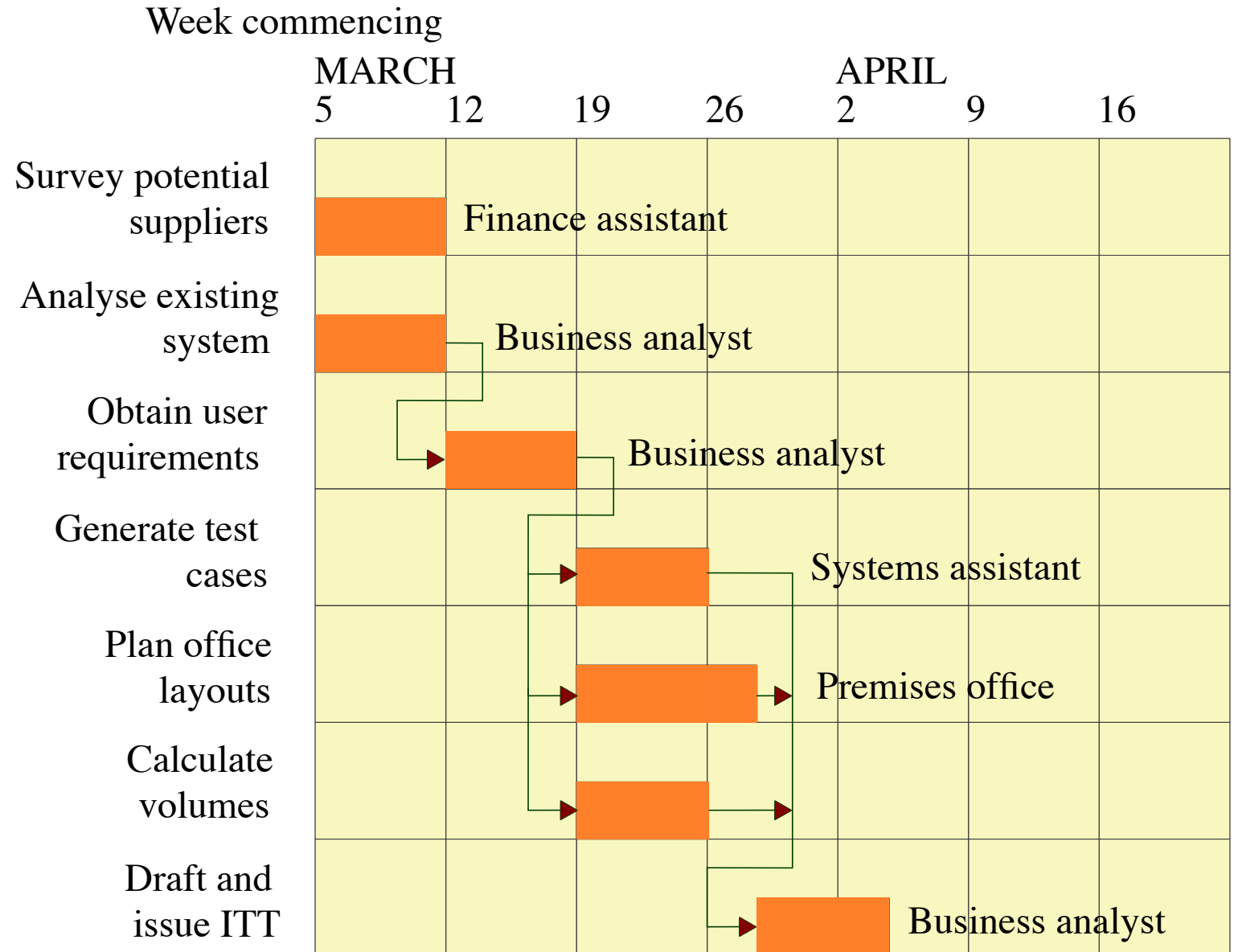
- e.g. staff not being available until a later date
- non-project activities



Gantt Charts



Gantt Charts



Step 8: Review/Publicize Plan

8.1 Review quality aspects of project plan

8.2 Document plan and obtain agreement



Step 9 and 10: Execute Plan and Create Lower Level Plans

9 Execute plan

10 Lower level plans

- Detailed planning of the later stages will need to be delayed because more information will be available nearer the start of the stage



Key Points

- Establish your objectives
- Think about the characteristics of the project
- Discover/set up the infrastructure to support the project (including standards)
- Identify **products** to be created and the **activities** that will create them
- Allocate resources
- Set up quality processes

SUMMARY

Selection of an Appropriate Project Approach

Subject Four



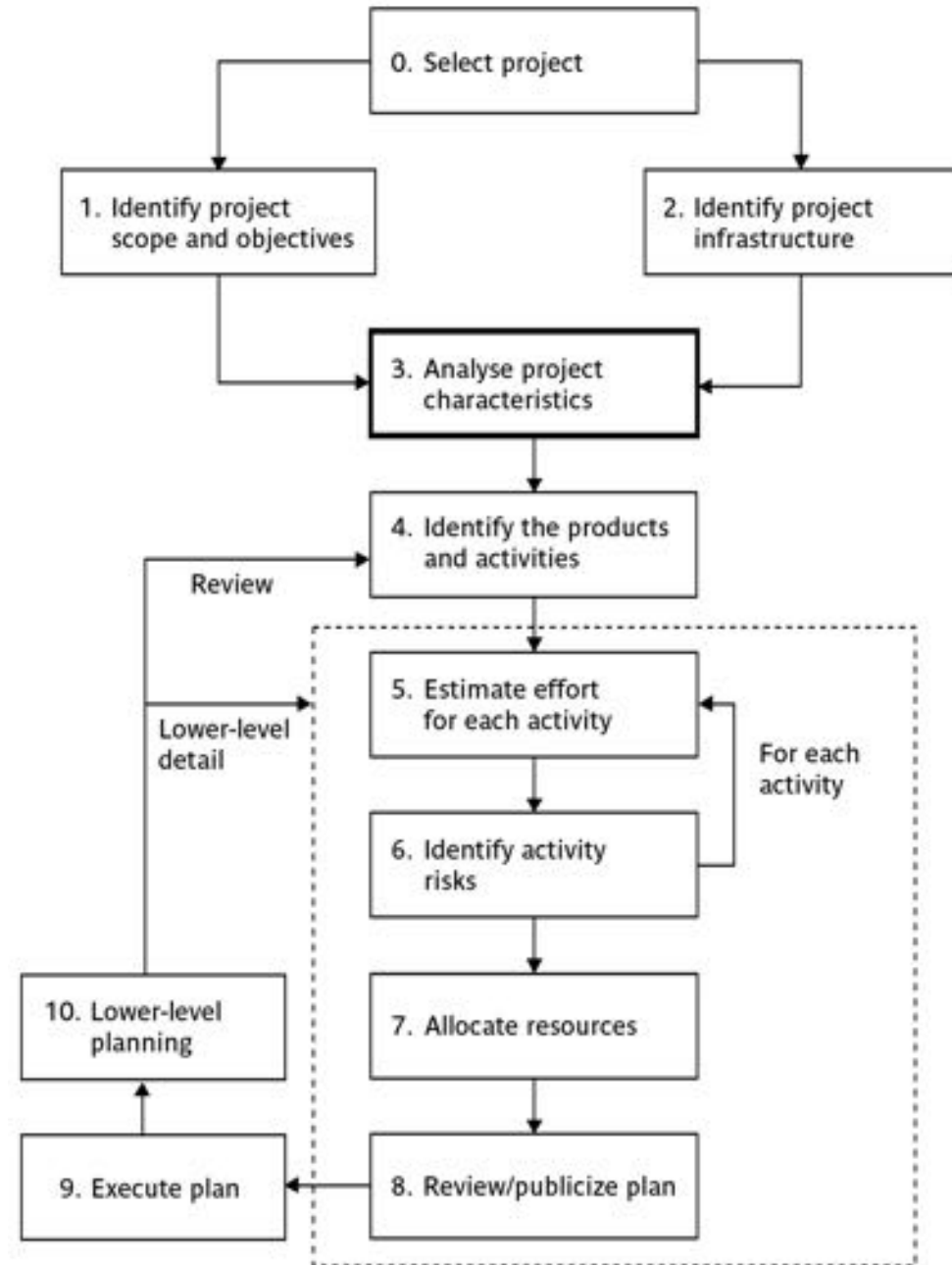
Outline of Lecture

- Building versus buying software
- Taking account of the characteristics of the project
- Process models
 - Waterfall
 - Prototyping and iterative approaches
 - Incremental delivery
 - Agile approaches

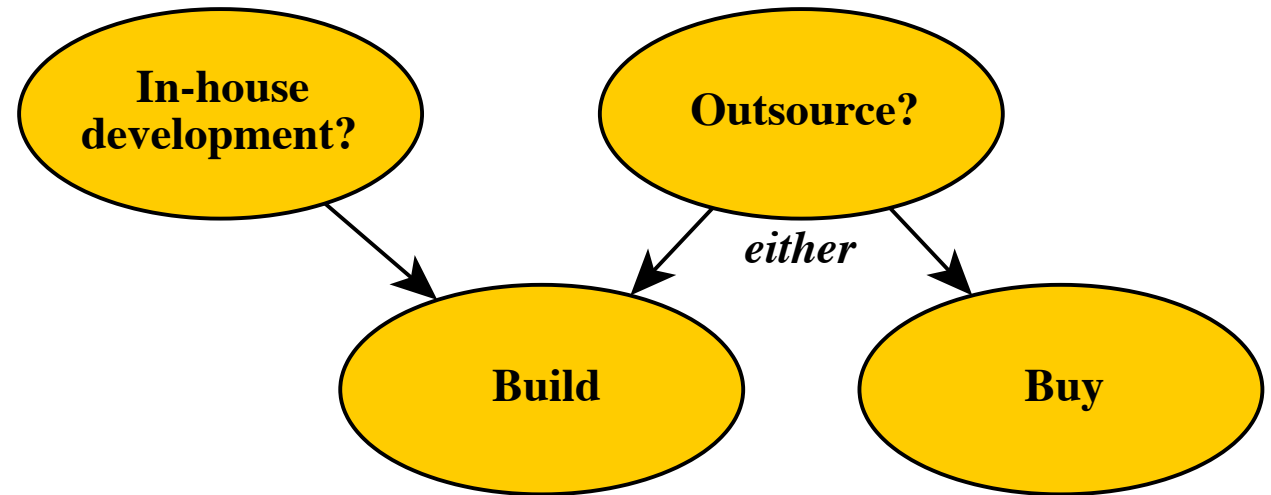
Selection of Project Approaches

- This lecture concerned with choosing the right approach to a particular project: variously called *technical planning, project analysis, methods engineering* and *methods tailoring*
- In-house: often the methods to be used dictated by organizational standards
- Suppliers: need for tailoring as different customers have different needs

Project Analysis in Step Wise



Build or Buy ?



Some Advantages 🥰 of Off-The-Shelf (OTS) Software

- Cheaper as supplier can spread development costs over a large number of customers
- Software already exists
- Can be trialed by potential customer
- No delay while software being developed
- Where there have been existing users, bugs are likely to have been found and eradicated







Some Possible Disadvantages 🤪 of OTS Software

- Customer will have same application as everyone else
- No competitive advantage, *but* competitive advantage may come from the *way* application is used
- Customer may need to change the way they work in order to fit in with OTS application
- Customer does not own the code and cannot change it
- Danger of over-reliance on a single supplier







General Approach

- Look at risks and uncertainties 
 - Are requirements well understood?
 - Are technologies to be used well understood?
- Look at the type of application being built 
 - Information system? Embedded system?
 - Criticality? Differences between target and development environments?
- Clients' own requirements  
 - Need to use a particular method



Choice of Process Models

- Waterfall 
- Incremental delivery 
- Evolutionary development 
- Agile methods 



Structure vs Speed of Delivery

Structured Approach

- Also called 'heavyweight' approaches 🏆
- Step-by-step methods where each step and intermediate product is carefully defined
- Emphasis on getting quality right first time
 - UML and USDP
- Future vision: Model-Driven Architecture (MDA)

Structure vs Speed of Delivery

Agile Methods 🧑🏻

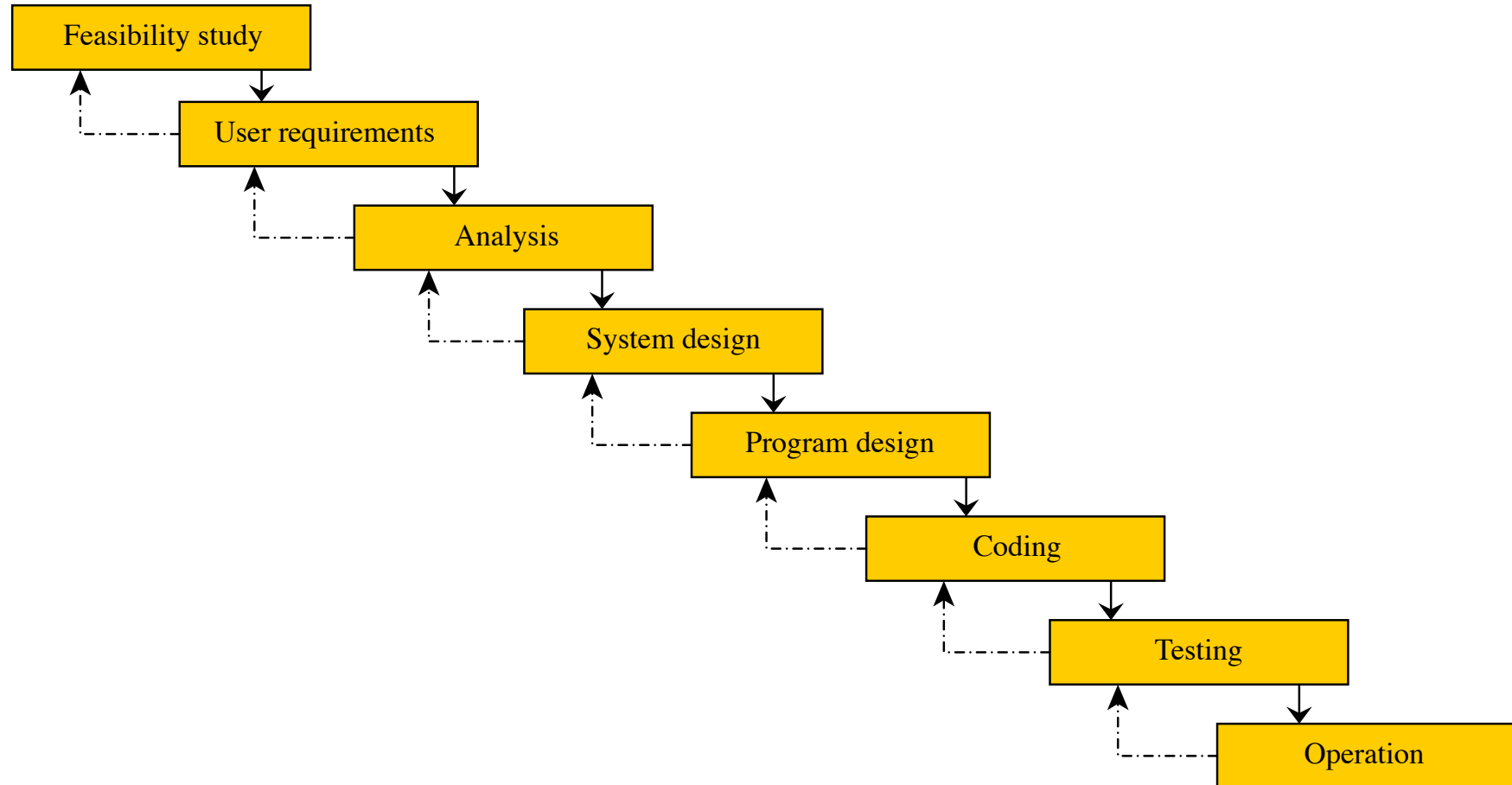
- Emphasis on speed of delivery rather than documentation
- **RAD**, Rapid Application Development
 - Emphasized use of quickly developed prototypes
- **JAD**, Joint Application Development
 - Requirements are identified and agreed in intensive workshops with users

A way of speeding up delivery is simply to deliver

less 🧑🏻



Waterfall

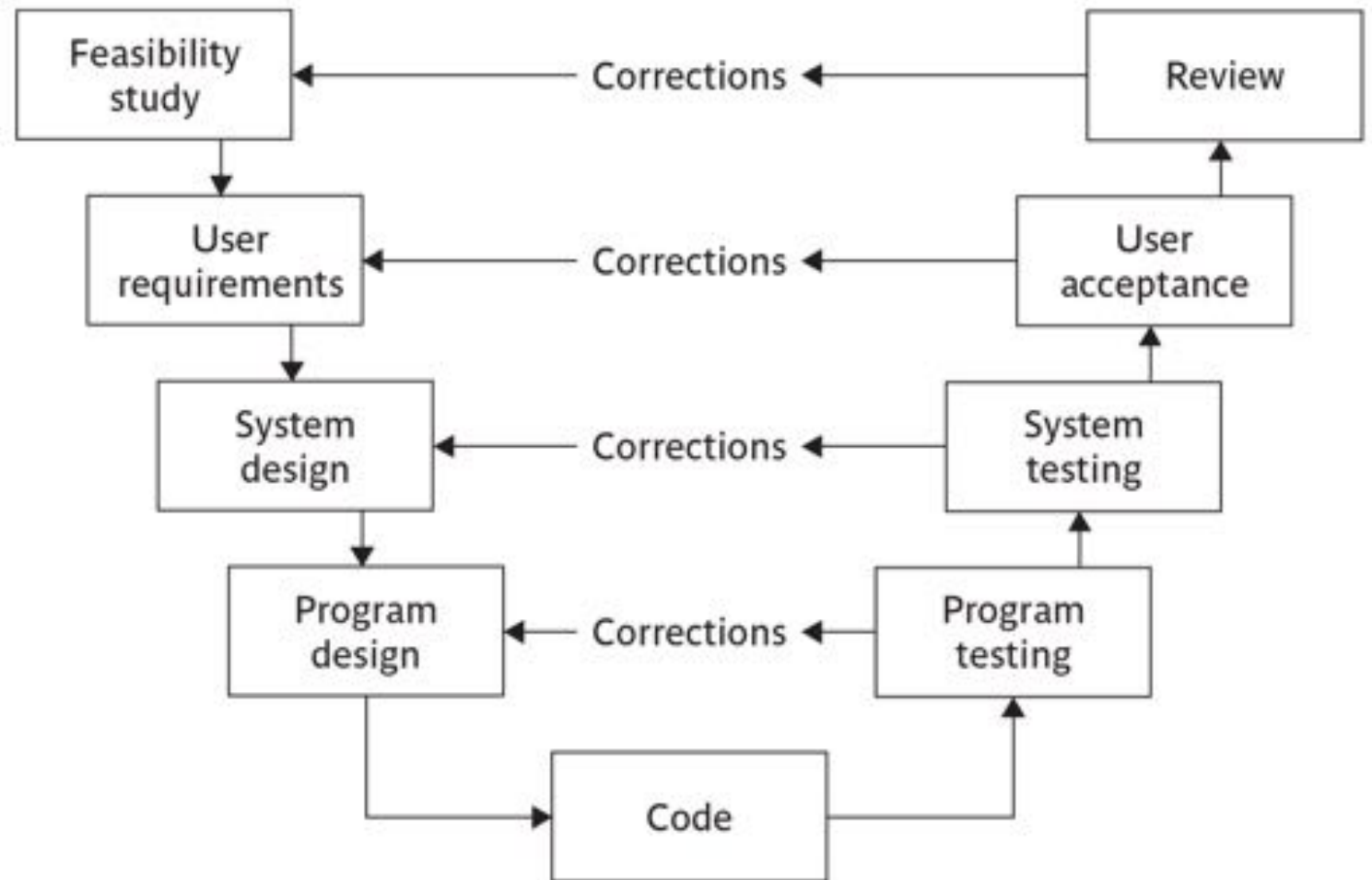


Waterfall

- The 'classical' model
- Imposes structure on the project
- Every stage needs to be checked and signed off
- Limited scope for iteration
- **V** model approach is an extension of waterfall where different testing phases are identified which check the quality of different development phases



Testing: V- Process Model



Evolutionary Delivery: Prototyping

Sprague and McNurlin

'An iterative process of creating quickly and inexpensively live and working models to test out requirements and assumptions'

- We can buy knowledge and reduce uncertainty
 - 'Throw away' prototypes
 - Evolutionary prototypes



Reasons for Prototyping

- Learning by doing
- Improved communication
- Improved user involvement
- Reduces maintenance costs
- Prototype can be used for producing expected results



Prototyping, some Dangers 🦁

- Users may misunderstand the role of the prototype
- Lack of project control and standards possible
- Additional expense of building prototype
- Focus on user-friendly interface could be at expense of machine efficiency



Other Ways of Categorizing Prototyping

- What is being learnt?
 - Organizational prototype
 - Hardware/software prototype ('experimental')
 - Application prototype ('exploratory')
- To what extent?
 - Mock-ups
 - Simulated interaction
 - Partial working models: vertical versus horizontal



What Is Being Prototyped?

- Human-computer interface
 - The physical vehicle for the prototype should be as similar as possible to the operational system
- Functionality
 - The precise way the system should function internally is not known (real world simulation)
 - The algorithms might need to be adjusted repeatedly



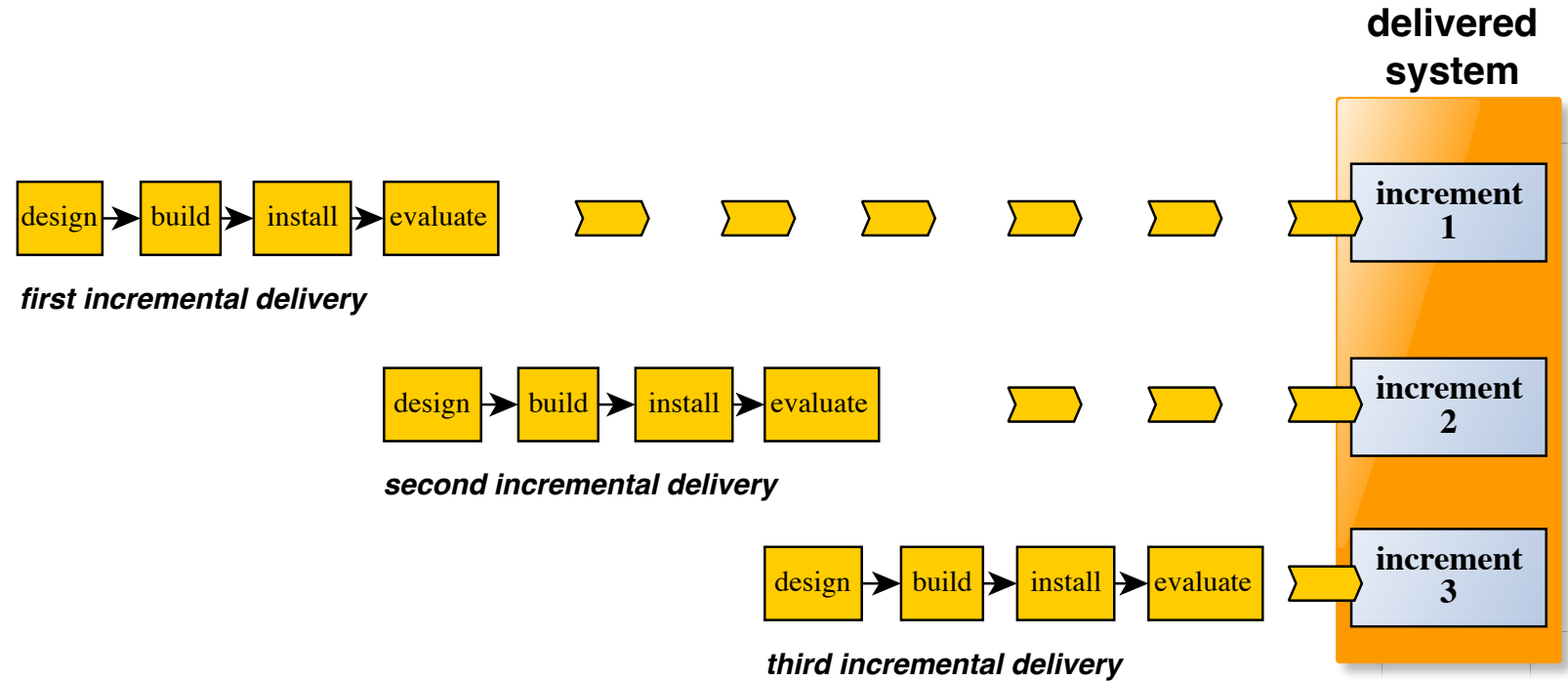
Controlling Changes during Prototyping

🤔 Follow all user suggestions?

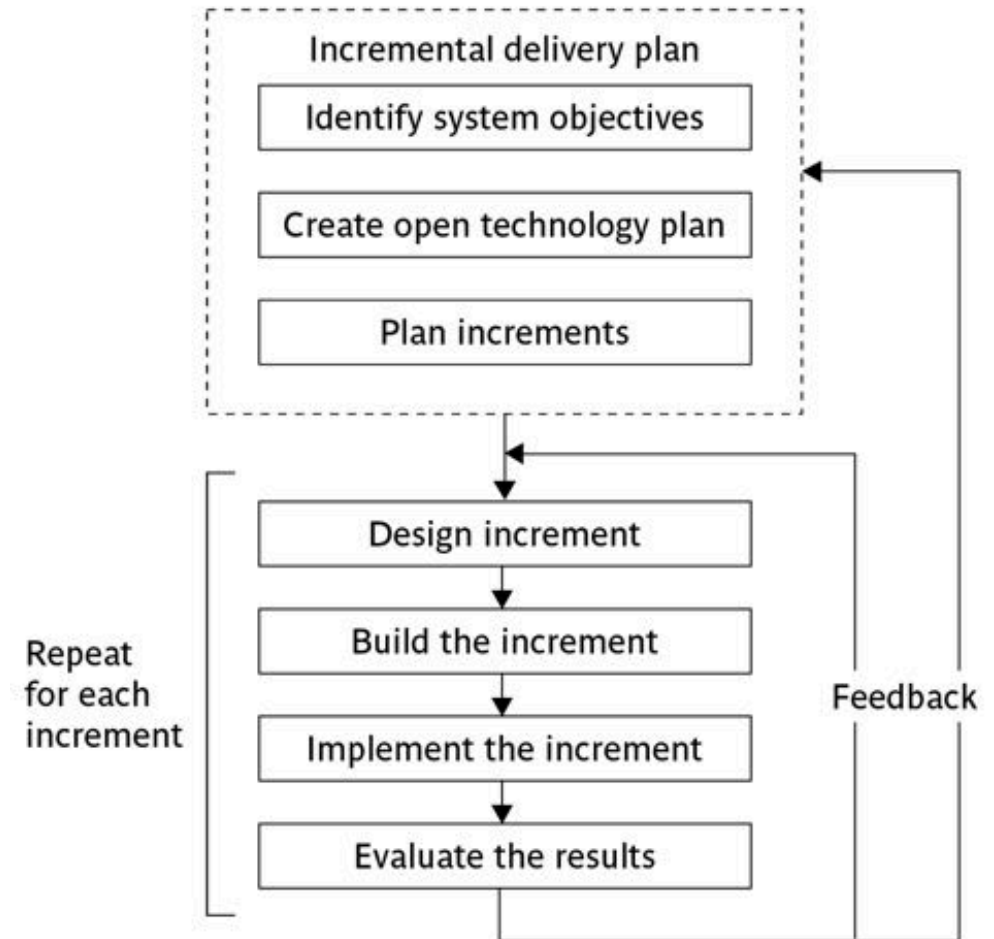
- 🏢 **Cosmetic** ($\approx 35\%$)
 - Simple changes to the layout of the screen or reports
- 🌴 **Local** ($\approx 60\%$)
 - The way that a screen or report is processed, but does not affect other parts
 - Backed-up, inspected retrospectively
- 🌐 **Global** ($\approx 5\%$)
 - Affects more than one part of the system
 - Must be subject of a design review before



Incremental Delivery



The Incremental Process



Incremental Approach

- Benefits 🍰
 - Feedback from early stages used in developing latter stages
 - Shorter development thresholds
 - User gets some benefits earlier
 - Project may be put aside temporarily
 - Reduces 'gold-plating'
- But there are some possible disadvantages 😞
 - Loss of economy of scale
 - 'Software breakage'



The Outline Incremental Plan

- Steps ideally 1% to 5% of the total project
- Ideal if a step takes one month or less
 - 🙌 Not more than three months
- Each step should deliver some benefit to the user
- Some steps will be physically dependent on others



Which Step First?

- Some steps will be pre-requisite because of physical dependencies
- Others may be in any order
- Value to cost ratios may be used
- V/C where
 - V is a score 1-10 representing value to customer
 - C is a score 0-10 representing value to developers

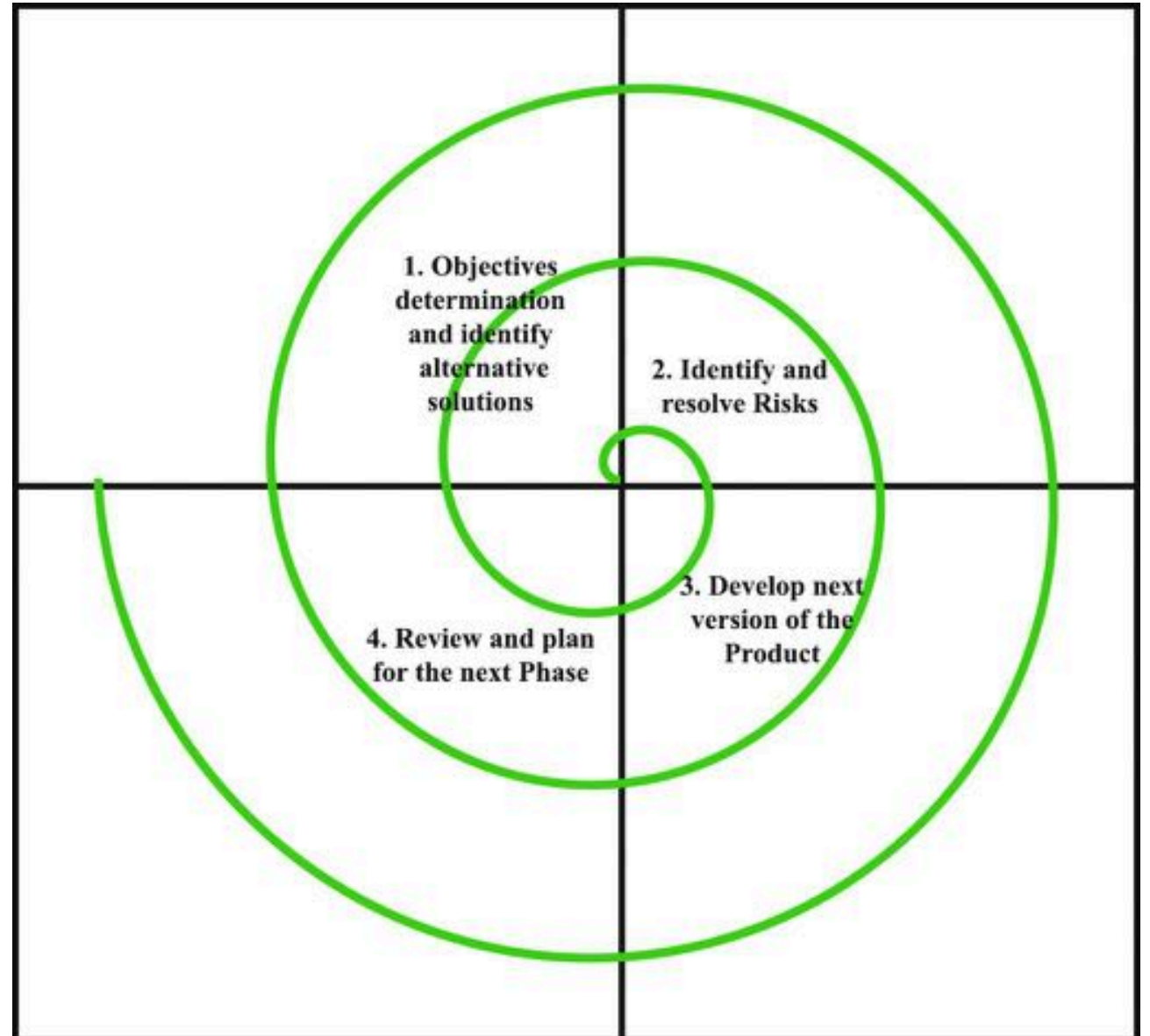


V/C Ratios: an Example

Step	Value	Cost	Ratio	Rank
Profit reports	9	1	9.00	2nd
Online database	1	9	0.11	5th
Ad Hoc enquiry	5	5	1.00	4th
Purchasing plans	9	4	2.25	3rd
Profit-based pay for managers	9	0	∞	1st



The Spiral Model



The Spiral Model

- Spiral model is one of the most important *Software Development Life Cycle models*, which provides support for Risk Handling
- It looks like a spiral with many loops
 - Number of loops unknown and depends on the project
 - Each loop as a **Phase** of the software development process
- The Radius represents the *cost* of the project, and the angular dimension represents the *progress* made in the current phase



Spiral Model as a Meta Model

- It subsumes all the other SDLC models
 - A single loop spiral actually represents the Iterative **Waterfall** Model
 - Uses the approach of **Prototyping** Model by building a prototype at the start of each phase as a risk handling technique
 - Can be considered as supporting the **Evolutionary** Model – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.



Advantages of Spiral Model

- *Risk Handling* - unknown risks can be analyzed and handled at every phase
- *Good for large projects* - recommended
- *Flexibility in Requirements* - change in requirements can be incorporated at later phase
- *Customer Satisfaction* - clients can see the development of the product at early phases



Disadvantages of Spiral Model

- *Complex* - is much more complex than other SDLC models
- *Expensive*- not suitable for small projects as it is expensive
- *Too much dependable on Risk Analysis* - Without very highly experienced expertise, it is going to be a failure to develop a project using this model
- *Difficulty in time management* - As the number of phases is unknown

'Agile' Methods 🧑🏫

- Structured development methods have some perceived disadvantages
 - Produce large amounts of documentation which can be largely unread
 - Documentation has to be kept up to date
 - Division into specialist groups and need to follow procedures stifles communication
 - Users can be excluded from decision process
 - Long lead times to deliver anything
- The answer? 'Agile' methods? ✨



'Agile' Methods 🧑🏻

- Crystal technologies
- Atern (formerly DSDM)
- Feature-driven development
- Scrum
- Extreme Programming (XP)



[2001] The Agile Manifesto 📖

- Individuals and interaction over processes and tools
- Working together over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

👉 <http://agilemanifesto.org>



Atern/DSDM Principles

- 1 Focus on business need
- 2 Delivery on time – use of time-boxing
- 3 Collaborate
- 4 Never compromise quality
- 5 Deliver iteratively
- 6 Build incrementally
- 7 Communicate continuously
- 8 Demonstrate control



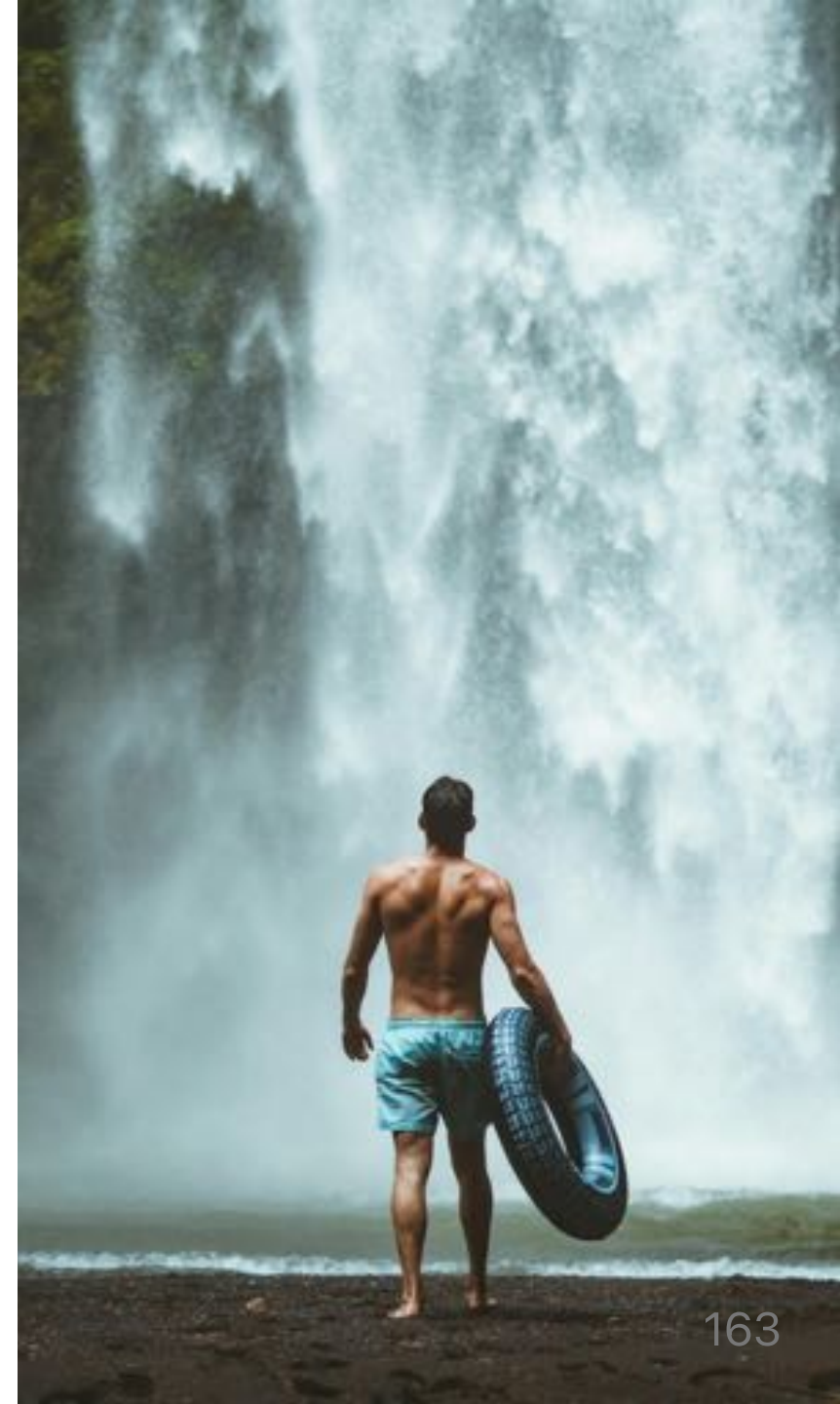
Atern/DSDM: Time-Boxing

- **Time-box** 🖱️ fixed deadline by which something has to be delivered 🕒
 - Typically 2 to 6 weeks
- **MoSCoW** priorities 📈
 - **Must have** - essential features
 - **Should have** - very important, but system could operate without
 - **Could have** - can be delayed with some inconvenience
 - **Want** - their delay to a later increment is accepted (but probably won't get!)






