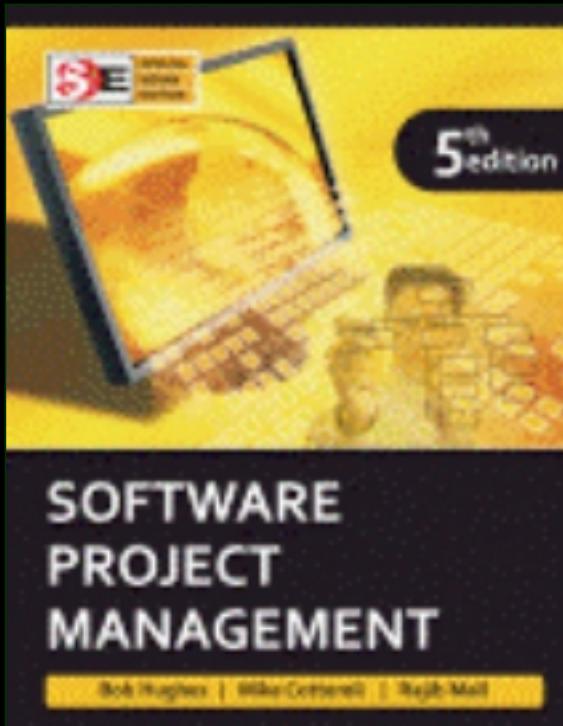


Software Project Management



Chapter One

An Introduction

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 e.g. UK government in 2003-4 spent £2.3 billions on contracts for ICT and only £1.4 billions on road building
- Project often fail – Standish Group claim only a third of ICT projects are successful. 82% were late and 43% exceeded their budget.
- Poor project management a major factor in these failures

What is a project?

Some dictionary definitions:

“A *specific plan or design*”

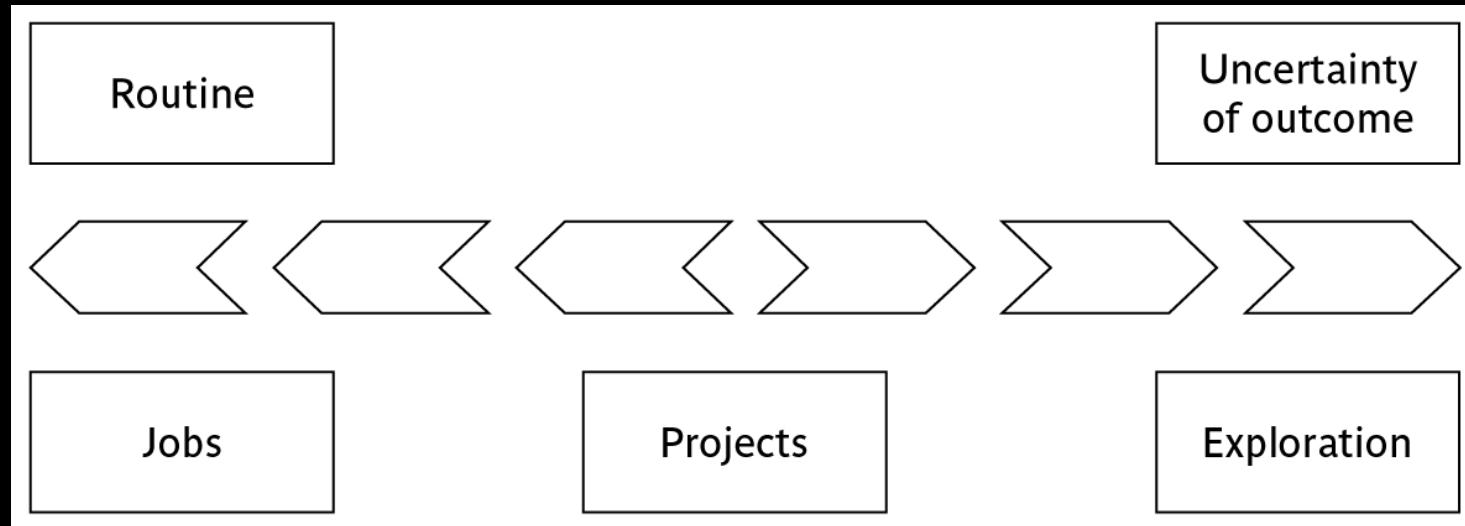
“A *planned undertaking*”

“A *large undertaking e.g. a public works scheme*”

Longmans dictionary

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

Jobs versus projects



‘Jobs’ – repetition of very well-defined and well understood tasks with very little uncertainty

‘Exploration’ – e.g. finding a cure for cancer: the outcome is very uncertain

Projects – in the middle!

Characteristics of projects

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

Are software projects really different from other projects?

Not really ...but

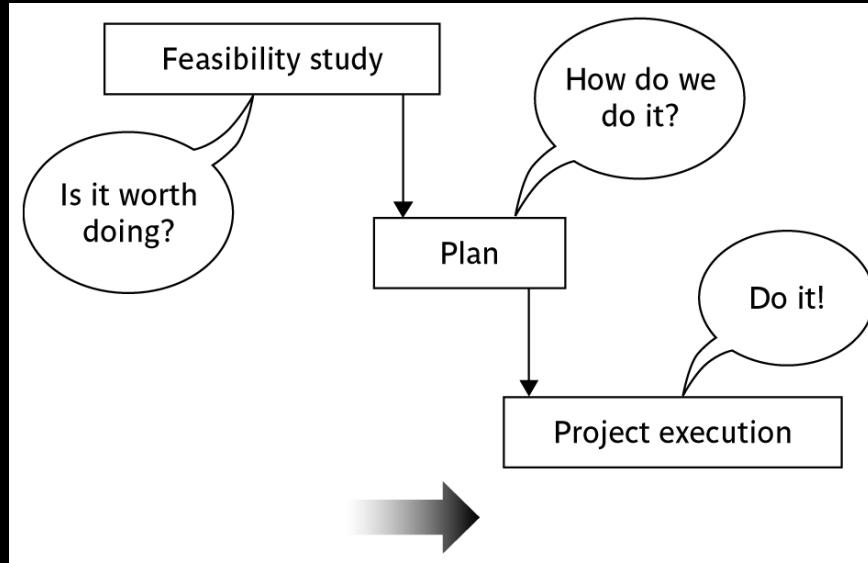
- Invisibility
 - Complexity
 - Conformity
 - Flexibility
- make software more problematic to build than other engineered artefacts.

Contract management versus technical project management

Projects can be:

- **In-house:** clients and developers are employed by the same organization
- **Out-sourced:** clients and developers employed by different organizations
- ‘Project manager’ could be:
 - ◆ a ‘contract manager’ in the client organization
 - ◆ a technical project manager in the supplier/services organization

Activities covered by project management



Feasibility study

Is project technically feasible and worthwhile from a business point of view?

Planning

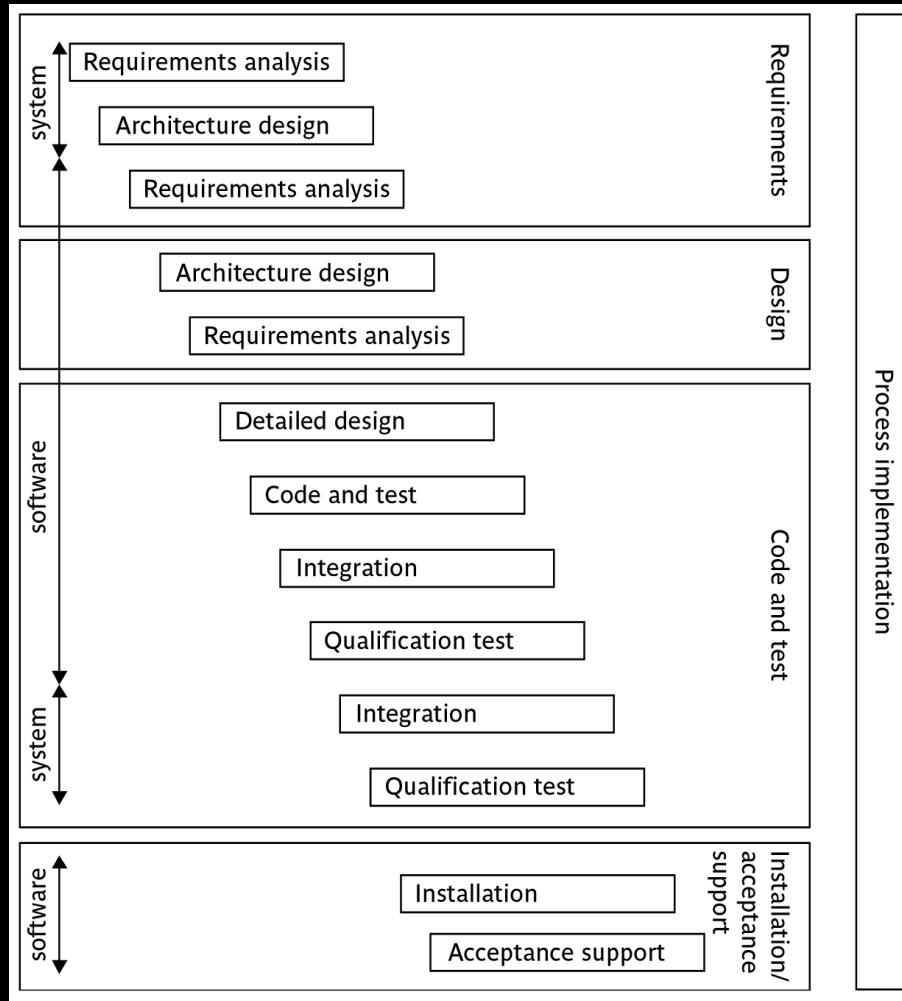
Only done if project is feasible

Execution

Implement plan, but plan may be changed as we go along

The software development life-cycle (ISO 12207)

converting 'requirements
into equivalents



Requirement elicitation
Resource requirement

Software requirement

Fulfilling the requirement

Setting up standing data
Implementing agreed
extensions and improvements

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 i.e. costs

ISO 12207 life-cycle

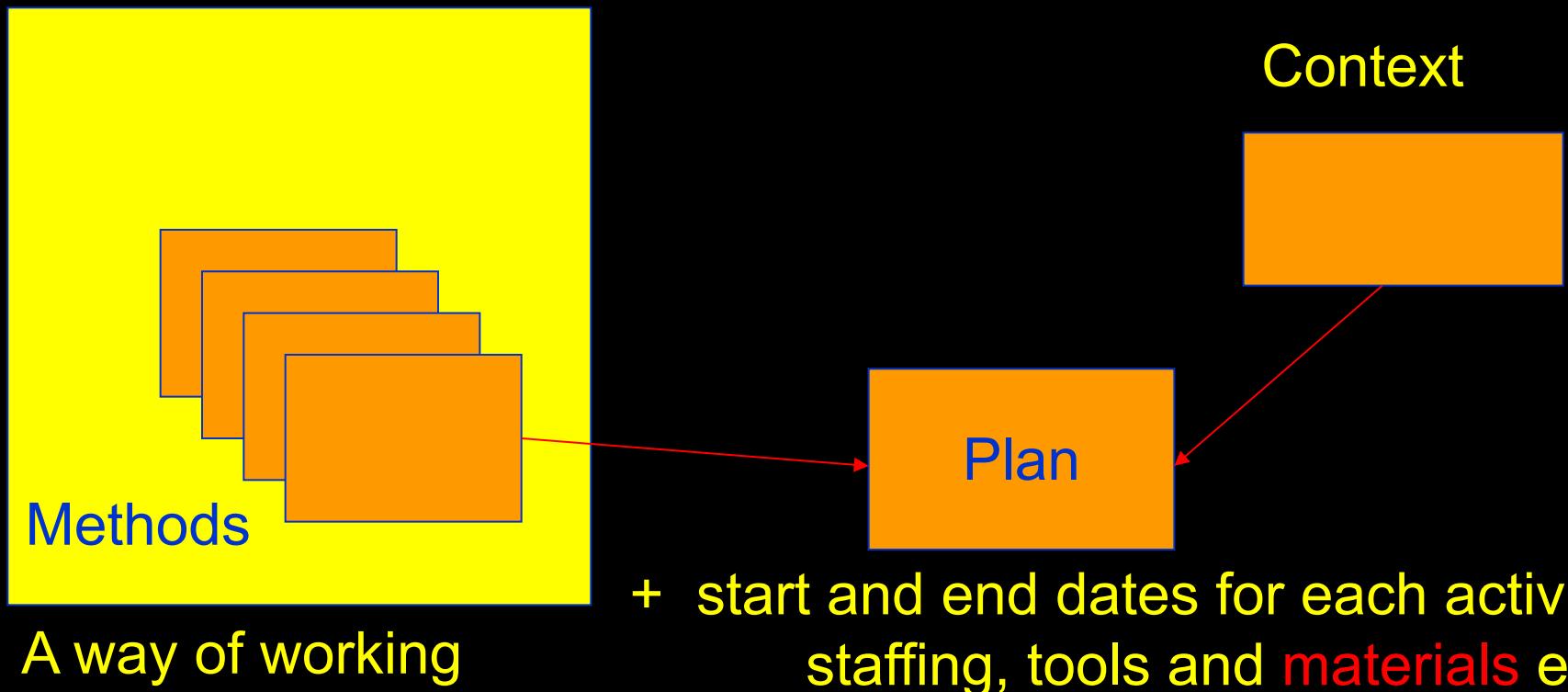
- Architecture design
 - ◆ Based on *system requirements*
 - ◆ Defines components of system: hardware, software, organizational
 - ◆ *Software requirements* will come out of this
- Code and test
 - ◆ Of individual components
- Integration
 - ◆ Putting the components together

ISO12207...continued

- 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 etc
- Acceptance support
 - ◆ Including maintenance and enhancement

Plans, methods and methodologies

Methodology = a set of methods



- ❑ Method relates to a type of activity
- ❑ Plan takes that method and converts it to a real activities
 - ✓ Its start and end dates
 - ✓ Who will carry it out
 - ✓ What tools and materials (also information) will be needed

The output from one method might be the input to another.
Group of methods or techniques are often grouped into *methodologies* such as object-oriented design.

Some ways of 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
- Product-development versus outsourced

Stakeholders

These are people who have a stake or interest in the project

In general, they could be *users/clients* or *developers/implementers*

They could be:

- Within the project team
- Outside the project team, but within the same organization
- Outside both the project team and the organization

Different stakeholders may have different objectives – need to define common project objectives

Setting objectives

Objectives focus on the desired outcomes of the project rather than tasks within it. They are the *post condition* of the project.

- Answering the question ‘*What do we have to do to have a success?*’
- Need for a *project authority*
 - ↳ 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

S – specific, that is, concrete and well-defined

M – measurable, that is, satisfaction of the objective can be objectively judged

A – achievable, that is, it is within the power of the individual or group concerned to meet the target

R – relevant, the objective must relevant to the true purpose of the project

T – time constrained: there is defined point in time by which the objective should be achieved

Goals/sub-objectives

These are steps along the way to achieving the objective

Informally, these can be defined by completing the sentence

To reach objective X, the following must be in place

A.....

B.....

C..... etc

Goals/sub-objectives continued

- Often a goal can be allocated to an individual.
- *Individuals might have the capability of achieving goal on their own, but not the overall objective e.g.*

Overall objective – user satisfaction with software product

Analyst goal – accurate requirements

Developer goal – reliable software

Measures of effectiveness

How do we know that the goal or objective has been achieved?

By a practical test, that can be objectively assessed.

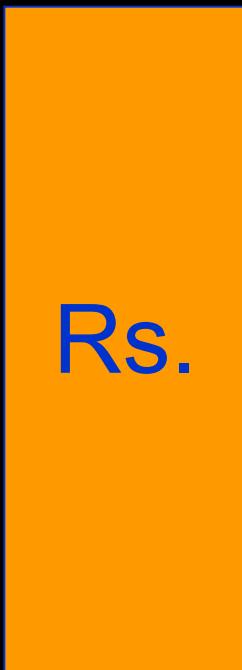
e.g. for user satisfaction with software product:

- Repeat business – they buy further products from us
- Number of complaints – if low etc. etc.

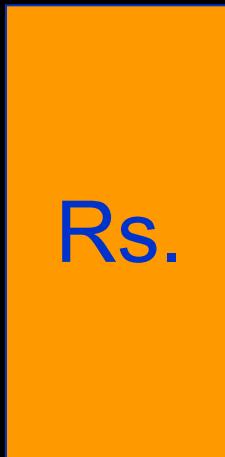
Measure of effectiveness is a practical method of checking that an objective has been met.

The business case

Benefits



Costs



- Benefits of delivered project must outweigh costs

Costs include:

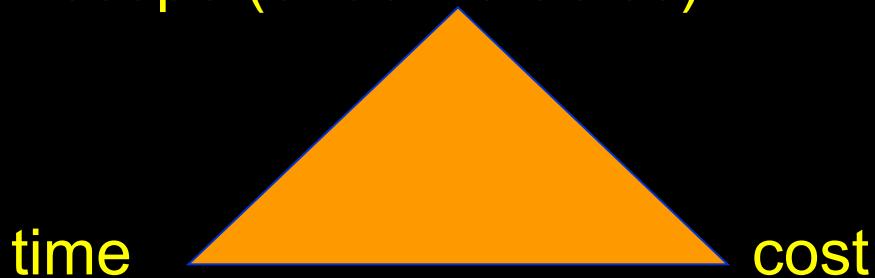
Development
Operation

Benefits

Quantifiable
Non-quantifiable

Project success/failure

- Degree to which objectives are met
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. Similarly costs and scope can be protected by adjusting other corners of the ‘project triangle’.

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?

This involves the following activities:

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

continued...

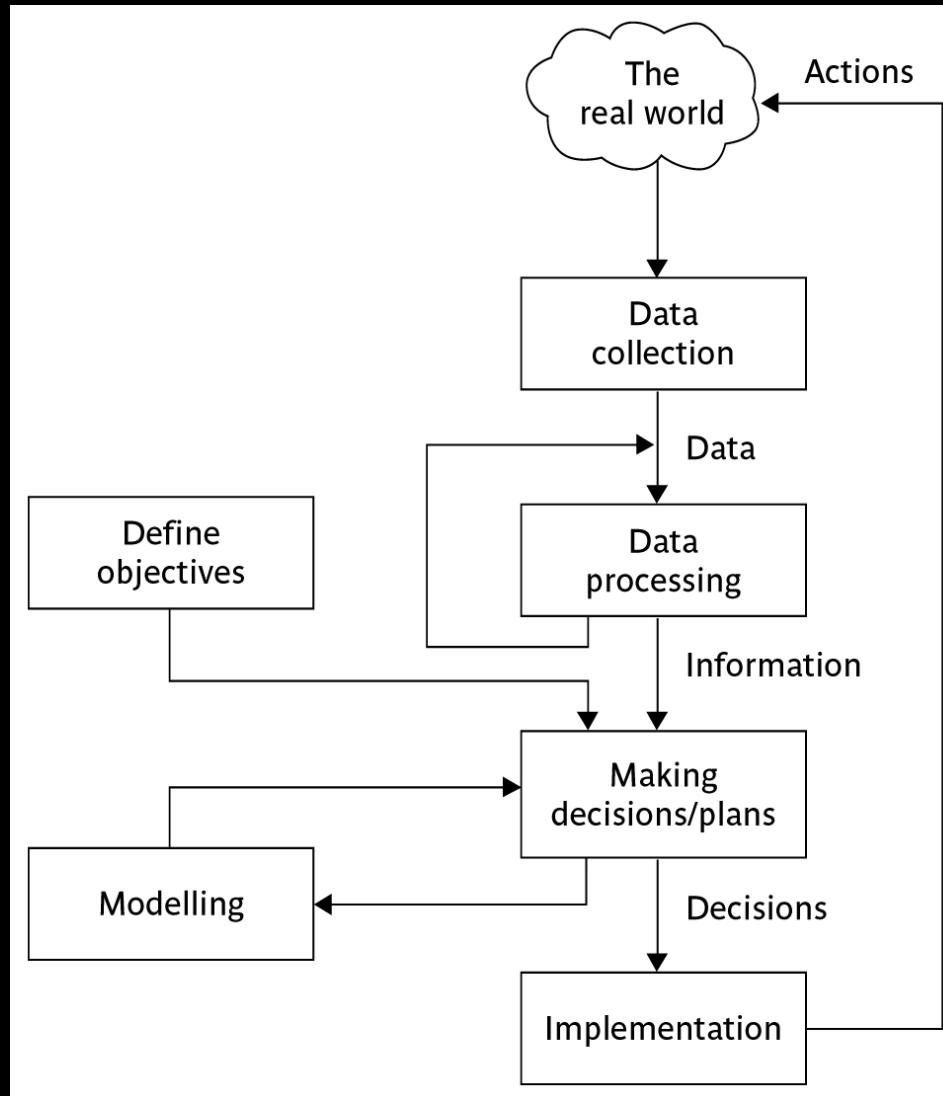
What is management? (continued)

- 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

Project Planning

- **Carried out before development starts.**
- **Important activities:**
 - ◆ **Estimation (cost, duration, effort)**
 - ◆ **Scheduling**
 - ◆ **Staffing**
 - ◆ **Risk management**
 - ◆ **Miscellaneous plans**

Management control



Management control

Data – the raw details

e.g. ‘*6,000 documents processed at location X*’

Information – the data is processed to produce something that is meaningful and useful

e.g. ‘*productivity is 100 documents a day*’

Comparison with objectives/goals

e.g. *we will not meet target of processing all documents by 31st March*

continued....

Management control - continued

Modelling – working out the probable outcomes of various decisions

e.g. if we employ two more staff at location X how quickly can we get the documents processed?

Implementation – carrying out the remedial actions that have been decided upon

Traditional versus Modern Project Management

- Projects are increasingly being based on either tailoring some existing product or reusing certain pre-built libraries.
- Facilitating and accommodating client feedbacks
- Facilitating customer participation in project development work
- Incremental delivery of the product with evolving functionalities.

- **Planning incremental delivery**

Old-Step wise execution (initiation, plan, monitor, control) New-RAD, Adaptive short term planning, incremental deliveries, extreme project management

- **Quality management**

Increase of awareness on product quality, assessment of project progress and tracking quality

- **Change management**

Old- change in requirement not entertained, Now- customer suggestions are actively considered, customer feed back taken, product development is carried out with greater functionalities.

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!