Extreme Programming

- Increments of 1 to 3 weeks
 - Customer can suggest 🙋 improvement at any point
- Argued that distinction between design and building of software are artificial
- Code to be developed to meet current needs only
- Frequent re-factoring to keep code structured



Extreme Programming - (ii)

- Developers work in pairs 
- Test cases and expected results devised before software design 
- After testing of increment, test cases added to a consolidated set of test cases 



Extreme Programming Core Values

1. Communication and feedback 🗣️

- Best face-to-face
- Provide users with frequent working increments

2. Simplicity 🔍

- Simplest design that implements requirements
- Not spend efforts in future possible needs

3. Responsibility 🙌

- Developers ultimately responsible for software quality

4. Courage 🙏

- Throw away work done and start from scratch



Core Practices in XP

- The planning exercise
- Small releases
- Metaphor
- Simple design
- Testing
- Refactoring



Core Practices in XP (ii)

- Pair programming
- Collective ownership
- Continuous integration
- Forty-hour weeks
- On-site customers
- Coding standards



Limitations of Extreme Programming

- Reliance on availability of high quality developers
- Dependence on personal knowledge – after development knowledge of software may decay making future development less easy
- Rationale for decisions may be lost e.g. which test case checks a particular requirement
- Reuse of existing code less likely



Scrum

- Named as an analogy to a rugby scrum – all pushing together 🏉
- Originally designed for new product development where 'time-to-market' is important 📊
- 'Sprints' increments of typically one to four weeks 🏃
- Daily 'scrums' – daily stand-up meetings of about 15 minutes 🧑🏻‍🤝‍🧑🏻



Scrum (ii)

- Unlike XP, requirements are frozen during a sprint
- At the beginning of the sprint there is a sprint planning meeting where requirements are prioritized
- At end of sprint, a review meeting where work is reviewed and requirements may be changed or added to



Grady Booch's Concern 🤔

- Booch, an OO authority, is concerned that with requirements driven projects:

'Conceptual integrity sometimes suffers because this is little motivation to deal with scalability, extensibility, portability, or reusability beyond what any vague requirement might imply'

- Tendency towards a large number of discrete functions with little common infrastructure?



'Rules of Thumb' about Approach to be Used 👍

IF uncertainty is high
THEN use evolutionary approach

IF complexity is high but uncertainty is not
THEN use incremental approach

IF uncertainty and complexity both low
THEN use waterfall

IF schedule is tight
THEN use evolutionary or incremental



Conclusion

- Examine each project carefully to see if it has characteristics which suggest a particular approach or process model
- Classic waterfall process model should lead to projects that are easy to control
- Prototyping may be able to reduce project uncertainties
- Incremental approach encourages the execution of a series of small, manageable 'mini-projects' but adds some cost

SUMMARY

Software Effort Estimation

Subject Five



What Makes a Successful Project?

Delivering:

- Agreed functionality 🤝
- On time 🕒
- At the agreed cost 💰
- With the required quality 👍

Stages:

1. Set targets 🤔
2. Attempt to achieve targets ➡️🎯



Some Problems with Estimating

- Subjective nature of much of estimating
 - It may be difficult to produce evidence to support your precise target
- Political pressures
 - Managers may wish to reduce estimated costs in order to win support for acceptance of a project proposal
- Changing technologies
 - These bring uncertainties, especially in the early days when there is a 'learning curve'
- Projects differ
 - Experience on one project may not be





Exercise

Calculate the productivity SLOC/wm of each project

Project	Design	Coding	Testing	Total wm (SLOC)
a	3.9	5.3	7.5	16.7 (6050)
b	2.7	13.4	6.5	22.6 (8363)
c	3.5	26.8	1.9	32.2 (13334)
d	0.8	2.4	0.7	3.9 (5942)

Exercise (ii)

Productivity rates in SLOC/m of each project

Project	Work-Month	SLOC	Productivity (SLOC/wm)
a	16.7	6050	362
b	22.6	8363	370
c	32.2	13334	414
d	3.9	5942	1524

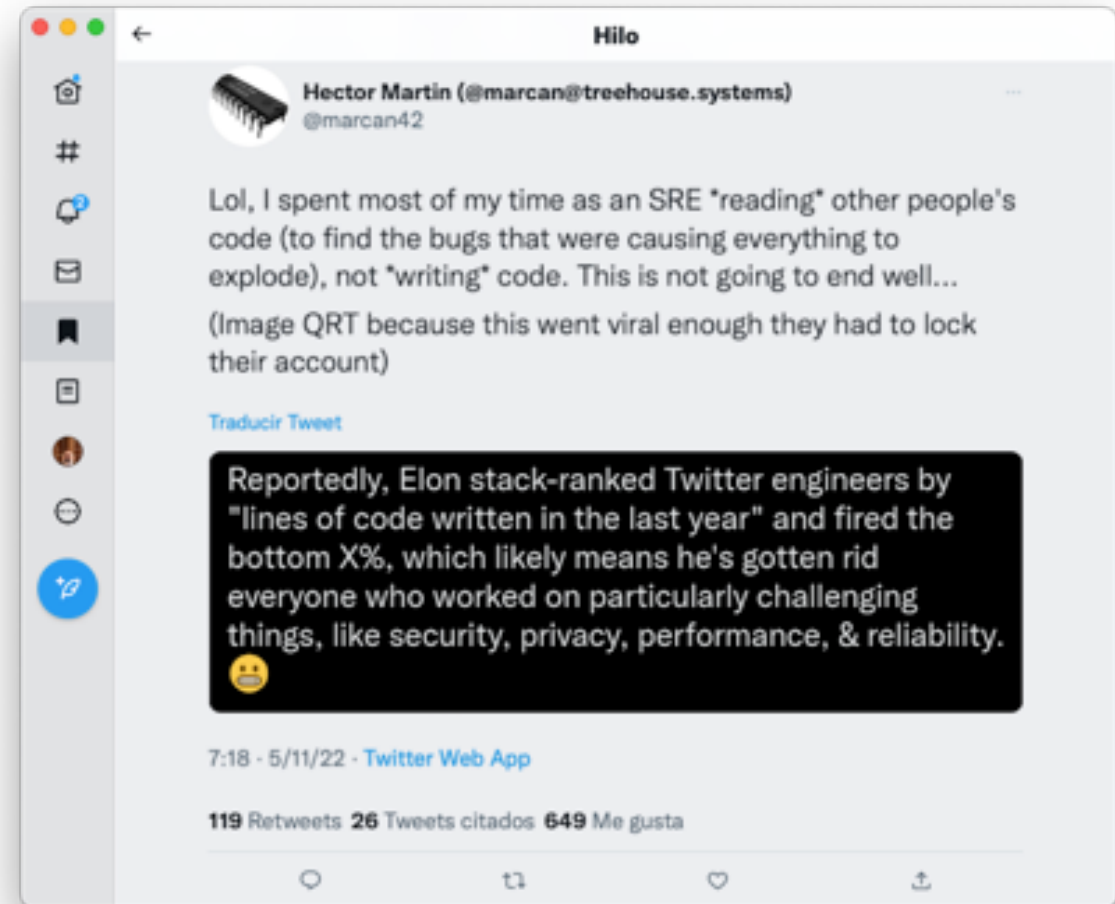


Exercise (iii)

If the average productivity is 668 LOC/month

Project	Estimated Wok-Month	Actual	Difference
a	$6050 / 668 = 9.1$	16.7	+7.6
d	$5942 / 668 = 8.9$	3.9	-5.0

Some Problems with Estimating



Over and Under-Estimating

- Parkinson's Law
 - ☞ *'Work expands to fill the time available'*
- Brooks' law
 - ☞ *'Putting more people on a late job makes it later'*
- Weinberg's Zeroth Law of reliability
 - ☞ *'A software project that does not have to meet a reliability requirement can meet any other requirement'*



Basis for Successful Estimating

- Information about **past projects**
 - Need to collect performance details about past project:
 - How big were they? How much effort/time did they need?
- Need to be able to **measure the amount of work** involved
 - Traditional size measurement for software is '*lines of code*' (KLOC) – but this can have problems



Table of SLOC Ratios (approx)

Language	SLOC Ratio
Assembly	~20x
C	~4-5x
C++	~1.5-3x
Java	~2-3x
Go	~1.5x
JavaScript	~1-1.5x
Python	1x



A Taxonomy of Estimating Methods

- **Bottom-up** - activity based, analytical
- **Top-down** - overall estimate broken down
- **Parametric or algorithmic models** - e.g. function points
- **Expert opinion** - just guessing?
- **Analogy** - case-based, comparative
- **Parkinson and 'price to win'**

 Boehm, *Software Engineering Economics*



Bottom-Up Estimating

1. 📄 Break project into smaller and smaller components
2. 🛑 Stop when you get to what one person can do in one/two weeks
3. 👤 Estimate costs for the lowest level activities
4. ⬆️ At each higher level calculate estimate by adding estimates for lower levels



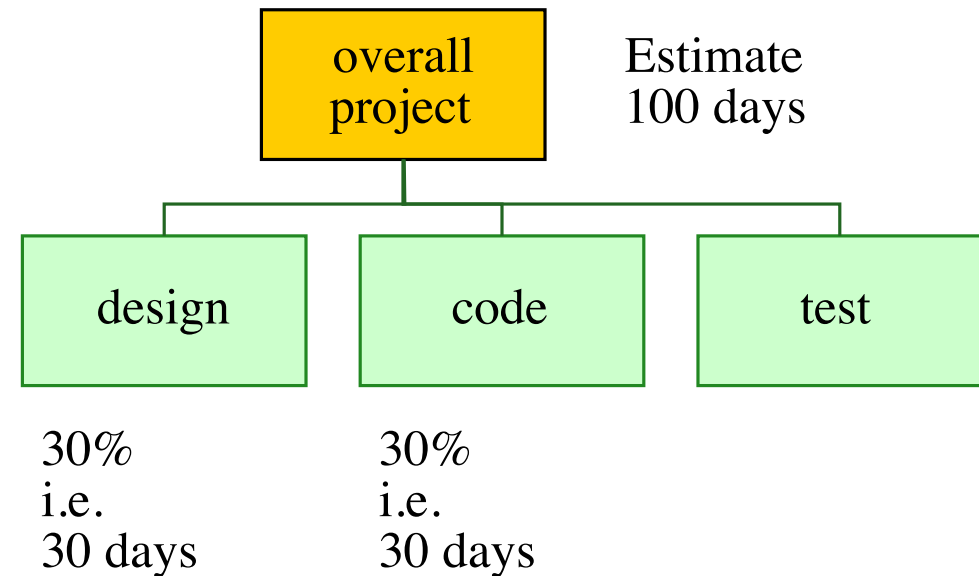
A Procedural Code-Oriented Approach

1. Envisage the number and type of software modules in the final system
2. Estimate the SLOC of each identified module
3. Estimate the work content, taking into account complexity and technical difficulty
4. Calculate the work-days effort



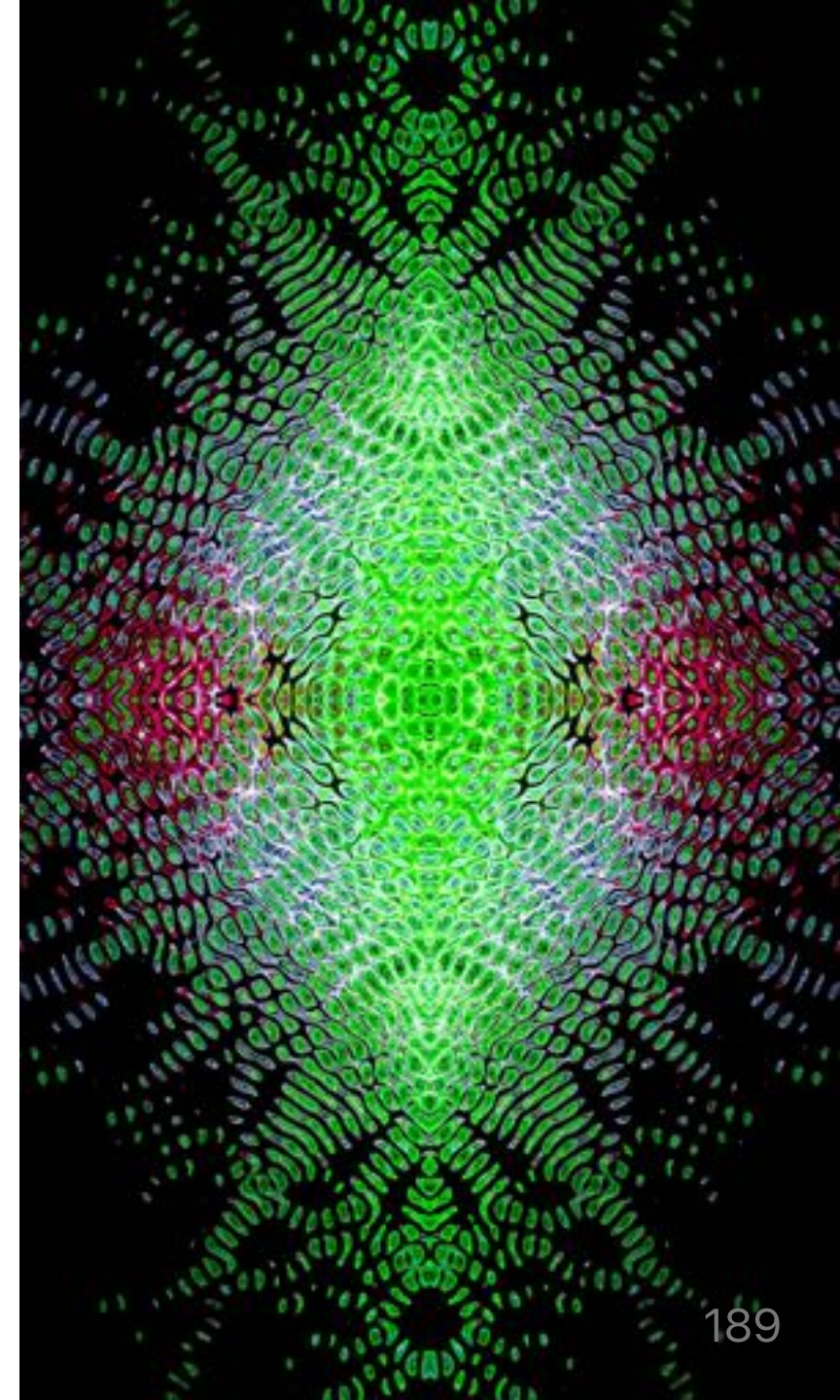
Top-Down Estimates

- Produce overall estimate using effort driver(s)
- Distribute proportions of overall estimate to components





Approximate Time Breakdown

Phase	Waterfall	Agile
Requirements	10-15%	05-10%
Design	15-20%	10-15%
Development (Coding)	30-40%	40-50%
Testing	25-30%	20-30%



Bottom-Up versus Top-Down

- **Bottom-up** 
 - Identify all tasks that have to be done
 - Add up the calculated effort for each activity to get an overall estimate
 - Use when you have no data about similar past projects
- **Top-down** 
 - Based on past project data
 - Produce overall estimate based on project cost drivers
 - Divide overall estimate between jobs to be



Algorithmic/Parametric Models

- **COCOMO** (lines of code) and **Function Points**
- Problem with COCOMO etc:
 - | Guess LOC → Algorithm → Estimate
- But what is desired is
 - | System Characteristics → Algorithm → Estimate



Parametric Models - the Need for Historical Data

- Simplistic model for an estimate

Estimated effort = system size /
productivity

- System size = lines of code
 - Productivity = lines of code per day
- Based on past projects



Parametric Models

- Some models focus on task or system size e.g.
Function Points
- FPs originally used to estimate Lines of Code, rather than effort
 - Number of file types
 - Number of input and output transaction types
 - → System size



Parametric Models

- Other models focus on productivity: e.g. **COCOMO**
- Lines of code (or FPs etc) an input
 - System size
 - Productivity factors
 - → Estimated effort

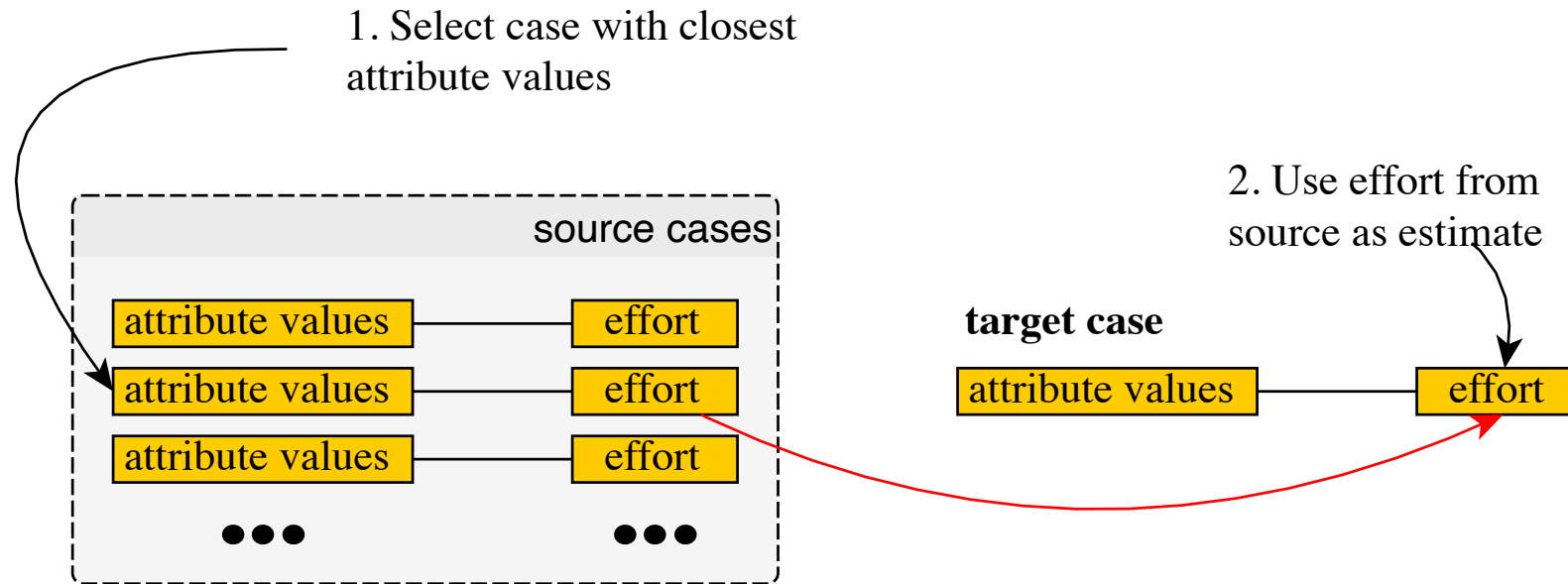


Expert Judgement

- Asking someone who is familiar with and knowledgeable about the application area and the technologies to provide an estimate
- Particularly appropriate where existing code is to be modified
- Research shows that experts judgement in practice tends to be based on analogy



Estimating by Analogy



Stages: Identify

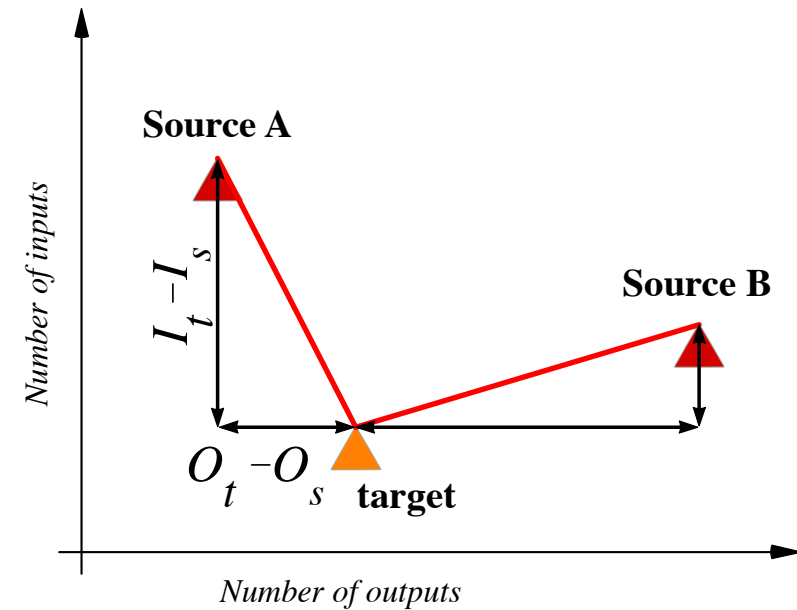
- Significant features of the current project
- Previous project(s) with similar features
- Differences between the current and previous projects
- Possible reasons for error (risk)
- Measures to reduce uncertainty



Machine Assistance for Source Selection

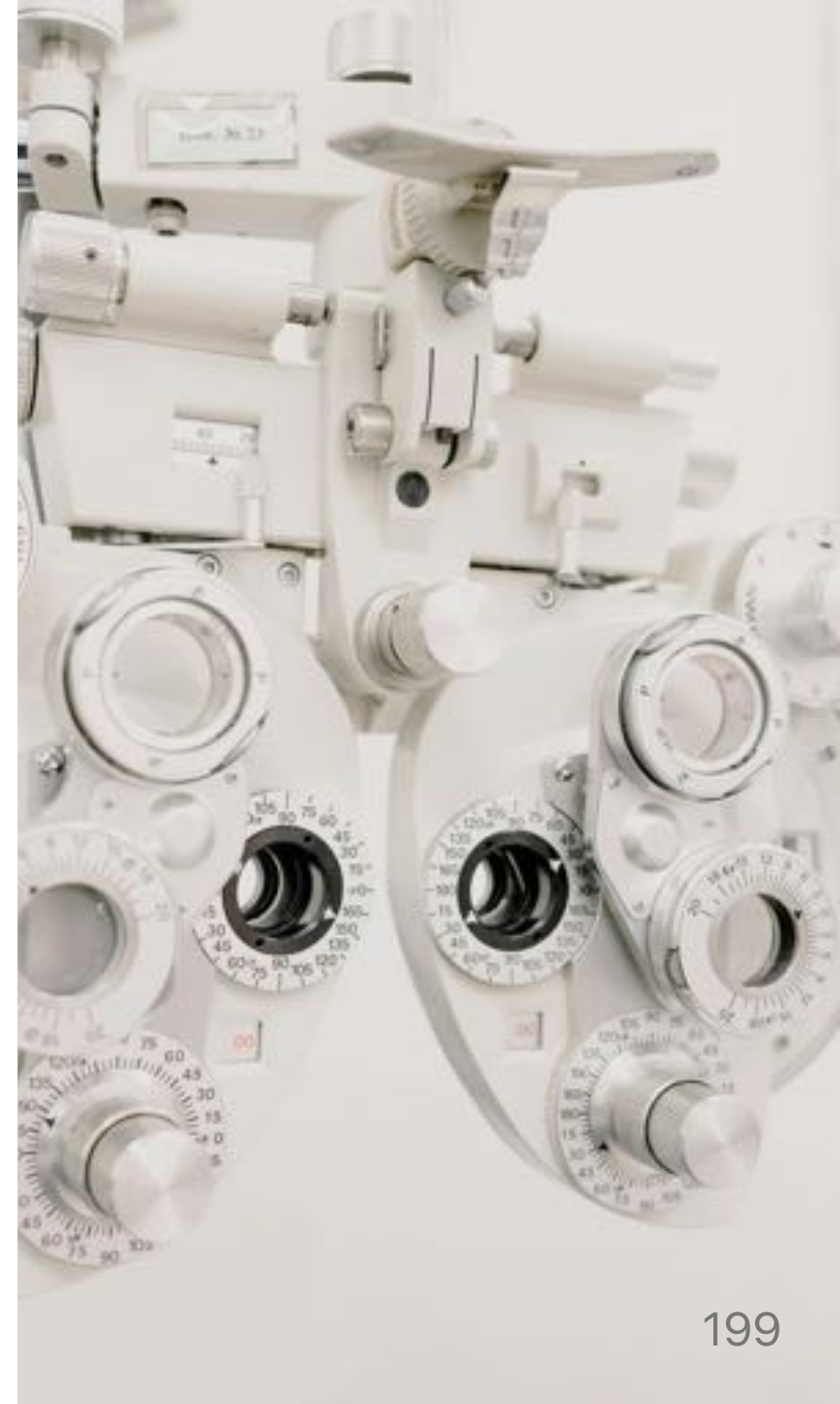
Euclidean distance =

$$\sqrt{(I_t - I_s)^2 + (O_t - O_s)^2}$$



Parametric Models

- We are now looking more closely at four parametric models:
 - Albrecht/IFPUG function points
 - Symons/Mark II function points
 - COCOMO81 and COCOMO II



Albrecht/IFPUG Function Points

- Albrecht worked at IBM and needed a way of measuring the relative productivity of different programming languages
- Needed some way of measuring the size of an application without counting lines of code
- Identified **five types of component** or **functionality** in an information system
- Counted occurrences of each type of functionality in order to get an indication of the size of an information system



Albrecht/IFPUG Function Points (ii)

Five function types

1. **Logical Interface File (LIF)** types – equates roughly to a datastore in systems analysis terms. Created and accessed by the target system
2. **External Interface File (EIF)** types – where data is retrieved from a datastore (also as output) which is actually maintained by a different application



Albrecht/IFPUG Function Points (ii)

3. **External Input (EI)** types – input transactions which update internal computer files
4. **External Output (EO)** types – transactions which extract and display data from internal computer files. Generally involves creating reports
5. **External Inquiry (EQ)** types – user initiated transactions which provide information but do not update computer files. Normally the user inputs some data that guides the system to the information the user needs



Albrecht Complexity Multipliers

External Users	Low Complexity	Medium Complexity	High Complexity
EI	3	4	6
EO	4	5	7
EQ	3	4	6
LIF	7	10	15
EIF	5	7	10

 <https://www.ifpug.org>



Example

Payroll application has:

- Transaction to input, amend and delete employee details – An EI that is rated of medium complexity
- A transaction that calculates pay details from timesheet data that is input – An EI of high complexity
- A transaction that prints out pay-to-date details for each employee - An EO of medium complexity

Example

- A file of payroll details for each employee – Assessed as of medium complexity LIF
- A personnel file maintained by another system is accessed for name and address details – A simple EIF

What would be the FP counts for these?

Example

FP Counts	
Medium EI	4 FPs
High complexity EI	6 FPs
Medium complexity EO	5 FPs
Medium complexity LIF	10 FPs
Simple EIF	5 FPs
Total	30 FPs

👉 If previous projects 5 FPs a day, it should take $30/5 = 6$ days

Function Points Mark II

- A simpler method
- For each transaction, count
 - Data items input (N_i)
 - Data items output (N_o)
 - Entity types accessed (N_e)

$$FPcount = W_i \times N_i + W_e \times N_e + W_o \times N_o$$

In practice:

$$FPcount = 0.58 \times N_i + 1.66 \times N_e + 0.26 \times N_o$$



Exercise

Calculate the number of unadjusted Mk II function points for the following transaction:

- The operator will input: Customer account number, Customer name, Address, Postcode, Customer type, and Renewal date
- All this information will be set up in a CUSTOMER record on the system's database
- If a CUSTOMER account already exists for the account number that has been input, an error message will be displayed

Exercise

The function types are:

- Input data types: 6
- Entities accessed: 1
- Output data types: 1

👉 $FP = (0.58 \times 6) + (1.66 \times 1) + (0.26 \times 1) = 5.4$

COCOMO (COnstructive COst Model)

- Based on industry productivity standards - database is constantly updated
- Allows an organization to benchmark its software development productivity
- Basic model: $effort = c \times size^k$
- Parameter c and k depend on the type of system: **organic, semi-detached, embedded**
- Size is measured in thousands of lines of code, **KLOC**
- **Effort** is measured in person-month (152 h)



The COCOMO Constants

System Type	c	k
Organic (broadly, information systems)	2.40	1.05
Semi-detached	3.00	1.12
Embedded (broadly, real-time)	3.60	1.20

- k exponentiation – ‘to the power of...’ adds disproportionately more effort to the larger projects takes account of bigger management overheads



Development Effort Multipliers (DEM)

- **Product attributes:** required reliability, database size, product complexity
- **Computer attributes:** execution time constraints, storage constraints, virtual machine (VM) volatility
- **Personnel attributes:** analyst capability, application experience, VM experience, programming language experience
- **Project attributes:** modern programming practices, software tools, schedule constraints



Using COCOMO Development Effort Multipliers (DEM)

An an example, for analyst capability:

- Assess capability as very low, low, nominal, *high* or very high
- Extract multiplier:
 - Very low: 1.46
 - Low: 1.19
 - Nominal: 1.00
 - High: 0.80
 - Very high: 0.71
- Adjust nominal estimate 🖱️ $32.6 \times 0.80 = 26.8$
staff months



Some Conclusions: How to Review Estimates

- Ask the following questions about an estimate
 - What are the task size drivers?
 - What productivity rates have been used?
 - Is there an example of a previous project of about the same size?
 - Are there examples of where the productivity rates used have actually been found?

SUMMARY

Activity Planning

Subject Six



Scheduling

🕒 'Time is nature's way of stopping everything happening at once'

Having

- **1** Worked out a method of doing the project
- **2** Identified the tasks to be carried
- **3** Assessed the time needed to do each task

➡ need to allocate dates/times for the start and end of each activity



Objectives of Activity Planning


These help us to:

- Assess the feasibility of the planned project **completion date**
- Identify when **resources** will need to be deployed to activities
- Calculate when **costs** will be incurred

This helps the co-ordination and motivation of the project team



When To Plan

- An ongoing process 
 - At feasibility study and project start-up, to estimate timescales and risks
 - After, to ensure resource availability and cash flow control
- Correct any drift





Project Schedule

- *Project Schedule* - when project plan shows each activity starting and ending dates and resource required
- Four stages:
 - i. To decide **what activities and its order**
 - ii. **Ideal activity plan and activity risk analysis**
 - iii. **Resource allocation**
 - iv. **Schedule production**



Defining Activities

Activities plans are based on some assumptions:

- A project:
 - Composed of a number of interrelated **activities**
 -  May start when at least one of its activities is ready to start
 -  Completed when all its activities are completed

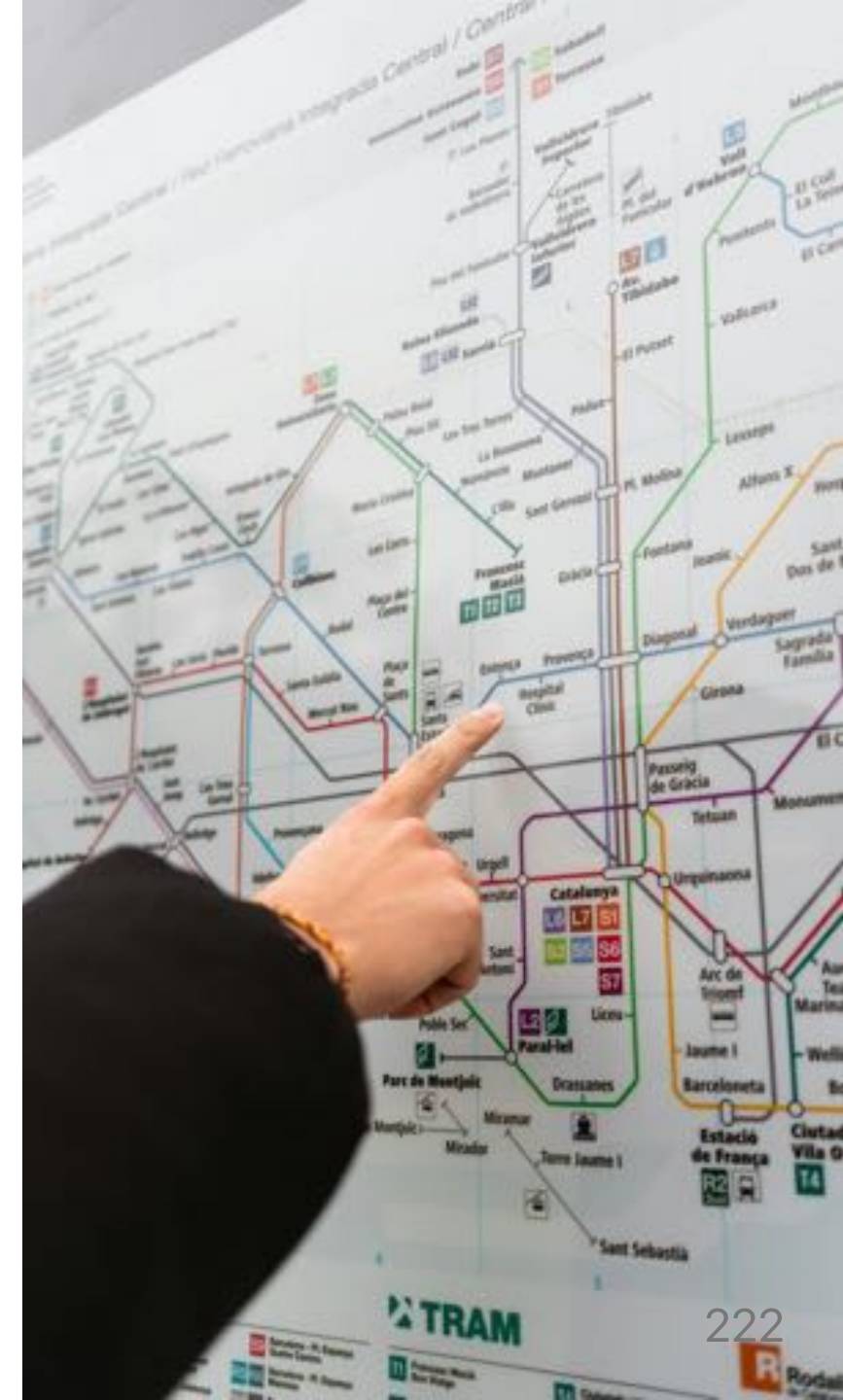


Defining Activities (ii)

An activity 📌

- Must have clearly defined **start** and **end**-points, producing a **deliverable**
- Must have a **duration** that can be forecast
- Must have **resource requirements** that can be forecast
 - Assumed to be constant throughout activity duration
- May be **dependent** on other activities being completed first (precedence networks)

● Any activity that does not meet these criteria must be redefined

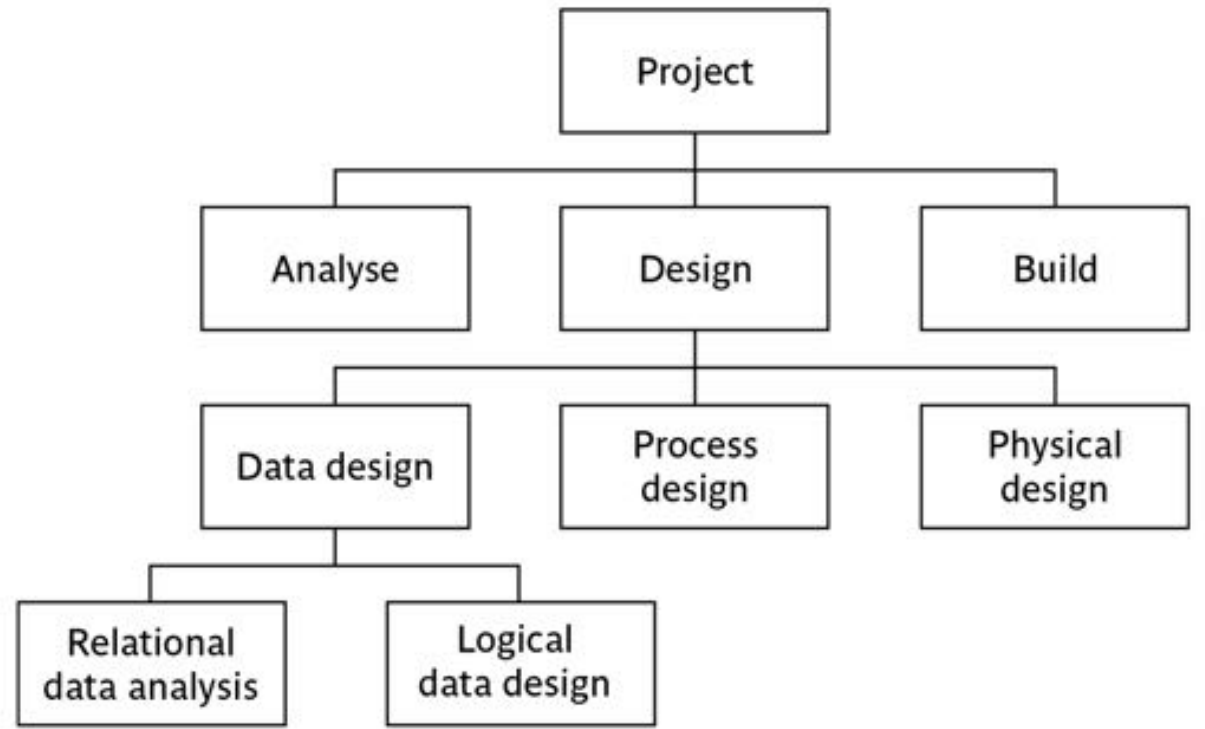


Identifying Activities

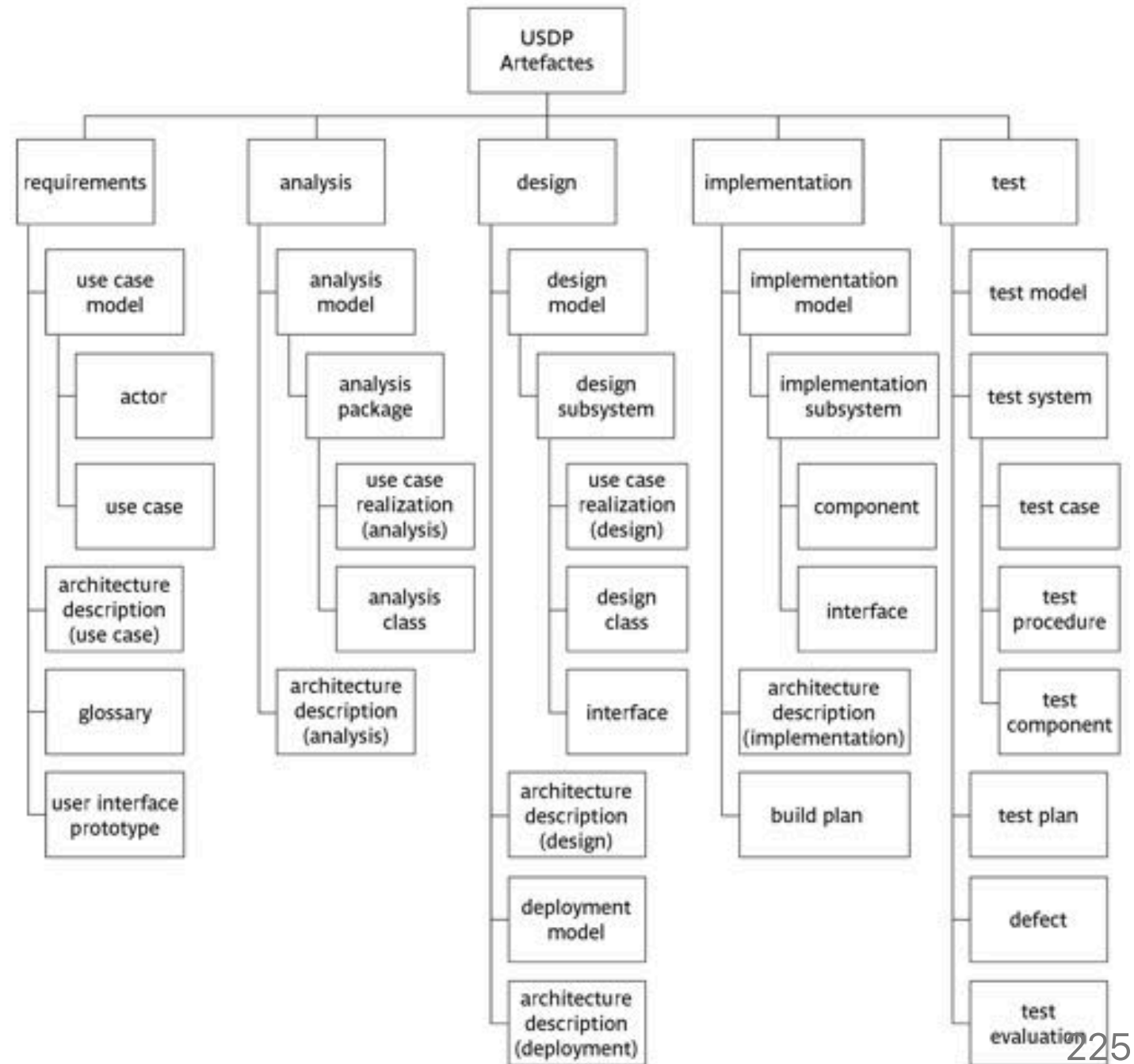
- Work-based
 - Draw-up a Work Breakdown Structure listing the work items needed
- Product-based approach
 - List the deliverable and intermediate products of project – *Product Breakdown Structure* (PBS)
 - Identify the order in which products have to be created
 - Work out the activities needed to create the products
- Hybrid approach



Work Breakdown Structure

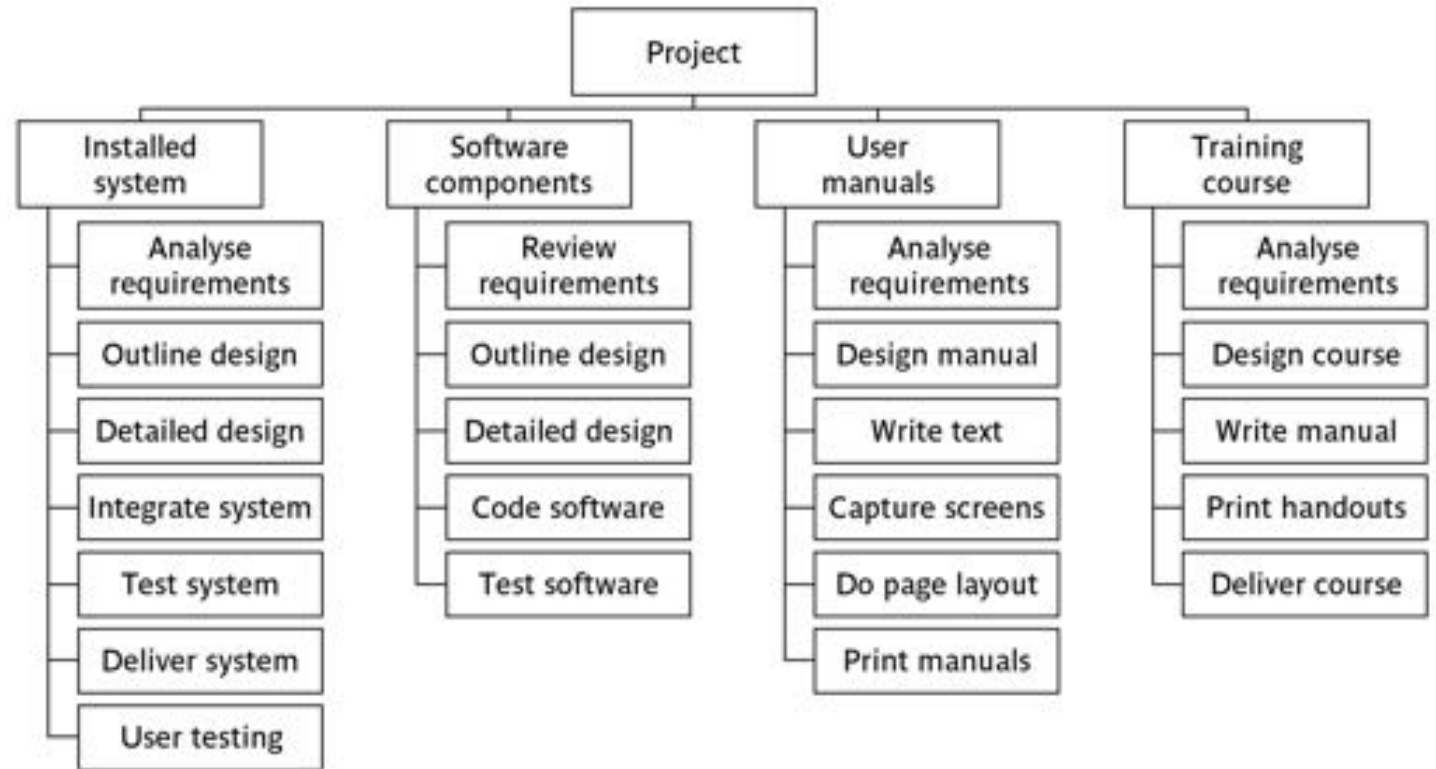


USDP Product Break-down



Hybrid Approach

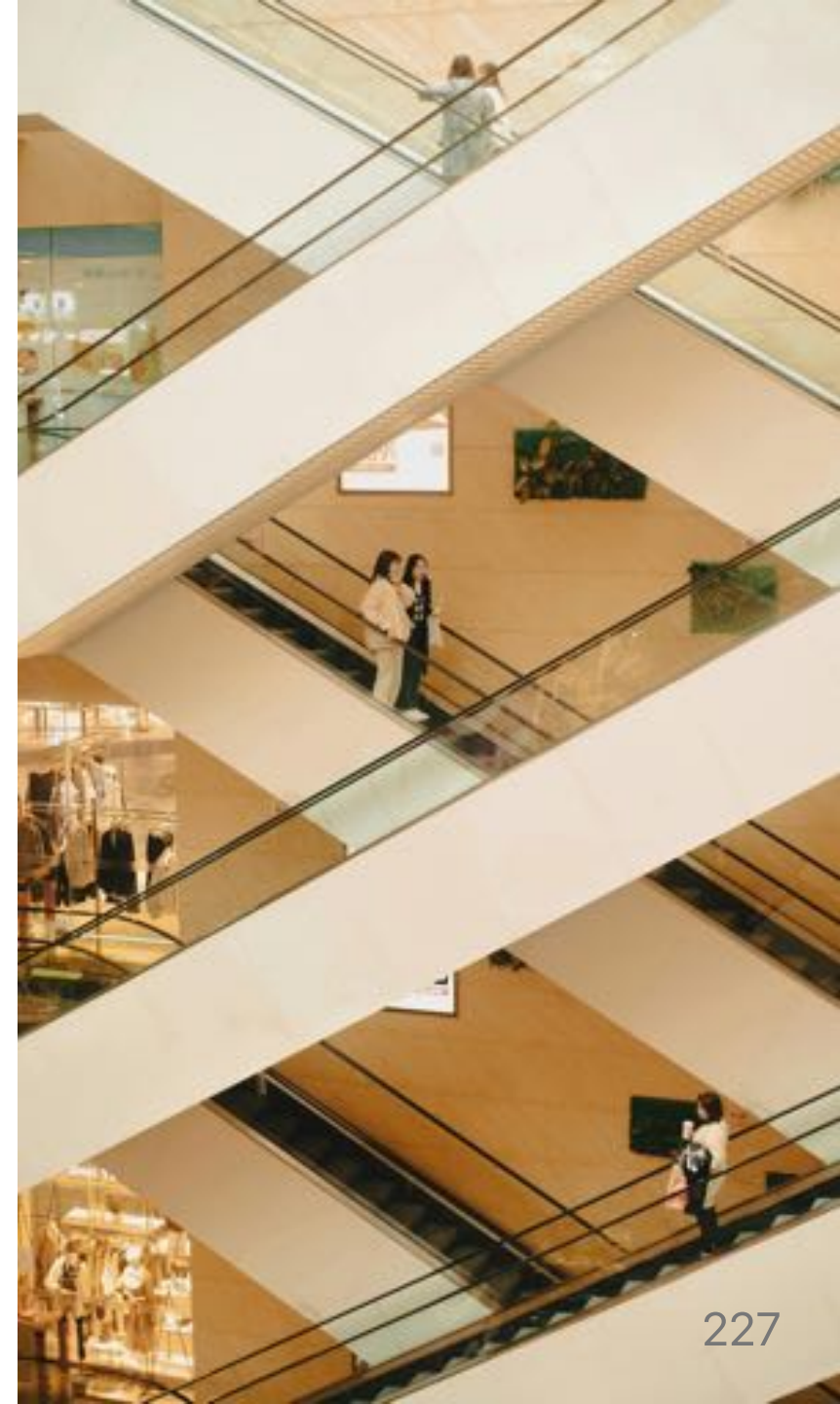
- WBS based upon project products
- Deliverables and activities




Hybrid Approach

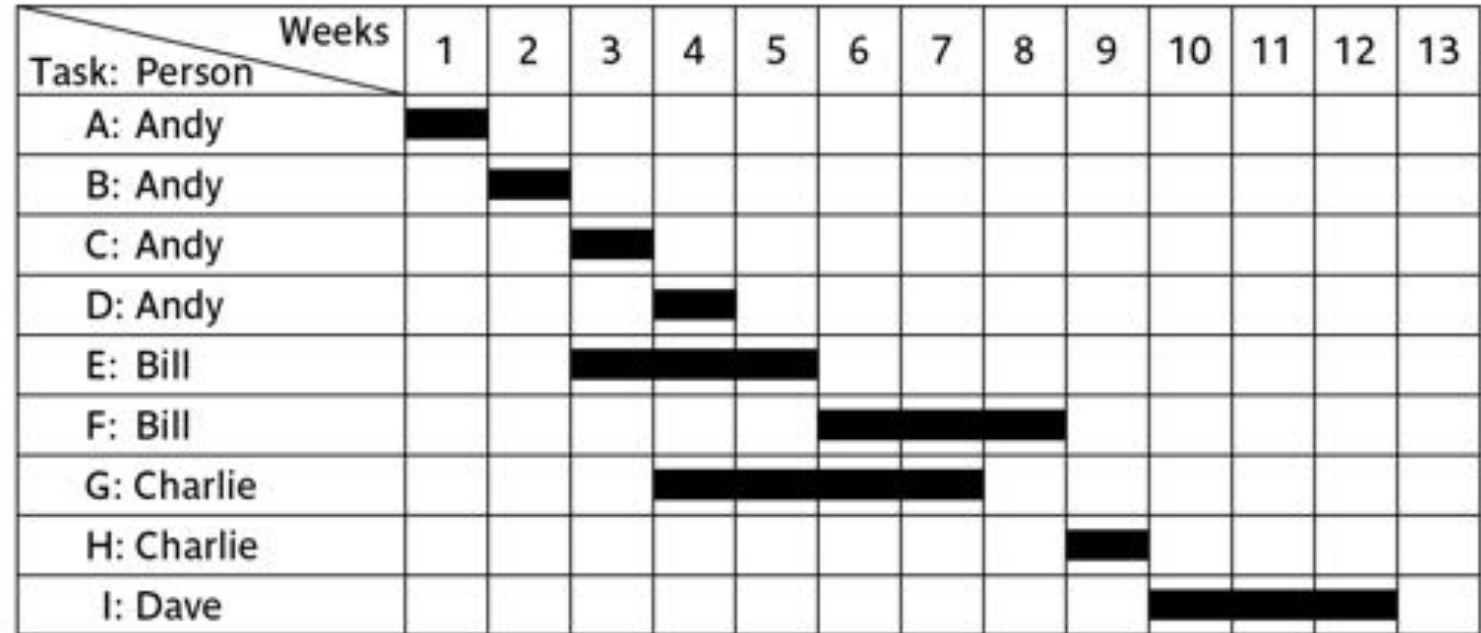
The IBM MITP approach:

- **L1: Project**
- **L2: Deliverables** - such as software, manuals and training courses
- **L3: Components** – key work items needed to produce the deliverables
- **L4: Work packages** - mayor work items, or collections of related tasks needed to produce the components
- **L5: Tasks** - tasks that will normally be the responsibility of a single person



The Final Outcome of the Planning Process

- A project plan as a bar chart 

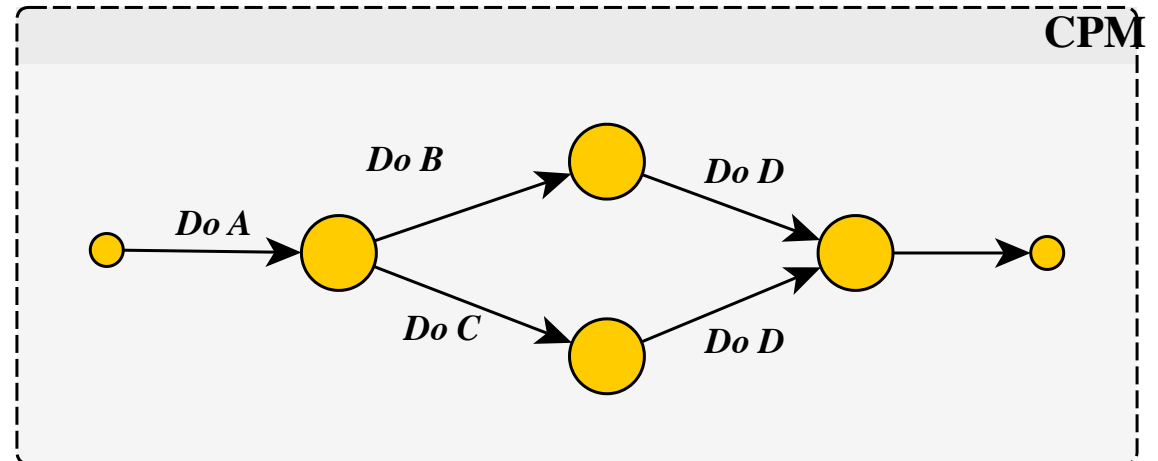
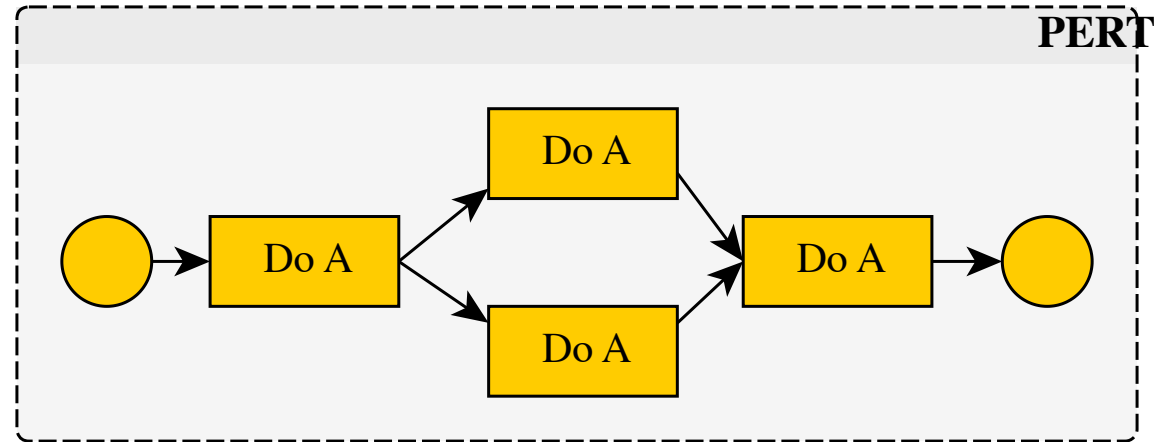


Activity key

A: Overall design
 B: Specify module 1
 C: Specify module 2
 D: Specify module 3
 E: Code module 1

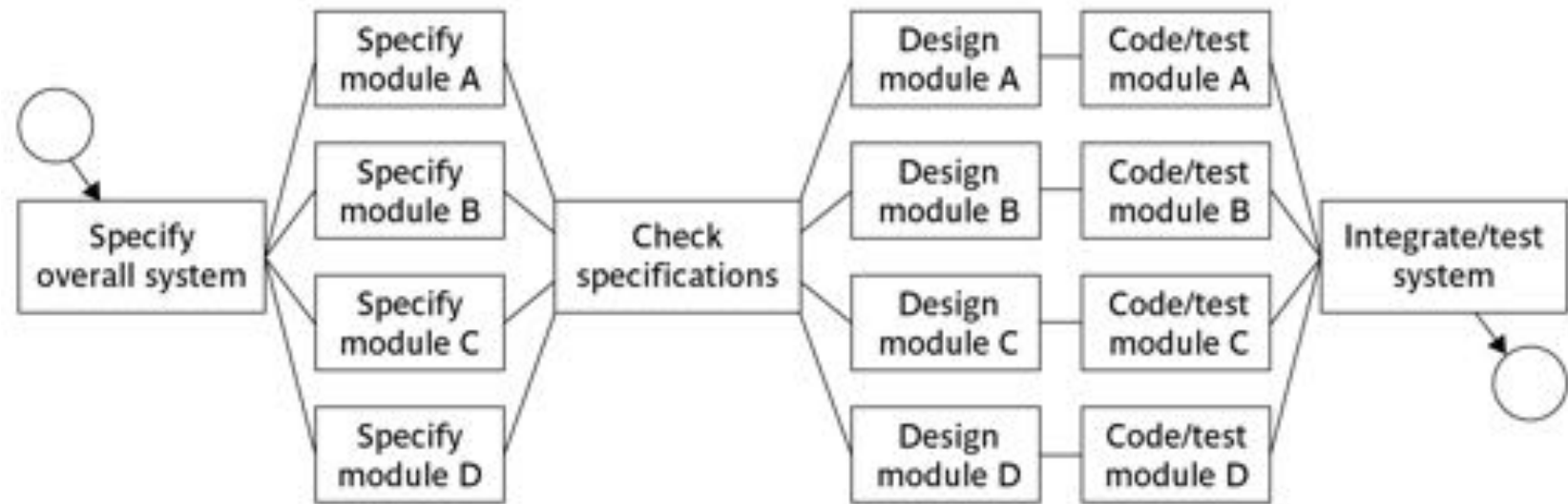
F: Code module 3
 G: Code module 2
 H: Integration testing
 I: System testing

Network Planning Models



Constructing Precedence Networks

- Activity-on-node



Precedence Networks Rules

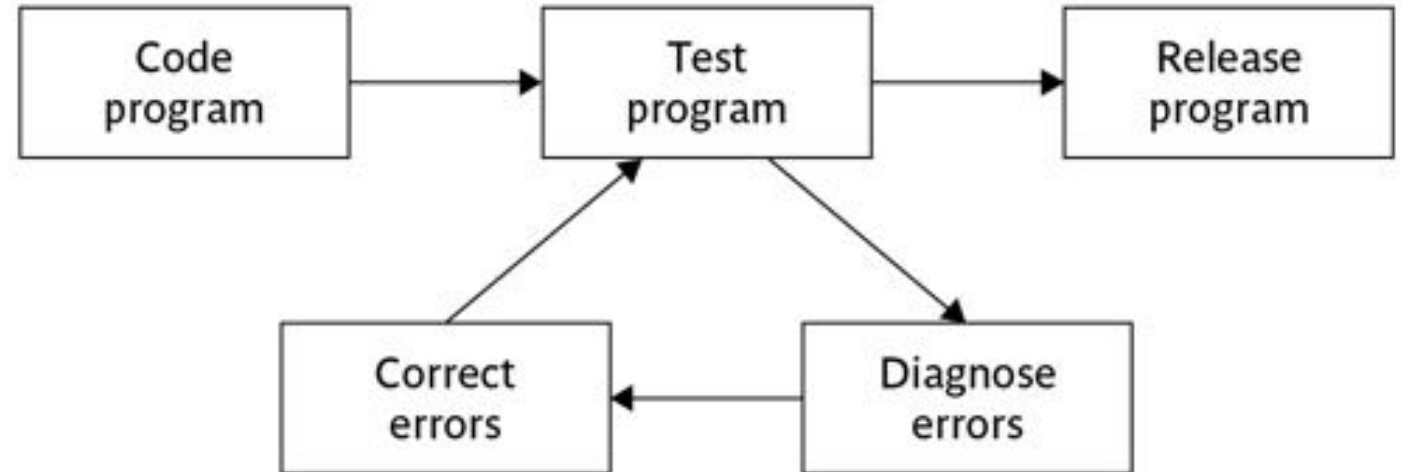
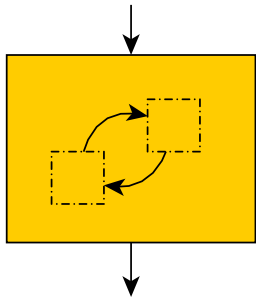
- Should have only one start node
- Should have only one end node
- A node has duration
- Links normally have no duration
- Precedents are the immediate preceding activities
- Time moves from left to right
- May no contain loops
- Should not contain dangles
- **Milestones** – ‘zero duration activities’, such as the start and end of the project, which indicate

transition points



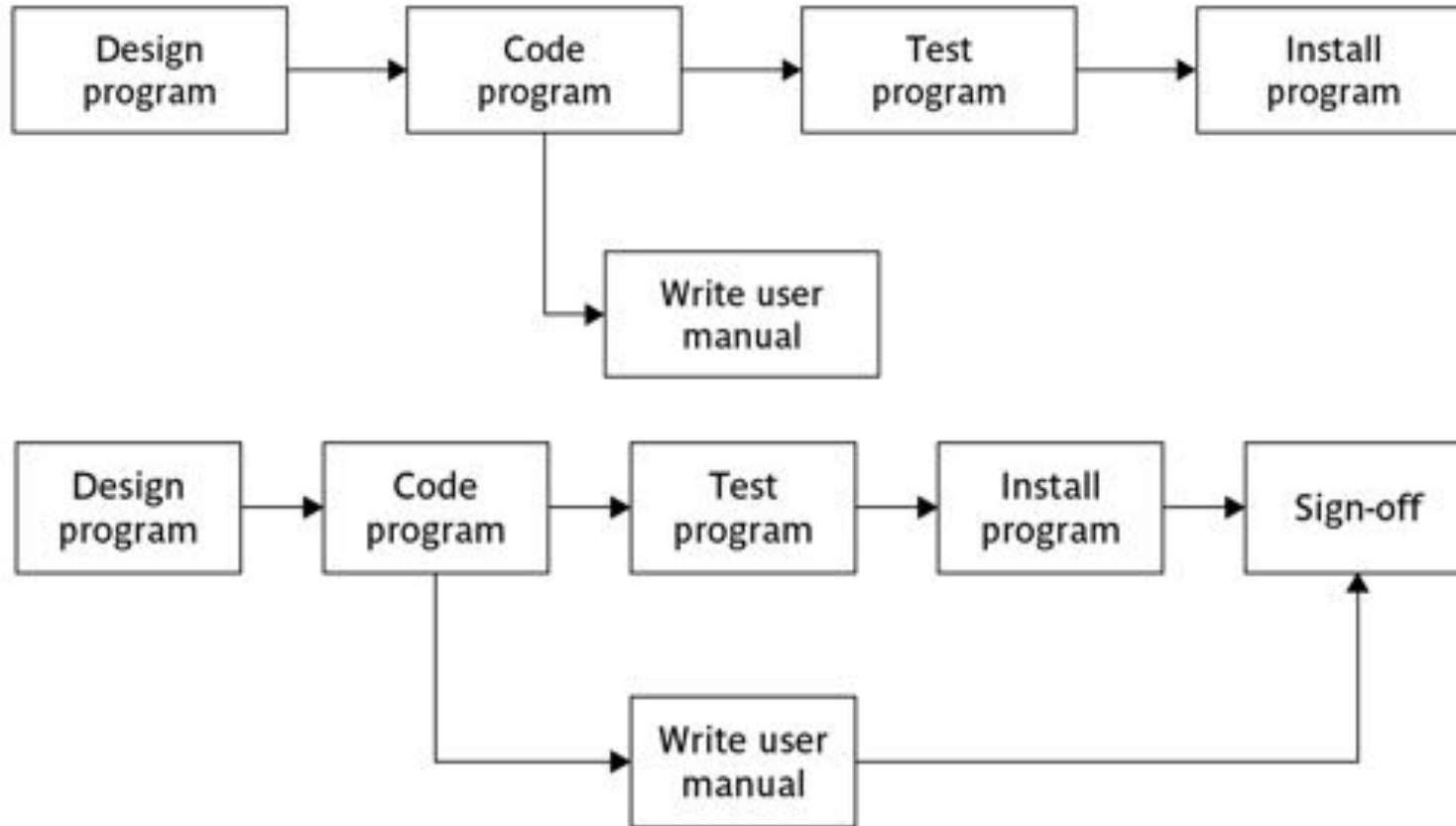
Drawing Up a PERT Diagram

- Loops represent an impossible sequence
- Deal with iterations by hiding them within single activities



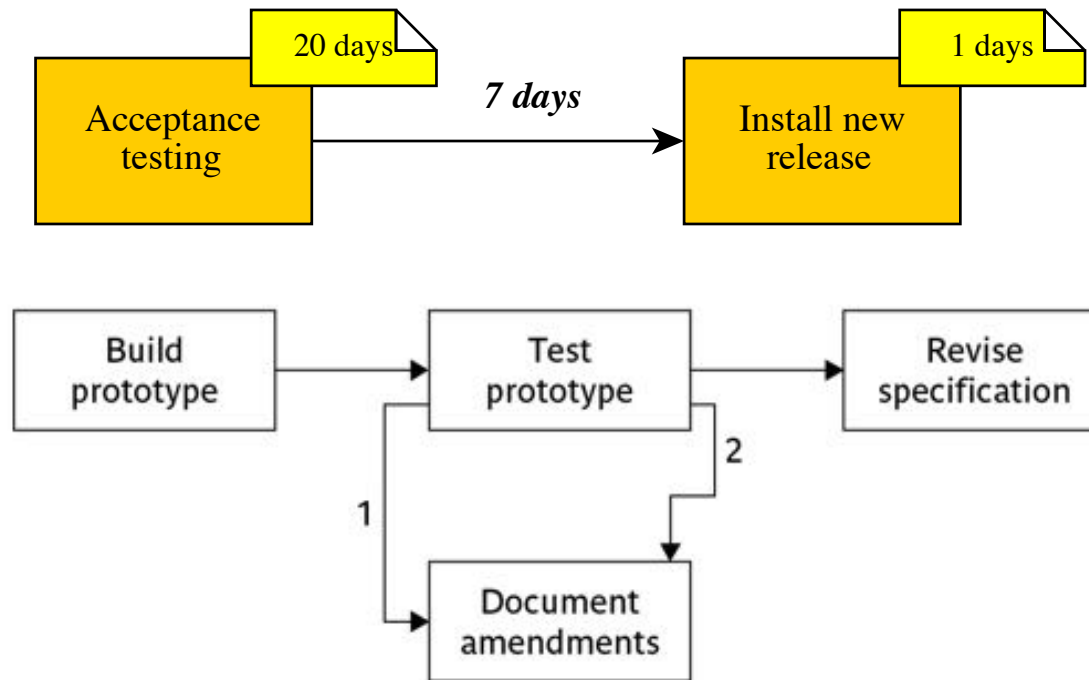
Drawing Up a PERT Diagram

- Dangling activity is likely to lead to errors, redraw it



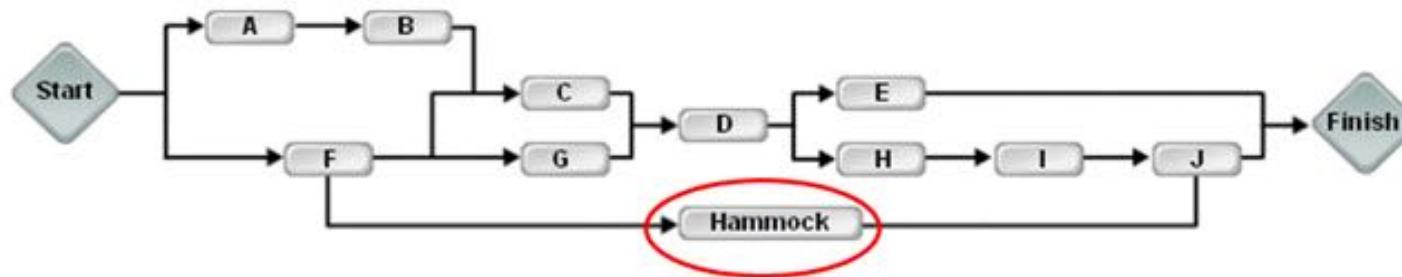
Drawing Up a PERT Diagram

- Lagged activities
 - Where there is a fixed delay between activities



Drawing Up a PERT Diagram

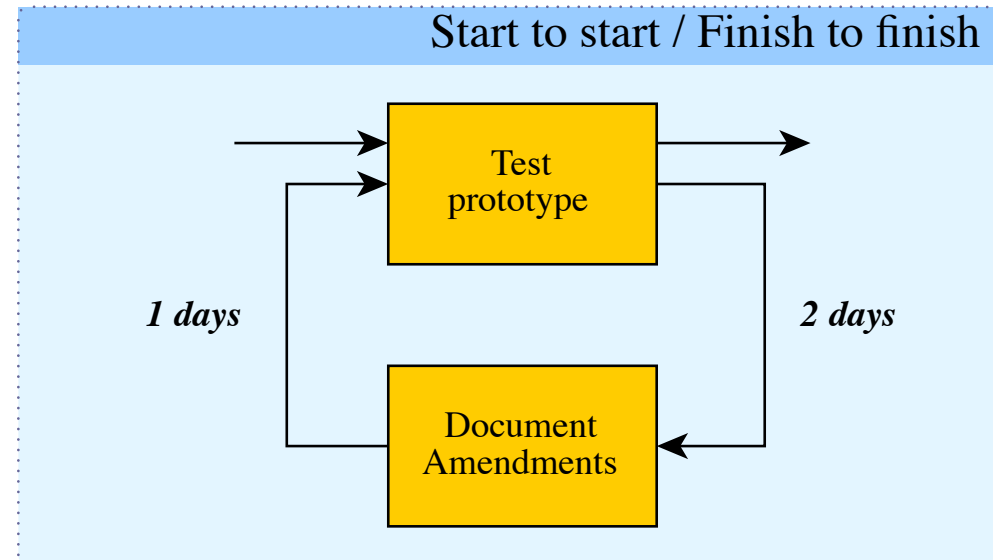
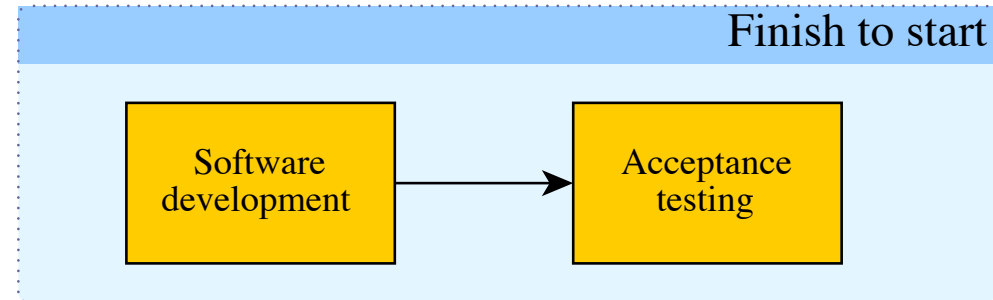
- Hammock activities
 - Activities with zero duration but assumed to start at the same time as the first 'hammocked' activity and to end at the same time as the last one.



- User for representing overhead costs or resources used at a constant rate over the duration of a set of activities

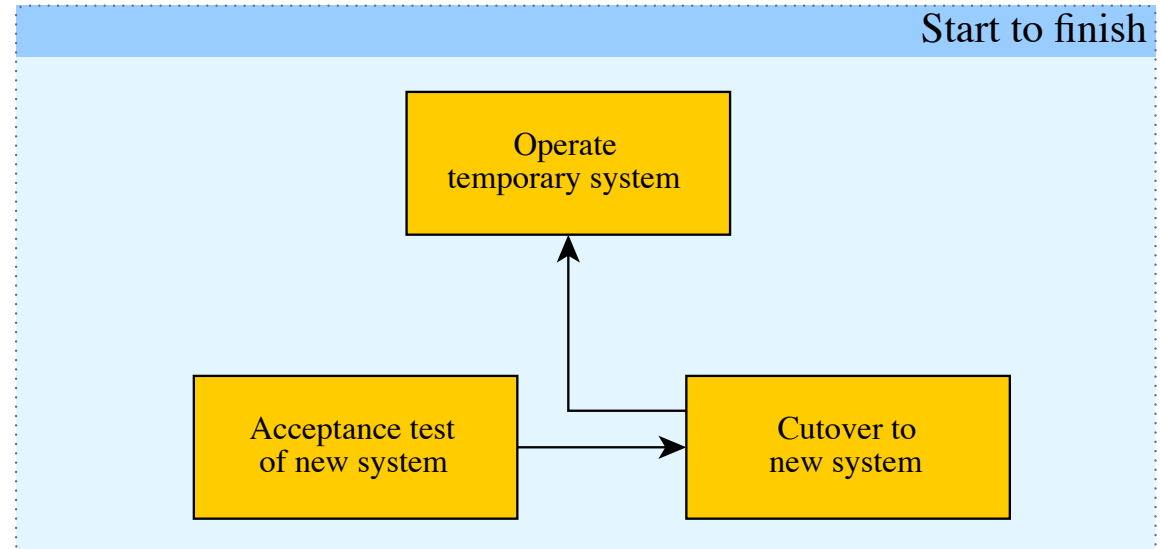
Types of Links Between Activities

- Finish to start
- Start to start/ Finish to finish



Types of Links Between Activities (ii)

- Start to finish



Start and Finish Times

- Earliest start (ES)
- Earliest finish (EF) = ES + duration
- Latest finish (LF) = latest task can be completed without affecting project end
- Latest start (LS) = LF - duration
- Float = LF - ES - duration





Example

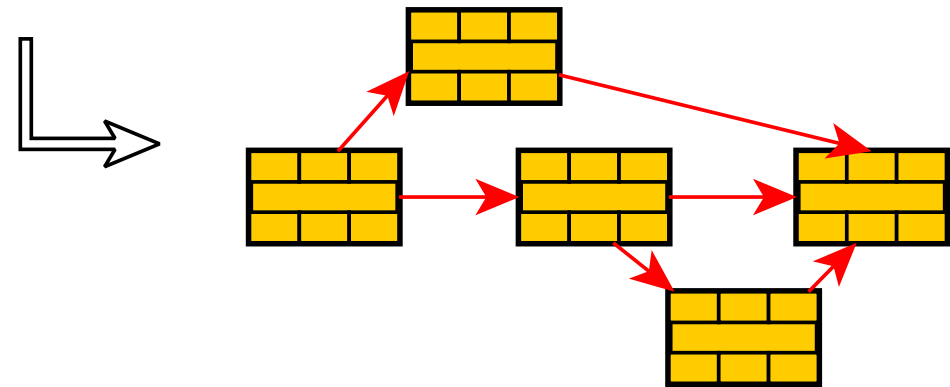
- Earliest Start = day 5
- Latest Finish = day 30
- Duration = 10 days
- Earliest Finish = ?
- Latest Start = ?
- Float = ?

What is it in this case?