Software Project Management



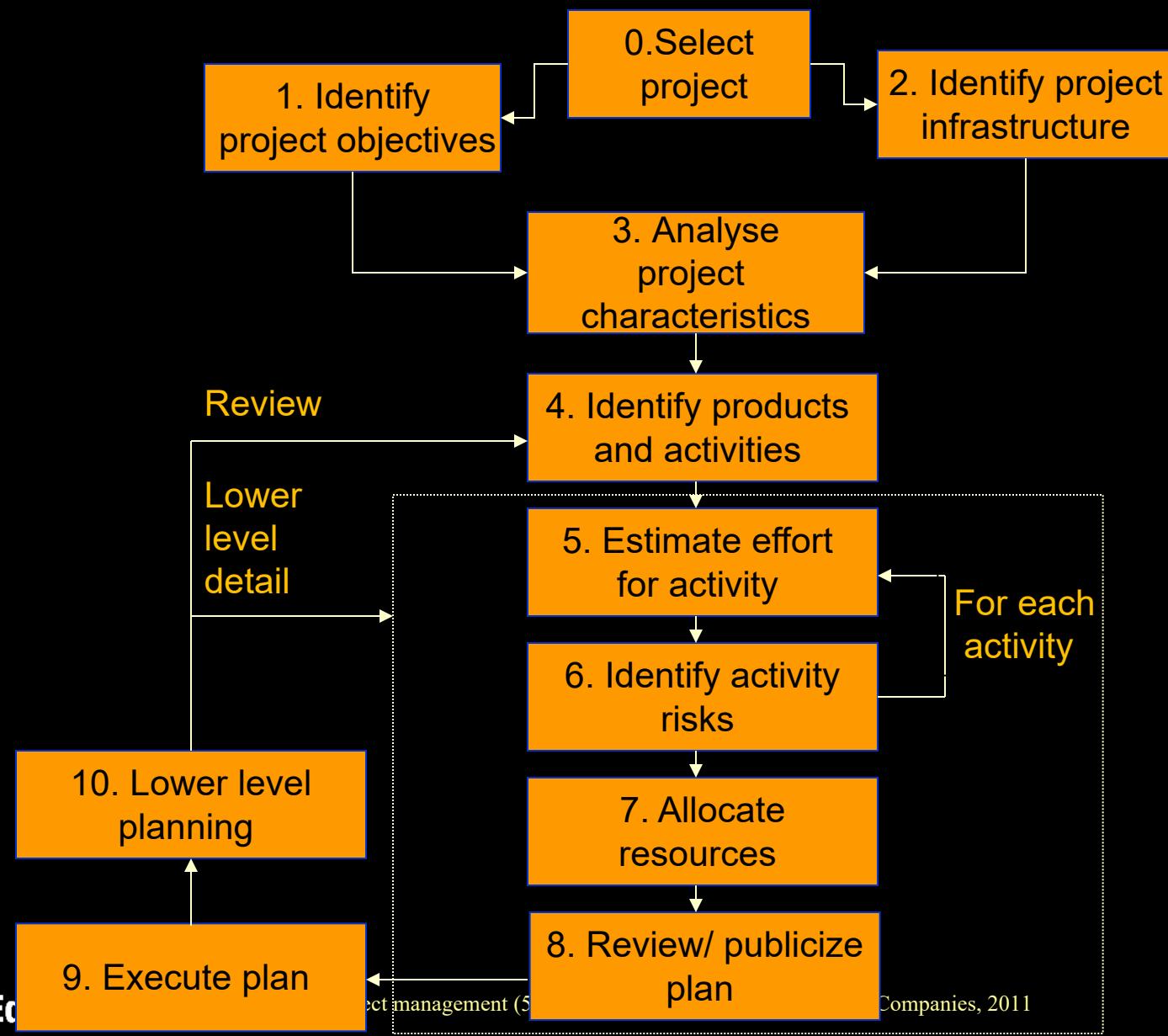
Chapter 3

Step Wise: An approach to planning software projects

‘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



A project scenario: Brightmouth College Payroll

- College currently has payroll processing carried out by a services company
- This is very expensive and does not allow detailed analysis of personnel data to be carried out
- Decision made to bring payroll ‘in-house’ by acquiring an ‘off-the-shelf’ application

Project scenario - continued

- The use of the off-the-shelf system will require a new, internal, payroll office to be set up
- There will be a need to develop some software ‘add-ons’ : one will take payroll data and combine it with time-table data to calculate the staff costs for each course run in the college
- The project manager is Brigette.

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 continued

- 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?’

Back to the scenario

- Project authority

Brigette finds she has two different clients for the new system: the finance department and the personnel office. A vice principal agrees to be official client, and monthly meetings are chaired by the VP and attended by Brigette and the heads of finance and personnel

These meetings would also help overcome communication barriers

Back to the scenario - continued

- Stakeholders
 - ◆ For example, personnel office would supply details of new staff, leavers and changes (e.g. promotions)
 - ◆ To motivate co-operation Brigette might ensure new payroll system produces reports that are useful to personnel staff

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?’
(change control and configuration management standards, quality standards and procedure manuals, measurement)
- 2.3. Identify project team organization
 - ◆ ‘where do I fit in?’
(SW developers, business analyst, business-to-customer web application group, database group)

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?
(tends to be more product driven and the underlying objectives always remain and must be respected)
- 3.2 Analyze other project characteristics (including quality based ones)
 - ◆ what is different about this project?
(information system, process control system, safety critical)

Step 3 continued

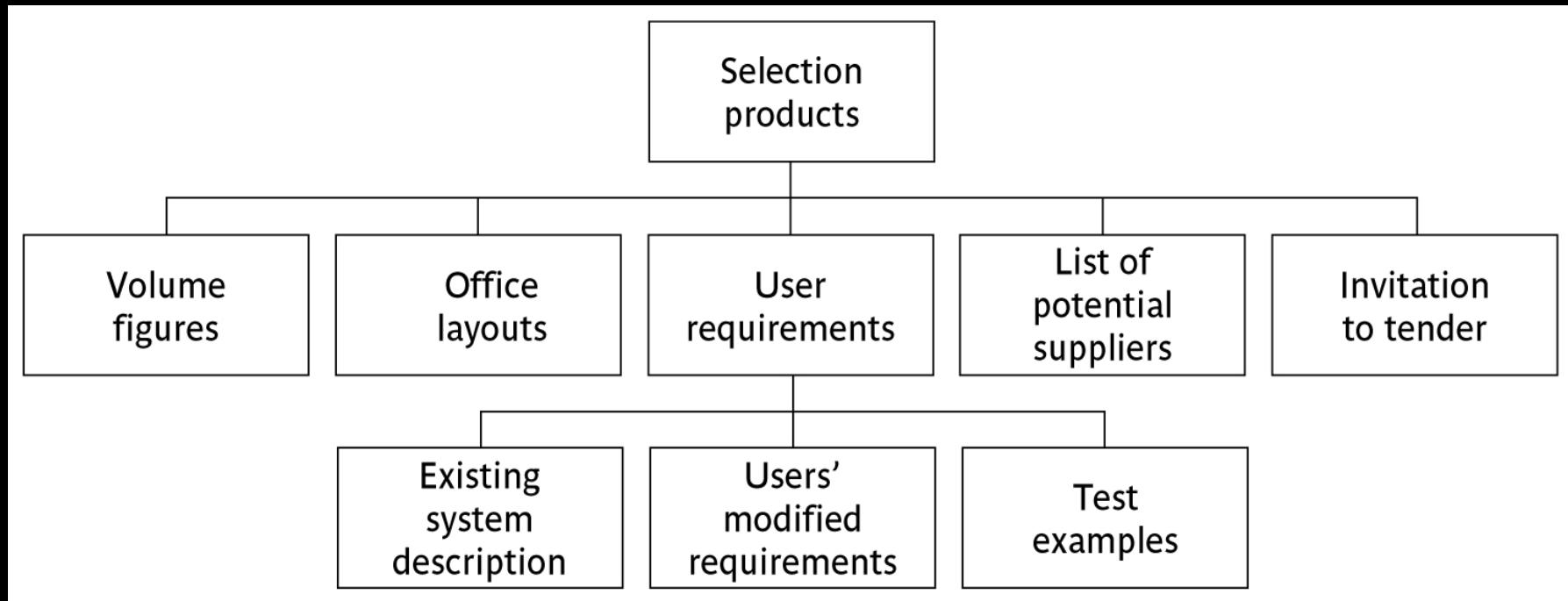
- Identify high level project risks
 - ◆ ‘what could go wrong?’
 - ◆ ‘what can we do to stop it?’
- Take into account user requirements concerning implementation
- Select general life cycle approach
 - ◆ waterfall? Increments? Prototypes?
- Review overall resource estimates
 - ◆ ‘does all this increase the cost?’

Back to the scenario

- Objectives vs. products
 - ◆ An objective-based approach has been adopted
- Some risks
 - ◆ There may not be an off-the-shelf package that caters for the way payroll is processed at Brightmouth College
- Answer?
 - ◆ Brigette decides to obtain details of how main candidate packages work as soon as possible; also agreement that if necessary processes will be changed to fit in with new system.

Step 4 Identify project products and activities

- 4.1 Identify and describe project products - ‘what do we have to produce?’



A fragment of a Product Breakdown Structure (PBS) for a system development task

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

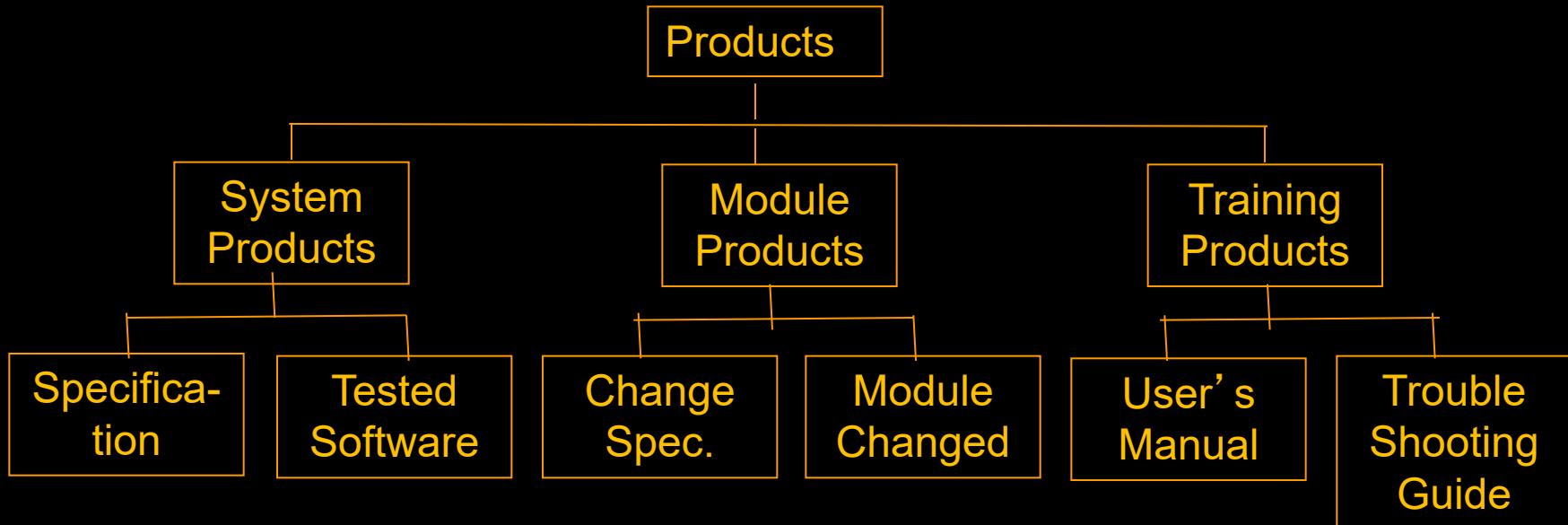
- The following are NOT normally products:
 - ◆ activities (e.g. ‘training’)
 - ◆ events (e.g. ‘interviews completed’)
 - ◆ resources and actors (e.g. ‘software developer’) - may be exceptions to this
- Products CAN BE *deliverable* or *intermediate*

Product description (PD)

- Product identity
- Description - what is it?
- Derivation - what is it based on?
- Composition - what does it contain?
- Format
- Relevant standards
- Quality criteria whether the product is acceptable
- *Create a PD for ‘test data’*

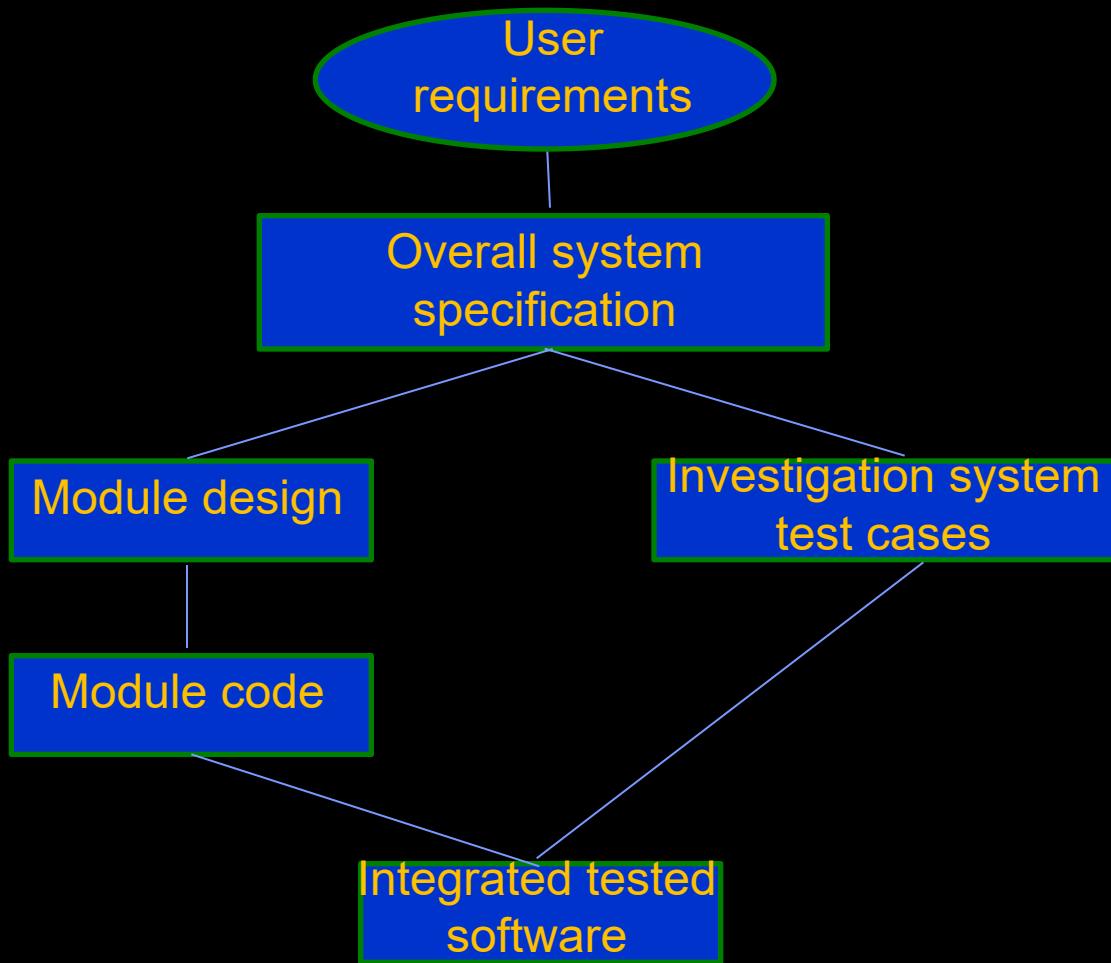
Exercise:

What would be the product breakdown structure of the deliverables of the vendor who would develop the Brightmouth College payroll software by customizing one of its existing products?



- 4.2 document generic product flows

- A program design must be created before the program can be written and program specification must exist.
- The relationships can be portrayed in a Product Flow Diagram (PFD).
- Flow on the diagram to be from top to bottom and left to right.
- User requirement is an oval.



PFD for a software development task

Exercise:

Draw up a possible Product Flow Diagram (PFD) based on Product Breakdown Structure (PBS) to identify some of the products of Brightmouth payroll project based on the information gathered. This is to be presented to potential supplier of the hardware as a part of an ‘invitation to the tender’ .

The volume of figures are the number of employees for whom the records will have to be maintained.

Step 4 continued

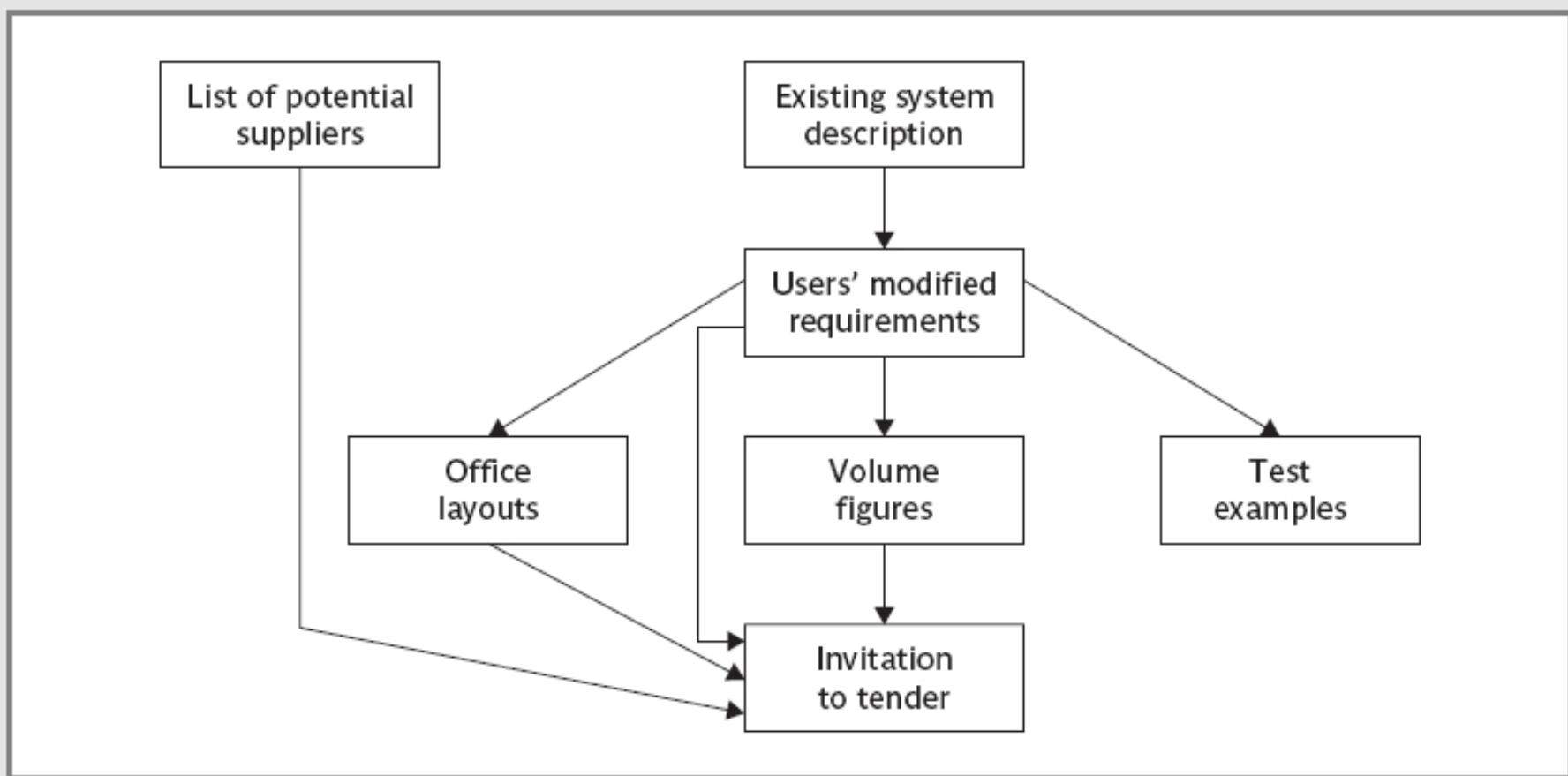


FIGURE B.1 Product Flow Diagram for the creation of an 'invitation to tender'

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

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

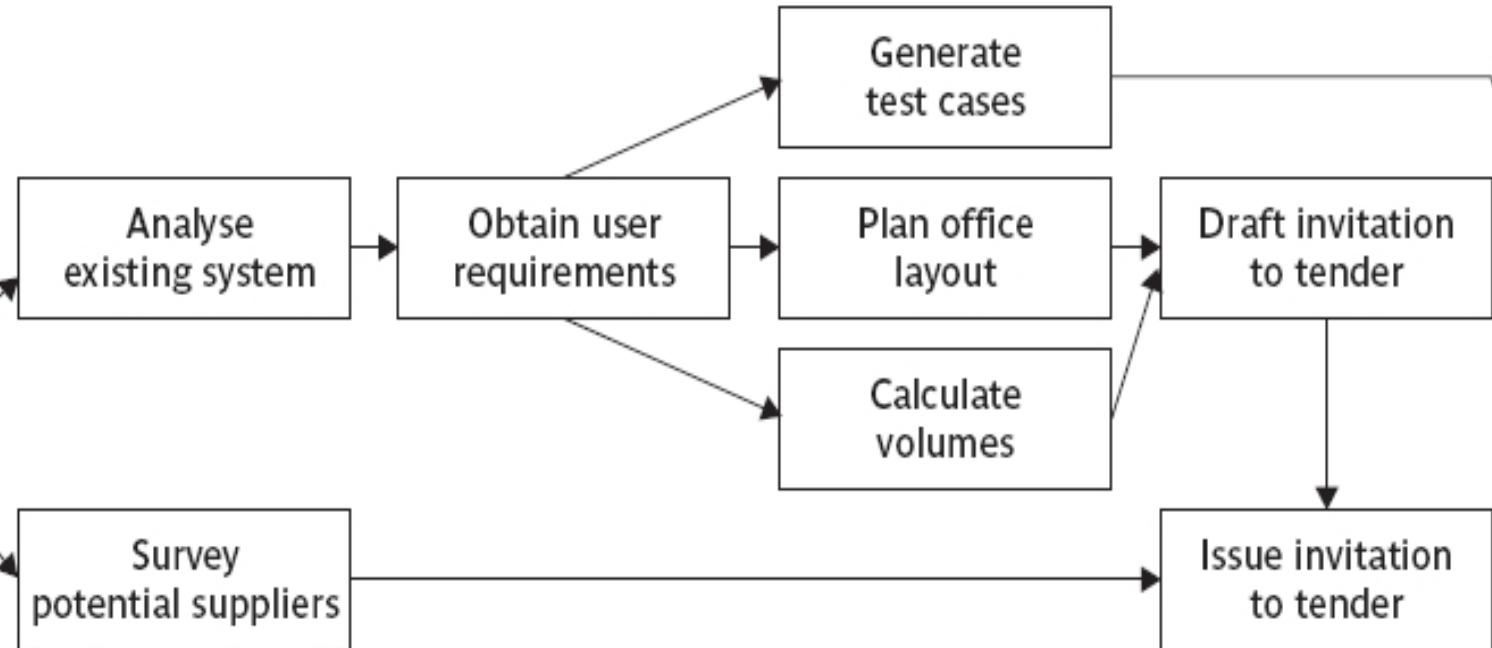
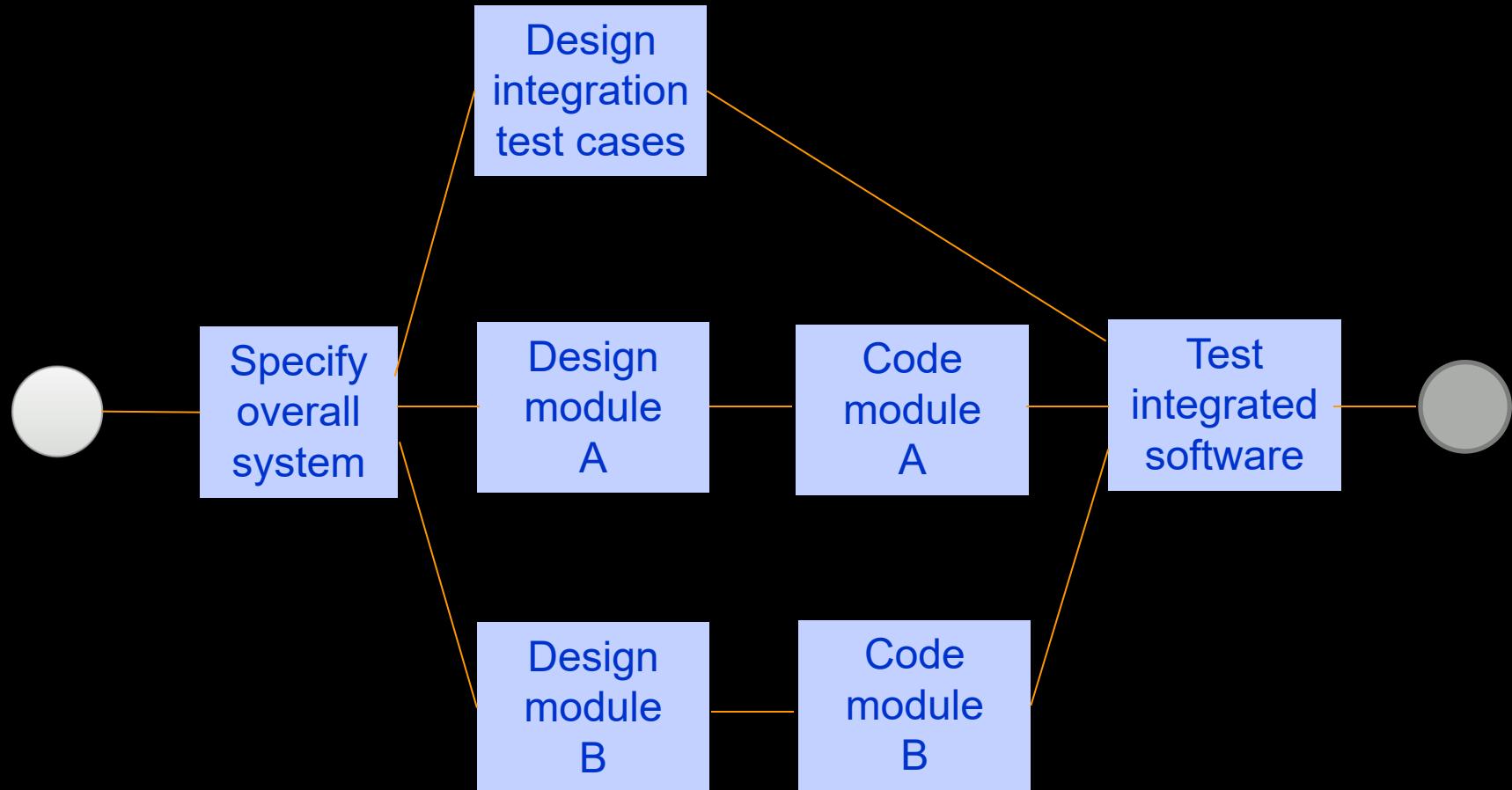