Notation

- An activity

<i>Earliest start</i>	<i>Duration</i>	<i>Earliest finish</i>
<i>Activity label, activity description</i>		
<i>Latest start</i>	<i>Float</i>	<i>Latest finish</i>



Exercise

- Complete for the previous example

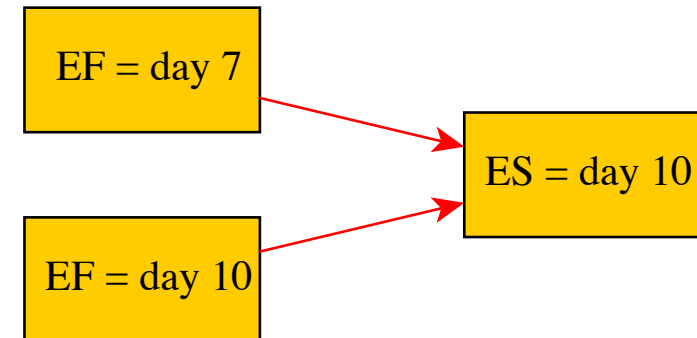
'Day 0' 🏁

- Note that in the last example, day numbers used rather than actual dates
- Makes initial calculations easier – not concerned with week-ends and public holidays
- For **finish** date/times Day 1 means at the END of Day 1.
- For a **start** date/time Day 1 also means at the END of Day 1.
- The first activity therefore begins at Day 0 i.e. the end of Day 0 i.e. the start of Day 1



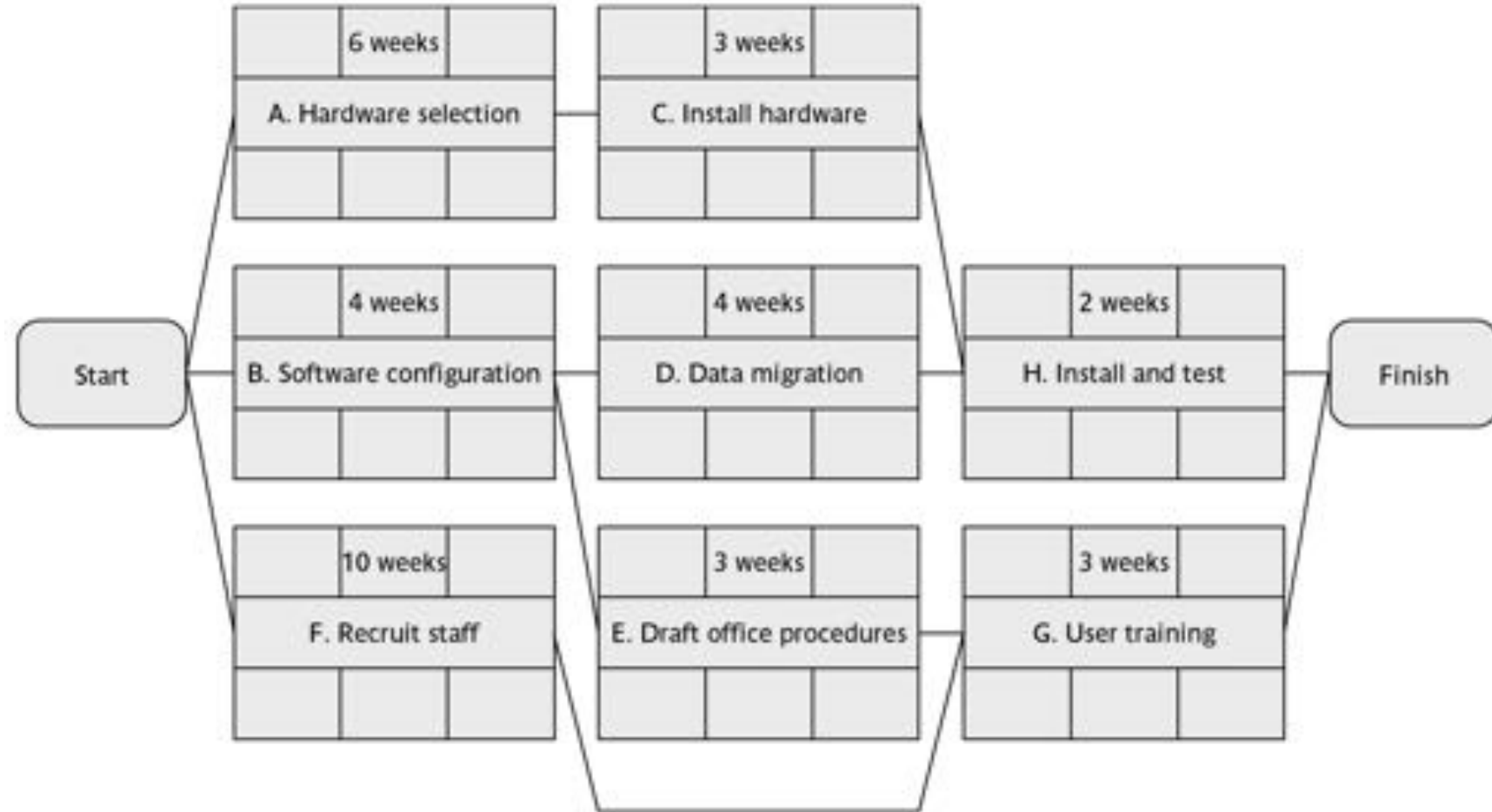
1st Step: Forward Pass

- Calculate the earliest dates on which each activity may be started and completed
 - Start at beginning (Day 0) and work forward following chains
- Earliest start date for the **current** activity = earliest finish date for the **previous**
- When there is more than one previous activity, take the **latest** earliest finish



Example

- An activity network

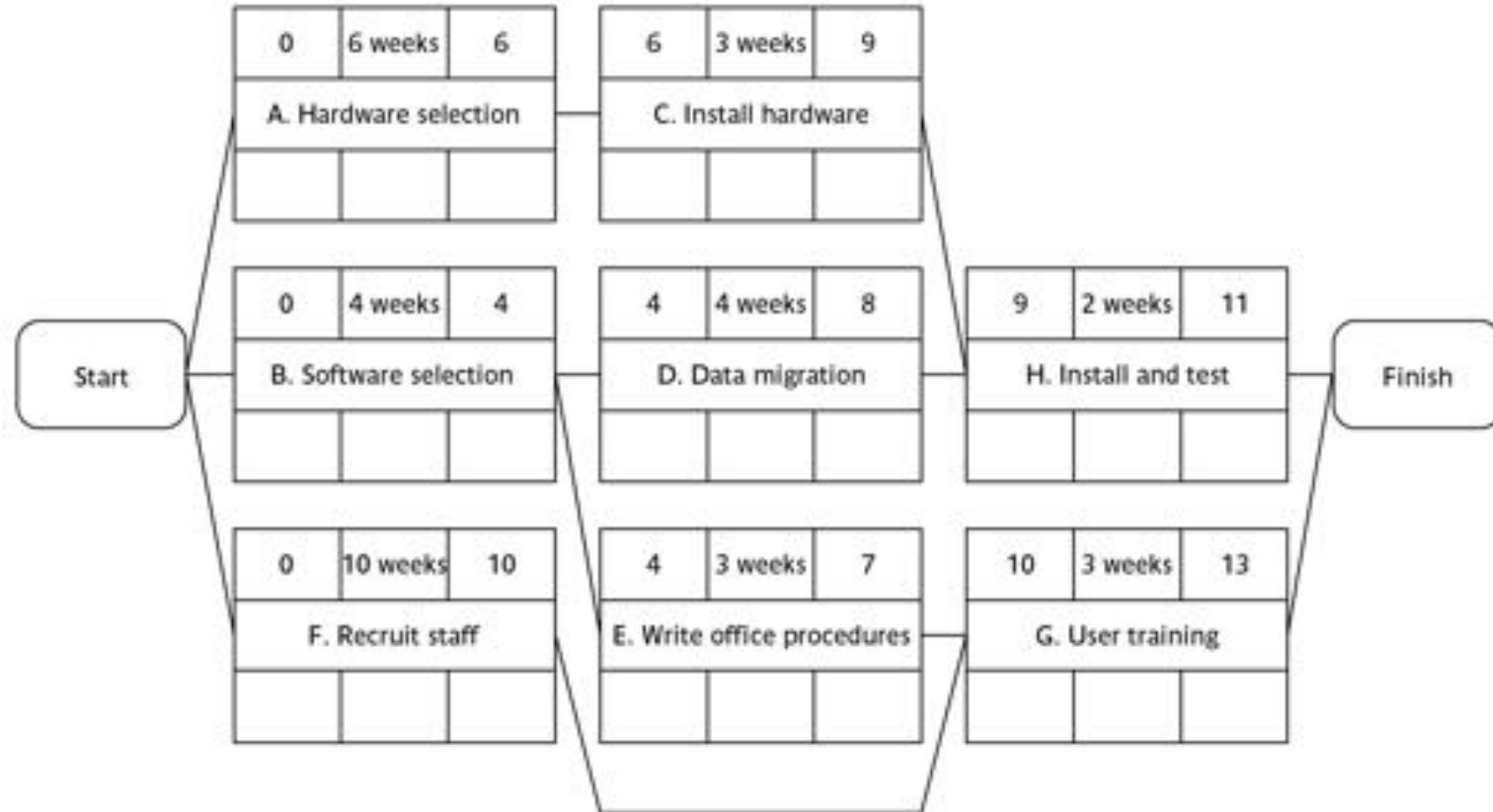


Complete the Table


Activity	ES	Duration	EF
A			
B			
C			
D			
E			
F			
G			
H			

Example

- Network after the forward pass



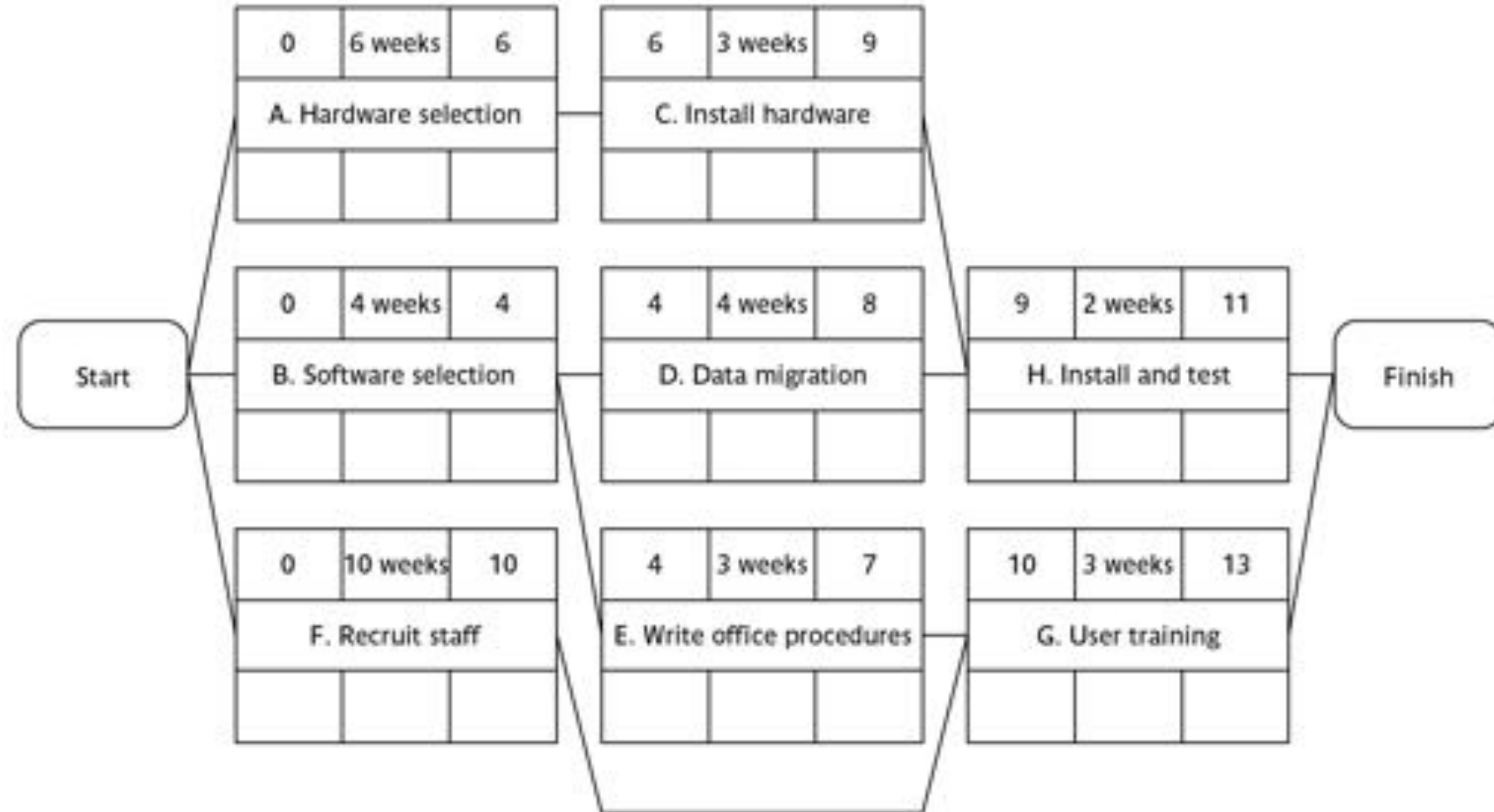
2nd Step: Backward Pass BACK

- Calculate the latest date at which each activity may be started and finished without delaying the end date of the project
 - Start from the **last** activity and work backwards
 -  BACK
- Latest finish (LF) for **last** activity = earliest finish (EF)
- Latest finish for **current** activity = Latest start for the **following**
 - More than one following activity - take the **earliest LS**



Exercise

- LS for All Activities?



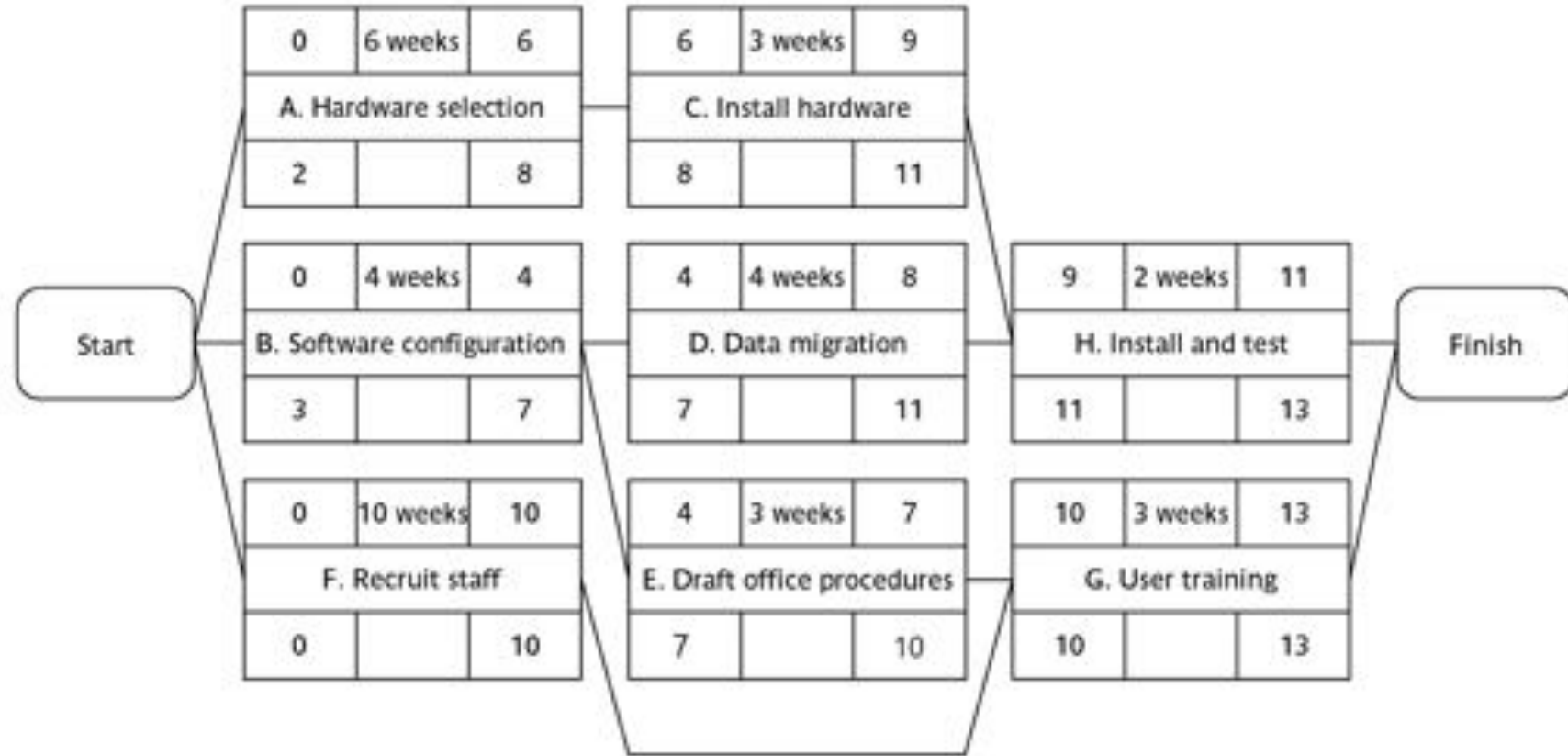


Complete the Table

Activity	ES	Dur	EF	LS	LF
A	0	6	6		
B	0	4	4		
C	6	3	9		
D	4	4	8		
E	4	3	7		
F	0	10	0		
G	10	3	10		
H	9	2	11		

Network After the Backward Pass

←
BACK

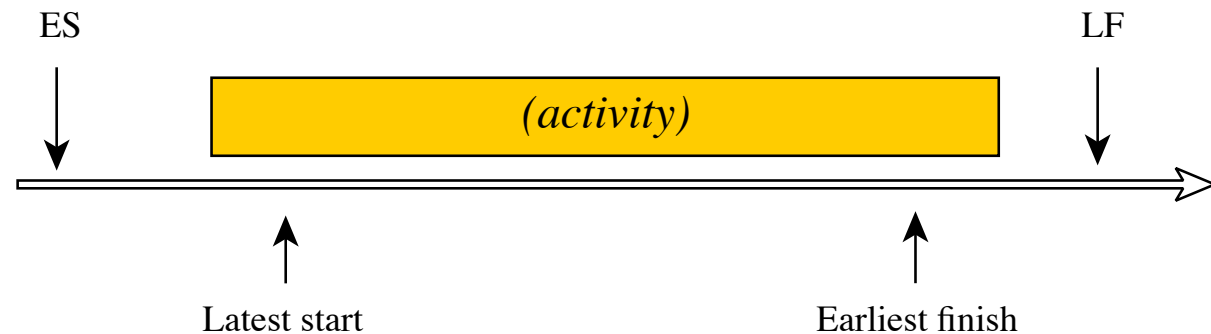


Float

- Difference between the earliest and latest start or finish dates



$$\text{Float} = \text{Latest finish} - \text{Earliest start} - \text{Duration}$$





Complete the Table

Activity	ES	Dur	EF	LS	LF	Float
A	0	6	6	2	8	
B	0	4	4	3	7	
C	6	3	9	8	11	
D	4	4	8	7	11	
E	4	3	7	7	10	
F	0	10	0	0	10	
G	10	3	10	10	13	
H	9	2	11	11	13	

Other Measures of Activity Float

- Although the total float is shown for each activity, it really 'belongs' to a path through the network
- **Free float** - The time by which an activity may be delayed without affecting any subsequent activity

$$FF_i = ES_{i+1} - EF_i$$

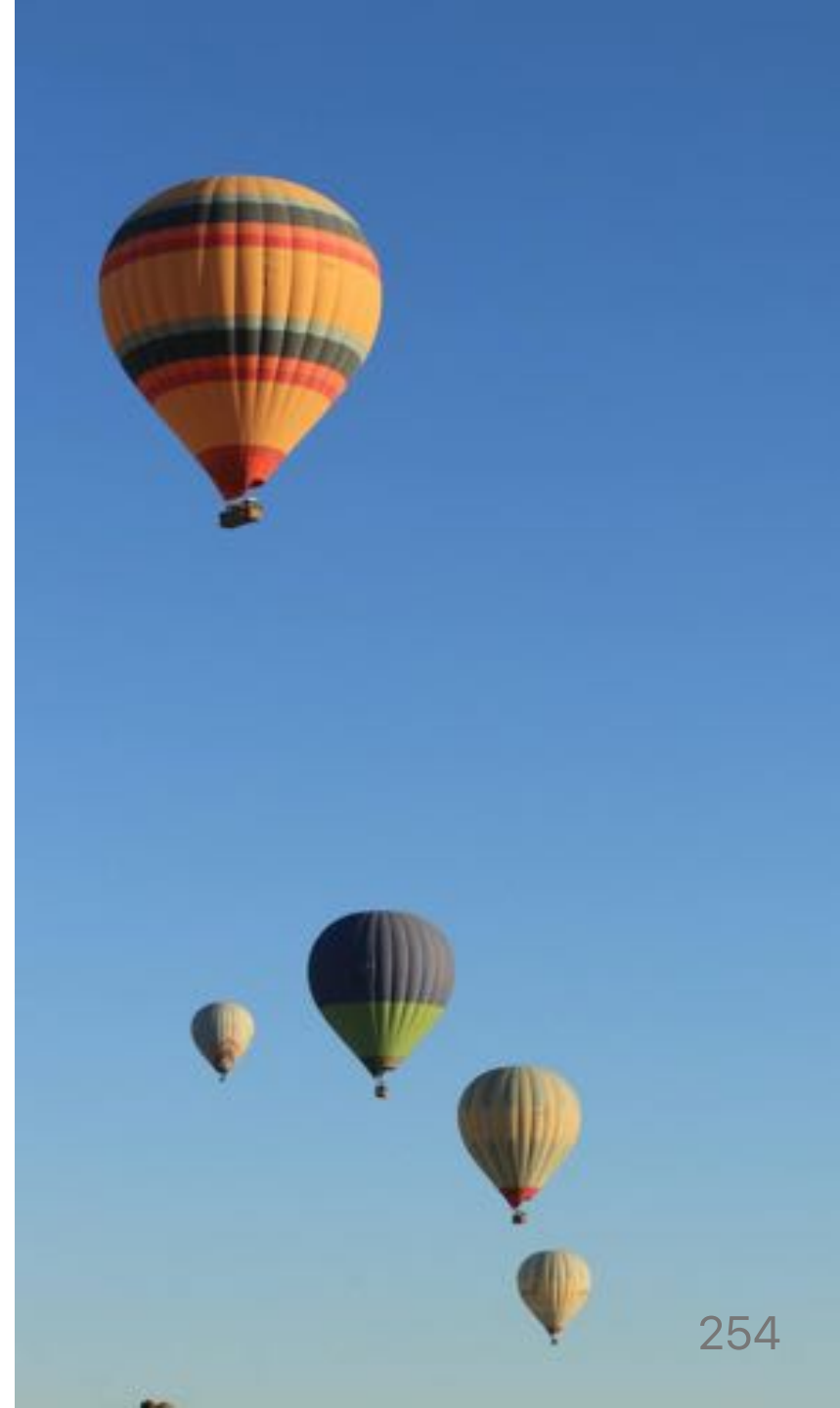
- **Interfering float** - How much the activity may be delayed without delaying the project's end date

$$IF = TF - FF$$



Total Float vs Free Float

- Float may occur when there are two or more activities happening concurrently
- Total float is shared between the activities in a **sequence**
 - Activities between a point of **path divergence** and **path convergence**
- Free float is only calculated on the last activity in an activity sequence

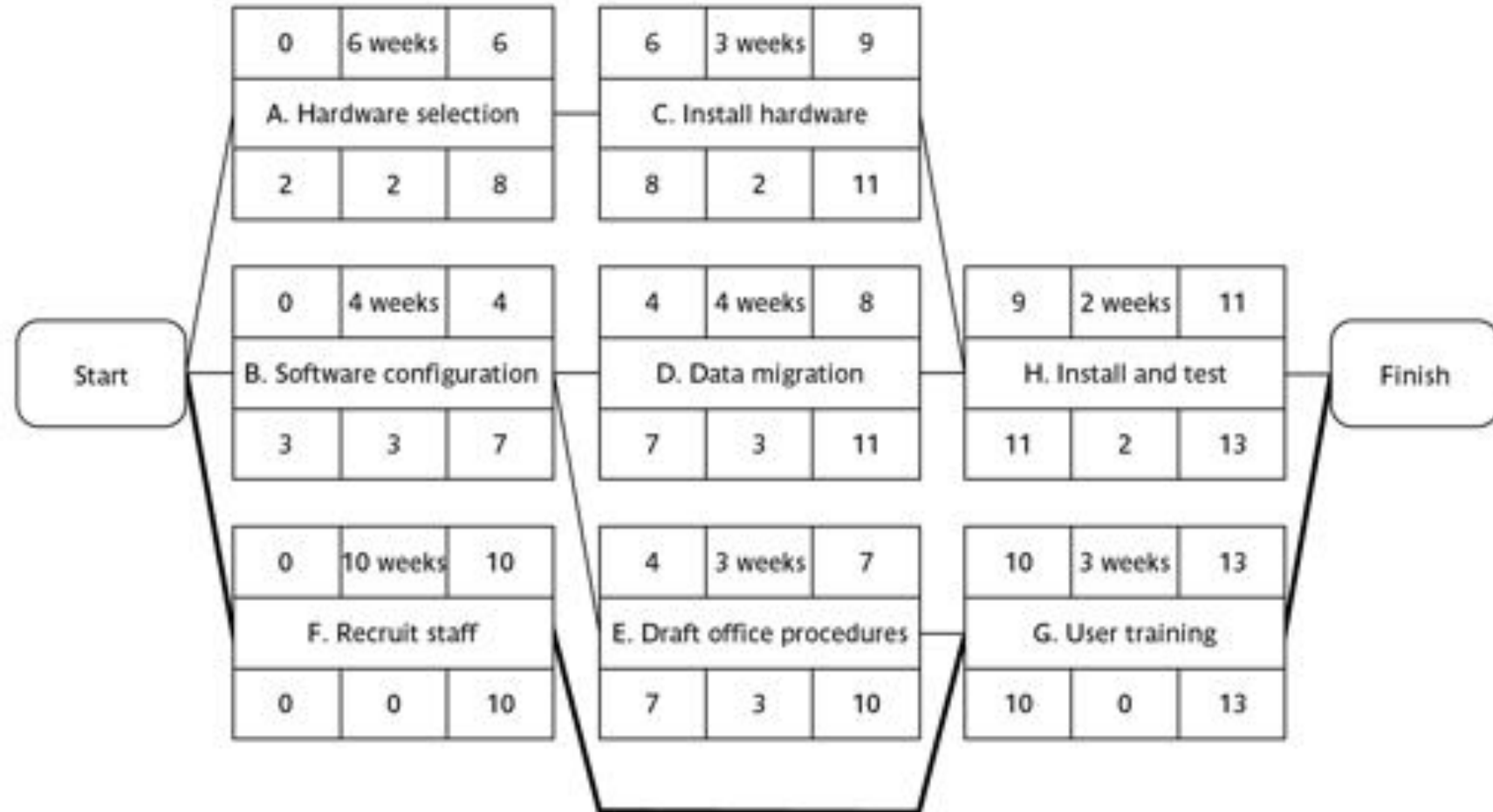


Critical Path

- Note the path through network with zero floats
- **Critical path:** any delay in an activity on this path will delay whole project
- Can there be more than one critical path?
- Can there be no critical path?
- Sub-critical paths



The Critical Path

Shortening the Project Duration

- To shorten the overall duration of a project attempt to reduce activity duration
 - Applying more resources to the task
 - The critical path indicates where we must look to save time
 - Check for any new critical path emerging
- There will come a point when we can no longer safely, or cost-effectively, reduce critical activity duration
- Questionate the logical sequencing of activities
 - Increasing the amount of parallelism in the network and the removal of bottlenecks



Identifying Critical Activities

- Activities that are not on the critical path may become critical
 - Use up some of their float
- Identify *near-critical* paths
 - Lengths 10-20% of the duration of the critical path
 - Total float of less than 10% of the project's uncompleted duration
- Cause of delays in completing the project
- Risk analysis, resource allocation and project monitoring



Conclusion

- Use of the critical path method to obtain an ideal activity path
- The plan tells us the order in which the activities should be executed and the earliest and latest they can start and finish
- Identify which activities are critical to meeting a target completion date
- Consider the resources allocated to each activity and its availability to schedule them: importance of the task and risks associated

SUMMARY

Risk Management

Subject Seven



Risk Management

This lecture will touch upon:

- Definition of **risk** and **risk management**
- Some ways of categorizing risk
- Risk management
- Risk identification – *What are the risks to a project?*
- Risk analysis – *Which ones are really serious?*
- Risk planning – *What shall we do?*
- Risk monitoring – *Has the planning worked?*
- We will also look at PERT risk and critical chains

Some Definitions of Risk

PRINCE2

The chance of exposure to the adverse consequences of future events

PM-BOK

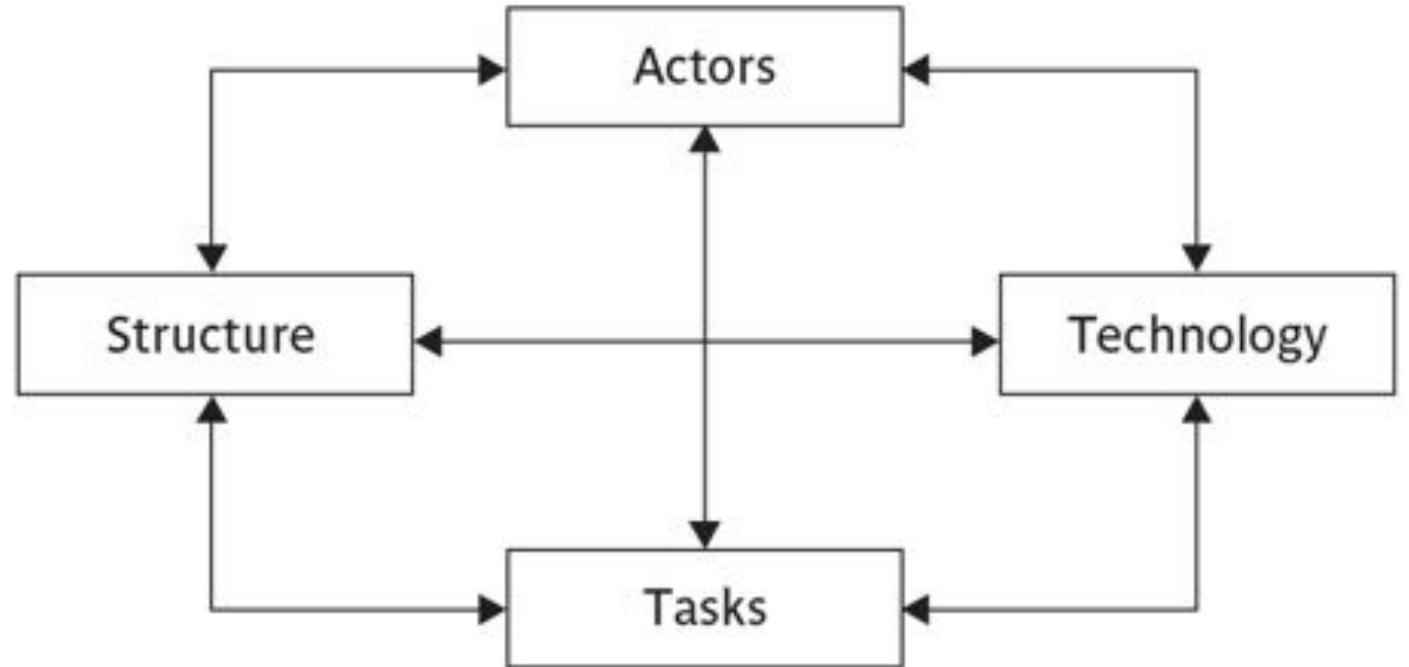
An uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives

- Risks relate to **possible future** problems, not current ones
- They involve a possible **cause** and its **effect(s)**

- If developer leaves, then task delayed

Categories of Risk

👉 Based on Lyytinen's *Sociotechnical Model of Risk*



A Framework for Dealing with Risk

The planning for risk includes these steps:

- 1** Risk identification – *What risks might there be?*
- 2** Risk analysis and prioritization – *Which are the most serious risks?*
- 3** Risk planning – *What are we going to do about them?*
- 4** Risk monitoring – *What is the current state of the risk?*



Risk Identification

Approaches to identifying risks include:

- Use of **checklists** – usually based on the experience of past projects
- **Brainstorming** – getting knowledgeable stakeholders together to pool concerns
- Causal mapping – identifying possible **chains of cause and effect**



Boehm's Top 10 Development Risks

Risk	Risk Reduction Techniques
Personnel shortfalls	Staffing with top talent; job matching; team building; training and career development; early scheduling of key personnel
Unrealistic time and cost estimates	Multiple estimation techniques; design to cost; incremental development; recording and analysis of past projects; standardization of methods



Boehm's Top 10 Development Risks (ii)

Risk	Risk Reduction Techniques
Developing the wrong software functions	Improved software evaluation; formal specification methods; user surveys; prototyping; early user manuals
Developing the wrong user interface	Prototyping; task analysis; user involvement



Boehm's Top Ten Risk (iii)

Risk	Risk Reduction Techniques
Gold plating	Requirements scrubbing, prototyping, design to cost
Late changes to requirements	Change control, incremental development
Shortfalls in externally supplied components	Benchmarking, inspections, formal specifications, contractual agreements, quality controls



Boehm's Top Ten Risk (iv)

Risk	Risk Reduction Techniques
Shortfalls in externally performed tasks	Quality assurance procedures, competitive design
Real time performance problems	Simulation, prototyping, tuning
Development technically too difficult	Technical analysis, cost-benefit analysis, prototyping, training



Risk Prioritization

Risk exposure

$$RE = \text{potential damage} \times \text{probability of occurrence}$$

Ideally

- **Potential damage:** a money value
 - e.g. a flood would cause 0.5€ millions of damage
- **Probability:** 0.00 (absolutely no chance) to 1.00 (absolutely certain)
 - e.g. 0.01 (one in hundred chance)

$$RE = 0.5M€ \times 0.01 = 5,000€$$

Crudely analogous to the amount needed for an



Risk Exposure Assessment

- Difficulties to estimate loss and probability with precision
- Use relative scales in the range 1 to 10

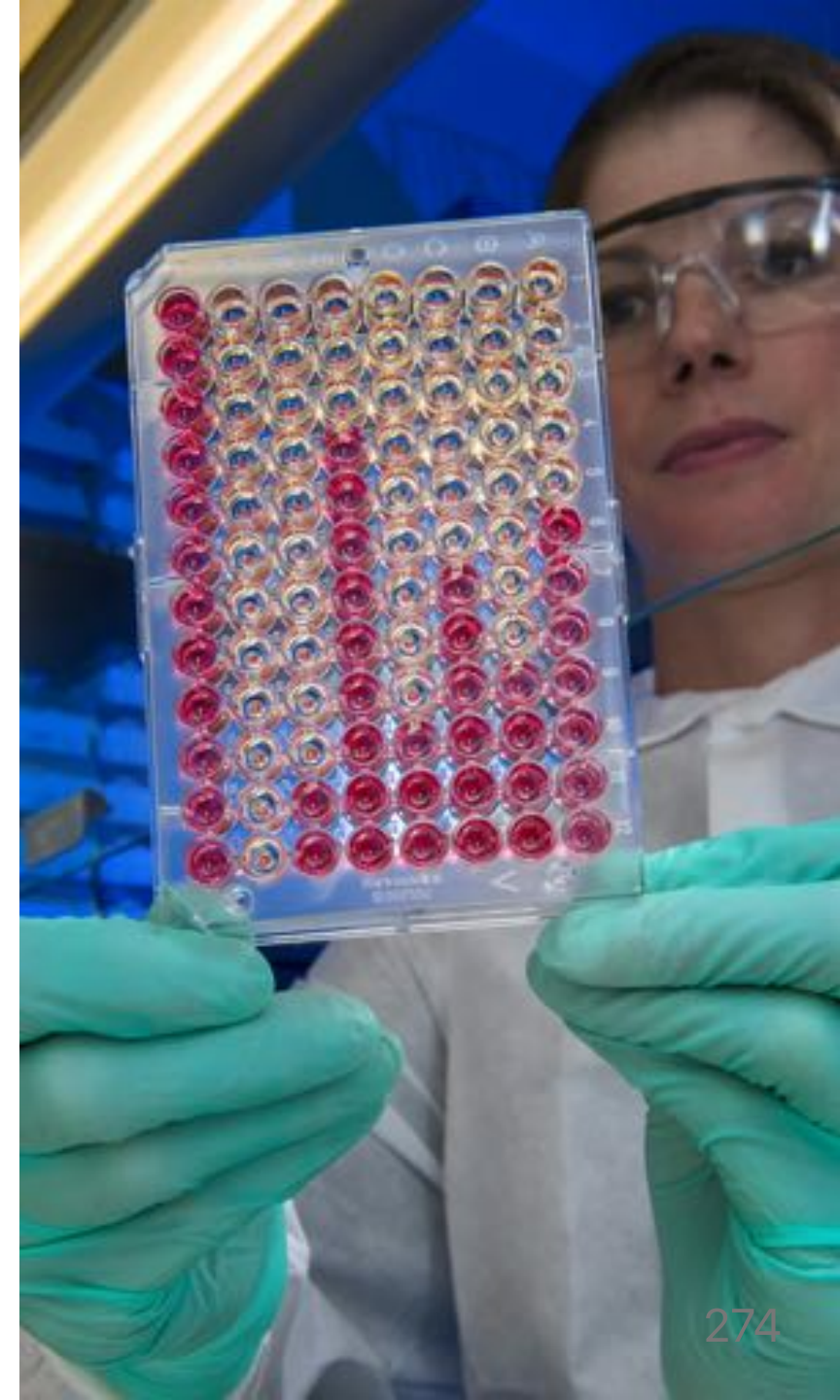
Ref	Hazard	Prob.	Impact	Risk
R1	Changes to requirements specification during coding	8	8	64
R2	Specification takes longer than expected	3	7	21

Risk Exposure Assessment (ii)

Ref	Hazard	Prob.	Impact	Risk
R3	Significant staff sickness affecting critical path activities	5	7	35
R4	Significant staff sickness affecting non-critical activities	10	3	30
R5	Module coding takes longer than expected	4	5	20
R6	Module testing demonstrates errors or deficiencies in design	4	8	32

Risk Probability: Qualitative Descriptors

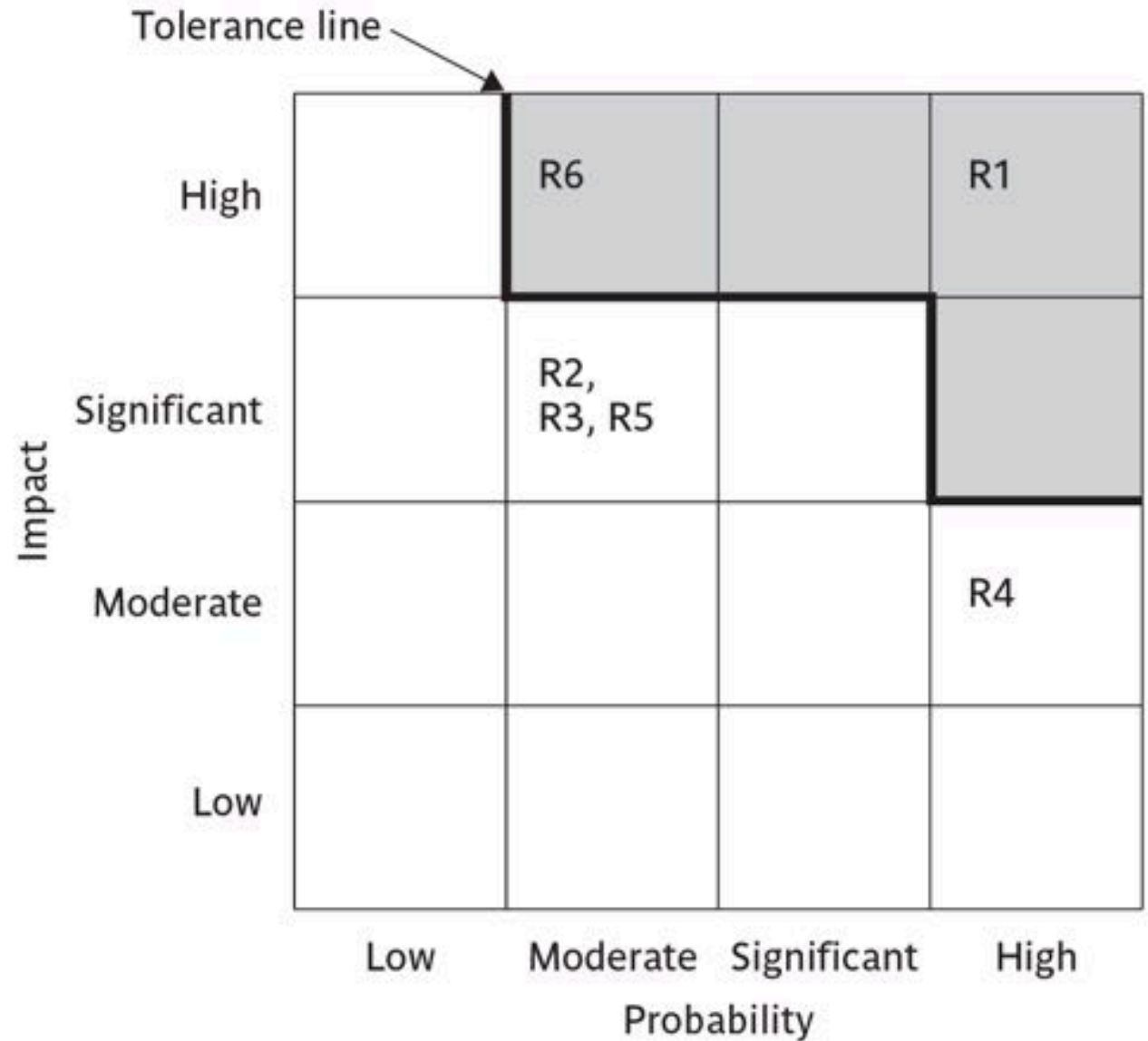
Probability Level	Range
High	Greater than 50% chance of happening
Significant	30-50% chance of happening
Moderate	10-29% chance of happening
Low	Less than 10% chance of happening



Qualitative Descriptors of Impact on Cost and Associated Range Values

Impact Level	Range
High	Greater than 30% above budgeted expenditure
Significant	20 to 29% above budgeted expenditure
Moderate	10 to 19% above budgeted expenditure
Low	Within 10% of budgeted expenditure

Probability Impact Matrix



Risk Planning

Risks can be dealt with by:

- Risk acceptance
- Risk avoidance
- Risk reduction
- Risk transfer
- Risk mitigation/contingency measures



Software Acquisition Risks

- Integration
- Upgrading
- No source code
- Supplier Failures or Buyouts

 Based on Fairley



Risk Management

- **Contingency plan** - A planned action to be carried out if the particular risk materializes
 - Cost 0 if the risk does not occur?
- Actions of the **risk reduction** incur some cost regardless of the risk materialized or not



Risk Reduction Leverage

$$RRL = \frac{(RE_{before} - RE_{after})}{\text{cost of risk reduction}}$$

- RE_{before} is risk exposure before risk reduction
 - e.g. 1% chance of a fire causing 200k€ damage
- RE_{after} is risk exposure after risk reduction
 - e.g. fire alarm costing 500€ reduces probability of fire damage to 0.5%
- $RRL = (1\% \text{ of } 200\text{k€}) - (0.5\% \text{ of } 200\text{k€}) / 500\text{€}$
 $= 2$
- As $RRL > 1.00$ therefore worth doing



Risk Register

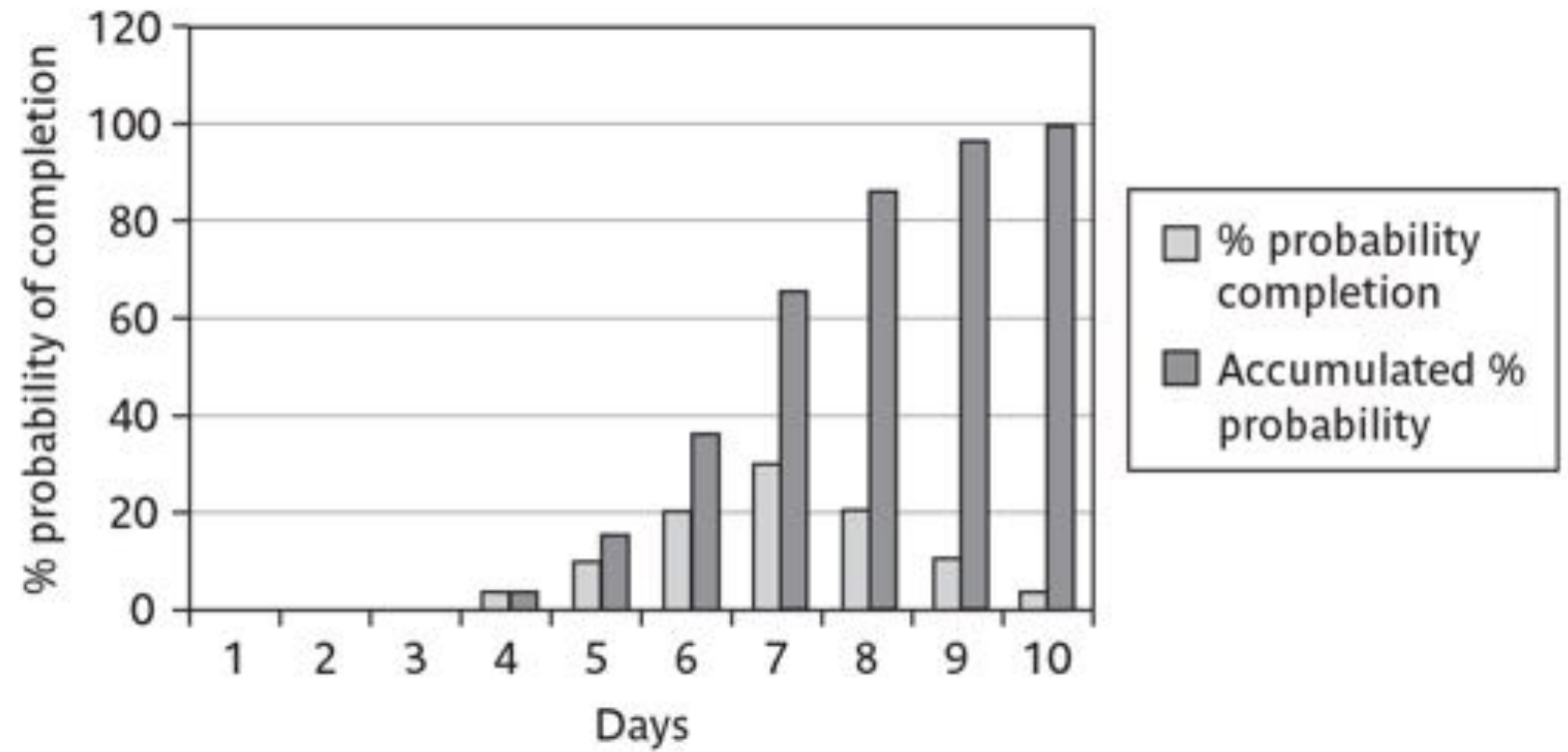
RISK RECORD				
Risk id	Risk title			
Owner	Date raised		Status	
Risk description				
Impact description				
Recommended risk mitigation				
Probability/impact values				
	Probability	Impact		
		Cost	Duration	Quality
Pre-mitigation				
Post-mitigation				
Incident/action history				
Date	Incident/action	Actor	Outcome/comment	

Evaluating Risk to the Scheduling

- Techniques which take account of the uncertainties in the durations of activities within a project
 - Pert
 - Monte Carlo simulation
- Drawback: tendency for developers to work to the schedule even if a task could be completed more quickly 🏃



Probability Chart



Using PERT to Evaluate the Effects of Uncertainty

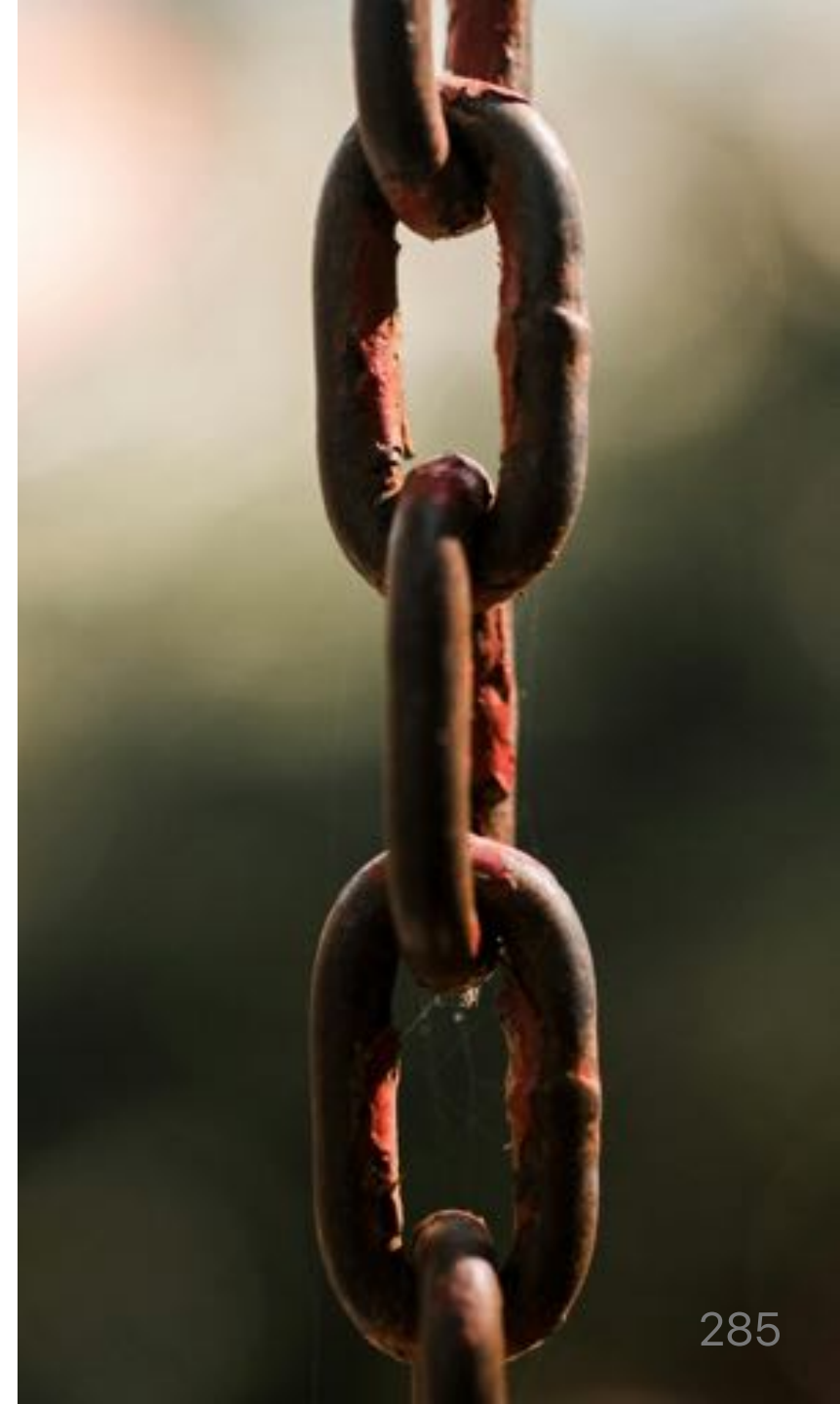
- Three estimates are produced for each activity
 - Most likely time (m)
 - Optimistic time (a)
 - Pessimistic (b)
- **Expected time:** $t_e = (a + 4m + b)/6$
- **Activity standard deviation:** $S = (b - a)/6$



A Chain of Activities



Task	<i>a</i>	<i>m</i>	<i>b</i>	<i>t_e</i>	<i>s</i>
A	10	12	16		
B	8	10	14		
C	20	24	38		



A Chain of Activities

- What would be the expected duration of the chain A + B + C?

Answer: $12.66 + 10.33 + 25.66 = 48.65$

- What would be the standard deviation for A + B + C?

Answer: $\sqrt{1^2 + 1^2 + 3^2} = 3.32$



Assessing the Likelihood of Meeting a Target

- Say the target for completing A+B+C was 52 days (T)
- Calculate the z value thus

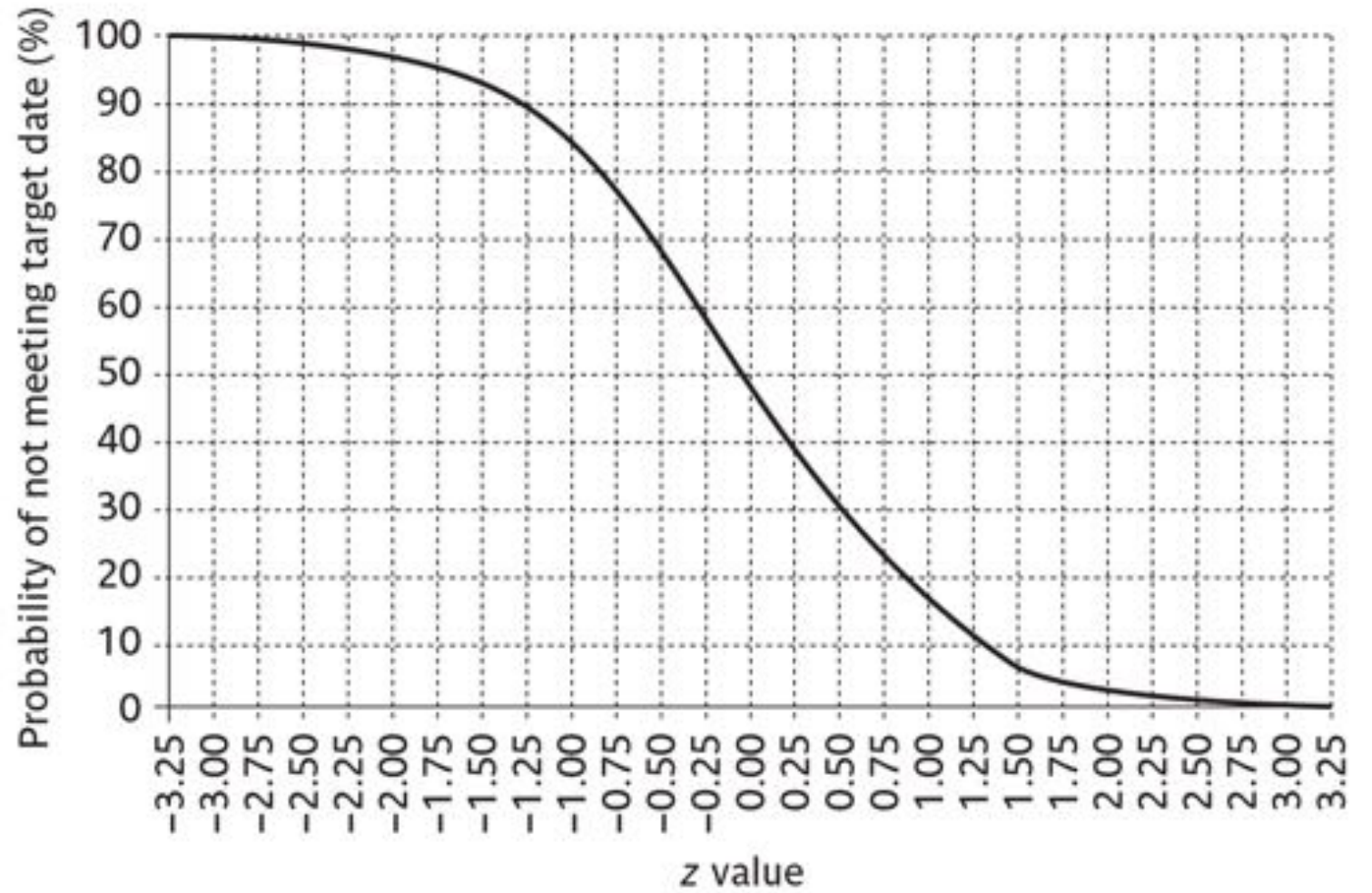
$$z = \frac{T - t_e}{s}$$

In this example $z = (52 - 48.65)/3.32 = 1.01$

- Look up in table of z values - see next overhead

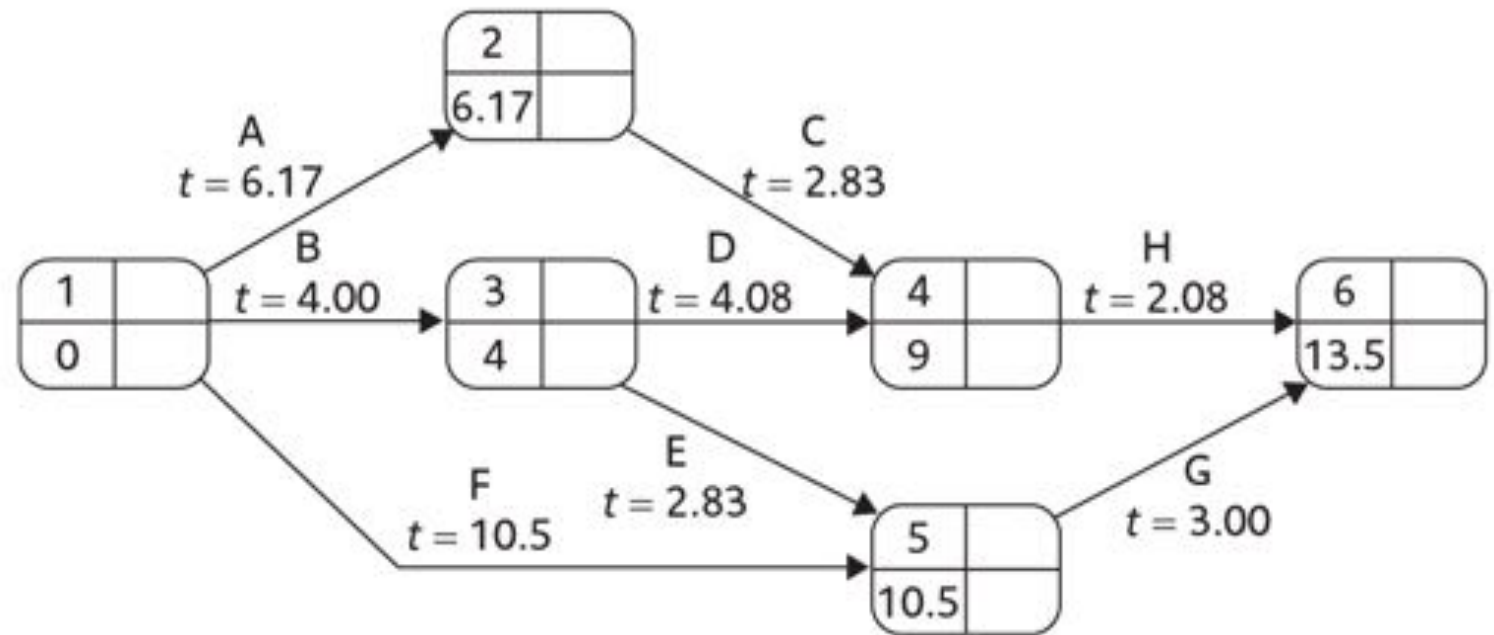


Graph of z Values



Using Expected Durations

PERT event	Label
Even num.	Target date
Te	S

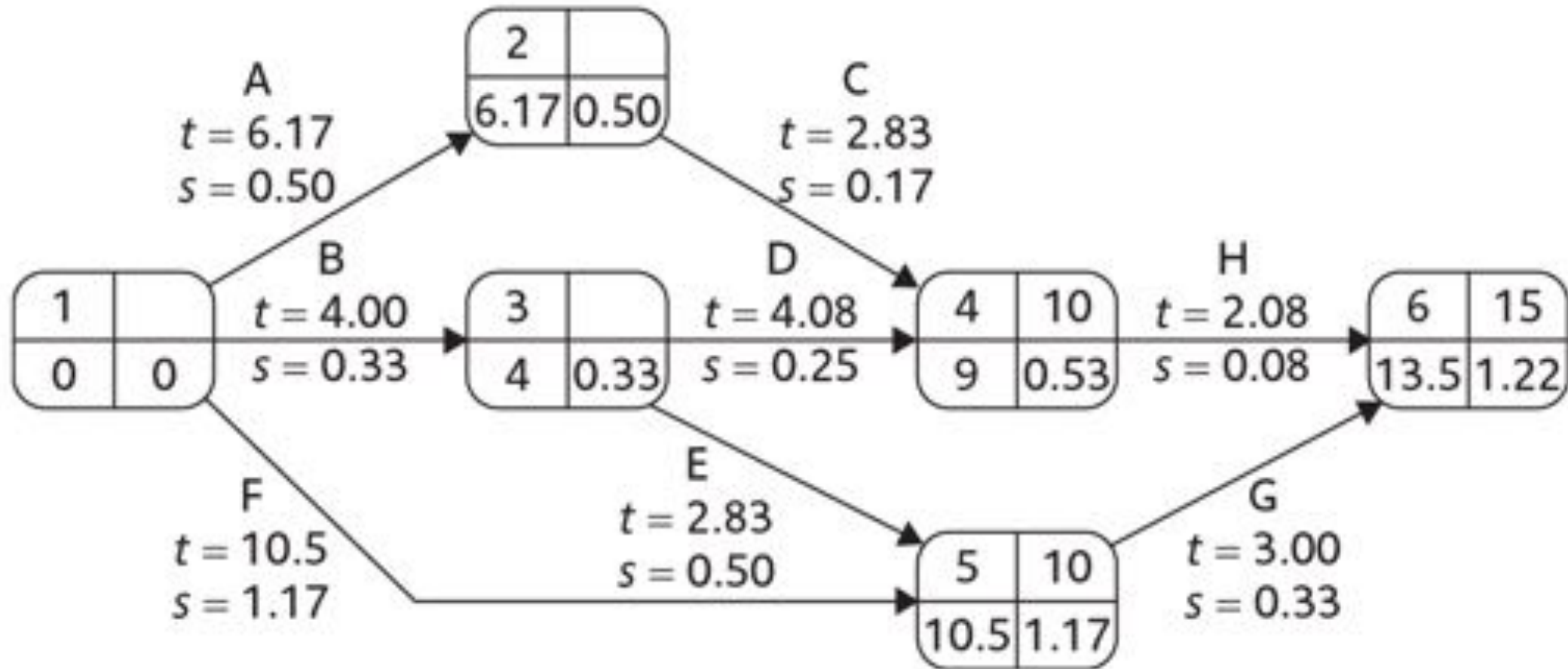


Expected Times and Std. Deviations

Activity	a	m	b	t_e	s
A	5	6	8	6.17	0.50
B	3	4	5	4.00	0.33
C	2	3	3	2.83	0.17
D	3.5	4	5	4.08	0.25
E	1	3	4	2.83	0.50
F	8	10	15	10.50	1.17
G	2	3	4	3.00	0.33
H	2	2	2.5	2.08	0.08



Assessing the Likelihood of Meeting a Target

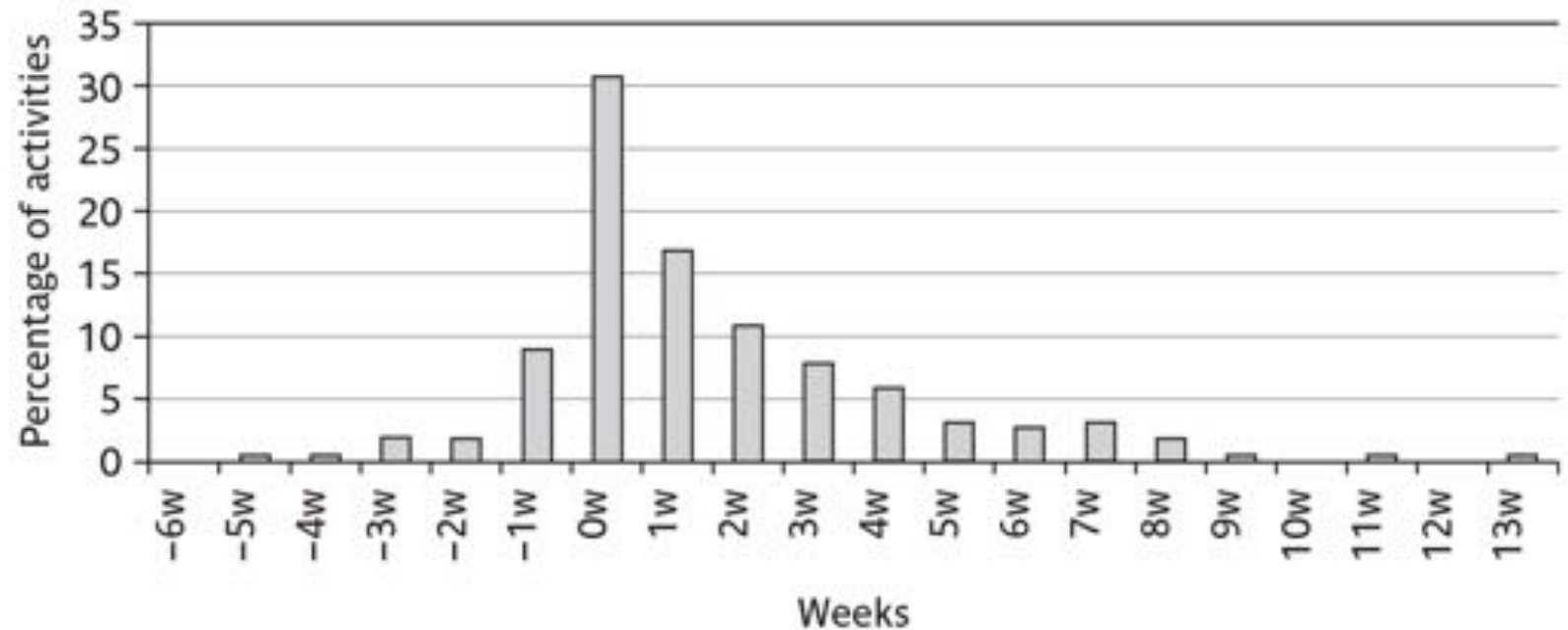


Exercise

- Calculate the z values for the events with target dates in the network and the probabilities of miss the target dates
- Find the probabilities of not achieving events 4 or 5 by their target dates of the end on week 10
- Calculate the likelihood of completing the project by week 14

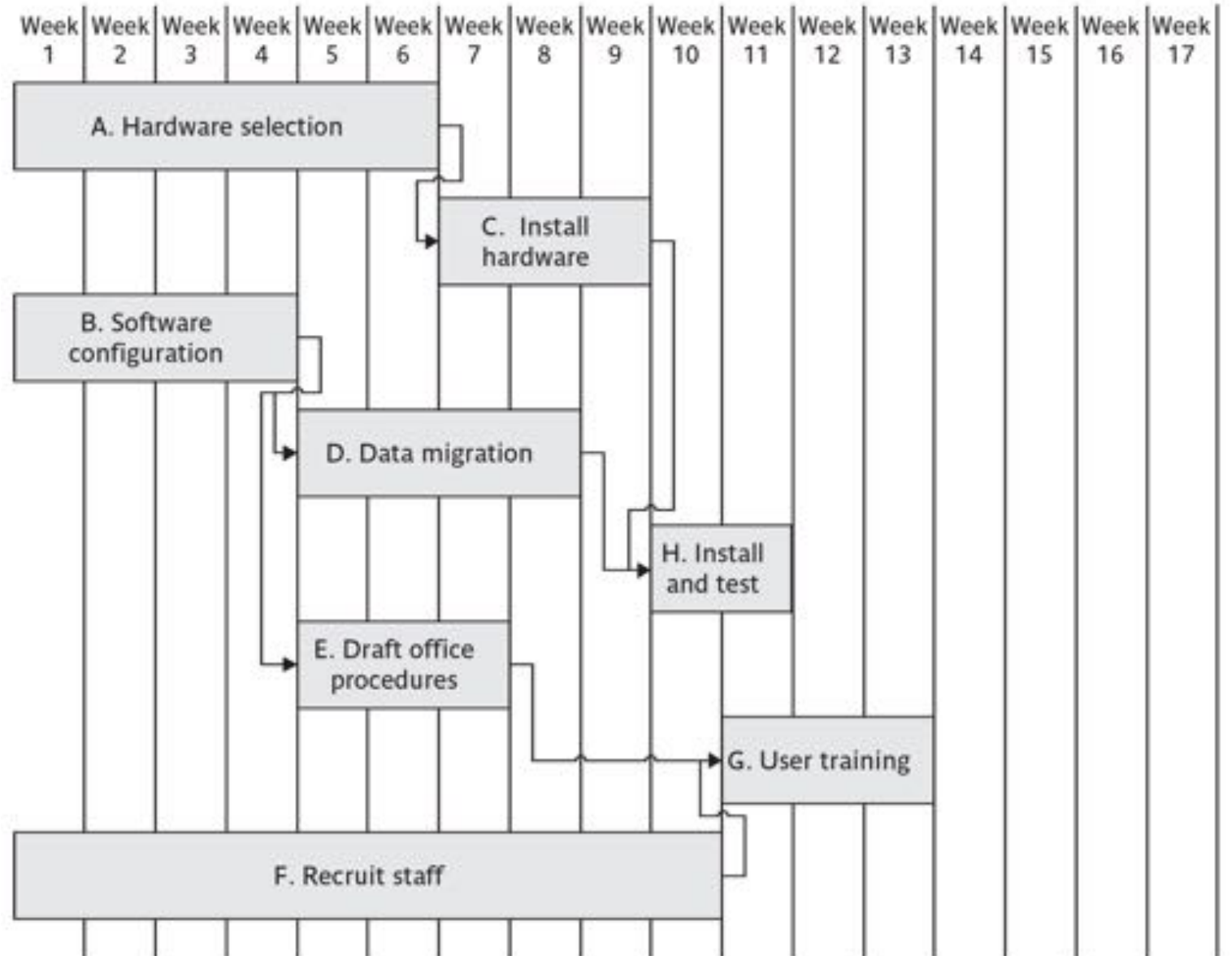
Critical Chain Concept

- Forecast of activity duration as a range of durations
- Choose one value in that range as the *target, the most likely*



Critical Chain Concept

Traditional planning approach



Critical Chain Approach

One problem with estimates of task duration:

- Estimators add a safety zone to estimate to take account of possible difficulties
 - Developers work to the estimate + safety zone, so time is lost
- No advantage is taken of opportunities where tasks can finish early – and provide a buffer for later activities



Critical Chain

1. Ask the estimators for two estimates

- Most likely duration: 50% chance of meeting this
- Comfort zone: additional time needed to have 95% chance

2. Schedule all activities using most likely values and starting all activities on **latest start dates**

- It does make every activity **critical**



Most Likely and Comfort Zone Estimates

Activity	Most Likely	Plus Comfort Zone	Comfort Zone
A	6	8	2
B	4	5	1
C	3	3	0
D	4	5	1



Most Likely and Comfort Zone Estimates (ii)

Activity	Most Likely	Plus Comfort Zone	Comfort Zone
E	3	4	1
F	10	15	5
G	3	4	1
H	2	2.5	0.5



Critical Chain (ii)

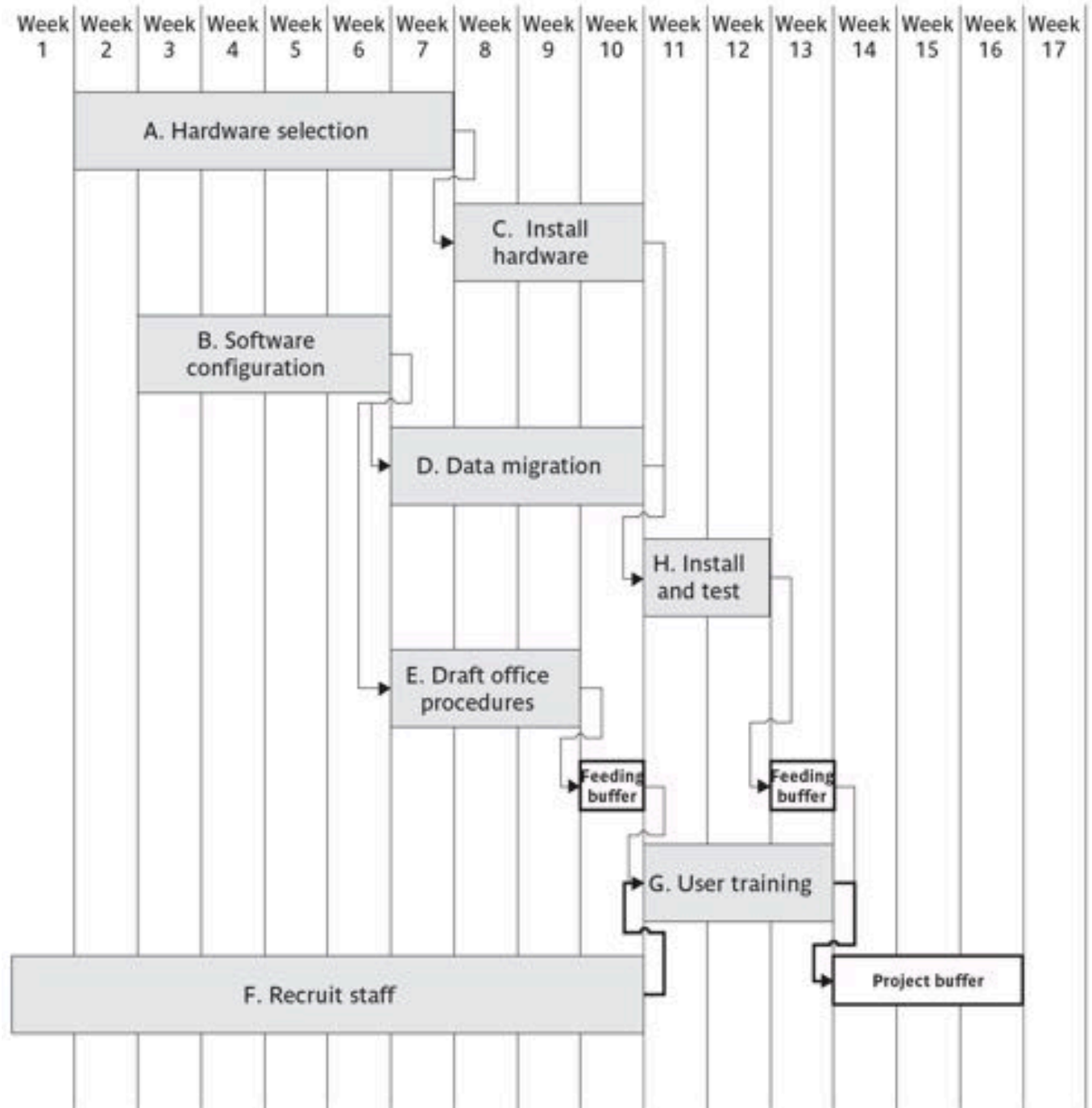
3. Identify the critical chain – longest chain of activities in the project, taking account of both task and resource dependencies
4. Put a project buffer at the end of the critical chain with duration 50% of sum of comfort zones of the activities on the critical chain.

Critical Chain (iii)

5. Where subsidiary chains of activities feed into critical chain, add feeding buffer
6. Duration of feeding buffer 50% of sum of comfort zones of activities in the feeding chain
7. Where there are parallel chains, take the longest and sum those activities



Plan Employing Critical Chain Concepts



Executing the Critical Chain-based Plan

- No **chain** of tasks is started earlier than scheduled, but once it has started is finished as soon as possible
 - This means the activity following the current one starts as soon as the current one is completed, even if this is early – the *relay race principle*



Executing the Critical Chain-Based Plan

- Buffers are divided into three categories:
 - ● Green: the first 33% → No action required
 - ● Amber : the next 33% → Plan is formulated
 - ● Red : last 33% → Plan is executed



Conclusion

- How to identify and manage the risk
- Assessing and prioritizing risks and drawing plans for addressing those risk before thy become problems
- Techniques for estimating the effect of risk on the project's activity network and schedule
- Many risk affecting software projects and be reduced by allocating more experienced staff on those activities affected

SUMMARY

Resource Allocation

Subject Eight



Introduction

The final result of resource allocation will normally be a number of schedules

- **Activity schedule** - indicating start and completion dates for each activity
- **Resource schedule** - indicating dates when resources needed + level of resources
- **Cost schedule** - showing accumulative expenditure

👉 The project manager must concentrate on those resources which, without planning, might not be available when required



Resources

- These include
 - Labour 
 - Equipment (e.g. workstations) 
 - Materials 
 - Space 
 - Services  
- Time  - elapsed time can often be reduced by adding other resources
- Money  - used to buy the other resources



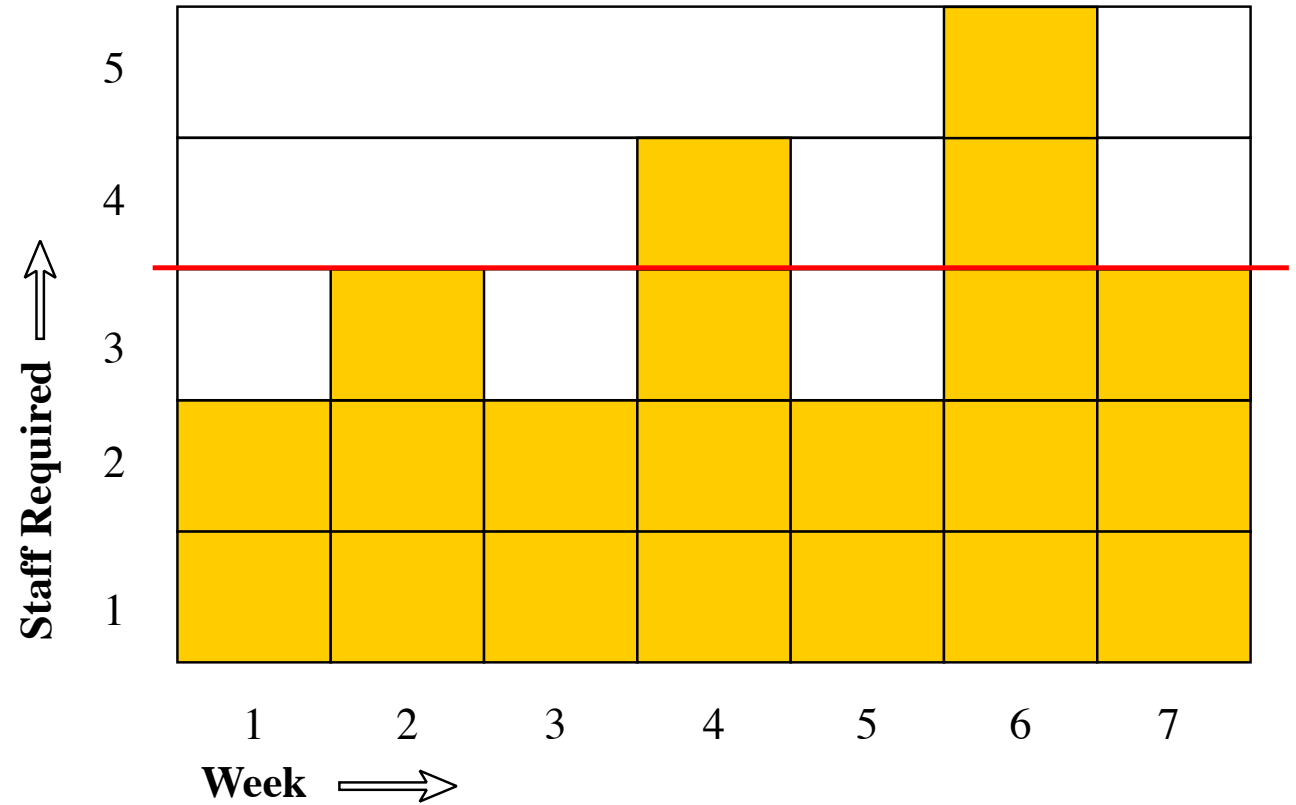
Resource Allocation

- Identify the resources needed for each activity and create a **resource requirement list** 📋
 - Resources required by each activity
 - Resources part of the infrastructure or required to support other resources
- Identify **resource types**
 - Individuals are interchangeable within the group
 - Different average productivity and cost




Resource Histogram

- Allocate resource types to activities and examine the **resource histogram**

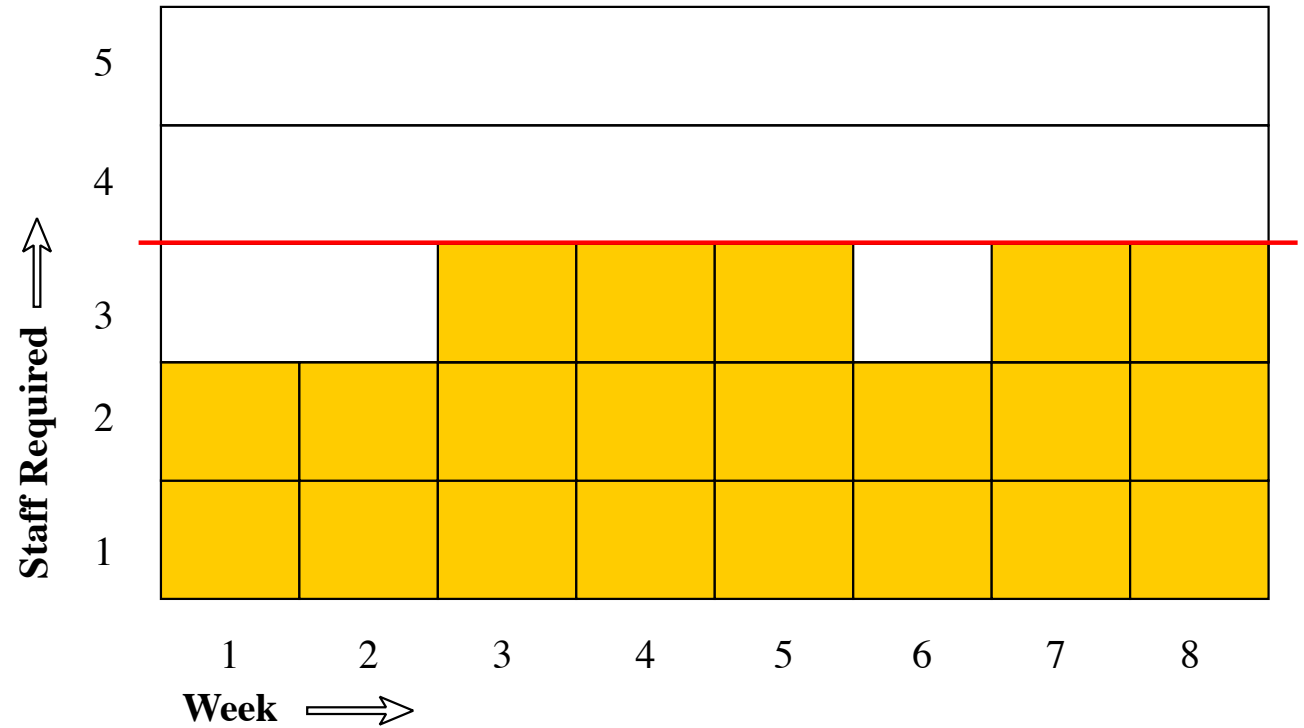


Resource Smoothing


- It is usually difficult to get specialist staff who will work odd days to fill in gaps
- Staff often have to be employed for a continuous block of time
- Therefore desirable to employ a constant number of staff on a project – who as far as possible are fully employed
- Hence need for **resource smoothing** 