FIGURE B.2 Brightmouth College payroll project activity network fragment

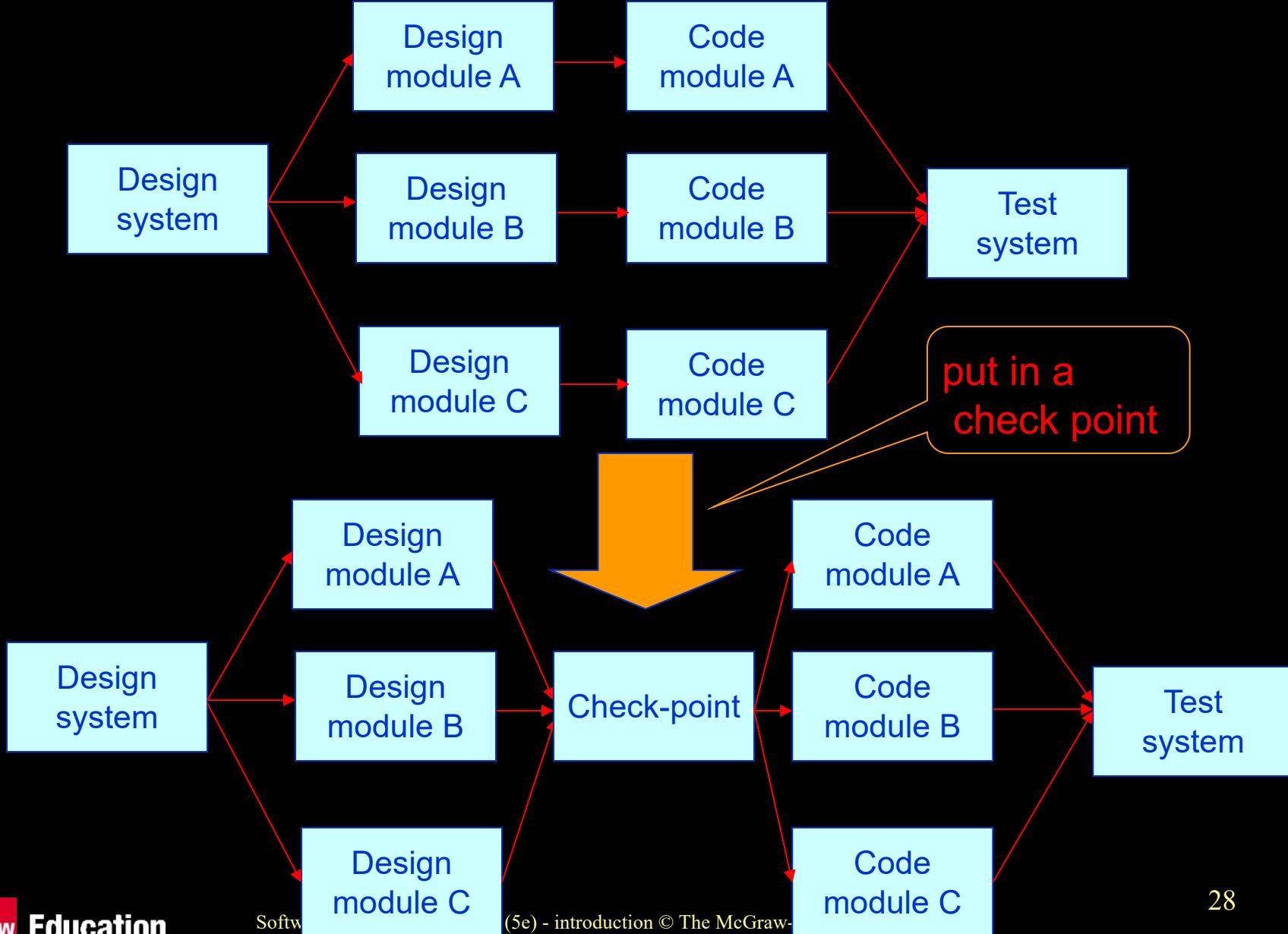


Another example of an activity network

Step 4.5 Add check-points if needed

- An activity will start as soon as the preceding ones are completed.
- Sometimes as per need, divide the project into stages and introduce a check point activities.
- Check point ensures the preceding activities together are complete and compatible.
- This may delay work on some elements of the project.
- There should be a trade-off between efficiency and quality.

Step 4.5 Add check-points ...cont



Step 5: Estimate effort for each activity

- 5.1 Carry out bottom-up estimates
 - ◆ distinguish carefully between *effort* and *elapsed time* (2 persons work for 2 days, the effort is 6 days. Elapsed time is between start and end of the task. Here the elapsed time is 2 days.)
- 5.2. Revise plan to create controllable activities
 - (If an activity involving system testing is to take 12 weeks, it is difficult after six weeks to judge whether 50% work is completed.)
 - ◆ 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 (most likely estimate)
- 6.2. Plan risk reduction and contingency measures
(contingency plans specify actions to be taken if risk materializes e.g. use a contract staff if a project member is not available or sick.)
 - ↳ risk reduction: activity, to stop risk occurring
 - ↳ contingency: action, if risk does occur

- 6.3 Adjust overall plans and estimates to take account of risks
 - ↳ e.g. add 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
(staff available for the project are identified and are provisionally allocated to tasks)
- 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

LT = lead tester
 TA = testing assistant
 ITT = Inf. Tech. transfer

Week
 commencing

MARCH

5

12

19

26

APRIL

2

9

16

Survey potential suppliers

Finance assistant

Analyse existing system

Business analyst

Obtain user requirements

Business analyst

Generate test cases

Systems assistant

Plan office layouts

Premises office

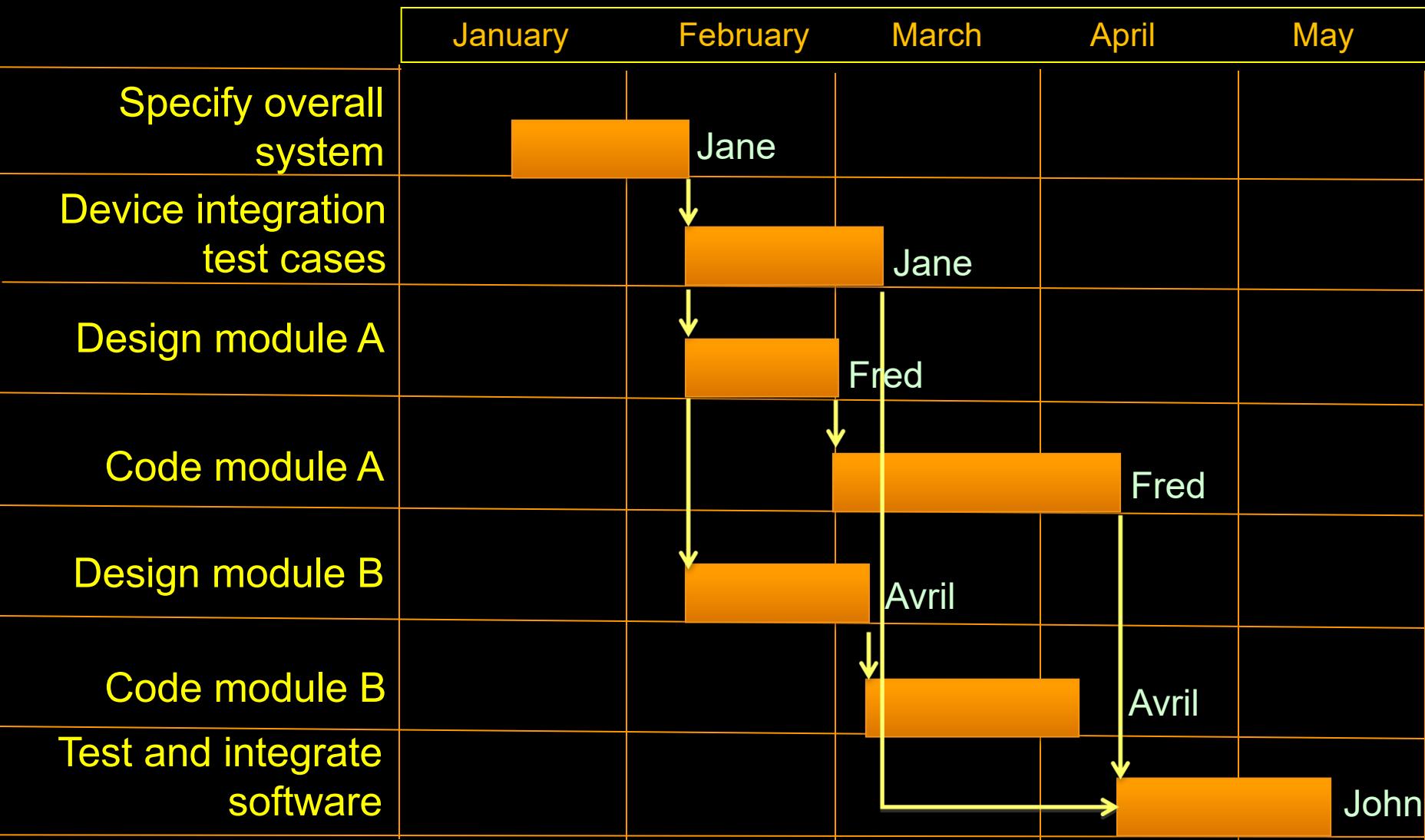
Calculate volumes

Systems assistant

Draft and issue ITT

Business analyst

Gantt chart showing the staff will carry out tasks



Step 8: Review/publicize plan

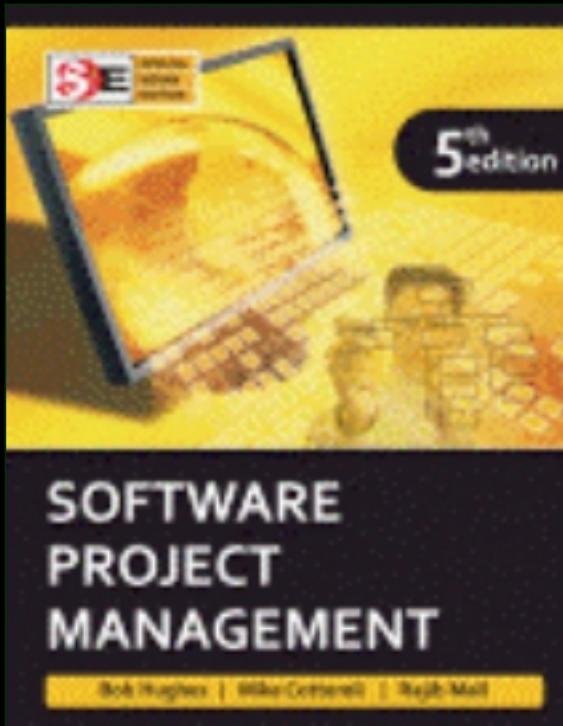
- 8.1 Review quality aspects of project plan
(ensure the quality criteria has been ensured before completion of the project)
- 8.2 Document plan and obtain agreement

Step 9 and 10: Execute plan and create lower level plans

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

Software Project Management



Chapter Two Project evaluation and programme management

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/
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 (e.g. a new computer game). The likely demand for the product would need to be assessed.

Content of the business case - continued

- **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 - continued

- **Outline implementation plan:** how the project is going to be implemented. This should consider the disruption to an organization that a project might cause.
- **Costs:** the implementation plan will supply information to establish these
- **Financial analysis:** combines costs and benefit data to establish value of project

Project portfolio management (PPM)

The concerns of project portfolio management include:

- 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 - continued

There are three elements to PPM:

1. 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

ICT-Information and communication technology

Project portfolio management - continued

2. Project portfolio management

Actual costing and performance of projects can be recorded and assessed

3. 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

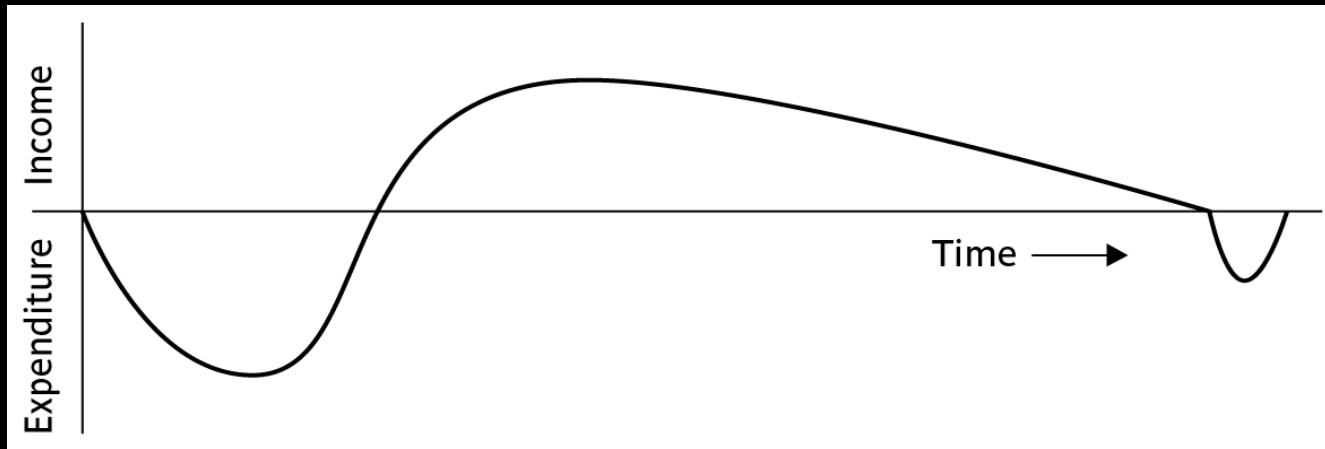
You may want to allow some work to be done outside the portfolio e.g. quick fixes

Cost benefit analysis (CBA)

This relates to an individual project. You need to:

- Identify all the costs which could be:
 - ◆ Development costs
 - ◆ Set-up
 - ◆ Operational costs
- Identify the value of benefits
- Check benefits are greater than costs

Product 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.

Net profit

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

'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

TABLE: A

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

Pay back period

This is the time it takes to start generating a surplus of income over outgoings. What would it be below?

| 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 |

Exercise:

Consider the four projects' cash flow given in TABLE: A and calculate the payback period for each of them.

Ans.

Project 1 ----- 5yrs

Project 2 ----- 5

Project 3 ----- 4

Project 4 ----- 4

Pay back period....cont

Advantages:

1. Simple to calculate
2. Not sensitive to small forecasting errors

Disadvantages:

1. Ignores the overall profitability
2. Totally ignores any income after breakeven.
(project 2 and 4 are better than project 3)

Return on investment (ROI)

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

In the previous example of project 1

- average annual profit
 - = $50,000/5$
 - = 10,000
- ROI
 - = $(10,000/100,000) \times 100$
 - = 10%

It provides a way of comparing the net profitability to the investment required.

Exercise:

Calculate the ROI for each of the other projects shown in TABLE: A and decide which, on the basis of criterion, is the most worthwhile.

Ans.

Project 1 10%

Project 2 2%

Project 3 10%

Project 4 12.5%

Return on investment.... Cont.

- Advantage

The return on investment provides a simple, easy-to-calculate measure of return on capital.

- Disadvantage

1. It takes no account of timing of the cash flow.
2. The rate of return bears no relationship to the interest rates charged by banks.
3. It is potentially very misleading.

Net present value

Would you rather I gave you Rs. 1,000 today or in 12 months time?

If I gave you Rs. 1,000 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 Rs. 1,000 in a year's time?

This figure is the *net present value* of Rs. 1,000 in one year's time

The annual rate by which we discount future earnings is known as *discount rate 10%*.

- The present value of Rs. 1,000 in a year's time is Rs. 910 i.e. Rs. 1000 in a year's time is the equivalent of Rs. 910 now.
- Rs. 1,000 received in two year's time would have a present value of approximately Rs. 830 i.e. Rs. 830 invested at the annual interest rate of 10% would yield approximately Rs. 1,000 in two years time.

For any future cash flow

$$\text{Present value} = \frac{\text{value_in_year_t}}{(1 + r)^t}$$

Discount factor

Discount factor = $1/(1+r)^t$

r is the interest rate (e.g. 10% is 0.10)

t is the number of years

In the case of 10% rate and one year

Discount factor = $1/(1+0.10) = 0.9091 \sim 0.91$

In the case of 10% rate and two years

Discount factor = $1/(1.10 \times 1.10) = 0.8294 \sim 0.83$

Applying discount factors (10%)

TABLE: C

| Year | Cash-flow (Project-1) | 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 |
| Net profit | 50,000 | NPV | 618 |

Exercise:

Assuming 10% discount rate, the NPV for project (TABLE: A) would be calculated as in TABLE: C. The net present value (NPV) for project 1 (TABLE:C), using 10% discount rate, is therefore Rs. 618. Using a 10% discount rate, calculate the NPV for project 2, 3 and 4 and decide which, on the basis of this, is the most beneficial to pursue.

Note: Refer to TABLE: A (slide no. 14).

Answer:

| Year | Discount factor | Discounted cash flow (Rs.) | | |
|------|-----------------|----------------------------|---------------|---------------|
| | | Project 2 | Project 3 | Project 4 |
| 0 | 1.00 | -1,000,000 | -100,000 | -120,000 |
| 1 | 0.90 | 181,820 | 27,273 | 27,273 |
| 2 | 0.82 | 165,280 | 24,792 | 24,792 |
| 3 | 0.75 | 150,260 | 22,539 | 22,539 |
| 4 | 0.68 | 136,600 | 20,490 | 20,490 |
| 5 | 0.62 | 186,270 | 18,627 | 46,568 |
| NPV | | -179,770 | 13,721 | 21,662 |

Exercise:

Calculate the NPV for each of the projects A, B and C shown in table below using each of the discount rate 8%, 10% and 12%. For each of the discount rate, decide which is the best project. What can you conclude from these results?

| Year | Project A (Rs) | Project B (Rs) | Project C (Rs) |
|------------|----------------|----------------|----------------|
| 0 | -8,000 | -8,000 | -10,000 |
| 1 | 4,000 | 1,000 | 2,000 |
| 2 | 4,000 | 2,000 | 2,000 |
| 3 | 2,000 | 4,000 | 6,000 |
| 4 | 1,000 | 3,000 | 2,000 |
| 5 | 800 | 9,000 | 2,000 |
| 6 | 500 | -6,000 | 2,000 |
| Net Profit | 4,000 | 5,000 | 6,000 |

NPV Discount Factors

TABLE: D

| Year | Discount rate (%) | | |
|---|-------------------|--------|--------|
|  | 8% | 10% | 12% |
| 1 | 0.9256 | 0.9091 | 0.8929 |
| 2 | 0.8573 | 0.8264 | 0.7972 |
| 3 | 0.7938 | 0.7513 | 0.7118 |
| 4 | 0.7350 | 0.6830 | 0.6355 |
| 5 | 0.6808 | 0.6209 | 0.5674 |
| 6 | 0.6302 | 0.5645 | 0.5066 |

TABLE: B (Effect on NPV of varying the discount rate)

| Year | Cash flow values (Rs.) | | |
|------------|------------------------|--------------|--------------|
| | Project A | Project B | Project C |
| 0 | -8,000 | -8,000 | -10,000 |
| 1 | 4,000 | 1,000 | 2,000 |
| 2 | 4,000 | 2,000 | 2,000 |
| 3 | 2,000 | 4,000 | 6,000 |
| 4 | 1,000 | 3,000 | 2,000 |
| 5 | 500 | 9,000 | 2,000 |
| 6 | 500 | -6,000 | 2,000 |
| Net profit | 4,000 | 5,000 | 6,000 |
| NPV @ 8% | 2,111 | 2,365 | 2,421 |
| NPV @ 10% | 1,720 | 1,818 | 1,716 |
| NPV @ 12% | 1,356 | 1,308 | 1,070 |

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
- There is a Microsoft Excel function which can be used to calculate

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

Example of a project risk matrix

| Risk | Importance | Likelihood |
|--|------------|------------|
| Client rejects proposed look and feel of site | H | — |
| Competitors undercut prices | H | M |
| Warehouse unable to deal with increased demand | M | L |
| Online payment has security problems | M | M |
| Maintenance costs higher than estimated | L | L |
| Response times deter purchasers | M | M |

TABLE 2.5 A fragment of a basic project/business risk matrix for an e-commerce application

Cost-benefit analysis

BuyRight's income forecast

| Sales | Annual sales income (Rs.) <i>i</i> | Probability <i>p</i> | Expected value (Rs.) <i>i X p</i> |
|-----------------|---------------------------------------|-------------------------|--------------------------------------|
| High | 800,000 | 0.1 | 80,000 |
| Medium | 650,000 | 0.6 | 390,000 |
| Low | 100,000 | 0.3 | 30,000 |
| Expected income | | | 500,000 |

Development costs are estimated = Rs. 750,000

Sales levels are expected to be constant for 4 years

Annual marketing and product maintenance cost = Rs. 200,000

Would you advise going ahead with the project?

Expected sales per year = Rs. 500,000
Annual costs per year = Rs. 200,000
Expected net income / year = Rs. 300,000
-do- for 4 years = $300,000 \times 4 = 1,200,000$

Investment (development cost) = Rs. 750,000

Expected profit = Rs. 450,000

If sales will drop what happens to the benefits and costs?