Resource Smoothing








Resource Clashes

- Where same resource needed in more than one place at the same time 🤦
- Can be resolved by:
 - Delaying one of the activities 
 - Taking advantage of float to change start date
 - Delaying start of one activity until finish of the other activity that resource is being used on - puts back project completion



Resource Clashes (ii)

- Can be resolved by:
 - Moving resource from a non-critical activity 
 - Split activities to fill troughs in the demand for a resource
 - Usually increases time  
 - Bringing in additional resource - increases costs  



Prioritizing Activities

There are two main ways of doing this:

- **Total float priority** – those with the smallest float have the highest priority
 - Recalculate floats (and hence reorder the list) each time an activity is delayed
- **Ordered list priority** – this takes account of the duration of the activity as well as the float – see next overhead



Burman's Priority List

Give priority to:

- 1 Shortest critical activities
- 2 Other critical activities
- 3 Shortest non-critical activities
- 4 Non-critical activities with least float
- 5 Non-critical activities

Unfortunately, resource smoothing is not always possible within planned timescales

- Increase the available resource levels or alter working method



Resource Usage

- Need to maximize % usage of resources
 - i.e. reduce idle periods between tasks
- Need to balance costs against early completion date
- Need to allow for contingency



Critical Path

- Scheduling resources can create new dependencies between activities
- It is best not to add dependencies to the activity network to reflect resource constraints
 - Makes network very messy
 - A resource constraint may disappear during the project, but link remains on network
- Amend dates on schedule to reflect resource constraints



Counting the Cost

- Concentrated on trying to complete the project by the **earliest completion date** with the **minimum resources**
 - Constraints on when activities can be carried out and increases the risk of not meeting target dates
- Additional costs of employing extra resources would need to be compared to the costs of delayed delivery and the effects in the scope and quality



Allocating Individuals to Activities

- The initial *resource types* for a task have to be replaced by actual individuals
- Factors to be considered:
 - Availability
 - Criticality
 - Risk
 - Training
 - Team building – and motivation

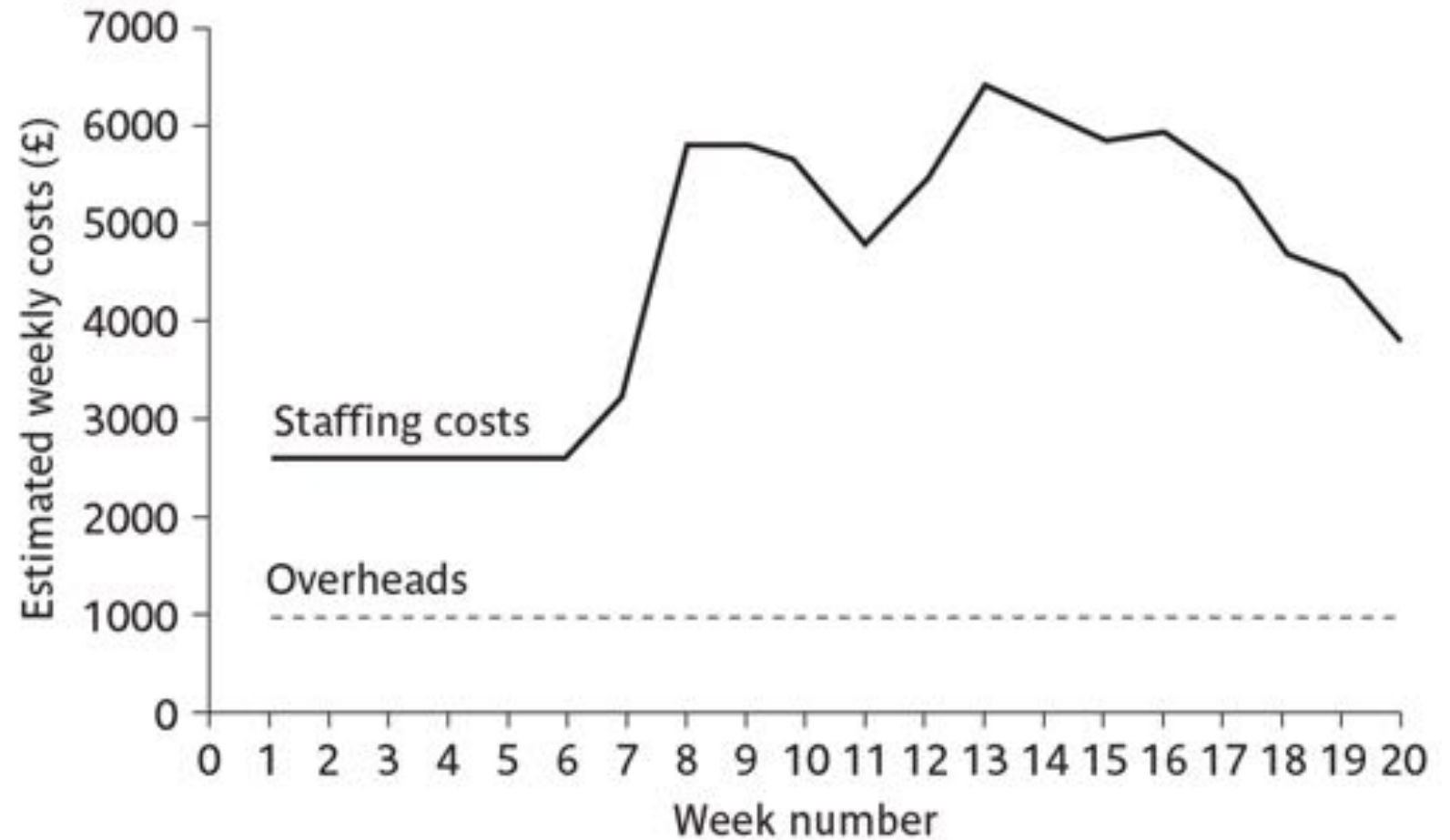


Cost Schedules

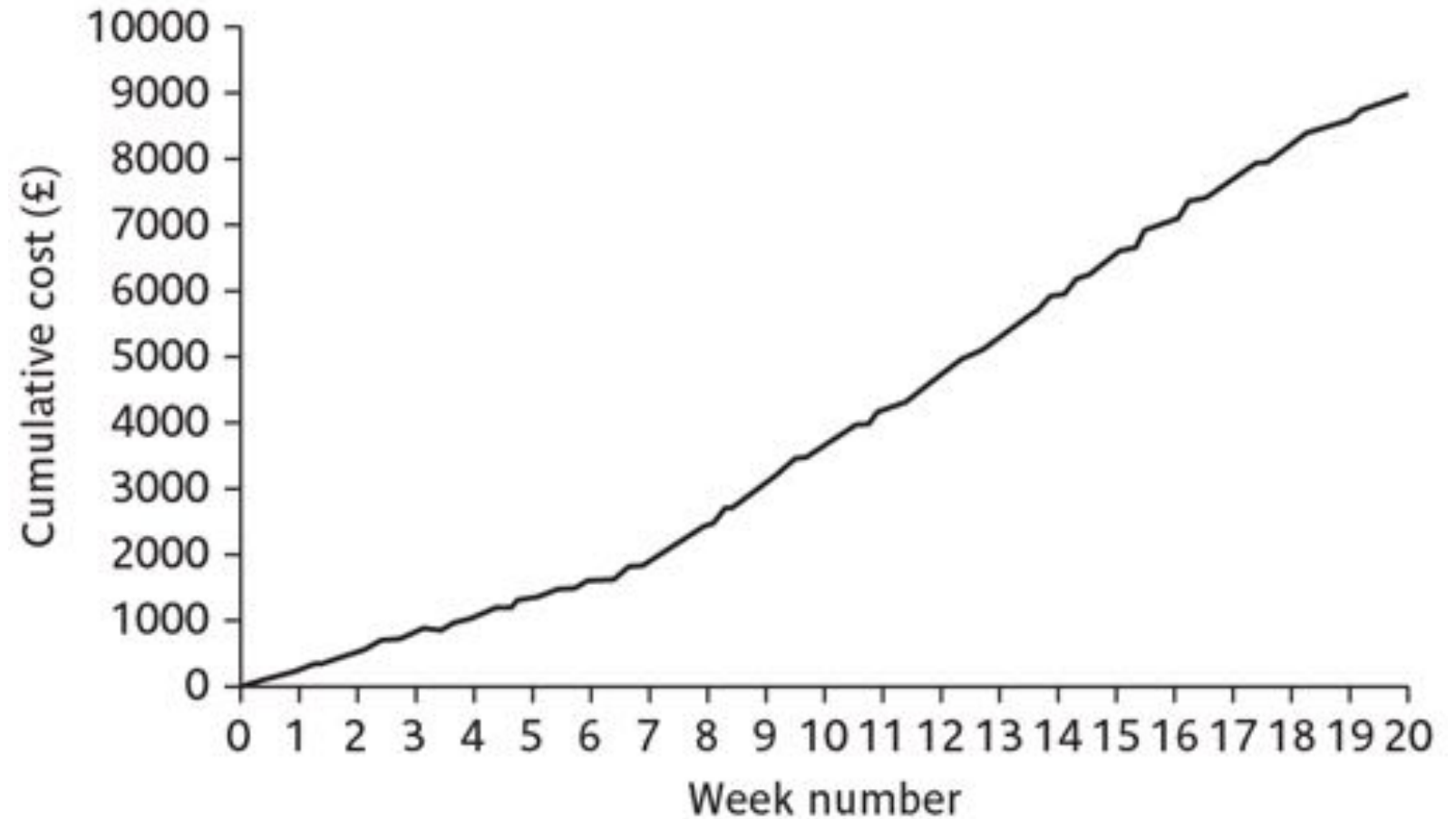
- Cost schedules can now be produced
- Costs include:
 - Staff costs
 - Overheads
 - Usage charges



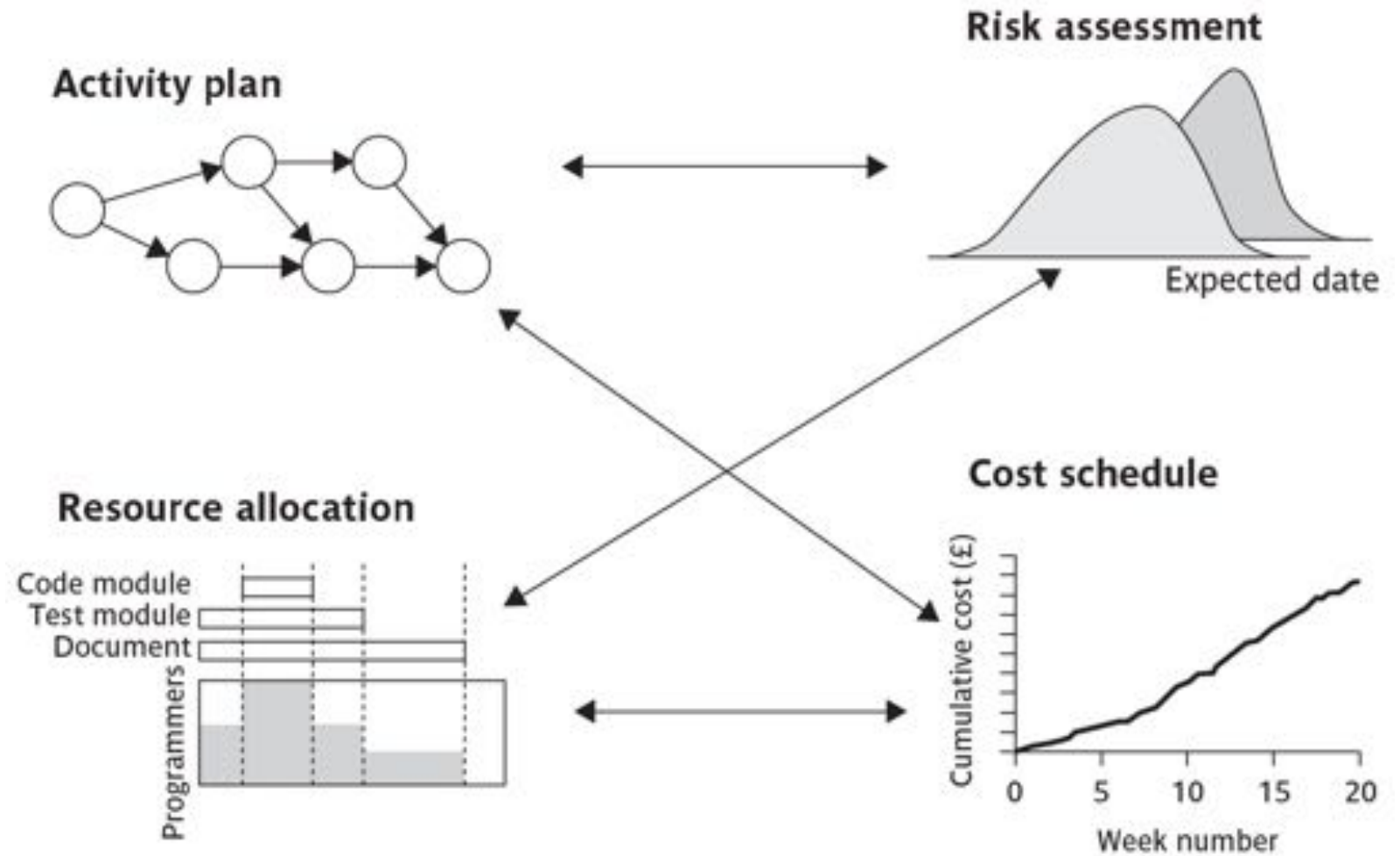
Cost Profile



Accumulative Costs



Balancing Concerns



Conclusion

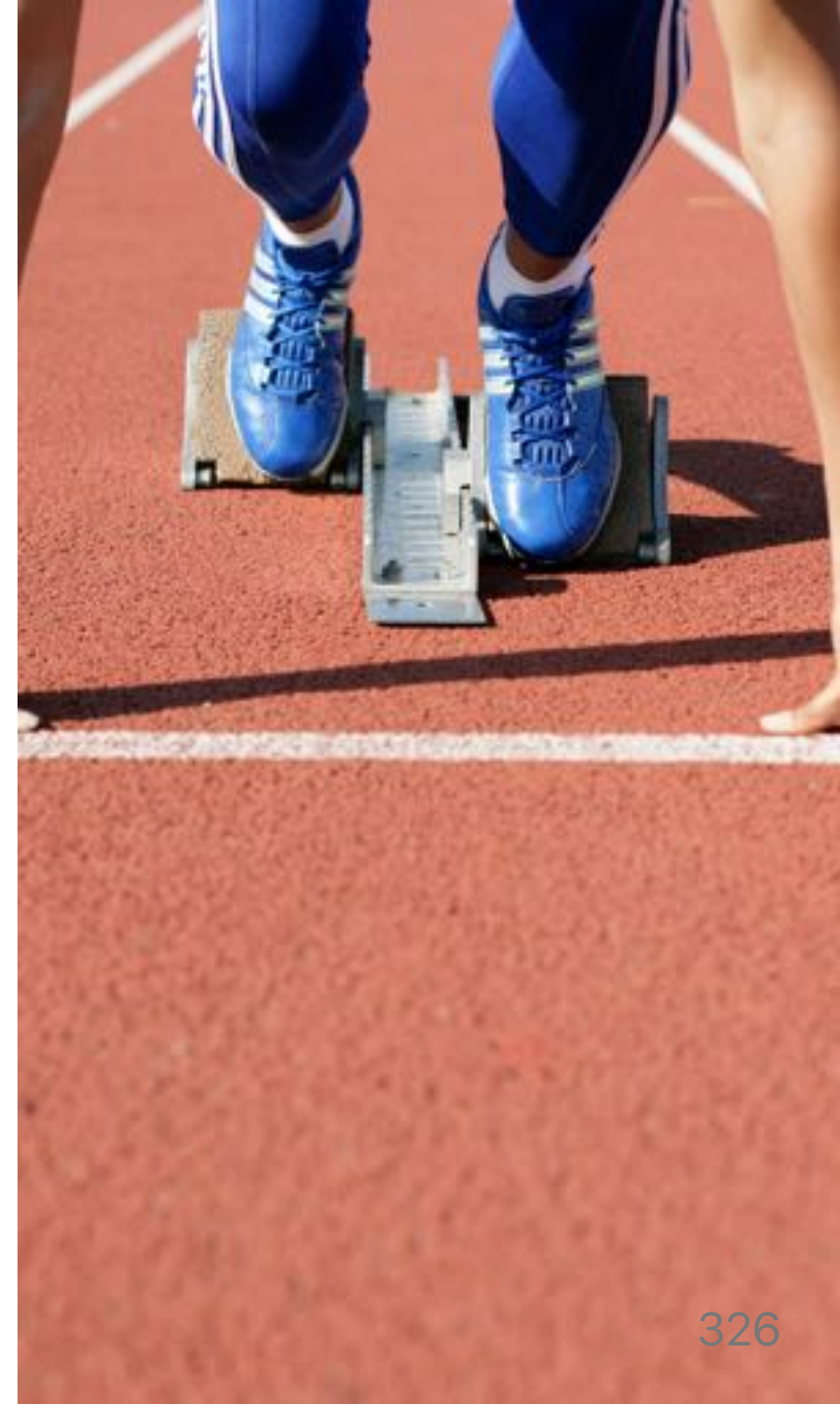
Conversion of an activity plan to a work schedule
allocating resources to project activities

- Identifying all the resources needed
- Arranging activity starts to minimize variations in resource levels
- Allocating resources to competing activities in a rational order of priority
- Taking care in allocating the right staff to critical activities

SUMMARY

Monitoring and Control

Subject Nine

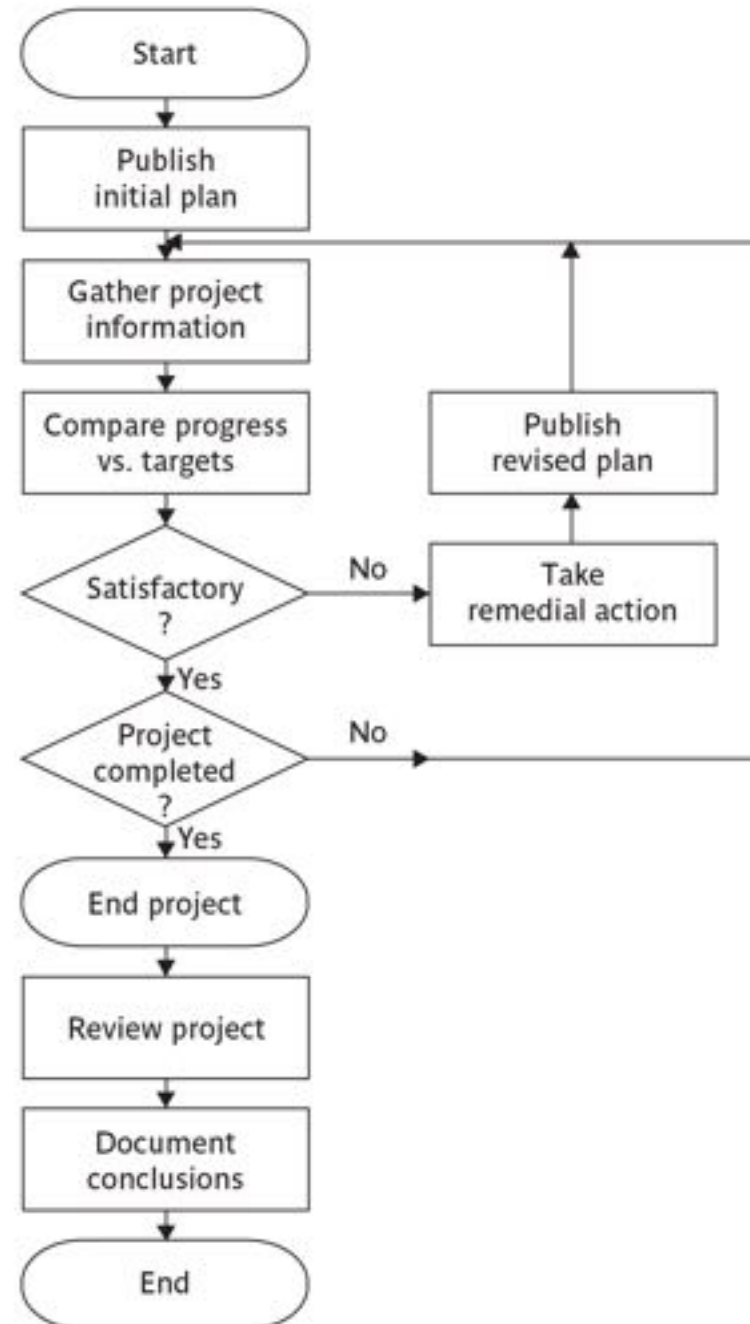


Introduction

- Schedules published and project started, be focused on progress
- Monitoring of what is happening
 - Comparing achievements against the schedule
 - Revision of plans or schedules where necessary
- Information gathering
- Actions to ensure project meets its targets



The Control Cycle 🧐

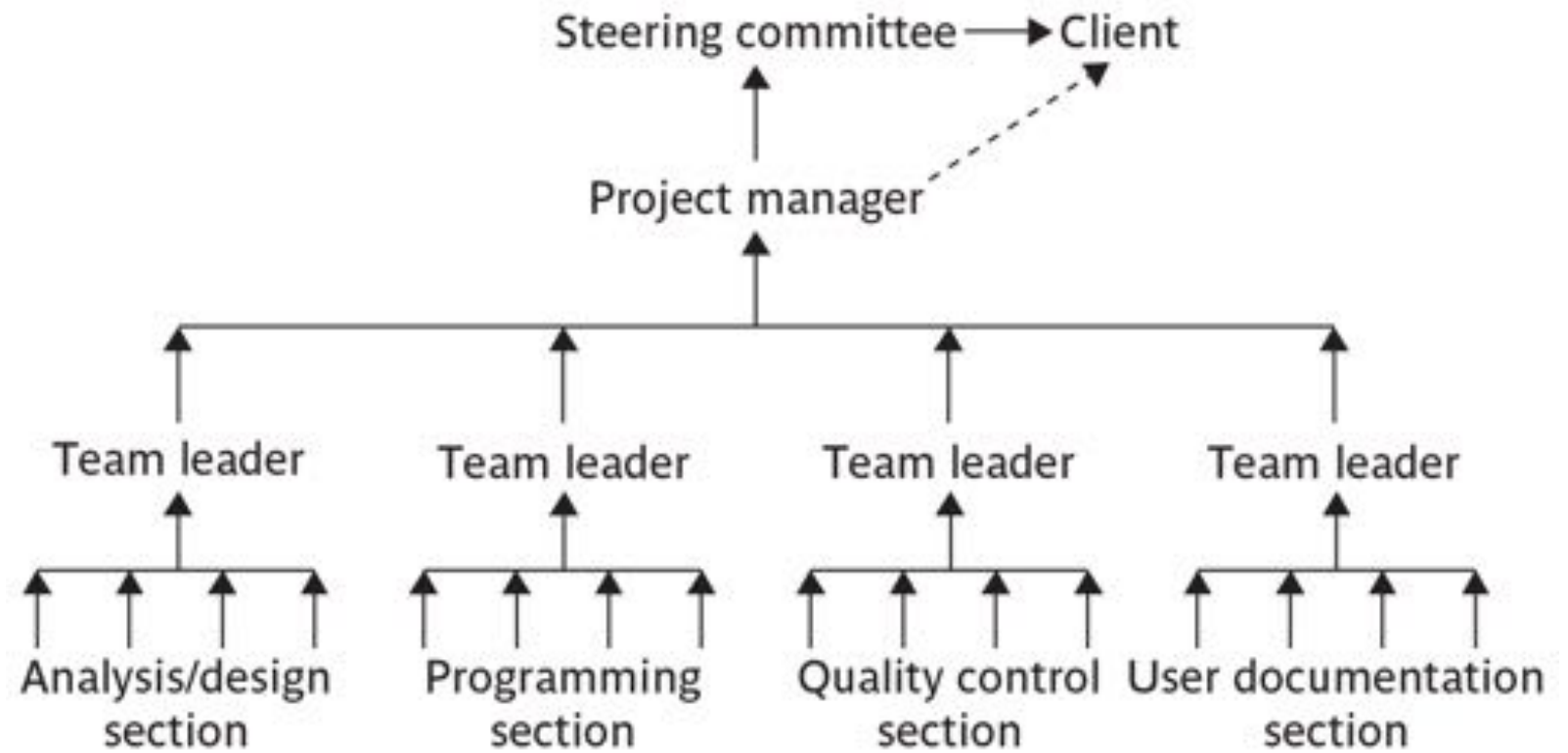


Shortfalls

- Delays in meeting target dates
- Shortfalls in quality
- Inadequate functionality
- Costs going over target



Responsibilities



Categories of Reporting

- **Oral formal regular:** Weekly or monthly progress meetings
 - Formal written minutes should be kept
- **Oral formal ad hoc:** End-of-stage review meetings
 - Likely to receive and generate written reports
- **Oral informal ad hoc:** Canteen discussion, social interaction
 - Early warning, must be backed up by formal reporting



Categories of Reporting (ii)

- **Written formal regular:** Job sheets, progress reports
 - Normally weekly using forms
- **Written formal ad hoc:** Exceptions reports, change reports



Assessing Progress

Checkpoints – predetermined times when progress is checked

- Event driven: check takes place when a particular event has been achieved
 - Production of a deliverable
- Time driven: date of the check is pre-determined
 - Regular, monthly



Taking Snapshots

- **Frequency** of reporting will depend upon the size and degree of risk of the project
- The higher the management level the longer the gaps between checkpoints
 - Team leader - daily
 - Project manager - weekly
- Individual developers - formal weekly collection of information
- Major progress reviews at particular points during the life of a project
 - **Review points or control points.**



Collecting Progress Details

- Need to collect data about
 - Achievements
 - Costs
- A big problem: how to deal with **partial completions**
 - *99% completion syndrome*
- Possible solutions
 - Control of products, not activities
 - Subdivide into lots of sub-activities



Weekly Timesheet

Time Sheet

Staff John Smith

Week ending 30/3/07

Rechargeable hours

Project	Activity code	Description	Hours this week	% complete	Scheduled completion	Estimated completion
P21	A243	Code mod A3	12	30	24/4/07	24/4/07
P34	B771	Document take-on	20	90	6/4/07	4/4/07

Total recharged hours	32
-----------------------	----

Non-rechargeable hours

Code	Description	Hours this week	Comment and authorization
Z99	Day in lieu	8	Authorized by RB

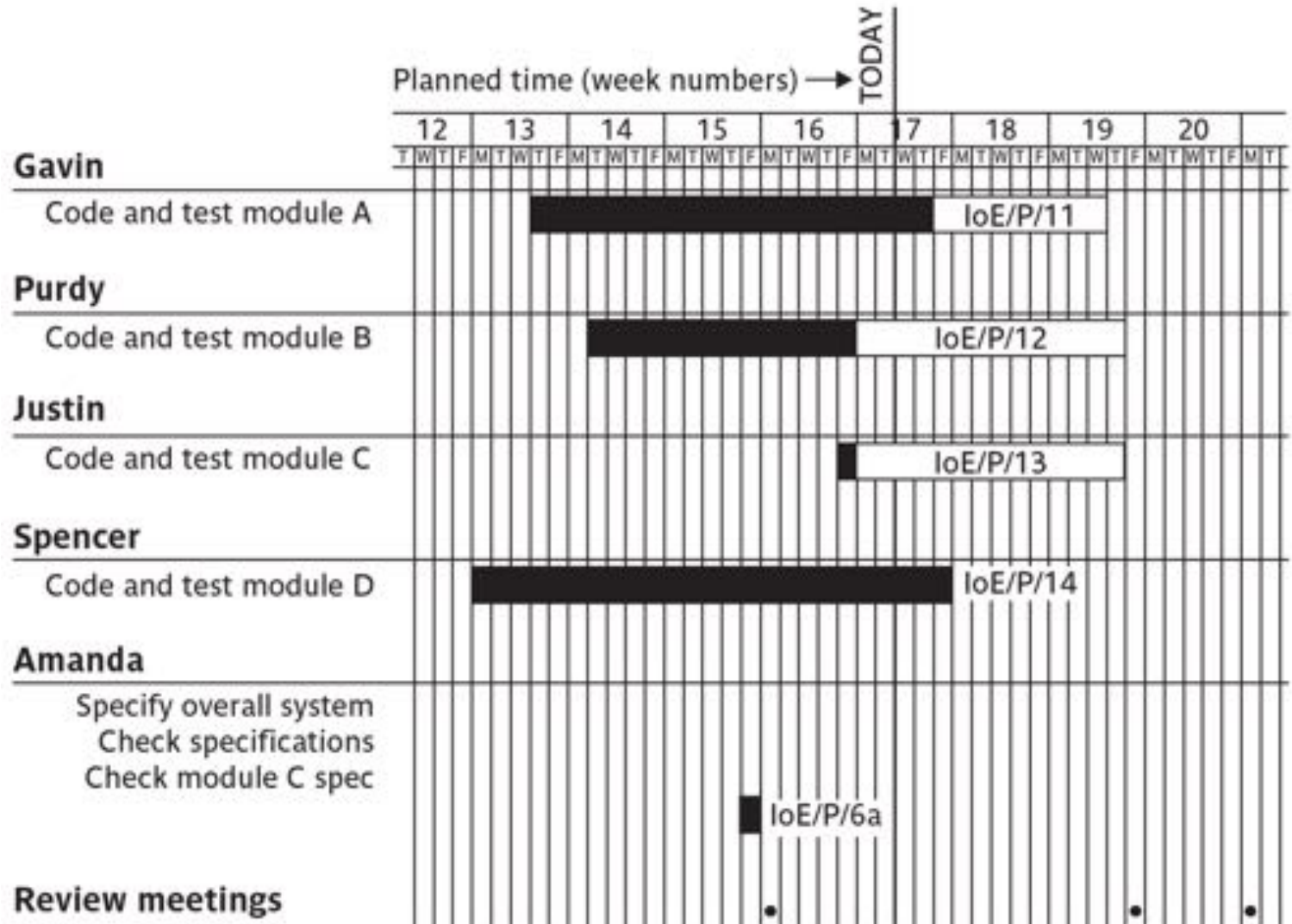
Total non-rechargeable hours	8
------------------------------	---

Red/Amber/Green (RAG) Reporting

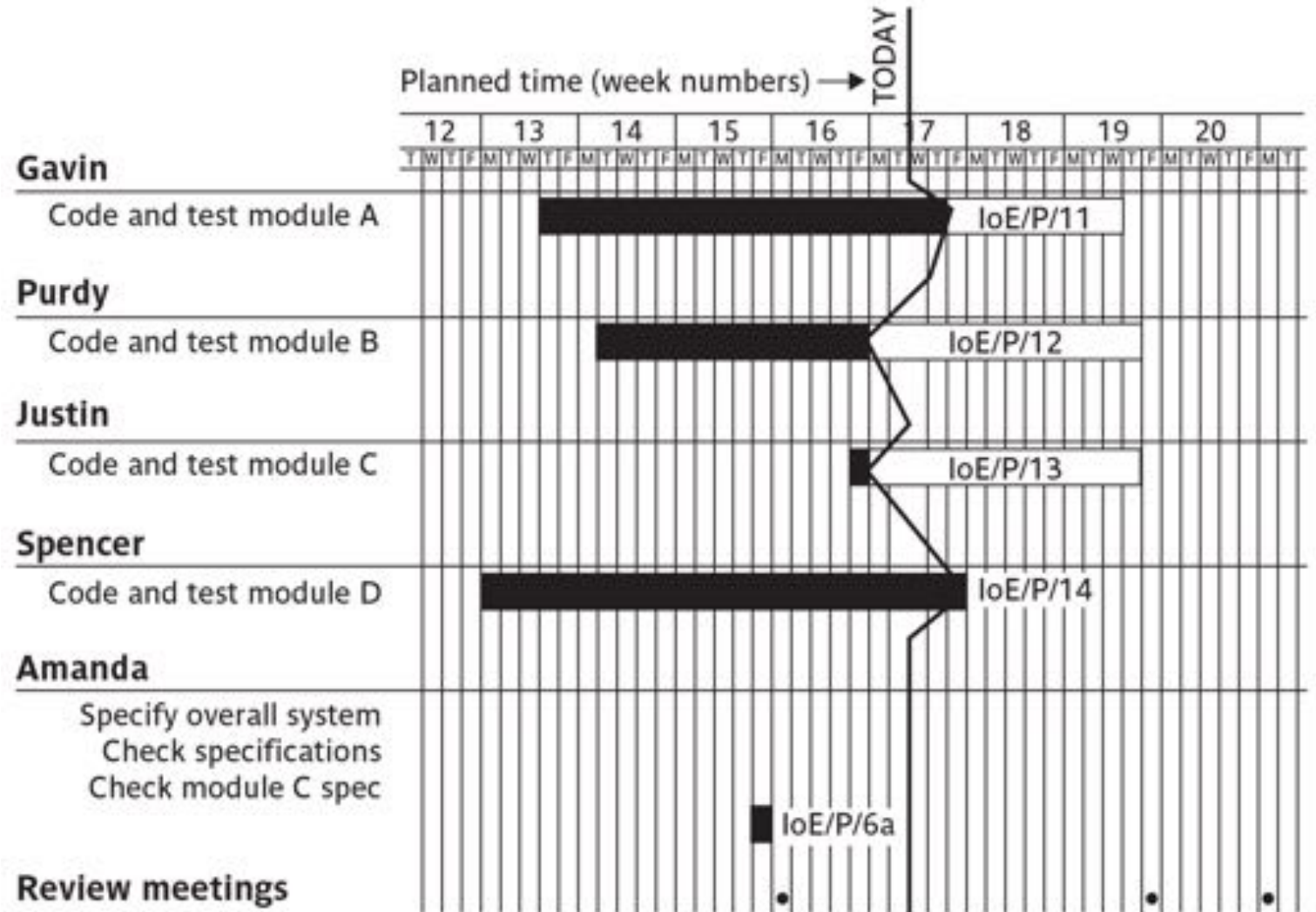
- Identify key tasks
- Break down into sub-tasks
- Assess subtasks as:
 - ● Green – *on target*
 - ● Amber – *not on target but recoverable*
 - ● Red – *not on target and recoverable only with difficulty*
- Status of **critical** tasks is particularly important



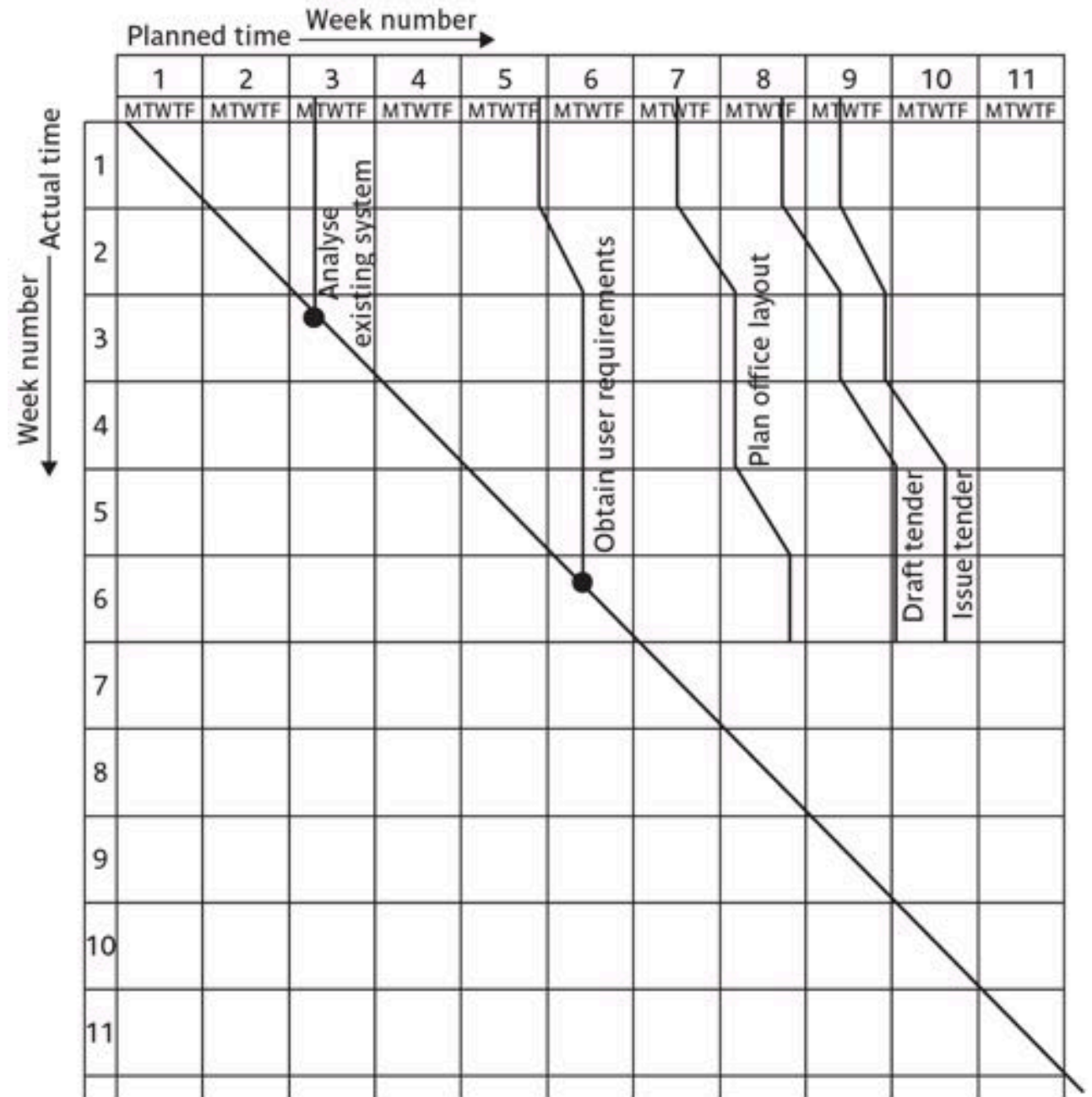
Gantt Charts



Slip Charts



The Timeline



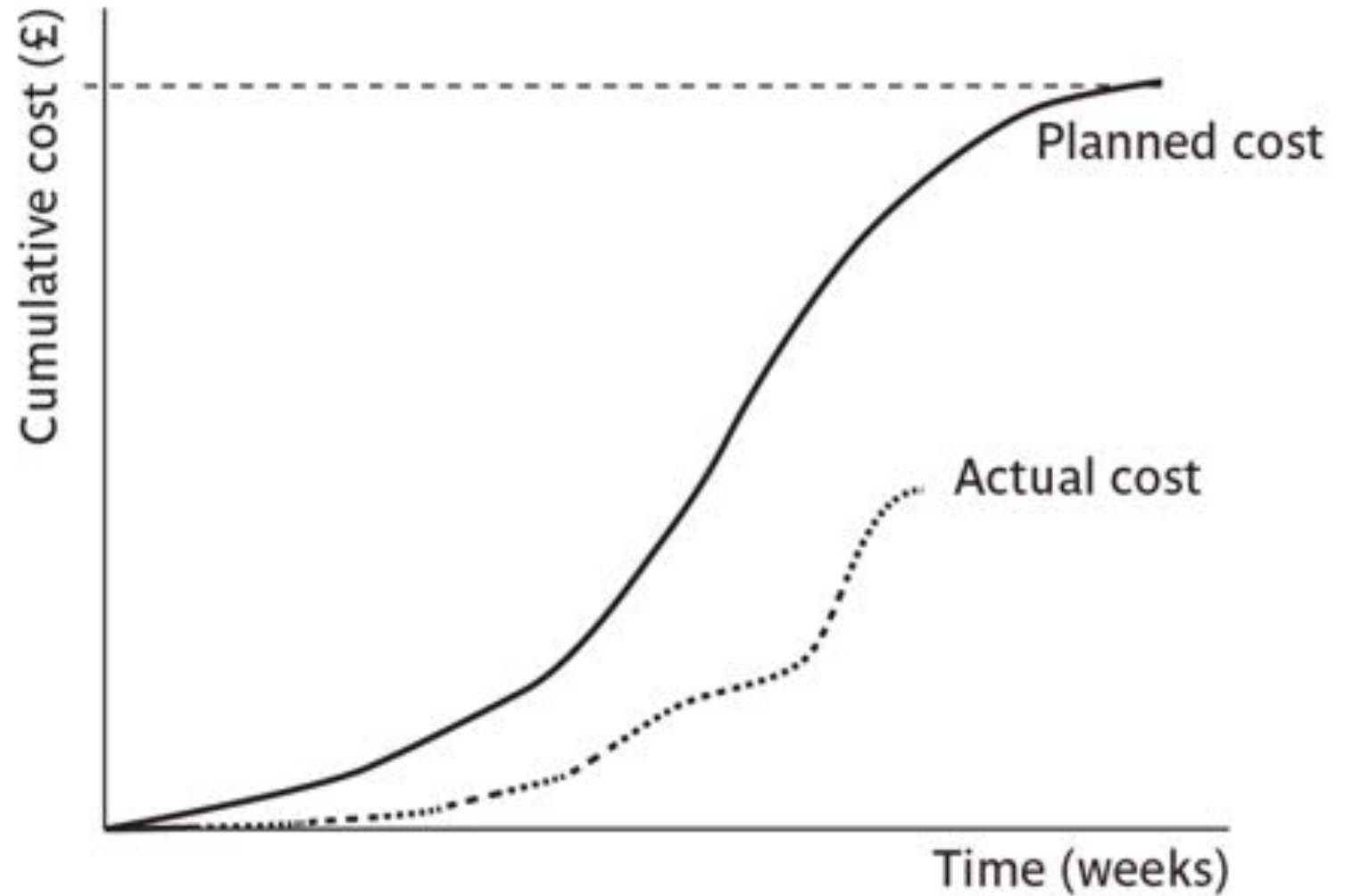
Cost Monitoring

- Important component of project control, not only in itself, but also because it provides an indication of the effort that has gone into a project
- A project could be late because the staff originally committed have not been deployed
 - Project will be **behind time** but **under budget**
- A project could be on time but only because additional resources have been added
 - Will be **over budget**
- Need to monitor both achievements and costs