Then go for,

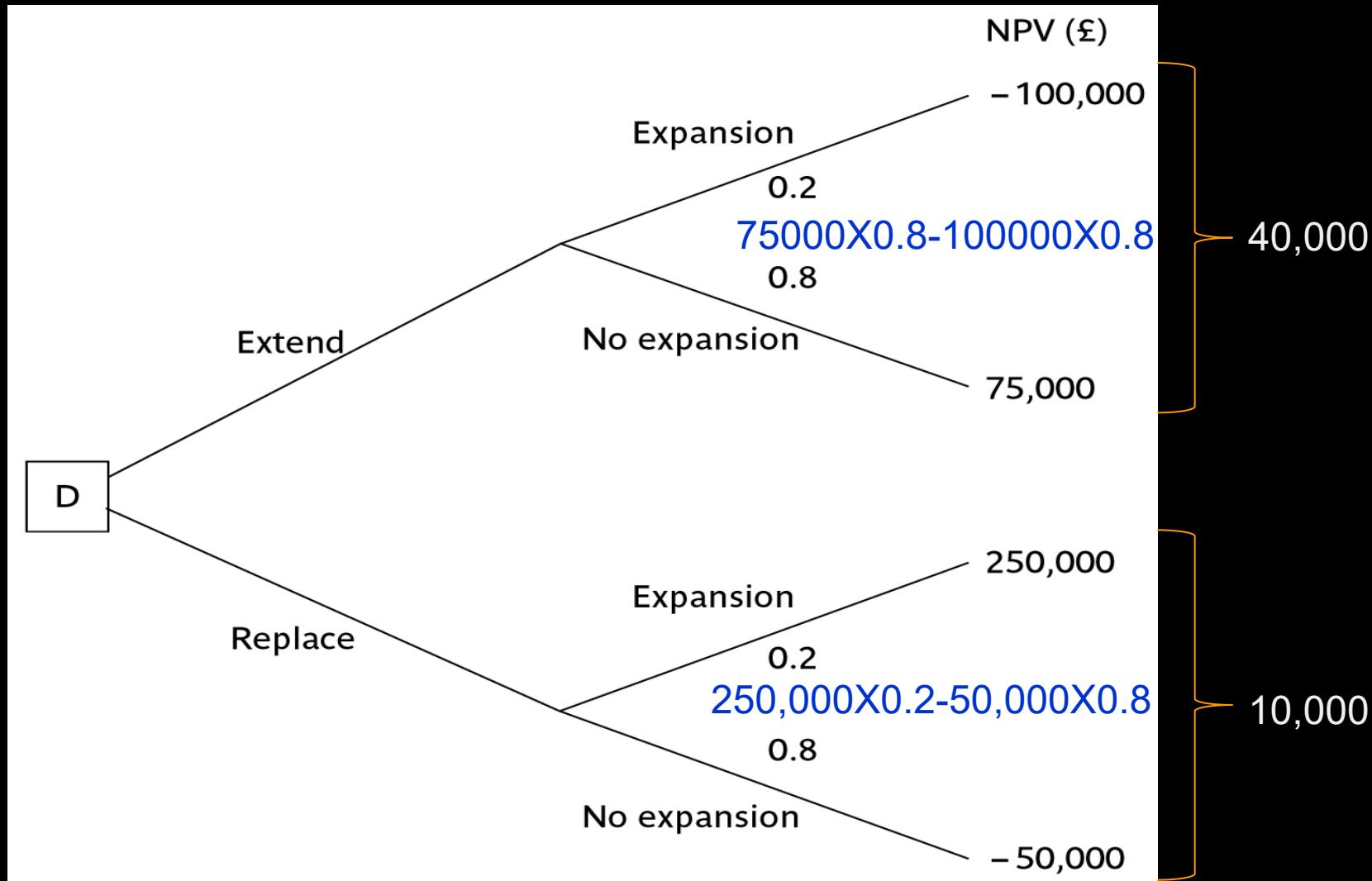
Risk profile analysis – sensitivity analysis

How a decision will affect future profitability of the project?

Decision trees

BuyRight software house

Decision Point



The company should therefore choose the option of extending the existing system

Programme management

- One definition:
'a group of projects that are managed in a co-ordinated way to gain benefits that would not be possible were the projects to be managed independently' Ferns

Programmes may be

- **Strategic** (e.g. merging two organization's computer system)
- **Business cycle programmes** (undertake the project within a particular planning cycle. Budgets and accounting period)
- **Infrastructure programmes** (different projects under one dept. Will require different distinct databases and information system) highway maintenance, refuse collection, education
- **Research and development programmes** (development of new product -risk of failure and potential returns, safe projects- where product is not radically different)
- **Innovative partnerships** (collaborative work on new technology in a 'pre-competitive' phase)

Programme managers versus 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

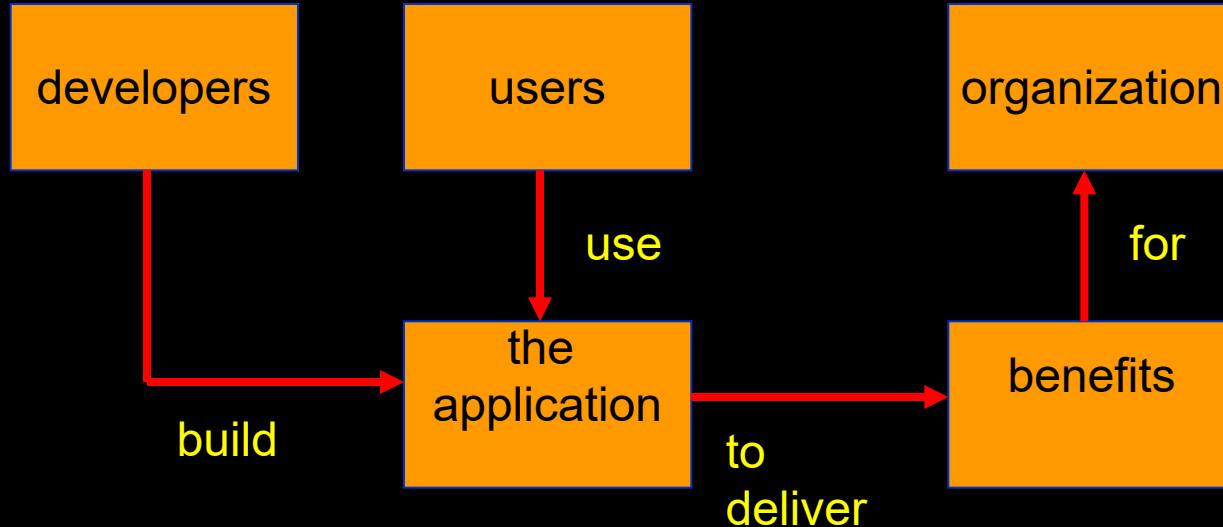
Strategic programmes

- Based on OGC approach
- Initial planning document is the **Programme Mandate** describing
 - ↳ The new services/capabilities that the programme should deliver
 - ↳ How an organization will be improved
 - ↳ Fit with existing organizational goals
- A **programme director** appointed a champion for the scheme

Next stages/documents

- **The programme brief** – equivalent of a feasibility study: emphasis on costs and benefits
- **The vision statement** – explains the new capability that the organization will have
- **The blueprint** – explains the changes to be made to obtain the new capability

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

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 - continued

- 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

Software Project Management



Chapter Four

Selection of an appropriate project approach

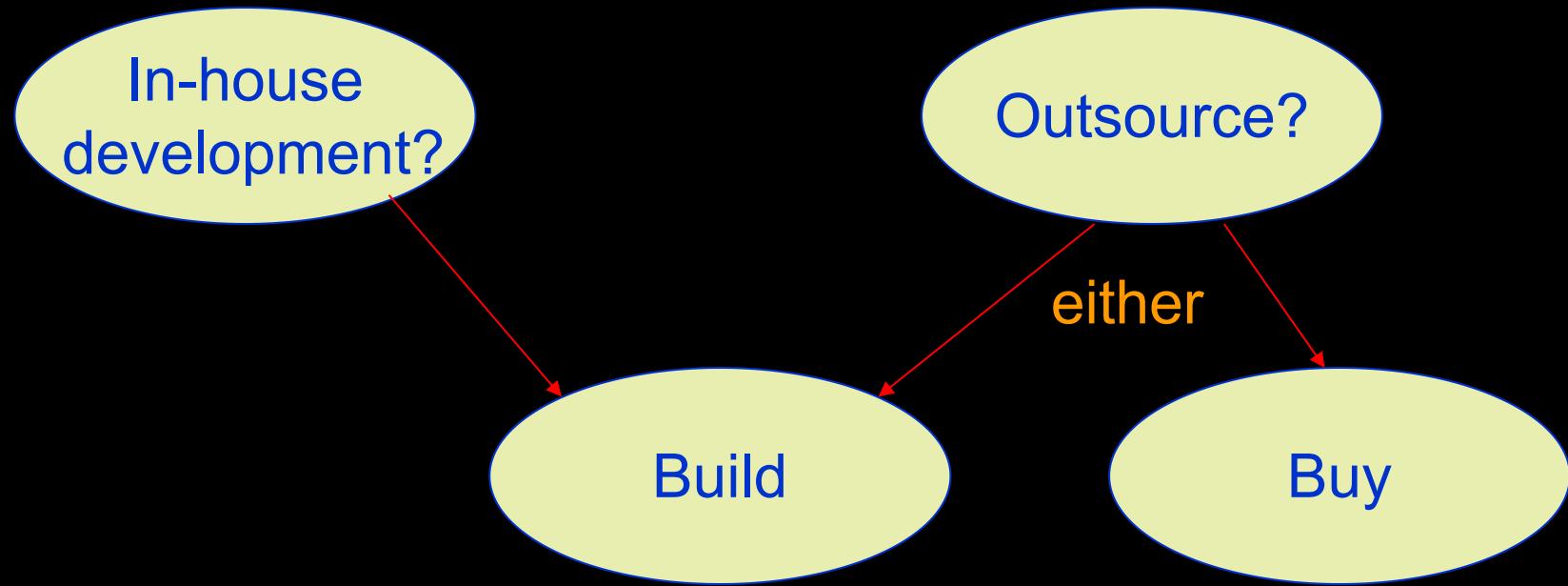
Outline of lecture

- 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

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 trialled 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 off-the-shelf

- 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 e.g.
 - ◆ are requirements well understood?
 - ◆ are technologies to be used well understood?
- Look at the type of application being built e.g.
 - ◆ information system? embedded system?
 - ◆ criticality? differences between target and development environments?
- Clients' own requirements
 - ◆ need to use a particular method

Structure versus 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
- Example: use of UML and USDP
- Future vision: Model-Driven Architecture (MDA). UML supplemented with Object Constraint Language (OCL), press the button and application code generated from the UML/OCL model

Note: USDP-Unified software development process

Structure versus speed of delivery

Agile methods

- Emphasis on speed of delivery rather than documentation
- RAD Rapid application development emphasized use of quickly developed prototypes (developer and user work together for few days, free from outside interruption, clean room development environment)
- JAD Joint application development. Requirements are identified and agreed in intensive workshops with users (intensive interaction with users, interviewing key personnel, creation of preliminary data and process)

Processes versus Process Models

(Process – a system in action, Process model- organizing the activities)

- Starting from the inception stage:
 - ◆ A product undergoes a series of transformations through a few identifiable stages
 - ◆ Until it is fully developed and released to the customer.
 - ◆ This forms its life cycle or development process.
- Life cycle model (also called a process model):
 - ◆ A graphical or textual representation of the life cycle.

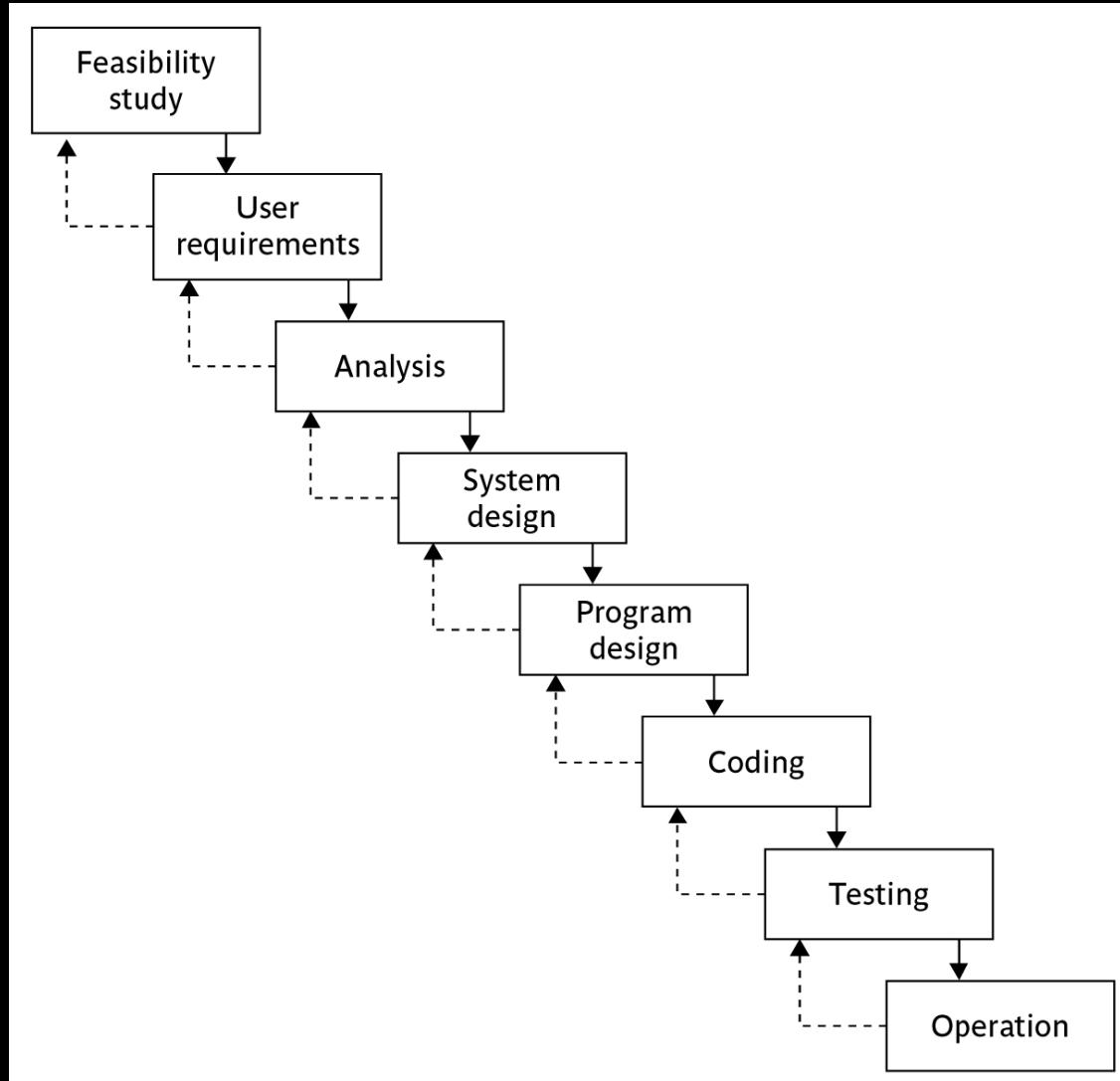
Choice of process models

- ‘waterfall’ also known as ‘one-shot’, ‘once-through’
- incremental delivery
- evolutionary development

Also use of ‘agile methods’ e.g. extreme programming

Waterfall

One-shot or once-through



Waterfall

- the ‘classical’ model
- imposes structure on the project
- every stage needs to be checked and signed off
- BUT
 - ◆ 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

Evolutionary delivery: prototyping

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

Sprague and McNurlin

main types

- ‘throw away’ prototypes
- evolutionary prototypes

what is being prototyped?

- human-computer interface
- functionality

Reasons for prototyping

- learning by doing
- improved communication
- improved user involvement
- a feedback loop is established
- reduces the need for documentation
- reduces maintenance costs i.e. changes after the application goes live
- prototype can be used for producing expected results

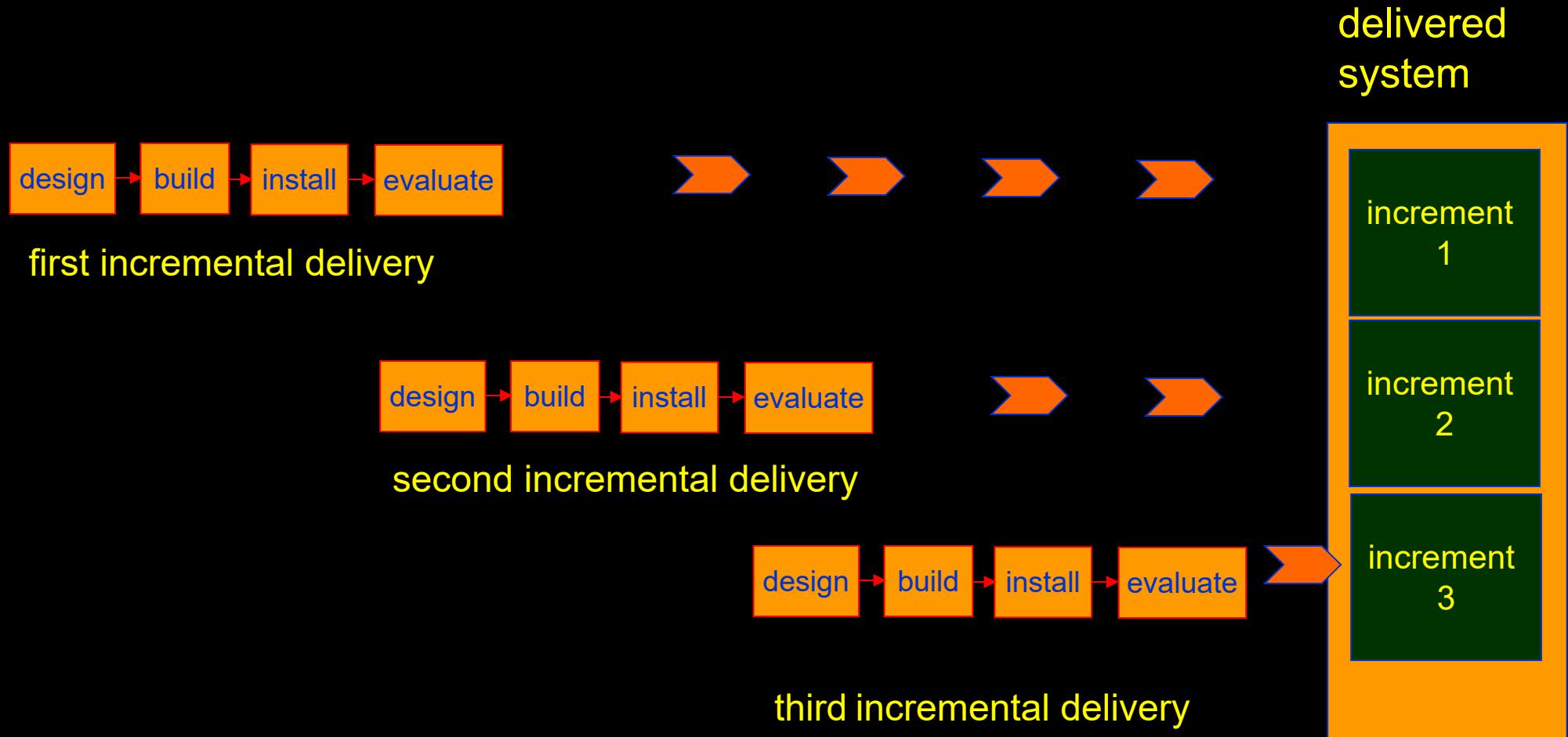
prototyping: some dangers

- 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

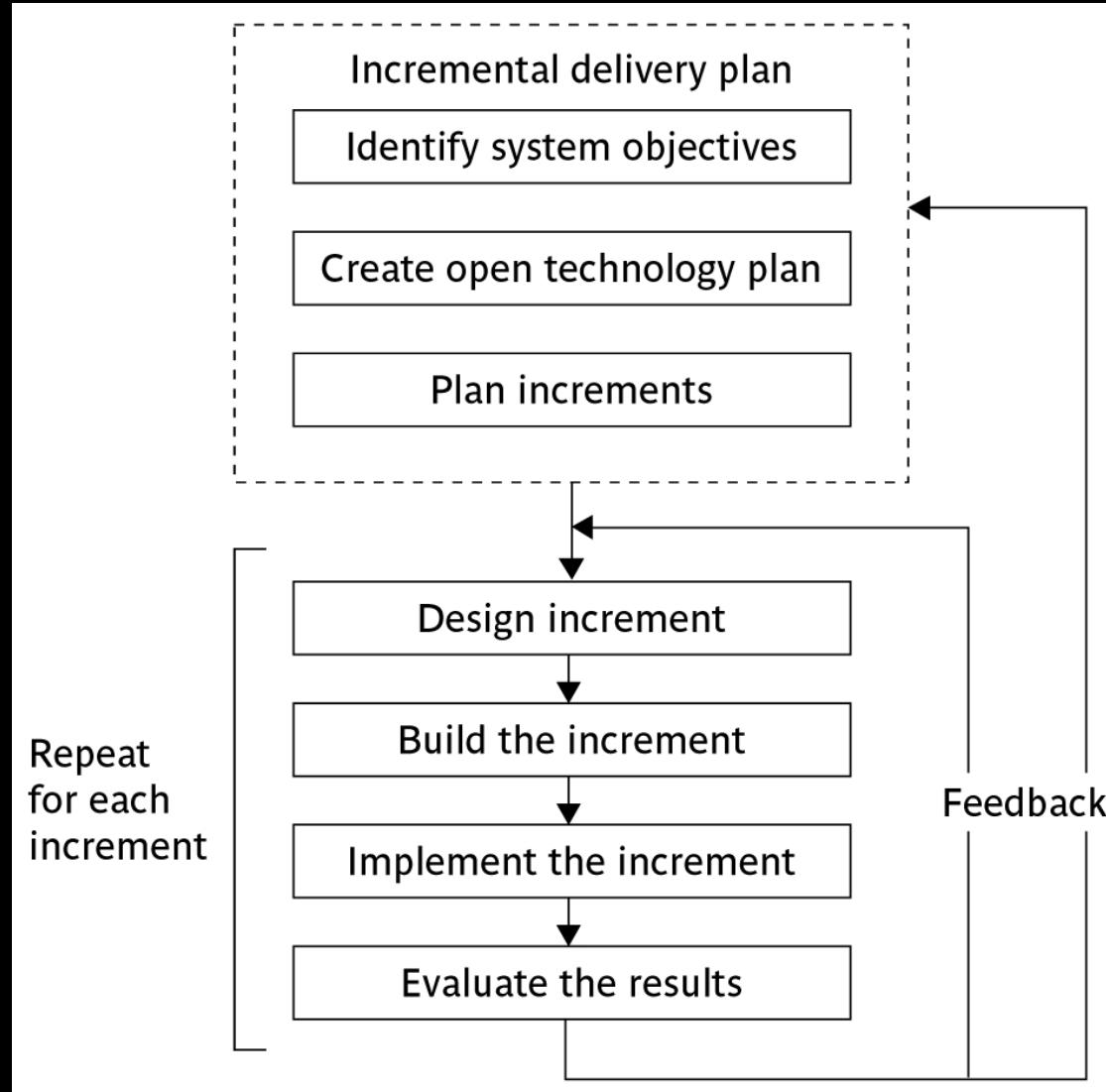
other ways of categorizing prototyping

- 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*
(vertical- some, but not all features are prototyped fully)
(horizontal- All features are prototyped but not in detail)

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’

Overview of incremental plan

- steps ideally 1% to 5% of the total project
- non-computer steps should be included
- 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 | |
|-----------------------------------|-------|------|-------|-----|
| profit reports | 9 | 1 | 9 | 2nd |
| online database | 1 | 9 | 0.11 | 5th |
| ad hoc enquiry | 5 | 5 | 1 | 4th |
| purchasing plans | 9 | 4 | 2.25 | 3rd |
| profit- based pay for managers | 9 | 0 | inf | 1st |

Genesis of ‘Agile’ methods

Structured development methods have several disadvantages

- ◆ produce large amounts of documentation which can largely remain 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 etc. etc

The answer? ‘Agile’ methods?

Agile Methods

- Agile is an umbrella term that refers to a group of development processes:
 - ◆ Crystal technologies
 - ◆ Atern (formerly DSDM)
 - ◆ Feature-driven development
 - ◆ Scrum
 - ◆ Extreme Programming (XP)
- Similar themes:
 - ◆ Some variations

Important Themes of Agile Methods

- Incremental approach:
 - ◆ At a time, only one increment is planned, developed, and then deployed at the customer site.
- Agile model emphasizes face-to-face communication over written documents.
- An agile project usually includes a customer representative in the team.
- Agile development projects usually deploy pair programming.

Atern/Dynamic system development method (DSDM)

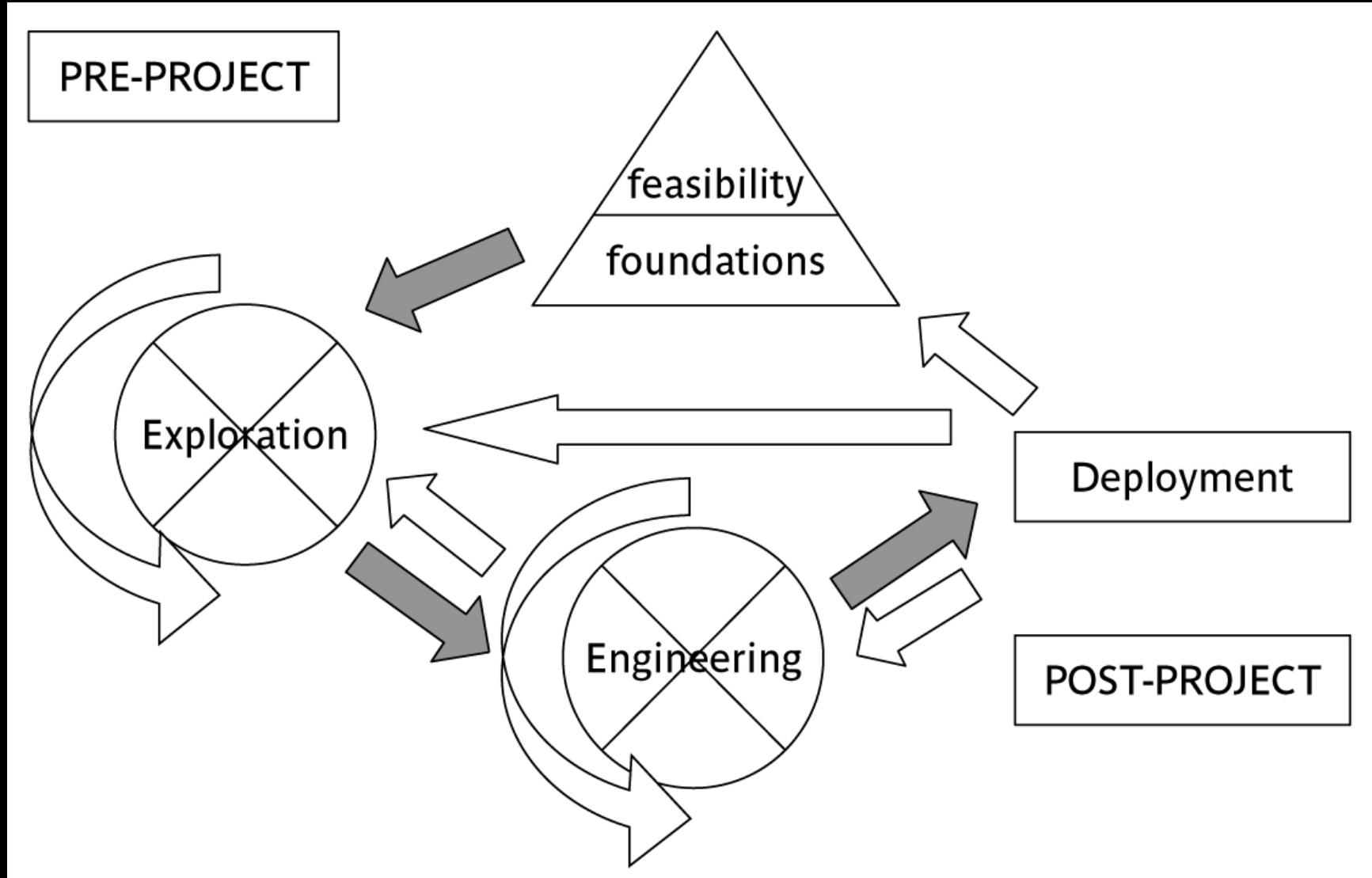
- UK-based consortium
- *arguably* DSDM can be seen as replacement for SSADM
- DSDM is more a project management approach than a development approach
- Can still use DFDs, LDSs etc!
- An update of DSDM has been badged as ‘Atern’

SSADM- Structured system analysis and design method

Six core 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

Atern/DSDM framework



Atern DSDM: time-boxing

- *time-box* fixed deadline by which *something* has to be delivered
- typically two to six weeks
- MOSCOW priorities
 - ↳ Must have - essential
 - ↳ Should have - very important, but system could operate without
 - ↳ Could have
 - ↳ Want - but probably won't get!

Extreme programming

- increments of one to three 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 - contd

- 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

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

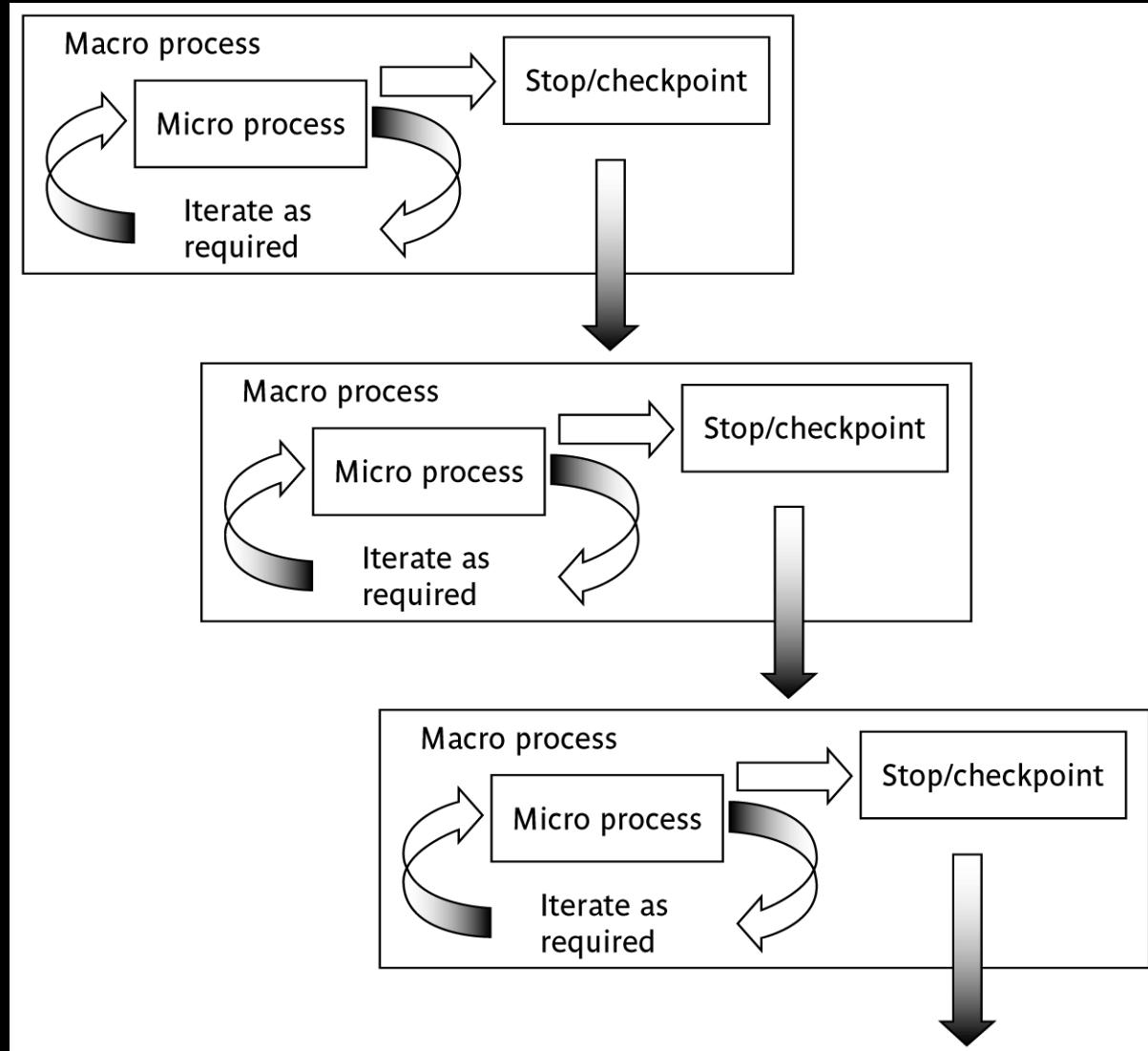
Grady Booch's concern

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

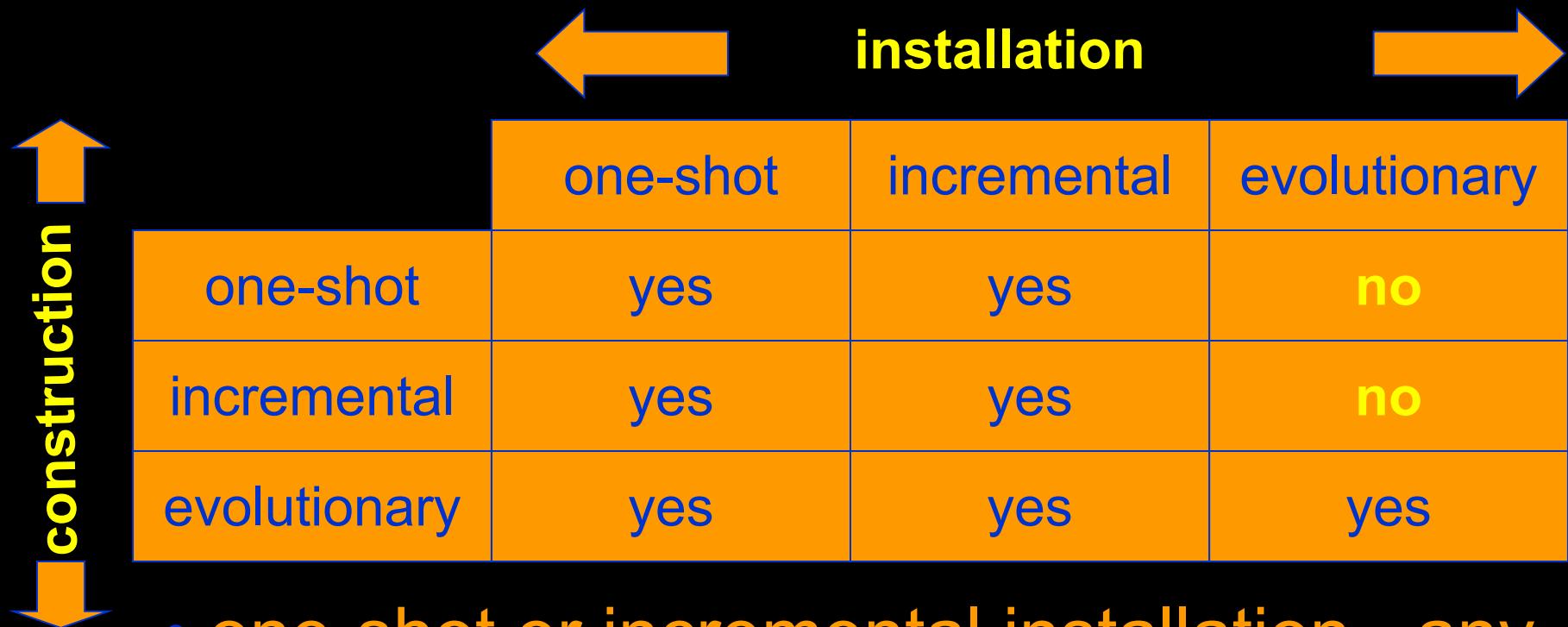
'Conceptual integrity sometimes suffers because this is little motivation to deal with scalability, extensibility, portability, or reusability beyond what any vague requirement might imply'

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

macro and micro processes



combinations of approach



- one-shot or incremental installation - any construction approach possible
- evolutionary installation implies evolutionary construction

‘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

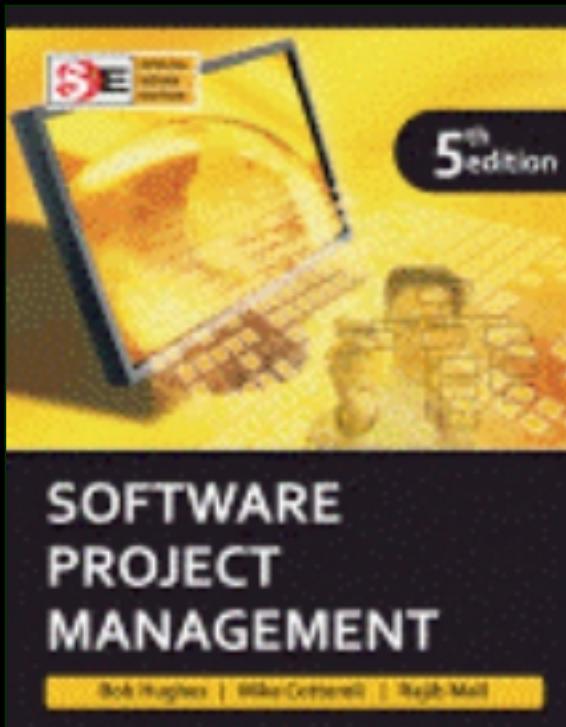
IF uncertainty and complexity both low

THEN use one-shot

IF schedule is tight

THEN use evolutionary or incremental

Software Project Management



Chapter Five

Software effort estimation

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

Difficulties in estimating due to complexity and invisibility of software.

BUT what if the targets are not achievable?

Some problems with estimating

- Subjective nature of much of estimating
 - ◆ Under estimating the difficulties of small task and over estimating large projects.
- Political pressures
 - ◆ Managers may wish to reduce estimated costs in order to win support for acceptance of a project proposal (over estimate to create a comfort zone)
- Changing technologies
 - ◆ these bring uncertainties, especially in the early days when there is a 'learning curve'. Difficult to use the experience of previous projects.
- Projects differ -lack of homogeneity of project experience
 - ◆ Experience on one project may not be applicable to another

Exercise:

Calculate the productivity (i.e. SLOC/work month) of each of the projects in Table next slide and also for the organization as a whole. If the project leaders for projects **a** and **d** had correctly estimated the source number of lines of code (SLOC) and then used the average productivity of the organization to calculate the effort needed to complete the projects, how far out would their estimate have been from the actual one.

| Project | Design | | Coding | | Testing | | Total | |
|---------|--------|-----|--------|-----|---------|-----|-------|-------|
| | wm | (%) | wm | (%) | wm | (%) | wm | SLOC |
| a | 3.9 | 23 | 5.3 | 32 | 7.4 | 44 | 16.7 | 6050 |
| b | 2.7 | 12 | 13.4 | 59 | 6.5 | 36 | 22.6 | 8363 |
| c | 3.5 | 11 | 26.8 | 83 | 1.9 | 6 | 32.2 | 13334 |
| d | 0.8 | 21 | 2.4 | 62 | 0.7 | 18 | 3.9 | 5942 |
| e | 1.8 | 10 | 7.7 | 44 | 7.8 | 45 | 17.3 | 3315 |
| f | 19.0 | 28 | 29.7 | 44 | 19.0 | 28 | 67.7 | 38988 |
| g | 2.1 | 21 | 7.4 | 74 | 0.5 | 5 | 10.1 | 38614 |
| h | 1.3 | 7 | 12.7 | 66 | 5.3 | 27 | 19.3 | 12762 |
| i | 8.5 | 14 | 22.7 | 38 | 28.2 | 47 | 59.5 | 26500 |

| Project | Work-month | SLOC | Productivity (SLOC/month) |
|----------------|--------------|----------------|------------------------------|
| a | 16.7 | 6,050 | 362 |
| b | 22.6 | 8,363 | 370 |
| c | 32.2 | 13,334 | 414 |
| d | 3.9 | 5,942 | 1,524 |
| e | 17.3 | 3,315 | 192 |
| f | 67.7 | 38,988 | 576 |
| g | 10.1 | 38,614 | 3,823 |
| h | 19.3 | 12,762 | 661 |
| i | 59.5 | 26,500 | 445 |
| Overall | 249.3 | 153,868 | 617 |

| Project | Estimated work-month | Actual | Difference |
|---------|----------------------|--------|------------|
| a | $6050 / 617 = 9.80$ | 16.7 | 6.90 |
| d | $5942 / 617 = 9.63$ | 3.9 | - 5.73 |

Over and under-estimating

- Parkinson's Law: 'Work expands to fill the time available'
- An over-estimate is likely to cause project to take longer than it would otherwise
- Weinberg's Zeroth Law of reliability: 'a software project that does not have to meet a reliability requirement can meet any other requirement'
- Brook's Law: 'putting more people on a late job makes it later'. If there is an overestimate of the effort required, this could lead to more staff being allocated than needed and managerial overheads being increased.

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’ (LOC) – but this can have problems
 - ◆ FP measure corrects to a great extent.

A taxonomy of estimating methods

- Bottom-up - activity based, analytical (WBS – insert, amend, update, display, delete, print)
- Parametric or algorithmic models (Top-down approach)
- Expert opinion - just guessing?
- Analogy - case-based, comparative
- Albrecht function point analysis

Parameters to be Estimated

- Size is a fundamental measure of work
- Based on the estimated size, two parameters are estimated:
 - ↳ Effort
 - ↳ Duration
- Effort is measured in person-months:
 - ↳ One person-month is the effort an individual can typically put in a month.

Measure of Work

- The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.
- Two metrics are used to measure project size:
 - ◆ Source Lines of Code (SLOC)
 - ◆ Function point (FP)
- FP is now-a-days favored over SLOC:
 - ◆ Because of the many shortcomings of SLOC.

Major Shortcomings of SLOC

- No precise definition (e.g. comment line, data declaration line to be included or not?)
- Difficult to estimate at start of a project
- Only a code measure
- Programmer-dependent
- Does not consider code complexity

Bottom-up versus top-down

- Bottom-up
 - ↳ use when no past project data
 - ↳ identify all tasks that have to be done – so quite time-consuming
 - ↳ use when you have no data about similar past projects
- Top-down
 - ↳ produce overall estimate based on project cost drivers
 - ↳ based on past project data
 - ↳ divide overall estimate between jobs to be done

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

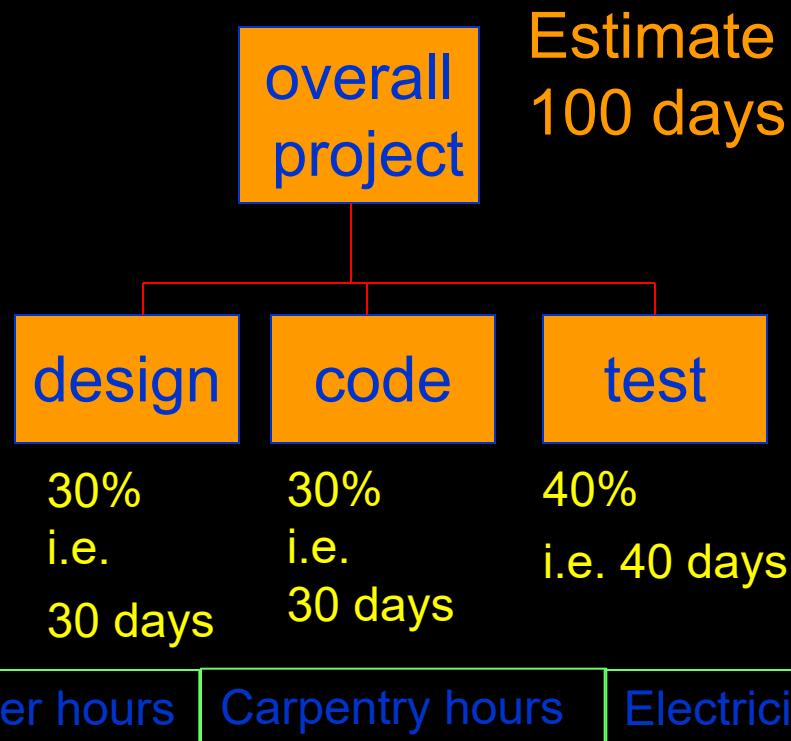
A procedural code-oriented approach

- a) Envisage the number and type of modules in the final system
- b) Estimate the SLOC of each individual module
- c) Estimate the work content
- d) Calculate the work-days effort

Top-down estimates

Normally associated with parametric (or algorithmic) model.

Ex: House building project



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

Algorithmic/Parametric models

- COCOMO (lines of code) and function points examples of these
- Problem with COCOMO etc:



but what is desired
is



Parametric models - the need for historical data

- simplistic model for an estimate
 $\text{estimated effort} = (\text{system size}) / (\text{productivity})$
- e.g.
 - system size = lines of code
 - productivity = lines of code per day
- productivity = (system size) / effort
 - ↳ based on past projects

Software size = 2 KLOC

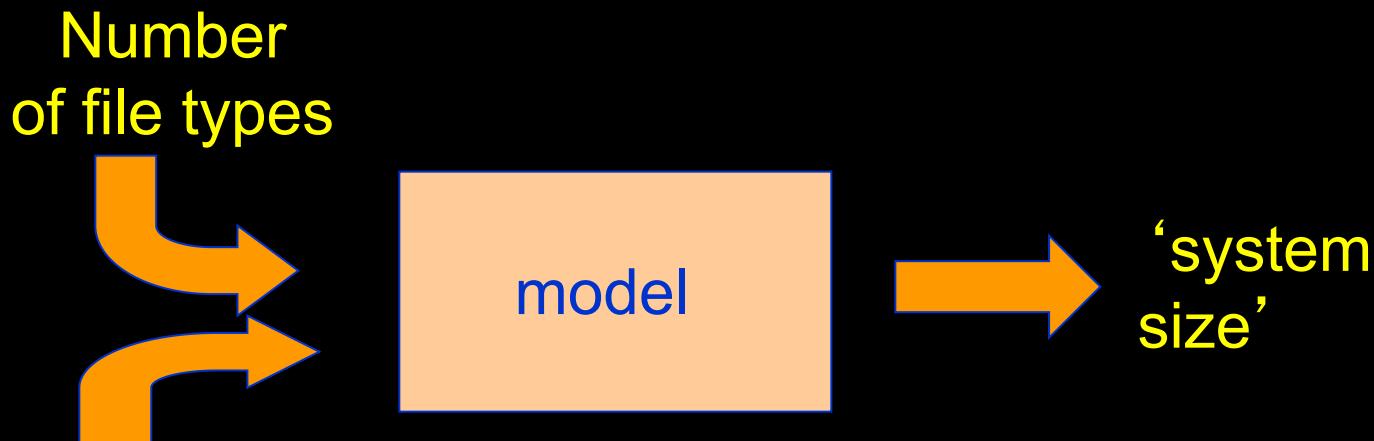
If A (expert) can take 40 days per KLOC 80 days

If B (novice) takes 55 days per KLOC 110 days

KLOC is a size driver, experience influences productivity

Parametric models

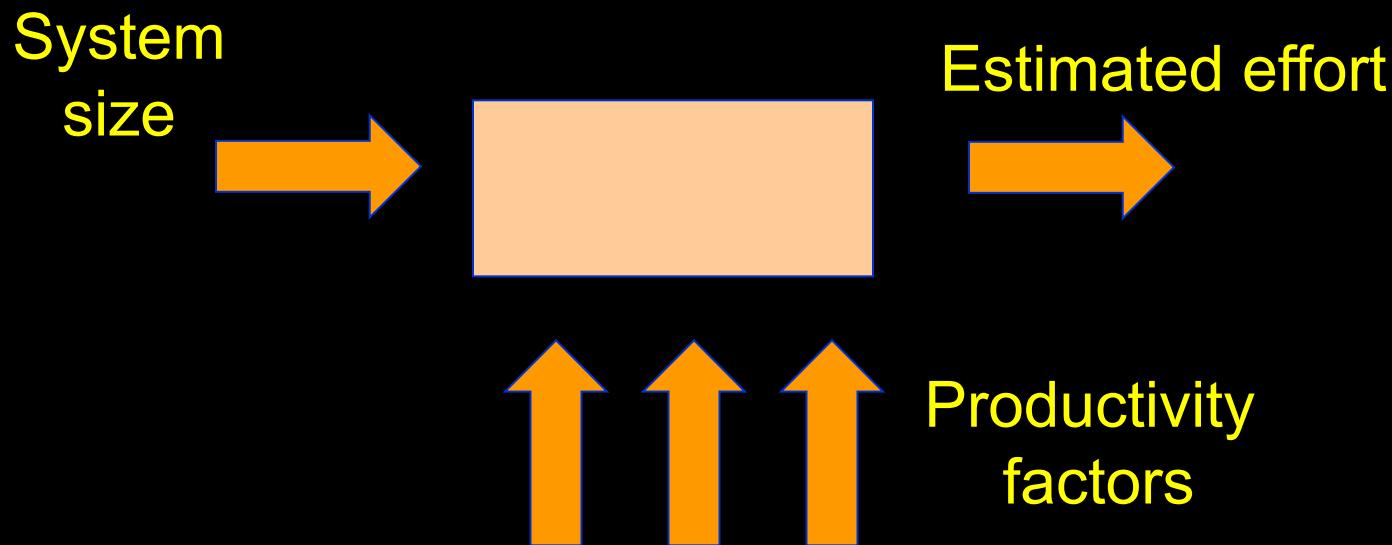
- Some models focus on task or system size e.g. Function Points
- FPs originally used to estimate Lines of Code, rather than effort



Numbers of input
and output transaction types

Parametric models

- Other models focus on productivity: e.g. COCOMO
- Lines of code (or FPs etc) an input



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

Note: Delphi technique – group decision making

Estimating by analogy (Case-based reasoning)

source cases-completed projects

| | |
|------------------|--------|
| attribute values | effort |

Use effort adjustment
from source as estimate

target case (new)

attribute values | ??????