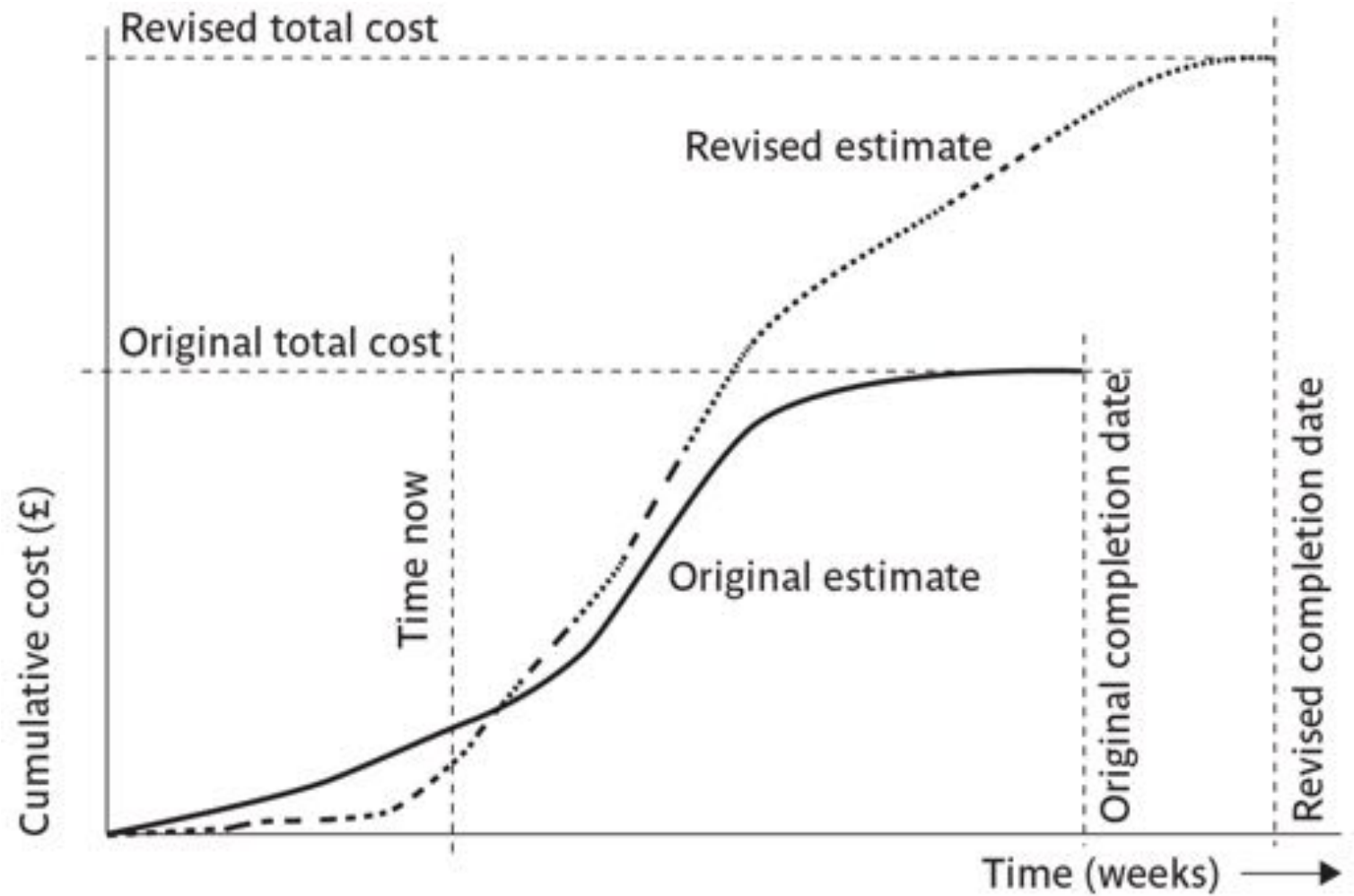


Tracking Cumulative Expenditure

- Is the project running late or it is on time but has shown substantial cost savings?



Tracking Cumulative Expenditure



Earned Value Analysis

- Method to measure the project's progress
 - 👉 **Earned value = budgeted cost of worked performed, BCWP**
- Enables the project manager to compute performance indices or burn rates for cost and schedule performance
 - How well the project is doing or performing relative to its original plans?
 - How the project will do in the future?

📖 Reichel, C. W. (2006) Earned value management systems (EVMS), PMI



EVM Foundational Concepts

- Allows to answer the following
 - 1 Where have we been?
 - 2 Where are we now?
 - 3 Where are we going?
- Three data sources
 - The budget (or **planned**) **value** of work scheduled
 - The **actual value** of work completed (*cost*)
 - The **earned value** of the work completed



Planned Value, PV

- How far along project work is supposed to be at any given point in the project schedule and cost estimate
- **Cumulative PV** is the sum of the approved budget for activities scheduled to be performed to date
- **Current PV** is the approved budget for activities scheduled to be performed during a given period
 - Days, weeks, months, etc



Actual Costs, AC

- Actual expenditures, the cost incurred for executing work on a project.
 - What you have spent
 - AC also called **Actual Cost of Work Performed (ACWP)**
- **Cumulative AC** is the sum of the actual cost for activities performed to date
- **Current AC** is the actual costs of activities performed during a given period



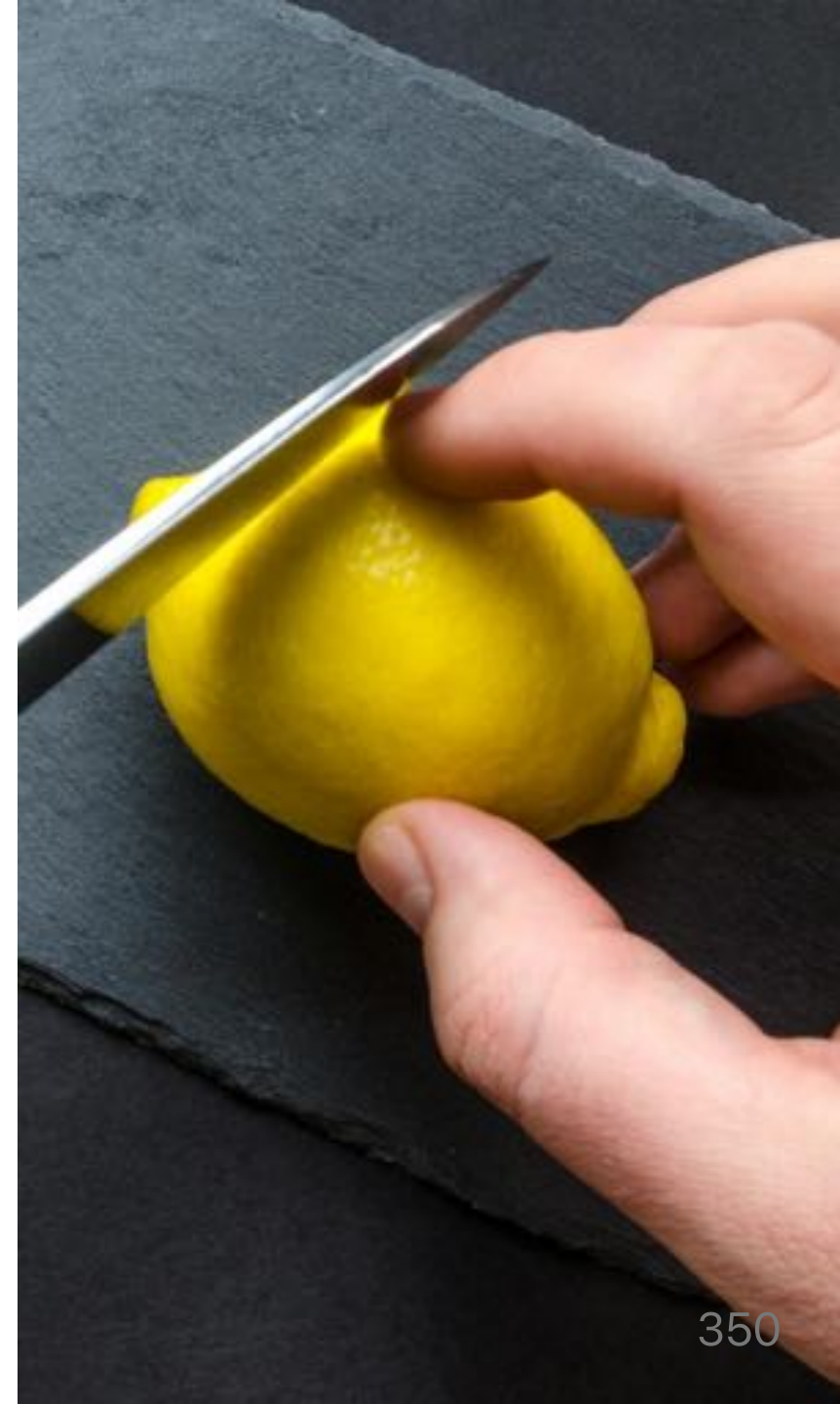
Earned Value, EV

- EV is the quantification of the “worth” of the work done to date
 - EV also called **Budgeted Cost of Work Performed (BCWP)**
- **Cumulative EV** is the sum of the budget for the activities accomplished to date
- **Current EV** is the sum of the budget for the activities accomplished in a given period



Accounting Conventions for EV

- Work completed allocated on the basis
 - **50/50** - half allocated at start, the other half on completion. These proportions can vary e.g. 0/100, 75/25 etc
 - **Milestone** - current value depends on the milestones achieved
 - **Units processed**
- Can use money values, or staff effort as a surrogate



Variance Analysis

PMI's PMBOK® Guide defines a **variance** as a quantifiable deviation, departure, or divergence away from a known baseline or expected value

- After PV, AC and EV are established, a variance analysis can be performed
 - **Schedule variance**
 - **Cost variance**



Schedule variance, SV

- Schedule Variance status does indicate the money value difference between work that is ahead or behind the plan
- $SV = EV - PV$
 - If =0, the project is on schedule 😊
 - If <1, the project is behind schedule 😞
 - If >1, the project is ahead of schedule 🎉



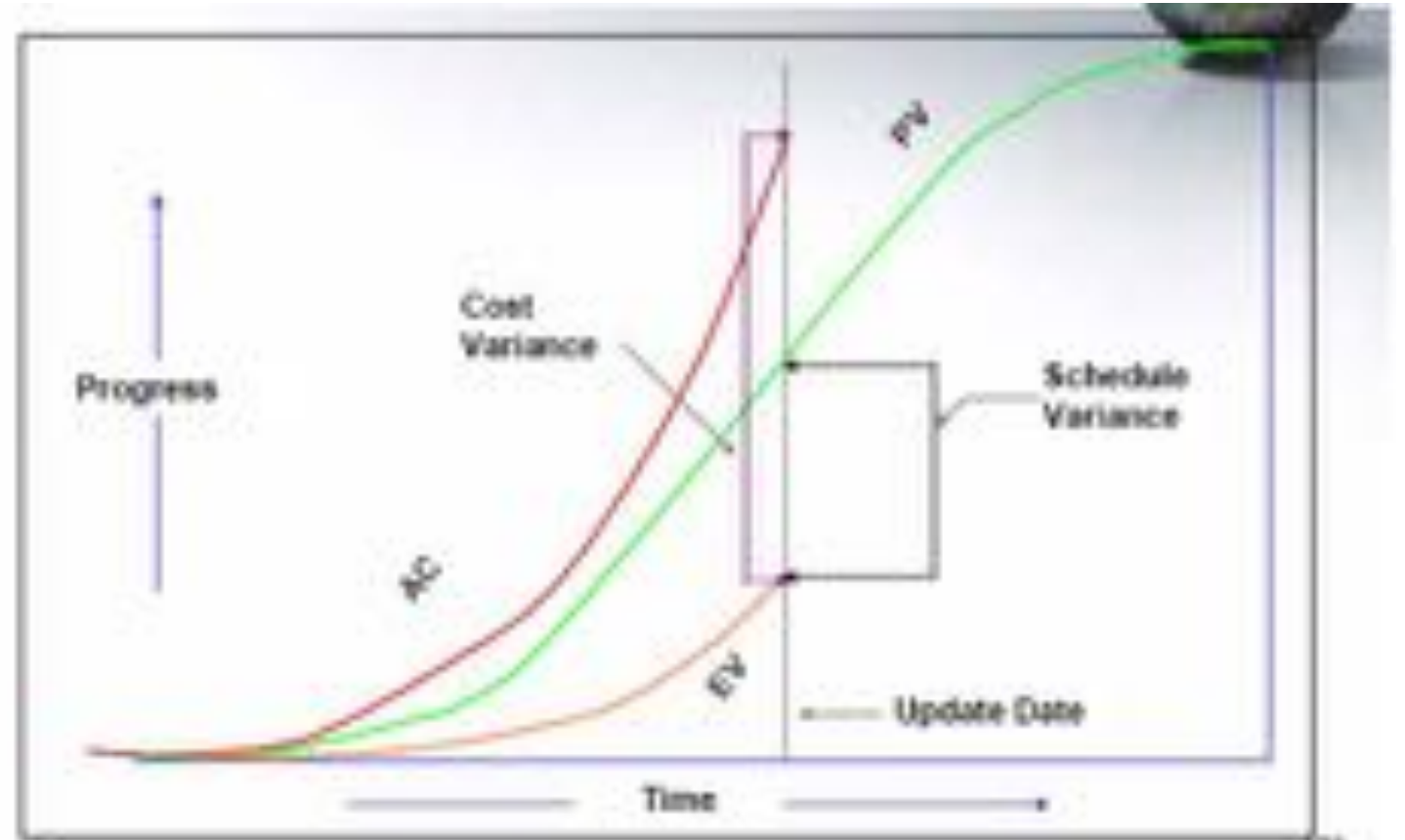
Cost Variance, CV

- $CV = EV - AC$
 - If =0, the project is on budget 😊
 - If <0, the project is over budget 😞
 - If >0, the project is under budget 🎉



Graphic Performance Report

(Reichel, 2006)



Performance Indexes

- Another analysis that can be done is performance of the project
 - Project's burn rate
- Two examination of performance are available
 - **Schedule Performance Index (SPI)**
 - **Cost Performance Index (CPI)**



Schedule Performance Index, SPI

The SPI is defined by PMI's PMBOK® Guide as a measure of schedule efficiency on a project

- $SPI = EV / PV$
 - If <1 , unfavorable condition 😞
 - If ≥ 1 , favorable condition 🎉
- An SPI of 1.1 means your project recognizing \$1.10 for every \$1.00 spent to date
 - 👉 Project will finish ahead of schedule



Cost Performance Index, CPI

The CPI is defined by PMI's PMBOK® Guide as a **measure of cost efficiency on a project**

- $CPI = EV / AC$
 - If <1 , unfavorable condition 😞
 - If ≥ 1 , favorable condition 🎉
- An CPI of \$0.90 means your project recognizing \$0.90 for every \$1.00 spent to date
 - 👉 Project will be over budget



Estimates to Complete

- When the project will be completed and how much it will cost to complete it?
- **Budget at Completion (BAC)** - the sum of all budgets allocated to a project scope
 - BAC must always equal the Project Total PV
- **Estimate at Completion (EAC)** - the actual cost to date plus an objective estimate of costs for remaining authorized work
- $EAC = BAC / CPI$



Effective Earned Value Management

- 1** Organize the project team and the scope of work, using a work breakdown structure
- 2** Schedule the tasks in a logical manner
- 3** Allocate the total budget resources to a time basis
- 4** Establish objective means for measuring work accomplishment
- 5** Control the project by analyzing cost and performance variances, assessing final costs, developing corrective actions, and controlling



Earned Value – An Example

- Tasks
 - Specify module : 5 days
 - Code module : 8 days
 - Test module : 6 days
- At the beginning of day 20, PV = 19 days
- If everything but testing completed EV = 13 days
- **Schedule Variance, $SV = EV - PV$**
 - $SV = 13 - 19 = -6$
- **Schedule Performance Indicator, $SPI = EV / PV$**
 - $SPI = 13 / 19 = 0.68$
- **SV negative or $SPI < 1.00$, project behind schedule**

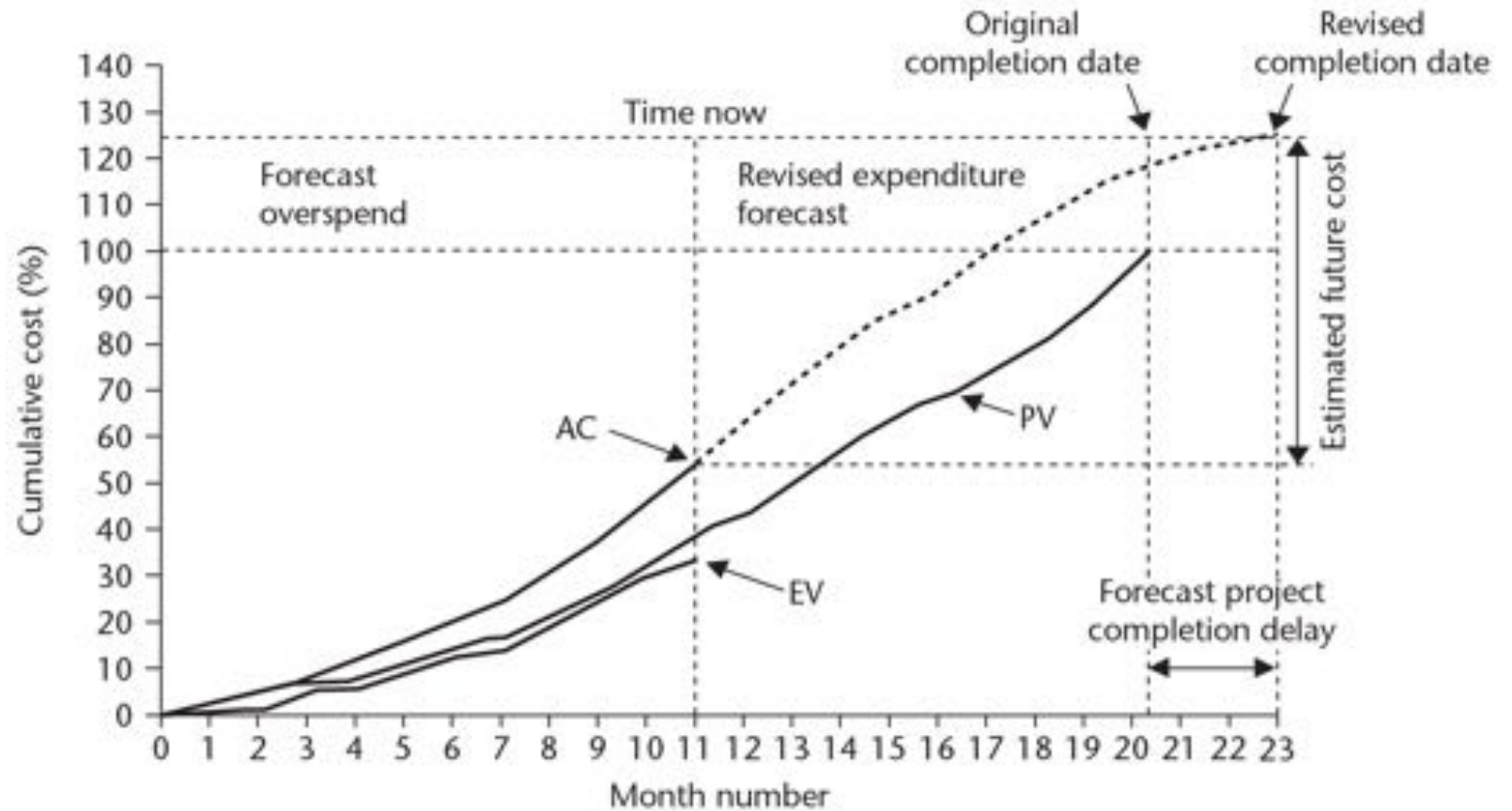
Earned Value Analysis – Actual Cost

- **Actual Cost (AC)** is also known as Actual Cost of Work Performed (ACWP)
- In previous example, if
 - Specify module actually took 3 days
 - Code module actually took 4 days
 - AC= 7 days
- **Cost Variance, $CV = EV - AC$**
 - $CV = 13 - 7 = 6$ days
- **Cost Performance Indicator, $CPI = EV / AC$**
 - $CPI = 13 / 7 = 1.86$
- CV positive or $CPI > 1.00$ means project within budget

Earned Value Analysis – Actual Costs

- CPI can be used to produce new cost estimate
- **Budget at Completion (BAC)** – current budget allocated to total costs of project
- **Estimate at Completion (EAC)** – updated estimate = BAC/CPI
 - e.g. say budget at completion is £19,000 and CPI is 1.86
 - $EAC = BAC/CPI = £10,215$ (projected costs reduced because work being completed in less time)
- Similarly a forecast of the actual duration of the project can be derived by dividing the original estimated duration by the SPI

Earned Value Chart with Revised Forecasts



Prioritizing Monitoring

We might focus more on monitoring certain types of activity e.g.

- Critical path activities
- Activities with no free float – if delayed later dependent activities are delayed
- Activities with less than a specified float
- High risk activities
- Activities using critical resources



Getting Back on Track: Options

- Renegotiate the deadline – if not possible then
 - Try to shorten critical path
 - Adding resources, work overtime
 - Re-allocate staff from less pressing work
 - Reduce scope / quality
 - Reconsider activity dependencies
 - Overlap the activities so that the start of one activity does not have to wait for completion of another
 - Split activities
- **Maintaining the business case!!**



Exception Planning

- Some changes could affect
 - Users
 - The business case (e.g. costs increase reducing the potential profits of delivered software product)
- These changes could be to
 - Delivery date
 - Scope
 - Cost
- In these cases an **exception report** is needed



Exception Planning (ii)

- First stage
 - Write an **exception report** for sponsors (perhaps through project board)
 - Explaining problems
 - Setting out options for resolution
- Second stage
 - Sponsor selects an option (or identifies another option)
 - Project manager produces an **exception plan** implementing selected option
 - Exception plan is reviewed and **accepted/rejected by sponsors/Project Board**

Change Control

- Assumed that the nature of the deliverables does not change, but
 - Changing user requirements
 - Inconsistencies in program specifications
- Products in change while are developed
- Final version is **baselined**
 - Foundation for further products
 - Changes need to be stringently controlled



Typical Change Control Process

- 1** One or more users might perceive the need for a change (RFC)
- 2** User management decide that the change is valid and worthwhile and pass it to development management
- 3** A developer is assigned to assess the practicality and cost of making the change
- 4** Development management report back to user management on the cost of the change; user management decide whether to go ahead



Change Control Process (ii)

- 5 One or more developers are authorized to make copies of components to be modified
- 6 Copies modified. After initial testing, a test version might be released to users for acceptance testing
- 7 When users are satisfied then operational release authorized – master configuration items updated



Configuration and Asset Management

Set of processes designed to ensure the quality of any product through the strict control of changes made to them and the constant availability of a stable version of each element



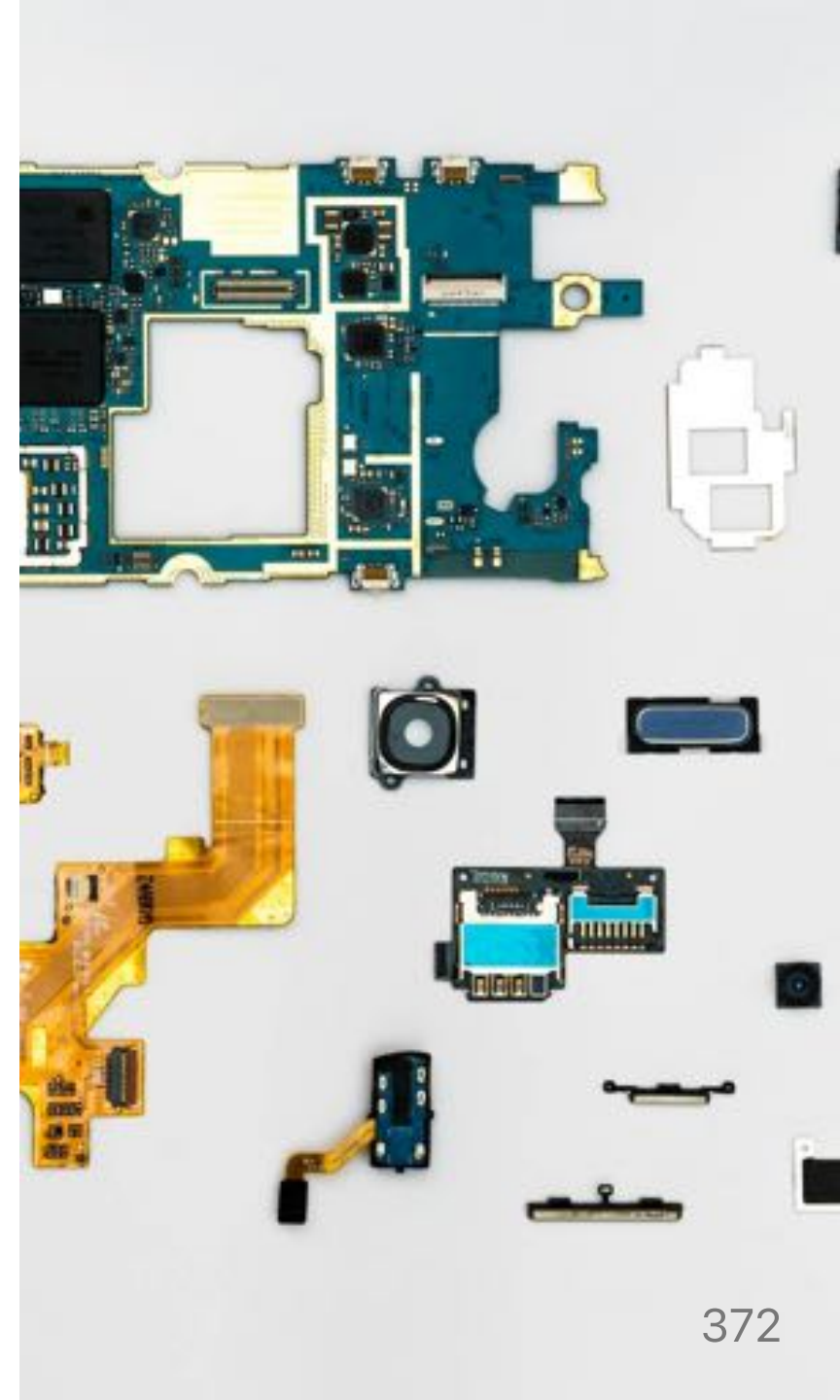
Software Configuration Elements

- Executable
- Source Code
- Data models
- Process models
- Requirements specifications
- Tests

And for each of these elements it will be stored at least:

- Name, version, status, location

👉 Métrica v3



The Role of Configuration Librarian

- Identifying items that need to be subject to change control
- Management of a central repository of the master copies of software and documentation
- Administering change procedures
- Maintenance of access records



Conclusion

- Planning is pointless unless the execution of the plan is monitored
- Activities too long subdivided to make them more controllable
- Progress measure through the delivery of products
- Costs and elapsed time need to be monitored
- Delayed projects can often be brought back on track

SUMMARY

Contract Management

Subject Ten



Introduction

- Buying goods and services is attractive when money is available but others types of resources don't
 - Staff time?
- Considerable staff time and attention still needed to manage contracted-out project
- Types of contracts, general steps to be followed, issues drafting a contract and other things to be done while contract is executed



Acquiring Software from External Supplier

This could be:

- **Bespoke system** - created specially for the customer 🧑🏻✂️
- **Off-the-shelf** - bought 'as is' 🌿
- **Customised off-the-shelf (COTS)** - a core system is customised to meet needs of a particular customer 🍵




Types of Contract

- Note difference between goods and services
- Often license to use software is bought rather than the software itself
- Regarding the way the payment is calculated
 - Fixed price contracts 🏛️
 - Time and materials contracts 🧑‍🔧
 - Fixed price per delivered unit 🧱



Fixed Price Contracts

- If there are no changes in the contract terms, there is a price to pay on completion
-  Detailed requirements analysis must have been carried out previously

Advantages to customer

- Known expenditure
- Supplier motivated to be cost-effective



Fixed Price Contracts

Disadvantages

- Supplier will increase price to meet contingencies
- Difficult to modify requirements
- Cost of changes likely to be higher
- Threat to system quality



Time and Materials 🧑‍🏭

- The customer is charged at a fixed rate per unit of effort
- The supplier may provide an initial estimate of the cost based on their current understanding of the customer's requirements, but this is not the basis for the final payment

👍 Advantages to customer

- Easy to change requirements
- Lack of price pressure can assist product quality



Time and Materials 🧑‍🏭

👎 Disadvantages

- Customer liability - the customer absorbs all the risk associated with poorly defined or changing requirements
- Lack of incentive for supplier to be cost-effective



Fixed Price per Unit Delivered 🧱

- Size of the system to be delivered is calculated or estimated at the outset of the project from requirements documents
 - Size in KLOC or FPs
- A price per unit is quoted



Fixed Price per Unit Delivered

FP Count	Design Cost/FP	Impl. Cost/FP	Total Cost/FP
to 2,000	\$242	\$725	\$967
2K - 2,5K	\$255	\$764	\$1,019
2,5K - 3K	\$265	\$793	\$1,058
3K - 3,5K	\$274	\$820	\$1,094
3,5K - 4K	\$284	\$850	\$1,134

Fixed Price/Unit Example

- Estimated system size 2,600 FPs
- Price
 - 2000 FPs x \$967 plus
 - 500 FPs x \$1,019 plus
 - 100 FPs x \$1,058
 - i.e. \$2,549,300
- What would be charge for 3,200 FPs?

Fixed Price/Unit

👍 Advantages for customer

- Customer understanding of how price is calculated
- Comparability between different pricing schedules
- Emerging functionality can be accounted for
- Supplier incentive to be cost-effective
- Life-cycle range



Fixed Price/Unit

👎 Disadvantages

- Difficulties with software size measurement - may need independent FP counter
- Changing (as opposed to new) requirements: how do you charge?



Additional Charges for Changed Functionality

	Pre-accepted testing handover	Post-acceptance testing handover
Additional PFs	100%	100%
Changed PFs	130%	150%
Deleted PFs	25%	50%

Some Figures

Company	Project	Personal	Outsourced
Multinational/Retail	ERP	1.5	3.1
Medium/ITC	ERP	0.6	0.06
Multinational/Medical	ERP	0.6	0.0
SpinOff/ITC	CMS	0.1	0.02
Medium/Building	Value-added tool	0.1	0.0

In M€

The Tendering Process 📣

According to the approach used in contractor selection:

- **Open tendering**
 - Any supplier can bid in response to the invitation to tender
 - All tenders must be evaluated in the same way
 - Government bodies may have to do this by local/international law (including EU and WTO requirements)

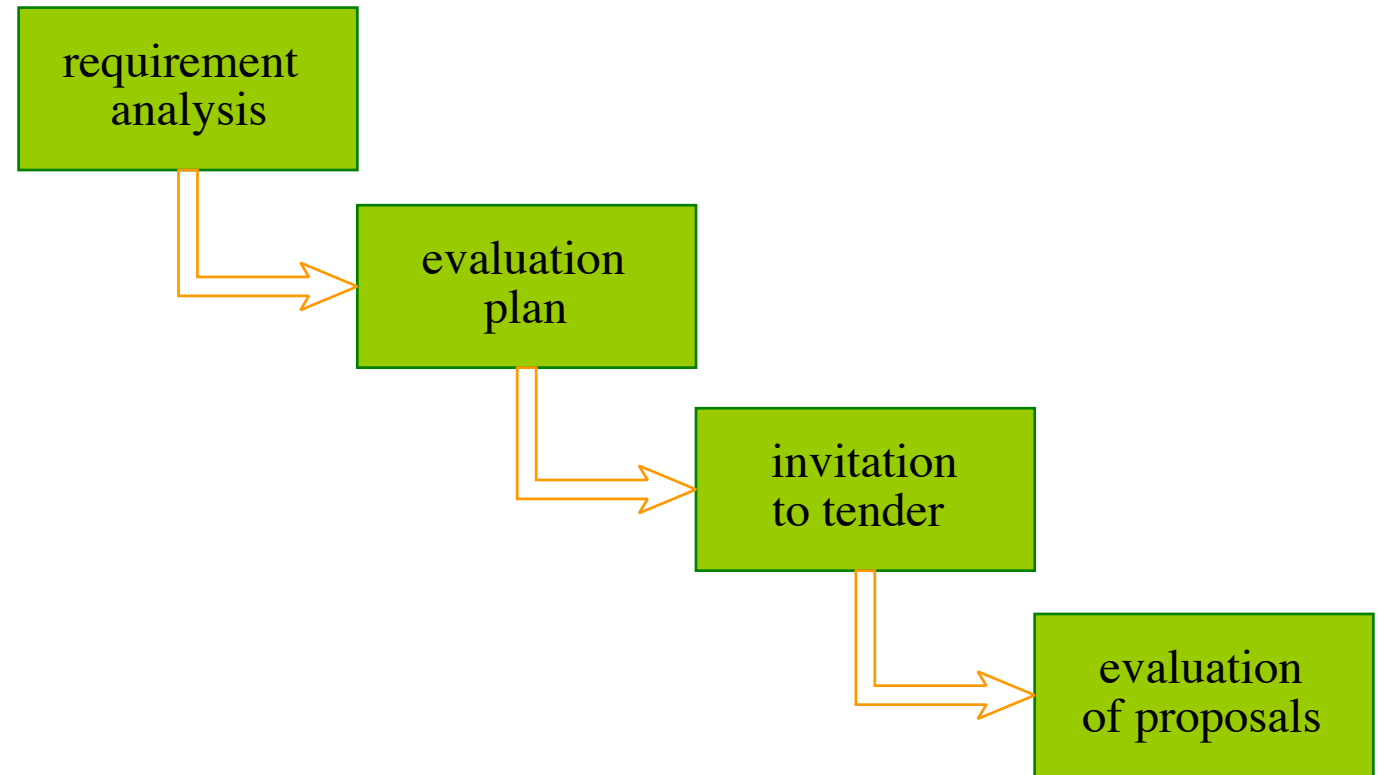


The Tendering Process

- **Restricted tendering process**
 - Bids only from those specifically invited
 - Can reduce suppliers being considered at any stage
- **Negotiated tendering procedure**
 - Negotiate with one supplier e.g. for extensions to software already supplied



Stages in Contract Placement



Requirements

- These will include
 - Functions in software, with necessary inputs and outputs
 - Standards to be adhered to
 - Other applications with which software is to be compatible
 - Quality requirements e.g. response times



Requirements Document: Sections

- Introduction
- Description of existing system and current environment
- Future strategy or plans
- System requirements
 - Mandatory features
 - Desirable features
- Deadlines
- Additional information required from bidders



Page 2 of 3 pages

TERMS AND CONDITIONS

For the purchaser's general purchasing conditions to be considered as accepted once the order, will only be considered as accepted once our acknowledgement is received. This offer is considered as final only after our acknowledgement.

Offer

This offer is exclusively to the supply of services specified here. The purchaser in no case will be allowed to put forward any other offer. This offer is accepted by our company.

Evaluation Plan

- How are proposals to be evaluated?
- Methods could include:
 - Reading proposals
 - Interviews
 - Demonstrations
 - Site visits
 - Practical tests



Evaluation Plan (ii)

- Need to assess **value for money** (VFM) for each desirable feature
- VFM approach an improvement on previous emphasis on accepting lowest bid
- Example:
 - Feeder file saves data input
 - 4 hours work a month saved at £20 an hour
 - System to be used for 4 years
 - If cost of feature £1000, would it be worth it?