Select case
with closest attribute
values

Estimating by analogy...cont.

- Use of ANGEL software tool (measure the Euclidean distance between project cases)
- Example:

Say that the cases are being matched on the basis of two parameters, the number of inputs to and the number of outputs from the application to be built. The new project is known to require 7 inputs and 15 outputs. One of the past cases, project A, has 8 inputs and 17 outputs.

The Euclidian distance between the source and the target is therefore $\sqrt{((7 - 8)^2 + (17 - 15)^2)} = 2.24$

- Exercise:

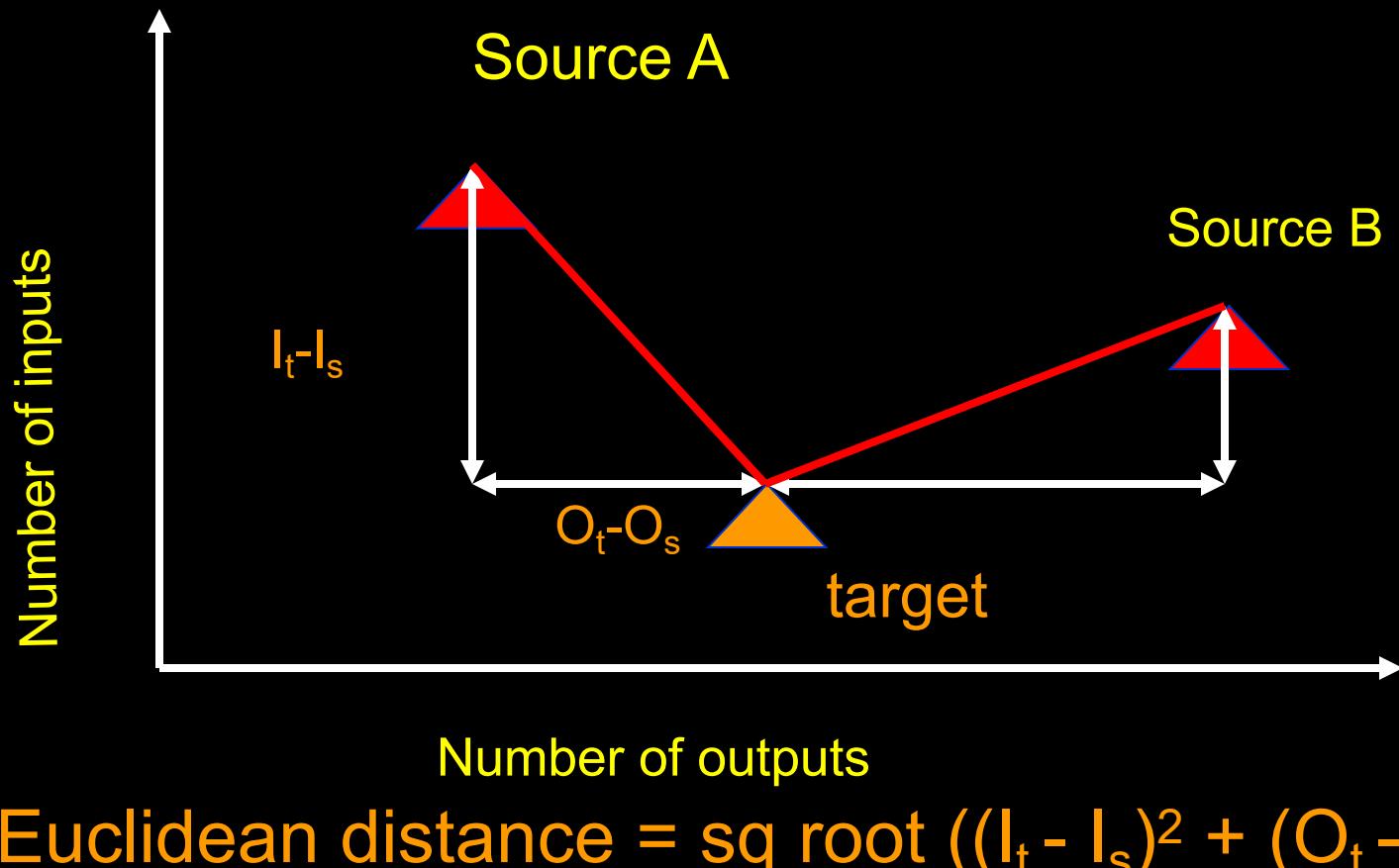
Project B has 5 inputs and 10 outputs. What would be the Euclidian distance between this project and the target new project being considered in the previous slide? Is project B a better analogy with the target than project A?

The Euclidian distance between project B and the target case is

$$\sqrt{((7-5)^2 + (15-10)^2)} = 5.39.$$

Therefore project A is a closer analogy.

Machine assistance for source selection (ANGEL)



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

Parametric models

We are now looking more closely at four parametric models:

Albrecht/IFPUG function points

Symons/Mark II function points

COSMIC function points

COCOMO81 and COCOMO II

COSMIC- common software measurement consortium

COCOMO- cost constructive model

IFPUG- international function point user group

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

Note: IFPUG- International FP User Group

Albrecht/IFPUG function points - continued

Five function types

1. Logical internal file (LIF) types – equates roughly to a data store in systems analysis terms. Created and accessed by the target system
(it refers to a group data items that is usually accessed together i.e. one or more record types)
PURCHASE-ORDER and PURCHASE-ORDER-ITEM
2. External interface file types (EIF) – where data is retrieved from a data store which is actually maintained by a different application.

Albrecht/IFPUG function points - continued

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

Table-1

| External user types | Low complexity | Medium complexity | High complexity |
|-------------------------------------|----------------|-------------------|-----------------|
| EI External input type | 3 | 4 | 6 |
| EO External output type | 4 | 5 | 7 |
| EQ External inquiry type | 3 | 4 | 6 |
| LIF Logical internal file type | 7 | 10 | 15 |
| EIF External interface file type | 5 | 7 | 10 |

With FPs originally defined by Albecht, the external user type is of high, low or average complexity is intuitive. Ex: in the case of **logical internal files** and **external interface files**, the boundaries shown in table below are used to decide the complexity level.

Table-2

| Number of record types | Number of data types | | |
|------------------------|----------------------|---------|---------|
| | < 20 | 20 – 50 | > 50 |
| 1 | Low | Low | Average |
| 2 to 5 | Low | Average | High |
| > 5 | Average | High | High |

Example

A **logical internal file** might contain data about purchase orders. These purchase orders might be organized into two separate record types: the main PURCHASE-ORDER details, namely purchase order number, supplier reference and purchase order date. The details of PURCHASE-ORDER-ITEM specified in the order, namely the product code, the unit price and number ordered.

- The number of record types for this will be 2
- The number of data types will be 6
- According to the previous table-2 file type will be rated as ‘Low’
- According to the previous table-1 the FP count is 7

Examples

Payroll application has:

1. Transaction to input, amend and delete employee details – an EI that is rated of medium complexity
2. A transaction that calculates pay details from timesheet data that is input – an EI of high complexity
3. A transaction of medium complexity that prints out pay-to-date details for each employee – EO
4. A file of payroll details for each employee – assessed as of medium complexity LIF
5. 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?

FP counts

Refer Table-1

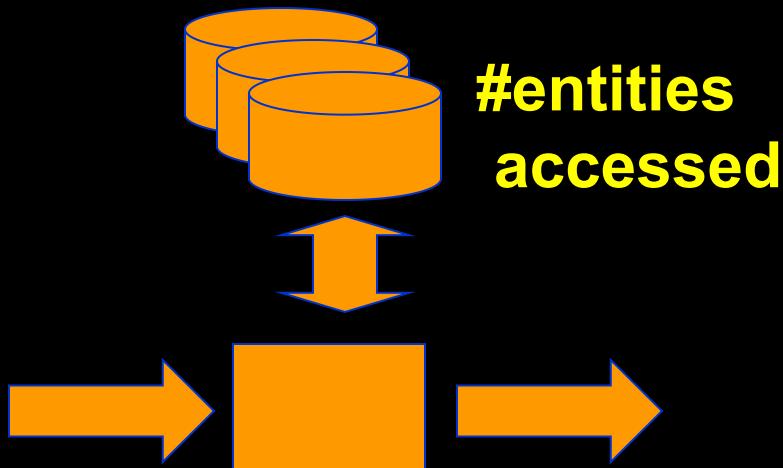
| | |
|-----------------------|--------|
| Medium EI | 4 FPs |
| High complexity EI | 6 FPs |
| Medium complexity EO | 5 FPs |
| Medium complexity LIF | 10 FPs |
| Simple EIF | 5 FPs |
| Total | 30 FPs |

If previous projects delivered 5 FPs a day, implementing the above should take $30/5 = 6$ days

Function points Mark II

- Developed by Charles R. Symons
- ‘Software sizing and estimating - Mk II FPA’ , Wiley & Sons, 1991.
- Builds on work by Albrecht
- Work originally for CCTA:
 - ↳ should be compatible with SSADM; mainly used in UK
- has developed in parallel to IFPUG FPs
- A simpler method

Function points Mk II continued



- For each transaction, count
 - data items input (N_i)
 - data items output (N_o)
 - entity types accessed (N_e)

$$\text{FP count} = N_i * 0.58 + N_e * 1.66 + N_o * 0.26$$

- UFP – Unadjusted Function Point – Albrecht
(information processing size is measured)
- TCA – Technical Complexity Adjustment
(the assumption is, an information system comprises transactions which have the basic structures, as shown in previous slide)
- For each transaction the UFPs are calculated:

$$W_i \times (\text{number of input data element types}) +$$
$$W_e \times (\text{number of entity types referenced}) +$$
$$W_o \times (\text{number of output data element types})$$

W_i, W_e, W_o are weightings derived by asking developers the proportions of effort spent in previous projects developing the code dealing with input, accessing stored data and outputs.

$W_i = 0.58, W_e = 1.66, W_o = 0.26$ (industry average)

Exercise:

A cash receipt transaction in an accounts subsystem accesses two entity types INVOICE and CASH-RECEIPT.

The data inputs are:

Invoice number

Date received

Cash received

If an INVOICE record is not found for the invoice number then an error message is issued. If the invoice number is found then a CASH-RECEIPT record is created. The error message is the only output of the transaction. Calculate the unadjusted function points, using industry average weightings, for this transaction.

$$(0.58 \times 3) + (1.66 \times 2) + (0.26 \times 1) = 5.32$$

Exercise:

In an annual maintenance contract subsystem is having a transaction which sets up details of new annual maintenance contract customers.

- | | | |
|----------------------------|------------------|----|
| 1. Customer account number | 2. Customer name | 3. |
| Address | 4. Postcode | 5. |
| Customer type | 6. 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 to the operator.

Calculate the number of unadjusted Mark II function points for the transaction described above using the industry average.

Answer:

The function types are:

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

UFP = Unadjusted function points

$$\begin{aligned} &= (0.58 \times 6) + (1.66 \times 1) + (0.26 \times 1) \\ &= 5.4 \end{aligned}$$

Function points for embedded systems

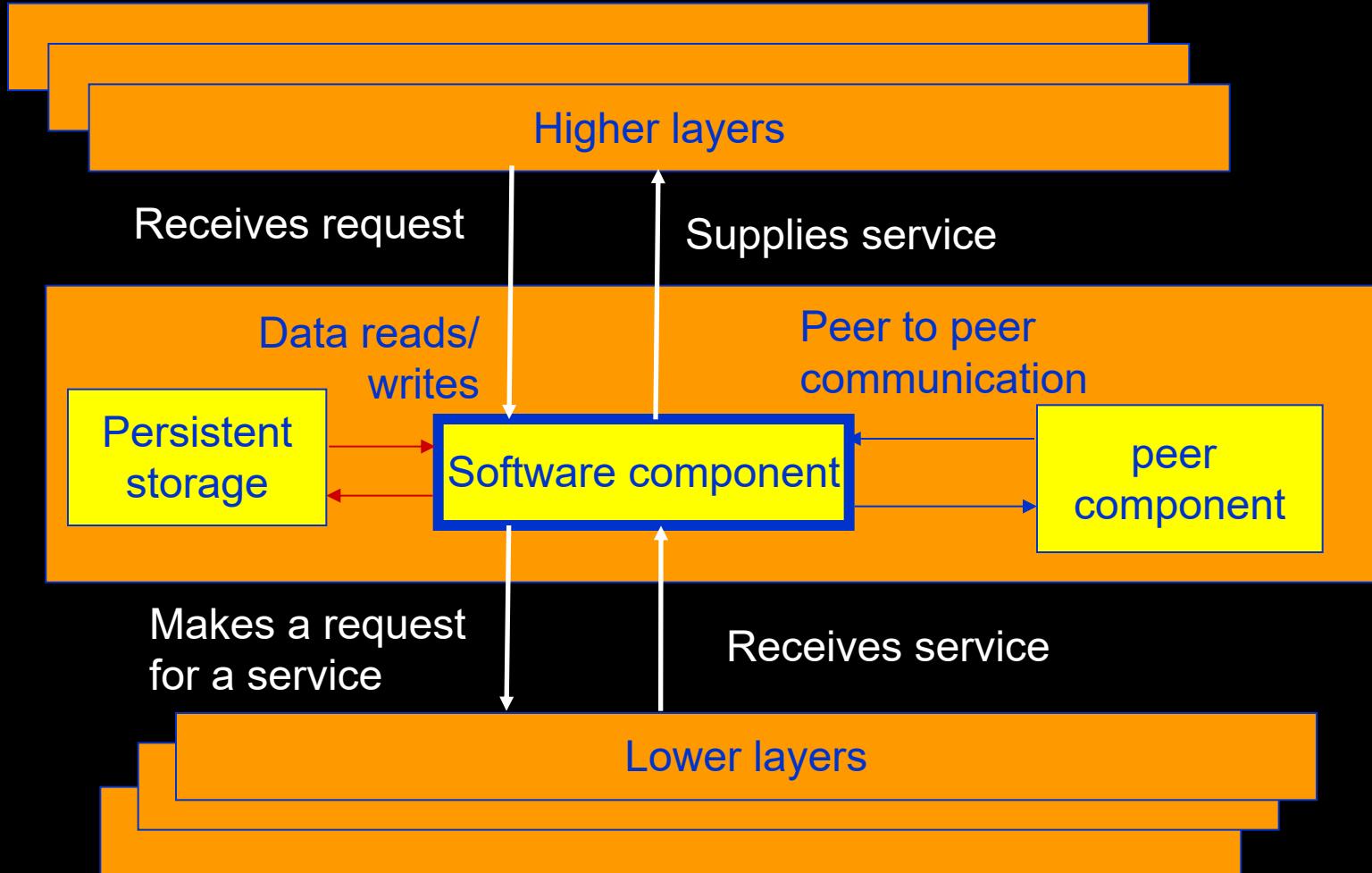
- Mark II function points, IFPUG function points were designed for information systems environments
- They are not helpful for sizing real-time or embedded systems
- COSMIC-FFPs (common software measurement consortium-full function point) attempt to extend concept to embedded systems or real-time systems
- FFP method origins the work of two interlinked groups in Quebec, Canada
- Embedded software seen as being in a particular ‘layer’ in the system
- Communicates with other layers and also other components at same level

The argument is

- existing function point method is effective in assessing the work content of an information system.
- size of the internal procedures mirrors the external features.
- in real-time or embedded system, the features are hidden because the software's user will probably not be human beings but a hardware device.

- COSMIC deals with by decomposing the system architecture into a hierarchy of software layers.
- The software component to be sized can receive requests the service from layers above and can request services from those below.
- There may be separate software components engage in peer-to-peer communication.
- Inputs and outputs are aggregated into data groups, where each data group brings together data items related to the same objects.

Layered software



COSMIC FPs

The following are counted:

(Data groups can be moved in four ways)

- Entries (E): movement of data into software component from a higher layer or a peer component
- Exits (X): movements of data out to a user outside its boundary
- Reads (R): data movement from persistent storage
- Writes (W): data movement to persistent storage

Each counts as 1 ‘COSMIC functional size unit’ (Cfsu).

The overall FFP count is derived by simply adding up the counts for each of the four types of data movement.

Exercise:

A small computer system controls the entry of vehicles to a car park. Each time a vehicle pulls up before an entry barrier, a sensor notifies the computer system of the vehicle's presence. The system examines a count that it maintains the number of vehicles currently in the car park. This count is kept on the backing storage so that it will still be available if the system is temporarily shut down, for example because of a power cut. If the count does not exceed the maximum allowed then the barrier is lifted and count is incremented. When the vehicle leaves the car park, a sensor detects the exit and reduce the count of vehicles.

Identify the entries, exits, reads and writes in this application.

| Data movement | Type |
|-------------------------------|------|
| Incoming vehicles sensed | |
| Access vehicle count | |
| Signal barrier to be lifted | |
| Increment vehicle count | |
| Outgoing vehicle sensed | |
| Decrement vehicle count | |
| New maximum input | |
| Set new maximum | |
| Adjust current vehicle count | |
| Record adjusted vehicle count | |

| Data movement | Type |
|-------------------------------|------|
| Incoming vehicles sensed | E |
| Access vehicle count | R |
| Signal barrier to be lifted | X |
| Increment vehicle count | W |
| Outgoing vehicle sensed | E |
| Decrement vehicle count | W |
| New maximum input | E |
| Set new maximum | W |
| Adjust current vehicle count | E |
| Record adjusted vehicle count | W |

Note: different interpretations of the requirements could lead to different counts. The description in the exercise does not specify to give a message that the car park is full or has spaces.

COCOMO81

- Based on industry productivity standards - database is constantly updated
- Allows an organization to benchmark its software development productivity
- **Basic model**
$$\text{effort} = c \times \text{size}^k$$
- C and k depend on the type of system: organic, semi-detached, embedded
- Size is measured in ‘kloc’ ie. Thousands of lines of code

Boehm in 1970, on a study of 63 projects, made this model. Of these only seven were business systems and so the model was based on other applications (non-information systems).

The COCOMO constants

| System type | c | k |
|---|-----|------|
| Organic (broadly, information systems, small team, highly familiar in-house environment) | 2.4 | 1.05 |
| Semi-detached (combined characteristics between organic and embedded modes) | 3.0 | 1.12 |
| Embedded (broadly, real-time, products developed has to operate within very tight constraints and changes to system is very costly) | 3.6 | 1.20 |

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

$$\text{effort} = c (\text{size})^k$$

effort – pm (person month) – 152 working hours

size – KLOC – kdsi (thousands of delivered source code instruction)

c and k – constants depending on whether the system is organic, semi-detached or embedded.

Organic : Effort = 2.4(KLOC)^{1.05} PM

Semidetached : Effort = 3.0(KLOC)^{1.12} PM

Embedded : Effort = 3.6(KLOC)^{1.20} PM

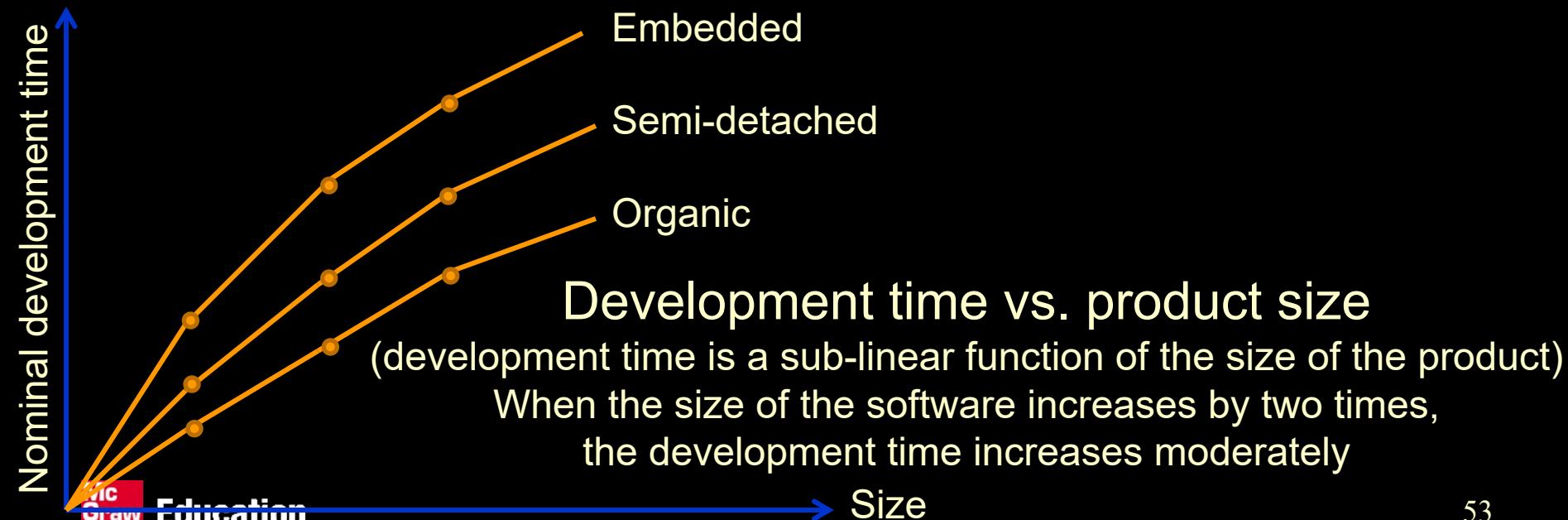
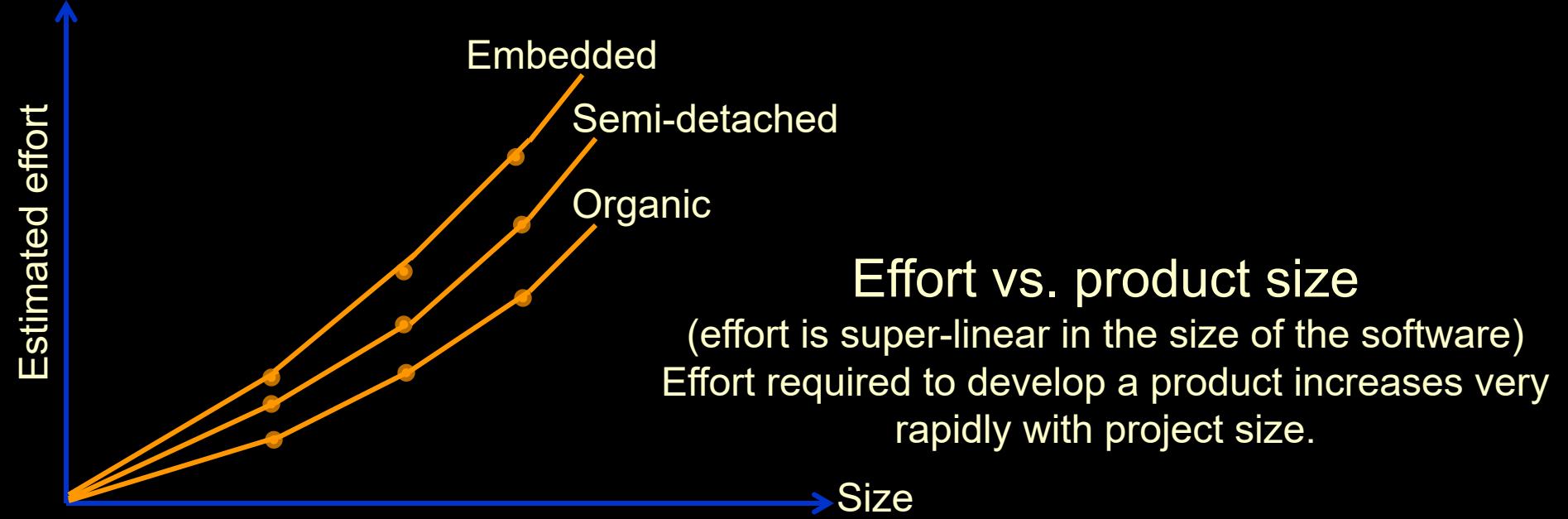
Estimation of development time

$$T_{dev} = a \times (\text{Effort})^b$$

Organic : $2.5(\text{Effort})^{0.38}$

Semidetached : $2.5(\text{Effort})^{0.35}$

Embedded : $2.5(\text{Effort})^{0.32}$



Exercise:

Assume that the size of an organic type software product is estimated to be 32,000 lines of source code. Assume that the average salary of a software developer is Rs.50,000 per month. Determine the effort required to develop the software product, the nominal development time, and the staff cost to develop the product.

$$\text{Effort} = 2.4 \times 32^{1.05} = 91 \text{ pm}$$

$$\text{Nominal development time} = 2.5 \times 91^{0.38} = 14 \text{ months}$$

Staff cost required to develop the product

$$91 \times \text{Rs. } 50,000 = \text{Rs. } 45,50,000$$

Ex-Two software managers separately estimated a given product to be of 10,000 and 15,000 lines of code respectively. Bring out the effort and schedule time implications of their estimation using COCOMO. For the effort estimation, use a coefficient value of 3.2 and exponent value of 1.05. For the schedule time estimation, the similar values are 2.5 and 0.38 respectively. Assume all adjustment multipliers to be equal to unity.