Invitation To Tender (ITT)

- Note that bidder is making an **offer** in response to ITT
- **Acceptance** of offer creates a **contract**
- Customer may need further information
- Problem of different technical solutions to the same problem



Contracts 🏆👤

- A project manager cannot be expected to be a legal expert
 - Needs advice
- BUT must ensure contract reflect true requirements and expectations of supplier and client



Contract Checklist

- Definitions – what words mean precisely e.g. **supplier, user, application**
- Form of agreement.
 - For example, is this a contract for a sale or a lease, or a license to use a software application? Can the license be transferred?
- Goods and services to be supplied
 - This could include lengthy specifications
- Timetable of activities
- Payment arrangements
 - Payments may be tied to completion of specific

tasks



Contract Checklist (ii)

- Ownership of software
 - Can client sell software to others?
 - Can supplier sell software to others?
 - Customer may want **exclusive use**
 - Does supplier retain the copyright?
- Where supplier retains source code for maintenance, may be a problem if supplier goes out of business
 - A copy of code could be deposited with an escrow service



Contract Checklist (iii)

- Environment
 - Where equipment is to be installed, who is responsible for various aspects of site preparation e.g. electricity supply?
 - In case of software, compatibility with existing software and OS
- Customer commitments
 - For example providing access, supplying information
- Standards to be met



Contract Checklist (iv)

- Acceptance procedures
 - User acceptance tests
- Project and quality management
 - Nature and frequency of progress meetings
- Timetable
 - Schedule of when the key parts of the project should be completed



Contract Checklist (v)

- Price and payment method
 - When the payments are to be made
 - Balance supplier incurred cost with customer's requirements satisfaction
- Miscellaneous legal requirements
 - Clauses
 - Liquidated damages
 - Legal jurisdiction
 - Arbitration



Contract Management

- Some terms of contract will relate to management of contract, for example,
 - Progress reporting
 - Decision points (milestones)
 - Could be linked to release of payments to the contractor
 - Quality reviews
 - Variations to the contract
 - How are changes to requirements dealt with?
 - Acceptance criteria



Acceptance 🤝

- Acceptance testing after the delivery
 - How long the testing can take?
- Asking to approve supplier's internal pre-acceptance testing
- Part or all payment conditioned to final testing
- Period of warranty

YES

Conclusion

- Successful contracting requires management time
- Easier to negotiate before a contract is signed
- Evaluate proposals comparing costs over the whole lifetime of the system
- A contract place obligations on the supplier as well as the customer
- Agreement on the management of the supplier-customer relationship

SUMMARY

Annex

- Software Copyright
- Authors' Rights
- Software Patent



Software Copyright ©

- **Software copyright** is used by software developers and proprietary software companies to prevent the unauthorized copying of their software
 - Free and open source licenses also rely on copyright law to enforce their terms
 - No such duty would apply when the software in question is in the public domain

Authors' Rights

- In general, software is protected as a literary work
- **Moral** and **economic** rights granted by law to authors by the simple fact of the creation of a literary, artistic, musical, scientific or didactic work, whether published or unpublished

Authors' Rights (ii)

- A work enters the public domain when the economic rights have expired
 - This usually happens after a period of time following the death of the author (*post mortem auctoris*), in European law, it is 70 years from the author's death
- Once that time has elapsed, the work can then be used freely, respecting the moral rights

EU Computer Programs Directive

Controls the legal protection of computer programs under the copyright law of the European Union 🇪🇺

- The most recent version is Directive 2009/24/EC
- Computer programs and any associated design material be protected under copyright as literary works
- The duration of the copyright was originally fixed at the life of the author plus fifty years in accordance with the Berne Convention standard for literary works

Ownership of Rights 🧑

- The author of a computer program shall be the natural person or group of natural persons who have created the program, or the legal entity that is recognized as the owner of the copyright in the cases expressly provided for in the Law on Intellectual Property

Ownership of Rights 🙋 (ii)

- In the case of collective works the person, whether natural person or legal entity, who publishes it and discloses it under his name shall be considered the author thereof, unless otherwise agreed
- The copyright in a computer program that is the unitary result of collaboration on the part of two or more authors shall be common property and shall accrue to all of them in such proportions as they may determine

Ownership of Rights 🙋 (iii)

- Where a salaried worker creates a computer program in the execution of his duties or on instructions given him by his employer, ownership of the economic rights in the computer program so created—including both the source program and the object program—shall accrue exclusively to the employer, unless otherwise agreed

🤔 What about the last year project at University?

Restricted Acts ✘

- The owner of the copyright has the exclusive right to authorize:
 - The temporary or permanent **copying** of the program, including any copying which may be necessary to load, view or run the program
 - The translation, adaptation or other **alteration** to the program
 - The **distribution** of the program to the public by any means, including rental

Exceptions to Restricted Acts !

- The legal owner of a program is assumed to have a license to **create any copies** necessary to use the program and to **alter** the program within its intended purpose (e.g. for error correction)
- The legal owner may also make a **back-up copy** for his or her personal use
- The program may also be **decompiled** if this is necessary to ensure it operates with another program or device, but the results of the decompilation may not be used for any other purpose without infringing the copyright in the program

Difficulties in the Protection of Software via Copyrights 🚧

- To officially prove the date of creation, the work has to be registered in the Intellectual Property Registry Office
 - This implies that the code has to be made public
 - This poses a problem for certain companies or individuals who wish to maintain a certain secrecy with their code
- Alternative methods to protect code:
 - Escrow contract, which consists of delivering a copy of the code to a notary

Software Patent

A software patent is a patent on a piece of software, such as a **computer program, libraries, user interface, or algorithm**

- A patent is a set of exclusionary rights granted by a state to a patent holder for a limited period of time, usually 20 years
- These rights are granted to patent applicants in exchange for their disclosure of the inventions
- Permission, where granted, is typically in the form of a license which conditions are set by the patent owner: it may be free or in return for a royalty payment or lump sum fee



European Patent Convention

The patentability of software, computer programs and computer-implemented inventions under the European Patent Convention (EPC)

- Under the EPC, *programs for computers* are **not regarded as inventions** for the purpose of granting European patents
- That does not mean that all inventions including some software are de jure not patentable

European Patent Convention

EPC excludes from patentability, in particular

1. Discoveries, scientific theories and mathematical methods
2. Aesthetic creations
3. Schemes, rules and methods for performing mental acts, playing games or doing business, and **programs for computers**
4. Presentations of information

Excludes patentability only to the extent to which a European patent application or European patent relates to such subject-matter or activities **as such**

Managing People in Software Environments

Subject Eleven



Main Topics

- What is organizational behaviour?
- Staff selection and induction
- Models of motivation – focus on the individual
- The dark side of motivation - stress
- The broader issues of health and safety
- Some ethical and professional concerns

Before Organizational Behaviour

- Frederick Taylor (1856-1915) '*the father of scientific management*'
- Focus:
 - To select the best people for the job 👁️
 - To instruct them in the best methods 🧑🏫
 - To give financial incentives in the form of higher wages to the best workers 💰
- One problem: **group norms** 👤👤



Hawthorne Effect

- 1920's – series of experiments at the Hawthorne Plant of Western Electric, Chicago
- Found that simply showing an interest in a group increased productivity

Management Approaches

- **Theory X:** there is a need for coercion, direction, and control of people at work 🖐️
- **Theory Y:** work is as natural as rest or play 🙌



Selecting the Best People

- Belbin distinguishes between **eligible** (having the right qualifications) and **suitable** candidates (can do the job)
- The danger is employ someone who is eligible but not suitable
- The best situation is to employ someone who is suitable but not eligible!
 - For example, these are likely to be cheaper and to stay in the job
- We should try to assess actual skills 🧑🎓 rather than past experience 🏠 and provide training to make good minor gaps in expertise



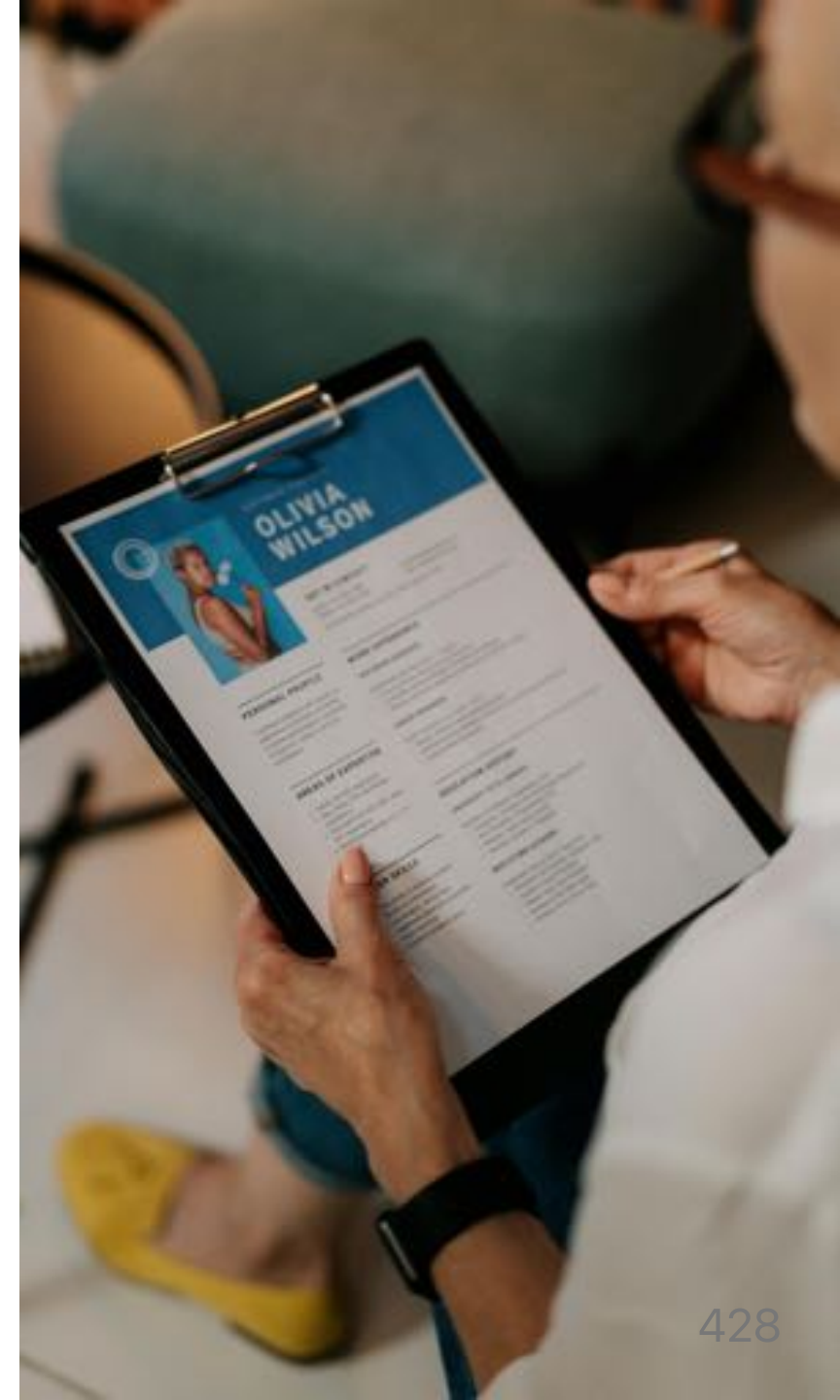
Do Good Software Developers Have Innate Characteristics?

- 1968 study – difference of 1:25 in time taken by different programmers to code program
- Other research found experience better than maths skills as a guide to software skills
- Some research suggested software developers less sociable than other workers 🤡
- Later surveys have found no significant social differences between IT workers and others 🧑 – this could be result of broader role of IT in organizations



A Selection Process

1. Create a job specification 🧑💻
 - Content includes types of task to be carried out
2. Create a job holder profile ✅
 - Describes the characteristics of the person who could do the job
3. Obtain applicants 🗣️
 - Identify the media that potential job holders are likely to consult. Elicit CVs







A Selection Process (ii)

4. Select potential candidates from CVs 📄
 - Do not waste everybody's time interviewing people whose CV clearly indicates are unsuitable
5. Further selection, including interview 💬
 - Selection processes could include aptitude tests, examination of work portfolios. Make sure selection processes map to the job holder profile
6. Other procedures 🕵️
 - E.g. taking up references, medicals etc



Instruction in the Best Methods

- The induction of new staff should be carefully planned 
 - Worst case where new recruit is simply ignored and not given any tasks !
- Good induction leads to new recruit becoming productive more quickly 
- Need to review staff progress frequently and provide feedback 
- Need to identify training that could enhance staff effectiveness 



Motivation 🙌

- Motivation and application can often make up for shortfalls in innate skills 📈
- Taylor's approach - financial incentives
- Abraham Maslow (1908-1970)
 - Motivations vary from individual to individual
 - Hierarchy of needs – as lower ones fulfilled, higher ones emerge 🏔️
 - Lowest level – food, shelter
 - Highest level – self-actualization



Herzberg

Herzberg suggested two sets of factors affected job satisfaction

1. **Hygiene or maintenance factors** - make you dissatisfied if they are not right 🤔
 - e.g. pay, working conditions
2. **Motivators** - make you feel the job is worthwhile 👍
 - e.g. a sense of achievement



Vroom

Vroom and colleagues identified three influences on motivation

1. Expectancy

– The belief that working harder leads to better performance

2. Instrumentality

– The belief that better performance will be rewarded

3. Perceived value of the reward



Oldham-Hackman Job Characteristics

Identified the following characteristics of a job which make it more *meaningful*

- Skill variety
- Task identity
- Task significance

Two other factors contributed to satisfaction

- Autonomy
- Feedback



Methods to Improve Job Satisfaction

- Set specific goals 📅
- Provide feedback on the progress towards meeting those goals 👉
- Consider job redesign ⚙️
 - Job enlargement 🗉
 - Job enrichment 📋



Stress

- Edward Yourdon quotes a project manager: *'Once a project gets rolling, you should be expecting members to be putting in at least 60 hours a week.... The project manager must expect to put in as many hours as possible'*
- 1960 study in US: people under 45 who worked more than 48 hours a week twice the risk of death from coronary heart disease
- XP practice – maximum 40 hour working week



Stress Can Be Reduced by Good Project Management

Good project management should lead to:

- Reasonable estimates of effort
- Good project control leading fewer unexpected crises
- Making clear what is expected of each team member – reduces **role ambiguity**
- Reduced **role conflict** where a person is torn between conflicting responsibilities
- Bullying tactics are a symptom of incompetent

project management



Health and Safety ❤️🛡️

- Apart from stress, health and safety less likely to be an issue compared to other engineering projects
- ... but sometimes IT infrastructure may be set up as other building work is going on
- UK law lays down that organizations employing over 5 staff should have a **written safety policy**
- Management of safety should be embedded in project management



Ethical and Professional Concerns 🌍

- Ethics relates to the moral obligation to respect the rights and interests of others – goes beyond strictly legal responsibilities
- Three groups of responsibilities
 - Responsibilities that everyone has
 - Responsibilities that people in organizations have
 - Responsibilities relating to your profession or calling
- 🗨️ 👤 👤 👤 *Facebook Employees Stage Virtual Walkout to Protest Trump Posts* 👤 👤 👤



Organizational Ethics

- There are some who argue that organizational ethics are limited
- **Stockholder theory** (e.g. Milton Friedman)
 - An employee's duty is to the owners of the business (which often means the stakeholders) above all others – although legal requirements must be met
- **Competitive relationships between businesses**
 - Competition may cause you to do things that could have a negative impact on the owners or employees of competitive businesses



Professional Ethics

- Professionals have knowledge about the technical domain that the general public does not
- Ethical duty of the expert to warn lay people of the risks involved in a particular course of action
- Many professions, or would be professions, have codes of conduct for their members e.g.
 - <http://www.bcs.org/upload/pdf/cop.pdf>
 - <http://www.ieee.org/web/aboutus/ethics>
 - http://www.acm.org/about/se_code



Conclusion

- People may be motivated by money, but they are motivated by other things as well
- Both staff selection and the identification of training need should be done in an orderly, structured, way where requirements are clearly defined first
- Thoughtful job design can increase staff motivation
- Undue pressure on staff can have short-term gains, but is harmful to both productivity and personal health in the longer term
- Project objectives should include, where appropriate, those relating health and safety

SUMMARY

Annex: Software Teams Organization

Based on Pablo Santos Conference

Professional Career

- Two paths
 - Individual Contributors, IC
 - Technical focused
 - Managers
 - Managerial focused
- Same career development options
- Jumps are possible

The Dual Ladder

IC	Manager
	Vicepresident (VP)
Distinguished Engineer	
Principal Engineer	Director
Staff Engineer	Senior Manager
	Manager / Technical Lead
Senior Engineer	
Junior Engineer / Engineer	

Engineering Coordination

- Senior Manager (Engineering)
 - Team Lead 1
 - ICs
 - Team Lead 2
 - ICs
- No more than 7 or 8 direct reports

Other Roles

- Technical Program Manager
 - Project Manager
- Product Managers
- Designers
- QA
- Support

Key Roles

- Engineers (software developers)
- Product Managers (PMs)
- Designers
- Technical Project Managers (TPM)
- Product Marketing Managers (PMM)

Product Manager (PM)

- The person who identifies the customer need and the larger business objectives that a product or feature will fulfill
 - Articulates what success looks like for a product
 - Rallies a team to turn that vision into a reality

Technical Project Manager (TPM)

- In small companies it is usually the **Technical Leader**
- In large companies there is a figure to coordinate different teams rather than the traditional vision of project management that we usually have
 - More a facilitator/coach than a boss who orders

Product Designers

- They are in charge of designing the product, but beyond the visual
 - Seeking to define how it should behave, in a usable (UX) and understandable way
 - Defining the overall experience: from the moment you register or download it, to the way you use it

Product Marketing Manager (PMM)

Only the PMM focuses on:

- The positioning of a product
- And the specific messaging created for delivering a USP (Unique Selling Proposition)
- By defining the product in the market and separating itself from the competition
- As it drives both prospect and customer engagement

Product Marketing Manager (ii)

Also, the PMM:

- Works with PM and developers
- Plans and executes product releases and launches
- Converts tech product info into marketing and sales messages




Other Marketing Roles

- Content marketers
 - Create blogs, videos, social media to attract interest, generate leads, increase brand awareness and engage with online audience
- Digital/online marketers, growth hackers and demand generation teams
 - Grow the business and reach audience using SEO, email marketing, ad campaigns, hosting webinars
- Business intelligence
 - Collects, stores and analyzes data to help strategic and operational changes

Other Marketing Roles (ii)


- Event coordinators/planners
 - Organize trade shows, conferences, events in general
- Copywriters
 - Write contents in various forms: blogs, paid ads, ebooks
- Graphics designers
 - Put together all the materials and create designs
- Field marketers
 - Run marketing ops in their local markets and languages, including building customer

Top Management

-  Chief Executive Officer - CEO
-  Chief Technical Officer - CTO
-  Chief Information Officer - CIO

Chief Executive Officer (CEO)

- Highest authority of management and administrative direction
- Typically reports to the board of directors and is charged with maximizing the value of the business
 - In the non-profit and government sector, aims at achieving outcomes related to the organization's mission

 *Director ejecutivo, director general, director gerente, jefe ejecutivo, principal oficial ejecutivo, primer ejecutivo, ejecutivo delegado o consejero delegado*

Chief Technical Officer (CTO)

- Executive-level position in a company whose occupation is focused on the scientific and technological issues within an organization
 - An evolution of the Research and Development manager (R&D)
- Also known as chief technology officer or chief technologist

 *Director de Tecnología*

Chief Information Officer (CIO)



- The most senior executive in an enterprise who works with information technology and computer systems, in order to support enterprise goals
 - Aligns information systems with company plans, prepares and manages budgets and coordinates technical teams
- Also known as chief digital information officer (CDIO) or information technology (IT) director

 *Director de Sistemas de Información*

Working in Teams

Subject Twelve




Introduction

- Human effort shared between individual software developers within *teams* and between groups of developers
- **Teams** - groups of people who working together
 - *co-located*
- **Project team** - *All* the people working on a project
- Individual and group tasks
- **Communication genres** refers to methods of communication



Becoming a Team

- Five basic stages of development:
 - Forming
 - Storming
 - Norming
 - Performing
 - Adjourning
- This can be accelerated through team-building exercises

 Classification associated with Tuckman and Jensen



Balanced Teams

- Meredith Belbin studied the performance of top executives carrying out group work at the Hendon Management Centre
- Tried putting the **best** people together in teams
 - Almost invariably did badly 😬
- Identified the need for a balance of skills and management roles in a successful team



Management Team Roles 🎭

- The **coordinator**
 - Good at chairing meetings
- The **'plant'**
 - An idea generator
- The **monitor-evaluator**
 - Good at evaluating ideas
- The **shaper**
 - Helps direct team's efforts
- The **team worker**
 - Skilled at creating a good working environment

👉 <https://www.belbin.com/about/belbin-team-roles>



Management Team Roles 🎭 (ii)

- The **resource investigator**
 - Adept at finding resources, including information
- The **completer-finisher**
 - Concerned with getting tasks completed
- The **implementer**
 - A good team player who is willing to undertake less attractive tasks if they are needed for team success
- The **specialist**
 - The 'techie' who likes to acquire knowledge for its own sake



Management Team Roles 🎭 (iii)

- When putting together a team selects the essential technical specialist first
 - The group roles of these individuals can then be assessed
- Any remaining team members can then be allocated with an eye on making the group roles more balanced



Group Performance

Some tasks are better carried out collectively while other tasks are better delegated to individuals

- **Additive tasks**
 - The effort of each participant is summed
- **Compensatory tasks**
 - The judgements of individual group members are summed
 - Errors of some compensated for by judgements of others



Group Performance (ii)

- **Disjunctive tasks**
 - There is only one correct answer
 - Someone must
 - Come up with right answer
 - Persuade the other that they are right
- **Conjunctive**
 - The task is only finished when all components have been completed



'Social Loafing'

- Tendency for some team participants to 'coast' and let others do the work
- Also tendency not to assist other team members who have problems
- Suggested counter-measures:
 - Make individual contributions identifiable
 - Consciously involve group members ('loafer' could in fact just be shy!)
 - Reward 'team players'



Barriers to Good Team Decisions

- Time-consuming
- Inter-personal conflicts – see earlier section on team formation
- Conflicts tend to be dampened by emergence of *group norms* – shared group opinions and attitudes
- *Risky shift* – people in groups are more likely to make risky decisions than they would as individuals

Delphi Approach

To avoid dominant personalities intruding the following approach is adopted

1. Enlist cooperation of experts
2. Moderator presents experts with problem
3. Experts send in their recommendations to the moderator
4. Recommendations are collated and circulated to all experts
5. Experts comment on ideas of others and modify their own recommendation if so moved
6. If moderator detects a consensus, stop; else back to

4



Team 'Heedfulness'

- Where group members are aware of the activities of other members that contribute to overall group success
- Impression of a 'collective mind'
- Some attempts to promote this:
 - Egoless programming
 - Chief programmer teams
 - XP
 - Scrum



Egoless Programming 🚗

- Gerry Weinberg noted a tendency for programmers to be protective of their code and to resist perceived criticisms by others of the code
- Encouraged programmers to read each others code
- Argued that software should become communal, not personal – hence 'egoless programming'



Chief Programmer Teams 🍳

- Fred Brooks was concerned about the need to maintain 'design consistency' in large software systems
- Appointment of key programmers, **Chief Programmers**, with responsibilities for defining requirements, designing, writing and test software code
- Assisted by a support team: **co-pilot** – shared coding, editor who made typed in new or changed code, program clerk who wrote and maintained documentation and tester

- **Problem** – finding staff capable of the chief



Extreme Programming 🚠

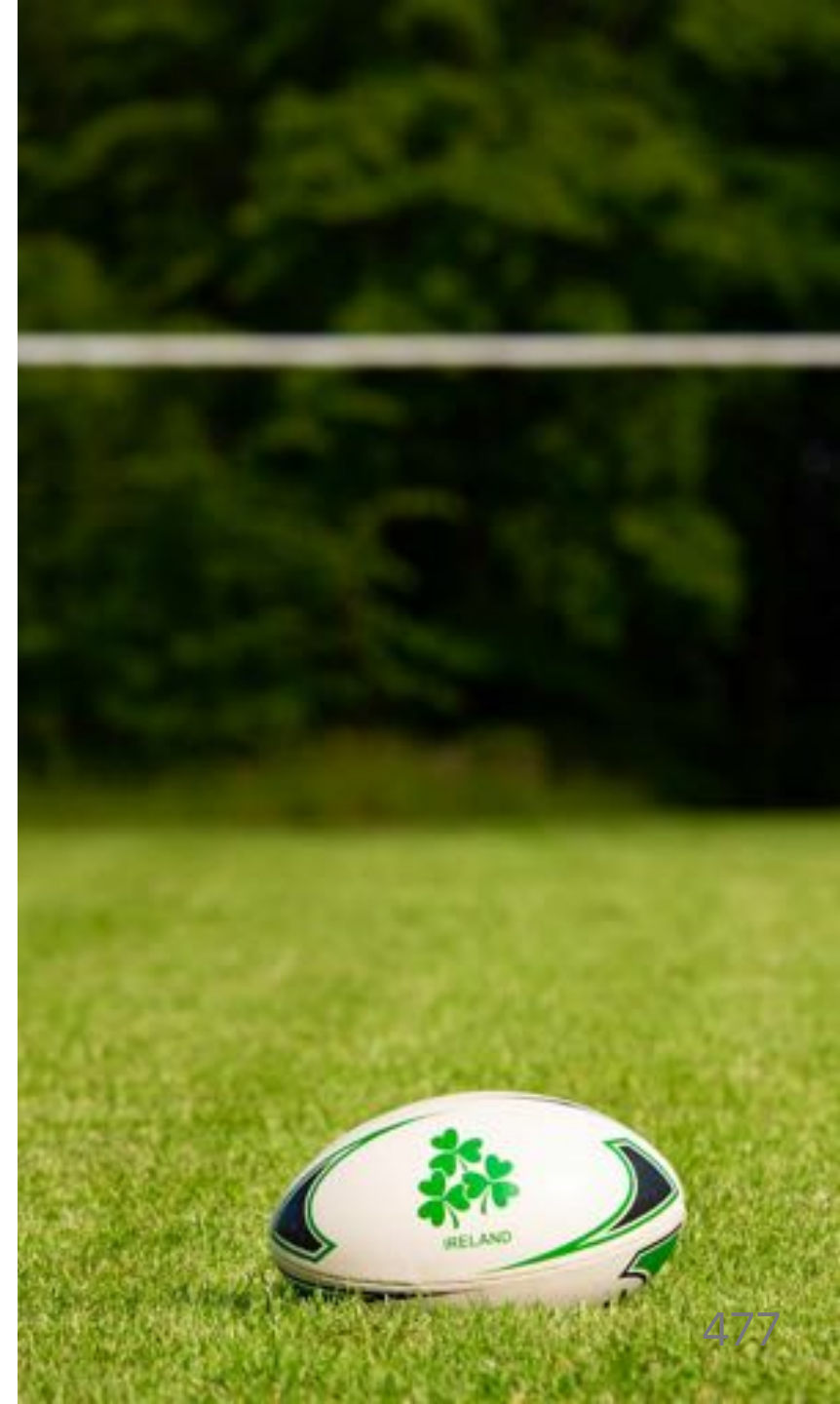
XP can be seen as an attempt to improve team heedfulness and reduce the length of communication paths (the time between something being recorded and it being used)

- Software code enhanced to be self-documenting
- Software regularly refactored to clarify its structure
- Test cases/expected results created *before* coding
 - acts as a supplementary specification
- Pair programming – a development of the co-pilot concept



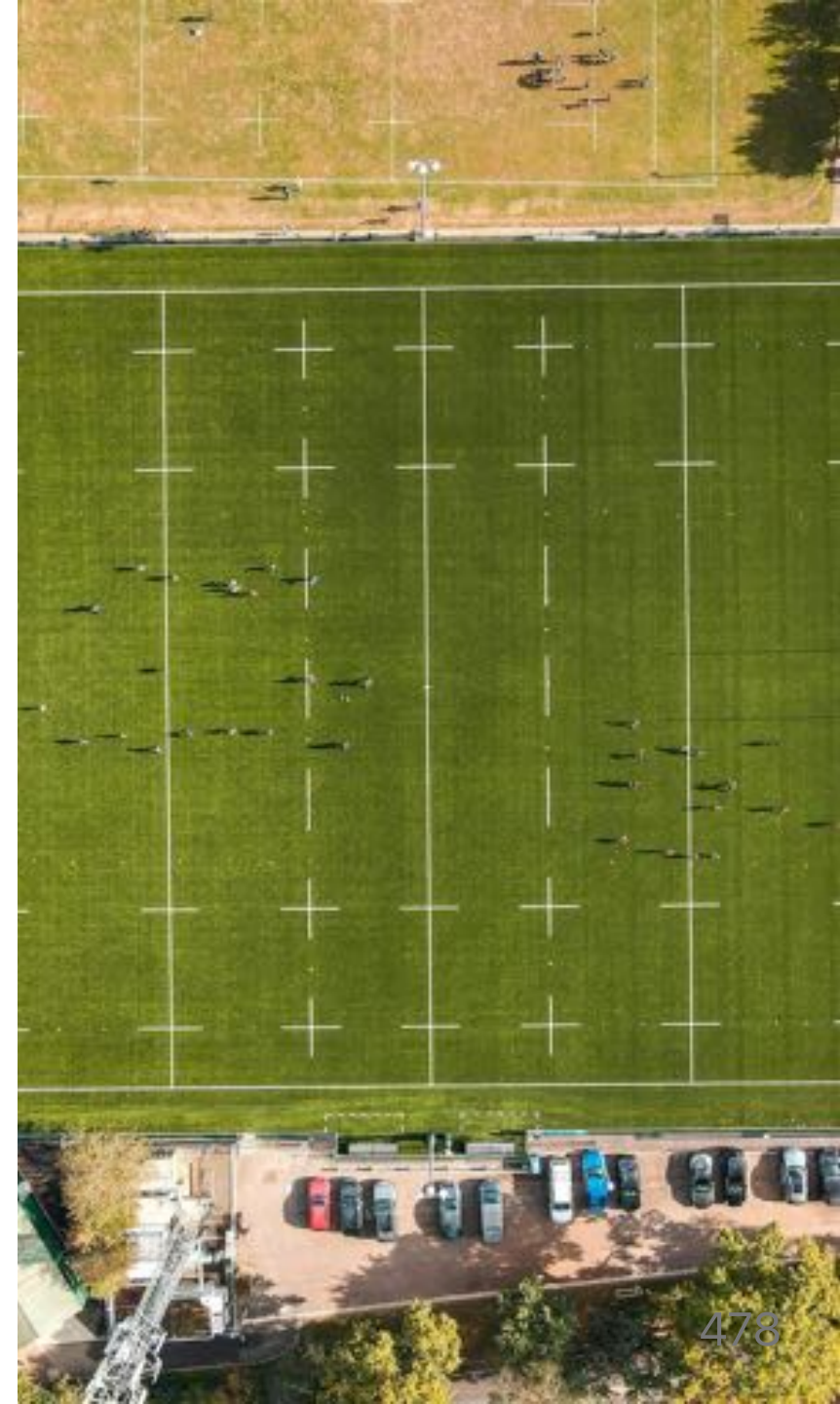
Scrum

- Named as an analogy to a rugby scrum – all pushing together
- Originally designed for new product development where 'time-to-market' is important
- There is no precise specification of the requirements of a particular client
- Having a product that is attractive to a number of customers is important
- Proposals for features are likely to evolve as ideas are tried out during development



Scrum (ii)

- The Scrum process starts with a system's architecture and planning phase
 - Chief architect defines the overall architecture of the product - chief programmer approach
- Required release date for the product and a set of the desired features of the product, each with a priority, would be defined at this stage
- 'Sprints' increments of typically one to four weeks



Scrum (iii)

- Unlike XP, requirements are frozen during a sprint
- At the beginning of the sprint there is a sprint planning meeting where requirements are prioritized
- At end of sprint, a review meeting where work is reviewed and requirements may be changed or added to
- Daily 'scrums' – daily stand-up meetings of about 15 minutes



Why 'Virtual Projects'? 🌴

- The physical needs of software developers (according to an IBM report)
 - 100 square feet of floor space ($9,3 \text{ m}^2$)
 - 30 square feet of work surface ($2,8 \text{ m}^2$)
 - Dividers at least 6 feet high to muffle noise ($1,8 \text{ m}$)
- Demarco and Lister found clear statistical links between noise and coding error rates
- One answer: Send the developers home!



Possible Advantages 👍

- Can use staff from developing countries – lower costs
- Can use short term contracts:
 - Reduction in overheads related to use of premises
 - Reduction in staff costs, training, holidays, pensions etc.
- Can use specialist staff for specific jobs



Further Advantages 👍

- Productivity of home workers can be higher – fewer distractions
- Can take advantage of time zone differences e.g. overnight system testing



Some Challenges

- Work requirements have to be carefully specified
- Procedures need to be formally documented
- Coordination can be difficult
- Payment methods need to be modified
 - piece-rates or fixed price, rather than day-rates



More Challenges

- Possible lack of trust when there is no face-to-face contact
- Assessment of quality of delivered products needs to be rigorous
- Different time zones can cause communication and co-ordination problems



Time/Place Constraints on Communication

	Same Place	Different Place
Same Time	Meetings, interviews	Telephone, Instant messaging
Different Time	Notice boards, Pigeon-holes	Email, Voicemail, Documents



Other Factors Influencing Communication Genres 🙌

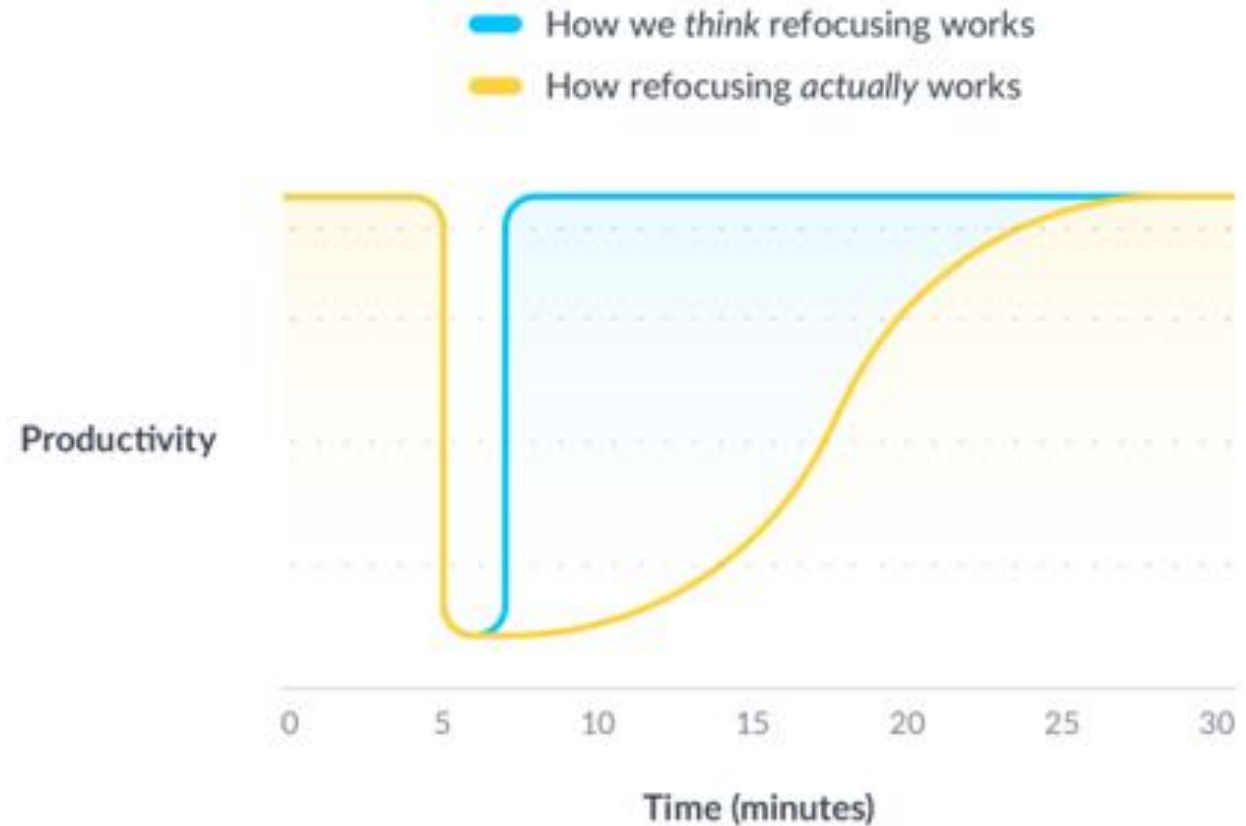
- Size and complexity of information
 - Favours documents
- Familiarity of context e.g. terminology
 - Where low, two-way communication favoured
- Personally sensitive
 - It has to be face-to-face communication here



Asynchronous Communication

👉 How to Make Asynchronous Communication Work for Your Team

What happens to our focus after an interruption?



Best Method of Communication Depends on Stage of Project

- Early stages 🥚
 - Need to build trust
 - Establishing context
 - Making important 'global' decisions
 - *Favours same time/same place*
- Intermediate stages 🐣
 - Often involves the parallel detailed design of components
 - Need for clarification of interfaces, etc
 - *Favours same time/different place*



Best Method of Communication Depends on Stage of Project (ii)

- Implementation stages 🤖
 - Design is relatively clear
 - Domain and context familiar
 - Small amounts of operational data need to be exchanged
 - *Favours different time/different place communications e.g. e-mail*
- Face to face coordination meetings ☕ – the 'heartbeat' of the project



Communications Plans

- As we have seen choosing the right communication methods is crucial in a project
- Therefore, a good idea to create a **communication plan**
- **Stages** of creating a communication plan
 - Identify all the major stakeholders for the project – see chapter 1
 - Create a plan for the project – see chapter 3
 - Identify stakeholder and communication needs for each stage of the project
 - Documented in a communication plan



Content of a Communication Plan



For each communication event and channel, identify:

- *What*. This contains the name of a particular communication event, e.g. 'kick-off meeting', or channel, e.g. 'project intranet site'
- *Who/target*. The target audience for the communication
- *Purpose*. What the communication is to achieve



Content of a Communication Plan 📋 (ii)

For each communication event and channel, identify (cont.):

- *When/frequency*. If the communication is by means of a single event, then a date can be supplied. If the event is a recurring one, such as a progress meeting then the frequency should be indicated
- *Type/method*. The nature of the communication, e.g., a meeting or a distributed document
- *Responsibility*. The person who initiates the communication



Leadership: Types of Power

Position power

- Coercive power – able to threaten punishment
- Connection power – have access to those who do have power
- Legitimate power – based on a person's title conferring a special status
- Reward power – able to reward those who comply



Leadership: Types of Power

Personal power

- Expert power: holder can carry out specialist tasks that are in demand
- Information power: holder has access to needed information
- Referent power: based on personal attractiveness or charisma



Leadership Styles

Decision making vs Implementation

	Autocrat	Democrat
Directive	makes decisions alone, close supervision of implementation	makes decisions participatively, close supervision of implementation
Permissive	makes decisions alone, gives latitude in implementation	makes decisions participatively, gives latitude in implementation

Leadership Styles 🚢

- **Task orientation** – focus on the work in hand
- **People orientation** – focus on relationships
- Where there is uncertainty about the way job is to be done or staff are inexperienced they welcome task oriented supervision
- Uncertainty is reduced – people orientation more important
- Risk that with reduction of uncertainty, managers have time on their hands and become more task oriented (interfering)



Conclusion

- Consideration should be given, when forming a new project team, to getting the right mix of people and to planning activities which promote team building
- Group working is more effective with some types of activity than others
- Different styles of leadership are needed in different situations
- Care should be taken to identify the most effective way of communication with project participants at key points in the project

SUMMARY