For 10,000 LOC

$$\text{Effort} = 3.2 \times 10^{1.05} = 35.90 \text{ PM}$$

$$\text{Schedule Time} = T_{\text{dev}} = 2.5 \times 35.90^{0.38} = 9.75 \text{ months}$$

For 15,000 LOC

$$\text{Effort} = 3.2 \times 15^{1.05} = 54.96 \text{ PM}$$

$$\text{Schedule Time} = T_{\text{dev}} = 2.5 \times 54.96^{0.38} = 11.46 \text{ months}$$

NB: Increase in size drastic increase in effort but moderate change in time.

COCOMO II

An updated version of COCOMO:

- There are different COCOMO II models for estimating at the ‘early design’ stage and the ‘post architecture’ stage when the final system is implemented. We’ll look specifically at the first.
- The core model is:

$$pm = A(\text{size})^{(sf)} \times (em_1) \times (em_2) \times (em_3) \dots$$

where **pm** = person months, **A** is 2.94, **size** is number of thousands of lines of code, **sf** is the scale factor, and **em** is an effort multiplier

$$sf = B + 0.01 \times \sum (\text{exponent driver ratings})$$

COCOMO II Scale factor

Boehm et al. have refined a family of cost estimation models. The key one is COCOMO II. It uses multipliers and exponent values. Based on five factors which appear to be particularly sensitive to system size.

Precedentedness (**PREC**). Degree to which there are past examples that can be consulted, else uncertainty

Development flexibility (**FLEX**). Degree of flexibility that exists when implementing the project

Architecture/risk resolution (**RESL**). Degree of uncertainty about requirements, liable to change

Team cohesion (**TEAM**). Large dispersed

Process maturity (**PMAT**) could be assessed by CMMI, more structured less uncertainty

– see Section 13.8

COCOMO II Scale factor values

| Driver | Very low | Low | Nominal | High | Very high | Extra high |
|--------|----------|------|---------|------|-----------|------------|
| PREC | 6.20 | 4.96 | 3.72 | 2.48 | 1.24 | 0.00 |
| FLEX | 5.07 | 4.05 | 3.04 | 2.03 | 1.01 | 0.00 |
| RESL | 7.07 | 5.65 | 4.24 | 2.83 | 1.41 | 0.00 |
| TEAM | 5.48 | 4.38 | 3.29 | 2.19 | 1.10 | 0.00 |
| PMAT | 7.80 | 6.24 | 4.68 | 3.12 | 1.56 | 0.00 |

Example of scale factor

- A software development team is developing an application which is very similar to previous ones it has developed.
- A very precise software engineering document lays down very strict requirements. PREC is very high (score 1.24).
- FLEX is very low (score 5.07).
- The good news is that these tight requirements are unlikely to change (RESL is high with a score 2.83).
- The team is tightly knit (TEAM has high score of 2.19), but processes are informal (so PMAT is low and scores 6.24)

Scale factor calculation

The formula for sf is

$$sf = B + 0.01 \times \Sigma \text{ scale factor values}$$

$$\begin{aligned} \text{i.e. } sf &= 0.91 + 0.01 \times (1.24 + 5.07 + 2.83 + 2.19 + \\ &6.24) \\ &= 1.0857 \end{aligned}$$

If system contained 10 kloc then estimate would be *effort*
 $= c (\text{size})^k = 2.94 \times 10^{1.0857} = 35.8 \text{ person-months}$

Using exponentiation ('to the power of') adds disproportionately more to the estimates for larger applications

$$B = 0.91(\text{constant}), \quad c = 2.94 \text{ (average)}$$

Exercise:

A new project has ‘average’ novelty for the software supplier that is going to execute it and thus given a nominal rating on this account for precedentedness. Development flexibility is high, requirements may change radically and so risk resolution exponent is rated very low. The development team are all located in the same office and this leads to team cohesion being rated as very high, but the software house as a whole tends to be very informal in its standards and procedures and the process maturity driver has therefore been given a rating of ‘low’ .

- (i) What would be the scale factor (sf) in this case?
- (ii) What would the estimate effort if the size of the application was estimated as in the region of 2000 lines of code?

Assessing the scale factors

| Factor | Rating | Value |
|--------|-----------|-------|
| PREC | nominal | 3.72 |
| FLEX | high | 2.03 |
| RESL | very low | 7.07 |
| TEAM | very high | 1.10 |
| PMAT | low | 6.24 |

- (i) The overall scale factor = $sf = B + 0.01 \times \sum (\text{exponent factors})$
= $0.91 + 0.01 \times (3.72 + 2.03 + 7.07 + 1.10 + 6.24)$
= $0.91 + 0.01 \times 20.16 = 1.112$
- (ii) The estimated effort = $c (\text{size})^k = 2.94 \times 2^{1.112} = 6.35 \text{ staff-months}$

Effort multipliers

(COCOMO II - early design)

As well as the scale factor effort multipliers are also assessed:

| | |
|------|------------------------------------|
| RCPX | Product reliability and complexity |
| RUSE | Reuse required |
| PDIF | Platform difficulty |
| PERS | Personnel capability |
| PREX | Personnel experience |
| FCIL | Facilities available |
| SCED | Schedule pressure |

Effort multipliers

(COCOMO II - early design)

Table-3

| | Extra low | Very low | Low | Nominal | High | Very high | Extra high |
|------|-----------|----------|------|---------|------|-----------|------------|
| RCPX | 0.49 | 0.60 | 0.83 | 1.00 | 1.33 | 1.91 | 2.72 |
| RUSE | | | 0.95 | 1.00 | 1.07 | 1.15 | 1.24 |
| PDIF | | | 0.87 | 1.00 | 1.29 | 1.81 | 2.61 |
| PERS | 2.12 | 1.62 | 1.26 | 1.00 | 0.83 | 0.63 | 0.50 |
| PREX | 1.59 | 1.33 | 1.12 | 1.00 | 0.87 | 0.74 | 0.62 |
| FCIL | 1.43 | 1.30 | 1.10 | 1.00 | 0.87 | 0.73 | 0.62 |
| SCED | | 1.43 | 1.14 | 1.00 | 1.00 | 1.00 | |

Example

- Say that a new project is similar in most characteristics to those that an organization has been dealing for some time
- **except**
 - ↳ the software to be produced is exceptionally complex and will be used in a safety critical system.
 - ↳ The software will interface with a new operating system that is currently in beta status.
 - ↳ To deal with this the team allocated to the job are regarded as exceptionally good, but do not have a lot of experience on this type of software.

Example -continued

Refer Table-3

| | | |
|------|------------|------|
| RCPX | very high | 1.91 |
| PDIF | very high | 1.81 |
| PERS | extra high | 0.50 |
| PREX | nominal | 1.00 |

All other factors are nominal

Say estimate is 35.8 person months

With effort multipliers this becomes $35.8 \times 1.91 \times 1.81 \times 0.5 = 61.9$ person months

Exercise:

A software supplier has to produce an application that controls a piece of equipment in a factory. A high degree of reliability is needed as a malfunction could injure the operators. The algorithms to control the equipment are also complex. The product reliability and complexity are therefore rated as very high. The company would like to take opportunity to exploit fully the investment that they made in the project by reusing the control system, with suitable modifications, on future contracts. The reusability requirement is therefore rated as very high. Developers are familiar with the platform and the possibility of potential problems in that respect is regarded as low. The current staff are generally very capable and are rated as very high, but the project is in a somewhat novel application domain for them so experience is rated as nominal. The toolsets available to the developers are judged to be typical for the size of company and are rated nominal, as is the degree of pressure to meet a deadline.

Given the data table-3

- (i) What would be the value for each of the effort multipliers?
- (ii) What would be the impact of all the effort multipliers on a project estimated as taking 200 staff members?

| Factor | Description | Rating | Effort multiplier |
|---------------|------------------------------------|---------------|--------------------------|
| RCPX | Product reliability and complexity | | |
| RUSE | Reuse | | |
| PDIF | Platform difficulty | | |
| PERS | Personnel capability | | |
| PREX | Personnel experience | | |
| FCIL | Facilities available | | |
| SCED | Required development schedule | | |

| Factor | Description | Rating | Effort multiplier |
|---------------|------------------------------------|---------------|--------------------------|
| RCPX | Product reliability and complexity | Very high | |
| RUSE | Reuse | Very high | |
| PDIF | Platform difficulty | Low | |
| PERS | Personnel capability | Very high | |
| PREX | Personnel experience | Nominal | |
| FCIL | Facilities available | Nominal | |
| SCED | Required development schedule | nominal | |

| Factor | Description | Rating | Effort multiplier |
|---------------|------------------------------------|---------------|--------------------------|
| RCPX | Product reliability and complexity | Very high | 1.91 |
| RUSE | Reuse | Very high | 1.15 |
| PDIF | Platform difficulty | Low | 0.87 |
| PERS | Personnel capability | Very high | 0.63 |
| PREX | Personnel experience | Nominal | 1.00 |
| FCIL | Facilities available | Nominal | 1.00 |
| SCED | Required development schedule | nominal | 1.00 |

New development effort multipliers (dem)

According to COCOMO, the major productivity drivers include:

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

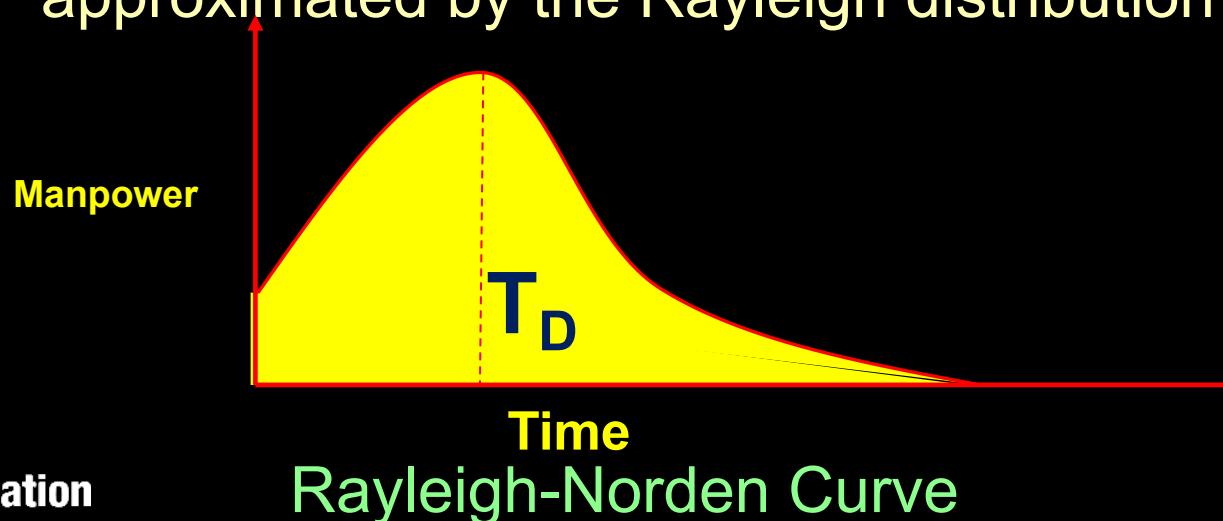
Project attributes: modern programming practices, software tools, schedule constraints

COCOMO II Post architecture effort multipliers

| Modifier type | Code | Effort multiplier |
|----------------------|-------|---|
| Product attributes | RELY | Required software reliability |
| | DATA | Database size |
| | DOCU | Documentation match to life-cycle needs |
| | CPLX | Product complexity |
| | REUSE | Required reusability |
| Platform attributes | TIME | Execution time constraint |
| | STOR | Main storage constraint |
| | PVOL | Platform volatility |
| Personnel attributes | ACAP | Analyst capability |
| | AEXP | Application experience |
| | PCAP | Programmer capabilities |
| | PEXP | Platform experience |
| | LEXP | Programming language experience |
| Project attributes | PCON | Personnel continuity |
| | TOOL | Use of software tools |
| | SITE | Multisite development |
| | SCED | Schedule pressure |

Staffing

- Norden was one of the first to investigate staffing pattern:
 - ◆ Considered general research and development (R&D) type of projects for efficient utilization of manpower.
- Norden concluded:
 - ◆ Staffing pattern for any R&D project can be approximated by the Rayleigh distribution curve



Putnam's Work

- Putnam adapted the Rayleigh-Norden curve:
 - ◆ Related the number of delivered lines of code to the effort and the time required to develop the product.
 - ◆ Studied the **effect of schedule compression**:

$$pm_{new} = pm_{org} \times \left(\frac{td_{org}}{td_{new}} \right)^4$$

Example

- If the estimated development time using COCOMO formulas is 1 year:
 - ◆ Then to develop the product in 6 months, the total effort required (and hence the project cost) increases by

16 times.

Why?

- The extra effort can be attributed to the increased communication requirements and the free time of the developers waiting for work.
- The project manager recruits a large number of developers hoping to complete the project early, but becomes very difficult to keep these additional developers continuously occupied with work.
- Implicit in the schedule and duration estimated arrived at using COCOMO model, is the fact that all developers can continuously be assigned work.
- However, when a large number of developers are hired to decrease the duration significantly, it becomes difficult to keep all developers busy all the time. The simultaneous work is getting restricted.

Exercise:

The nominal effort and duration of a project is estimated to be 1000 pm and 15 months. This project is negotiated to be £200,000. This needs the product to be developed and delivered in 12 months time. What is the new cost that needs to be negotiated.

The project can be classified as a large project. Therefore the new cost to be negotiated can be given by Putnam's formula as

$$\text{New Cost} = \text{£200,000} \times (15/12)^4 = \text{£488,281}$$

Boehm's Result

- There is a limit beyond which a software project cannot reduce its schedule by buying any more personnel or equipment.
 - ◆ This limit occurs roughly at 75% of the nominal time estimate for small and medium sized projects
 - ◆ If a project manager accepts a customer demand to compress the development schedule of a project (small or medium) by more than 25% , he is very unlikely to succeed.
 - ◆ The reason is, every project has a limited amount of activities which can be carried out in parallel and the sequential work can not be speeded up by hiring more number of additional developers.

Capers Jones' Estimating Rules of Thumb

- Empirical rules: (IEEE journal – 1996)
 - ◆ Formulated based on observations
 - ◆ No scientific basis
- Because of their simplicity:
 - ◆ These rules are handy to use for making off-hand estimates. Not expected to yield very accurate estimates.
 - ◆ Give an insight into many aspects of a project for which no formal methodologies exist yet.

Capers Jones' Rules

- *Rule 1: SLOC-function point equivalence:*
 - ◆ One function point = 125 SLOC for C programs.
- **Rule 2: Project duration estimation:**
 - ◆ Function points raised to the power 0.4 predicts the approximate development time in calendar months.
- **Rule 3: Rate of requirements creep:**
 - ◆ User requirements creep in at an average rate of 2% per month from the design through coding phases.

Illustration:

Size of a project is estimated to be 150 function points.

Rule-1: $150 \times 125 = 18,750$ SLOC

Rule-2: Development time = $150^{0.4} = 7.42 \approx 8$ months

Rule-3: The original requirement will grow by 2% per month i.e 2% of 150 is 3 FPs per month.

- If the duration of requirements specification and testing is 5 months out of total development time of 8 months, the total requirements creep will be roughly $3 \times 5 = 15$ function points.
- The total size of the project considering the creep will be $150 + 15 = 165$ function points and the manager need to plan on 165 function points.

Capers Jones' Rules

- Rule 4: Defect removal efficiency:
 - ◆ Each software review, inspection, or test step will find and remove 30% of the bugs that are present.
(Companies use a series of defect removal steps like requirement review, code inspection, code walk-through followed by unit, integration and system testing. A series of ten consecutive defect removal operations must be utilized to achieve good product reliability.)
- Rule 5: Project manpower estimation:
 - ◆ The size of the software (in function points) divided by 150 predicts the approximate number of personnel required for developing the application. (For a project size of 500 FPs the number of development personnel will be $500/125 = 4$, without considering other aspects like use of CASE tools, project complexity and programming languages.)

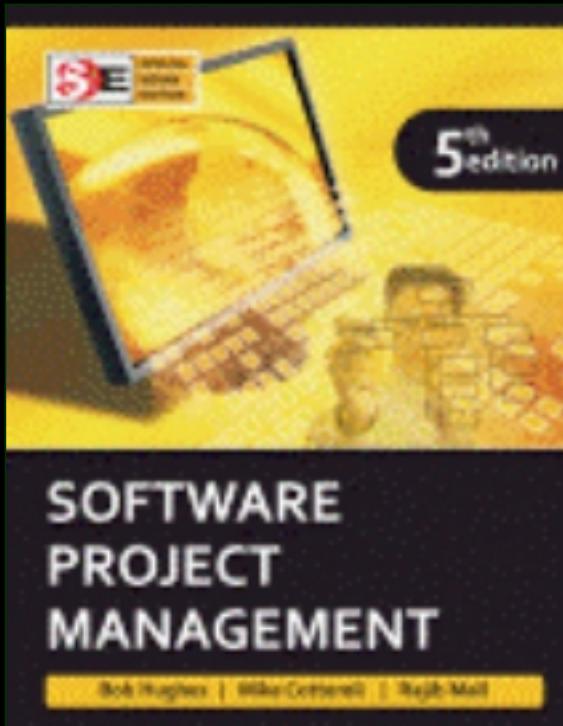
Capers' Jones Rules

- Rule 6: Software development effort estimation:
 - ◆ The approximate number of staff months of effort required to develop a software is given by the software development time multiplied with the number of personnel required. (using rule 2 and 5 the effort estimation for the project size of 150 FPs is $8 \times 1 = 8$ person-months)
- Rule 7: Number of personnel for maintenance
 - ◆ *Function points divided by 500 predicts the approximate number of personnel required for regular maintenance activities.* (as per Rule-1, 500 function points is equivalent to about 62,500 SLOC of C program, the maintenance personnel would be required to carry out minor fixing, functionality adaptation **ONE.**)

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?

Software Project Management



- Chapter Six
- Activity planning

Scheduling

‘Time is nature’s way of stopping everything happening at once’

Having

- ↳ worked out a method of doing the project
- ↳ identified the tasks to be carried
- ↳ assessed the time needed to do each task

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

Activity networks

These help us to:

- Ensure that the appropriate resources will be available precisely when required.
- Avoid different activities competing for the same resources at the same time.
- Calculate when costs will be incurred.
- Produce a timed cash flow forecast.
- Produce a detailed schedule showing which staff carry out each activity.
- Produce a detailed plan against which actual achievement may be measured.
- Plan the project during its life to correct drift from the target.

Also

- Activity plan will provide a target start and completion date for each activity.
- The start and completion of activities must be clearly visible and ensure each activity to produce some tangible product or ‘deliverable’ .
- The project will progress according to the plan.
- In case of deviation, identify the cause and plan to mitigate its effects.
- Activity plan provides a means of evaluating the consequences of not meeting the activity target dates and guide to bring the project back to target.

Defining activities

Activity networks are based on some assumptions:

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

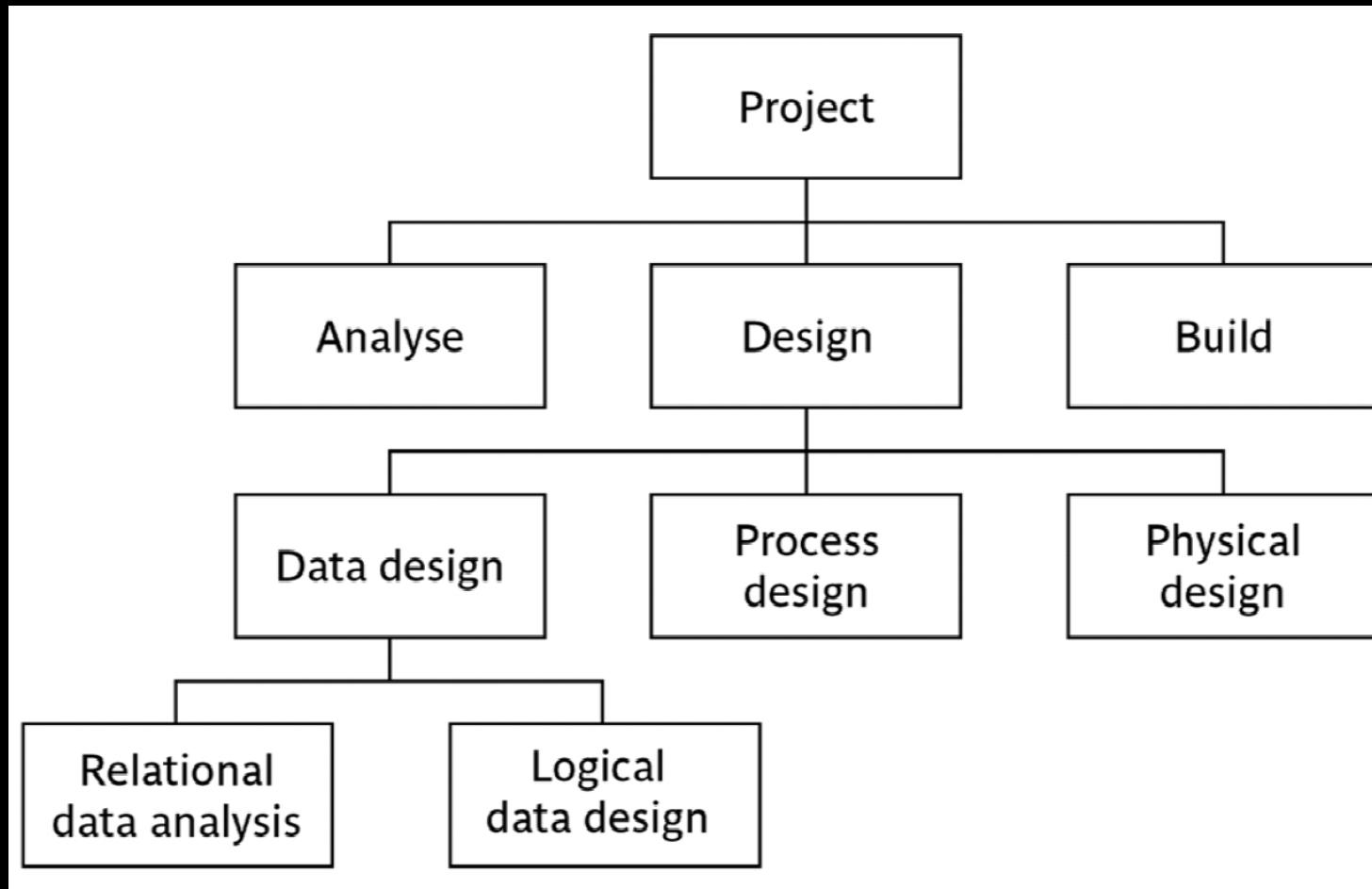
Defining activities -continued

- An activity
 - ◆ Must have clearly defined start and end-points
 - ◆ Must have resource requirements that can be forecast: these are assumed to be constant throughout the project
 - ◆ Must have a duration that can be forecast
 - ◆ May be dependent on other activities being completed first (precedence networks)

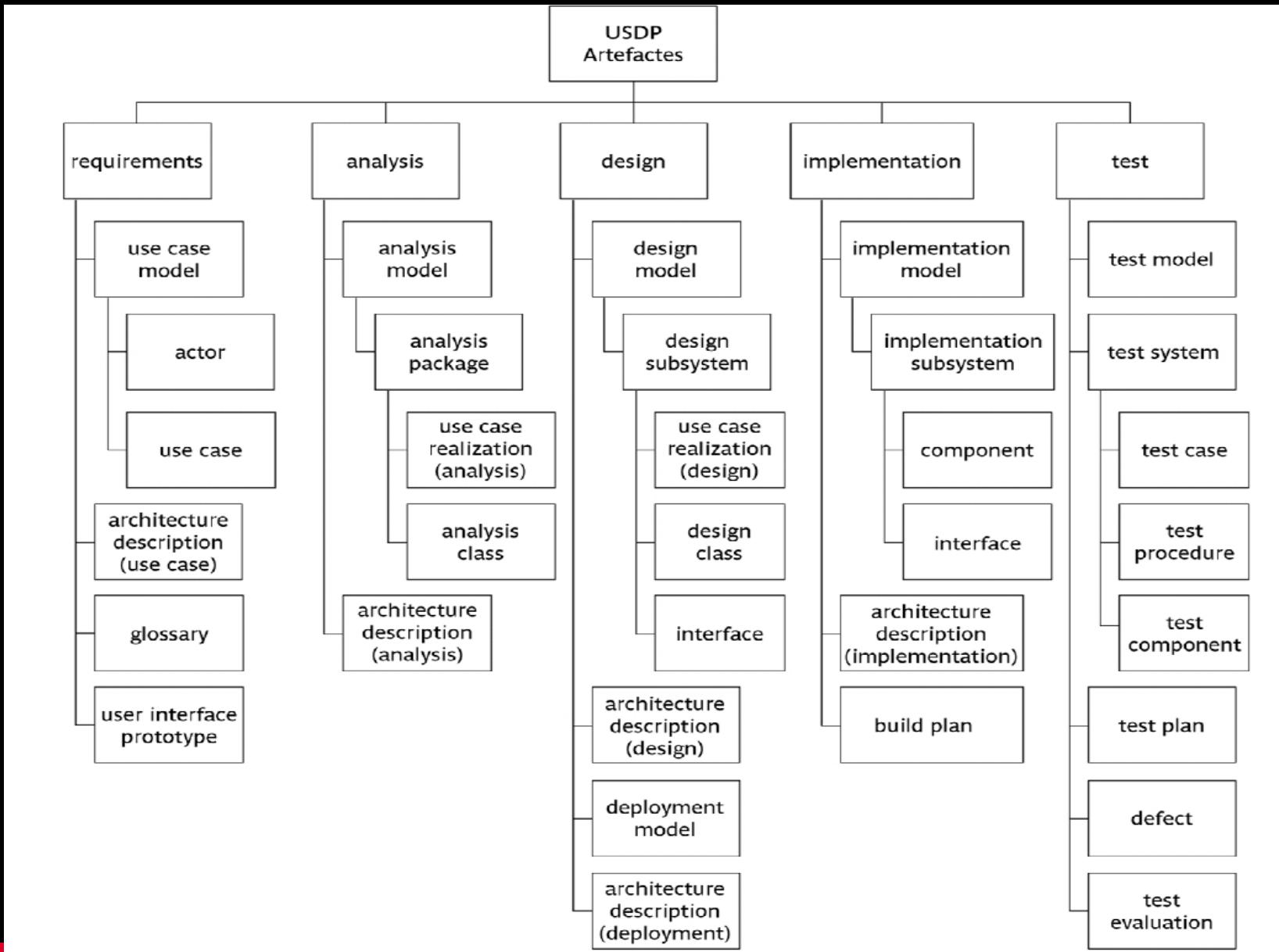
Identifying activities

- Work-based approach:
 - ✿ 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

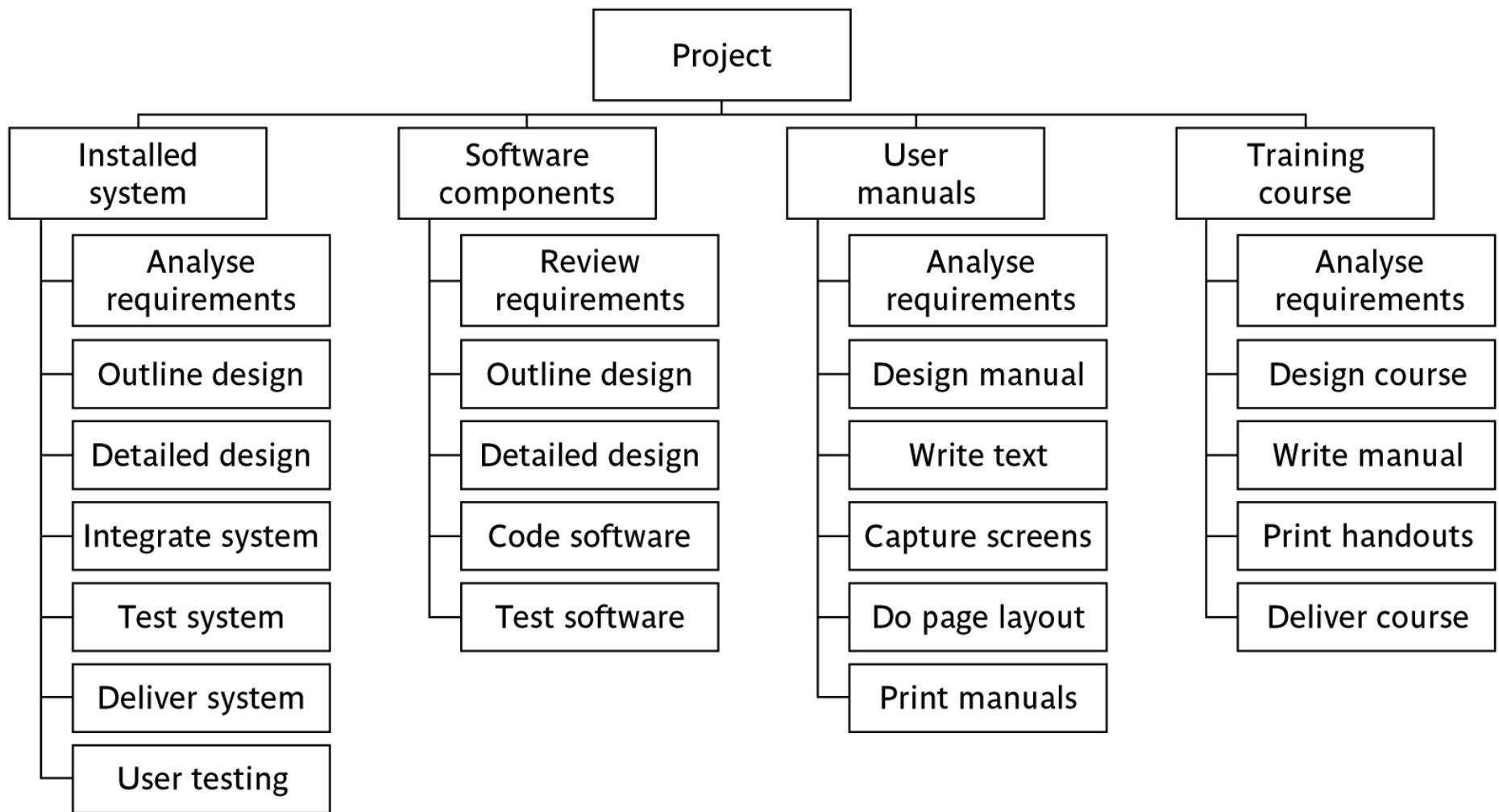
Activity based work-breakdown structure



USDP product breakdown structure

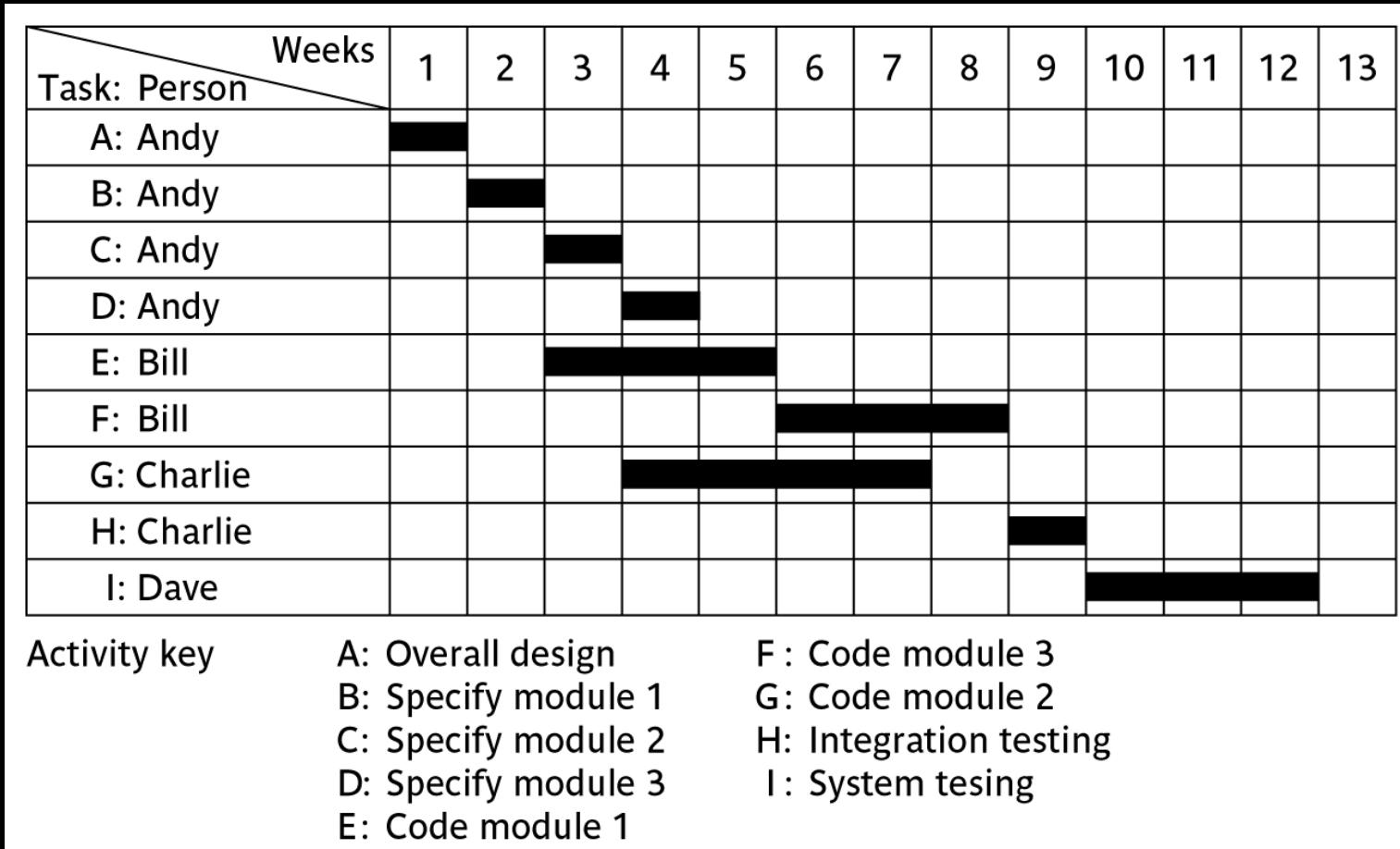


Hybrid approach



The final outcome of the planning process

A project plan as a bar chart



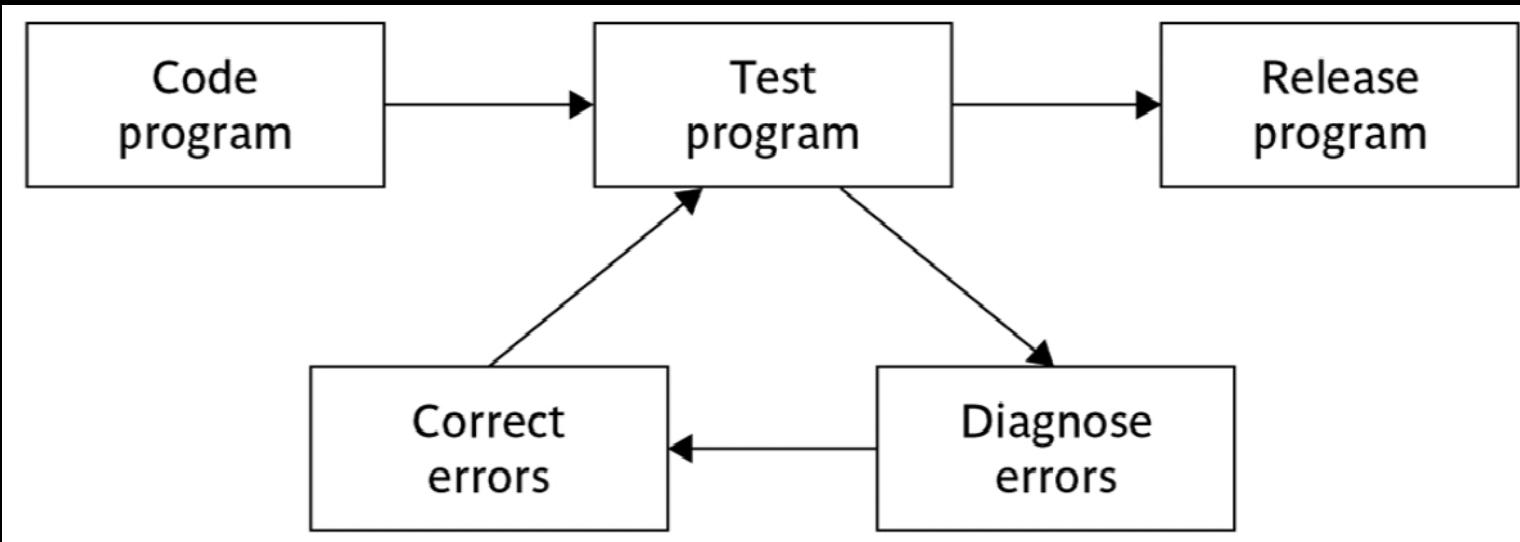
Network Planning Model

- Project scheduling techniques model the project activities and their relationships as a network
- In the network time flows from left to right.
- Developed in 1950s.
- Uses activity-on-arrow to visualize network.
- Circles and arrows are used.
- Technique is called precedence network.
- PERT (program evaluation review technique)
- CPM (critical path analysis)

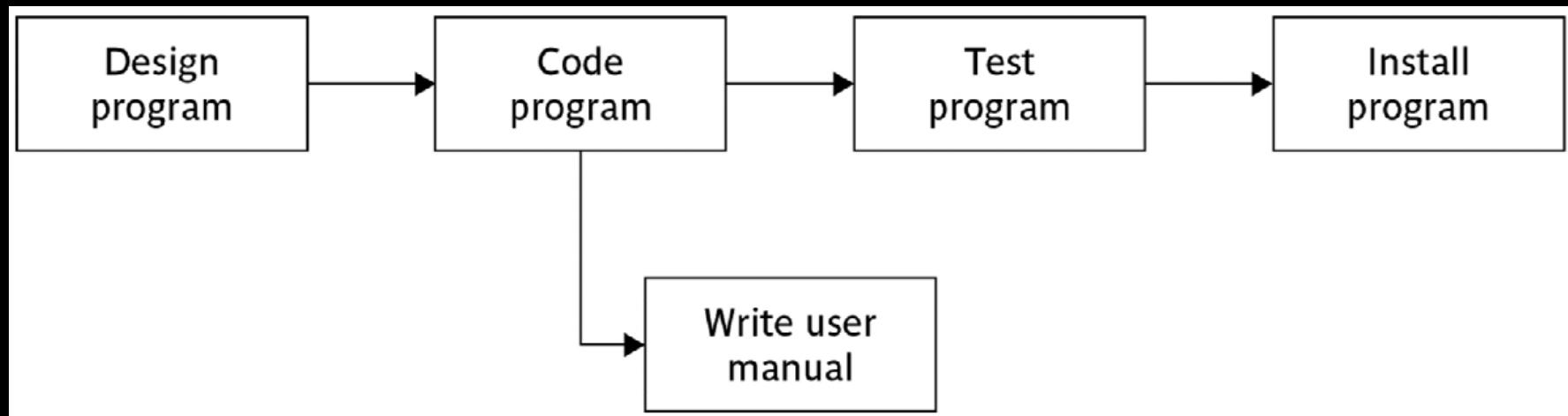
Constructing Precedence Network

- A project network should have only one **start** node.
- A project network should have only one **end** node.
- A node has a **duration**.
- Links normally have **no duration**.
- **Precedents** are the immediate preceding activities.
- **Time** moves from left to right.
- A network may not contain **loop**.
- A network should not contain **dangles**.

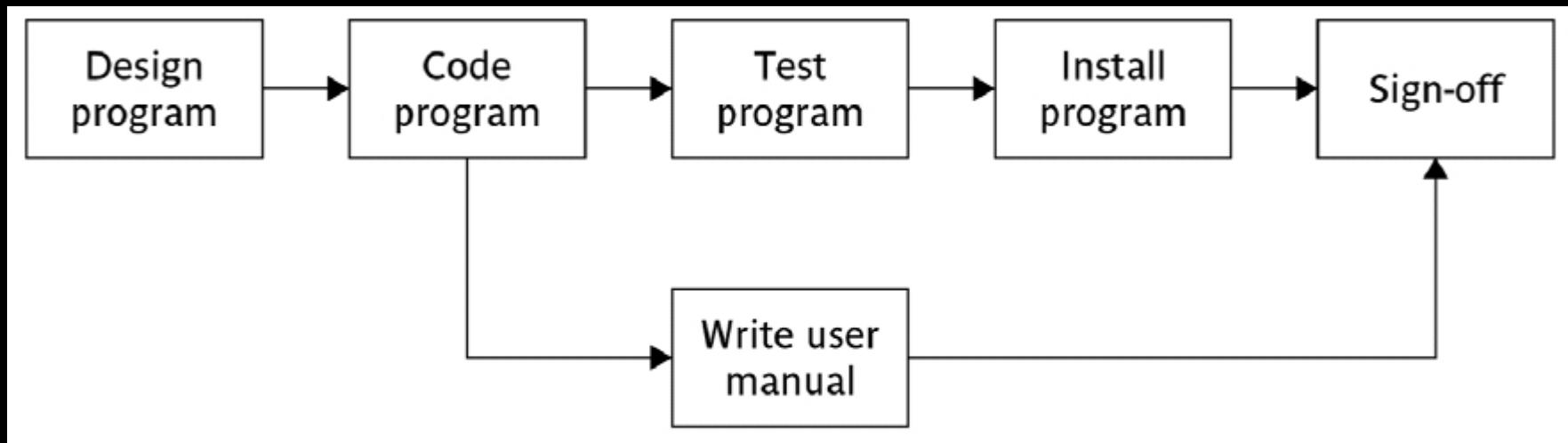
A loop represents an impossible sequence



A dangle

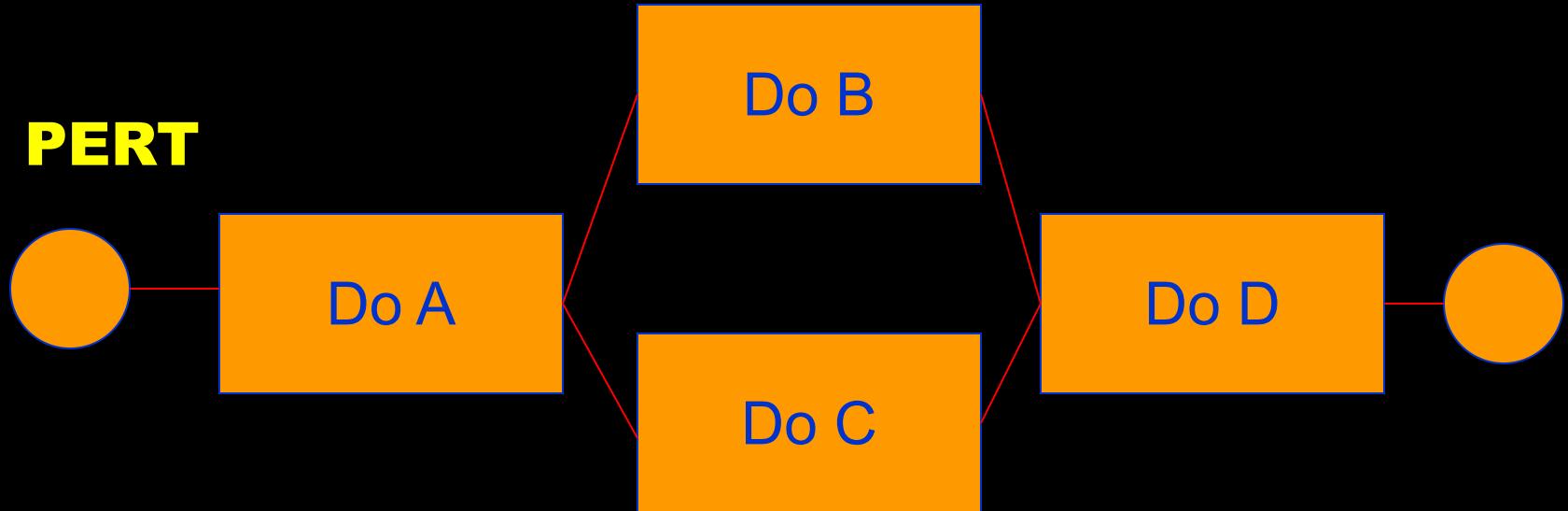


Resolving a dangle

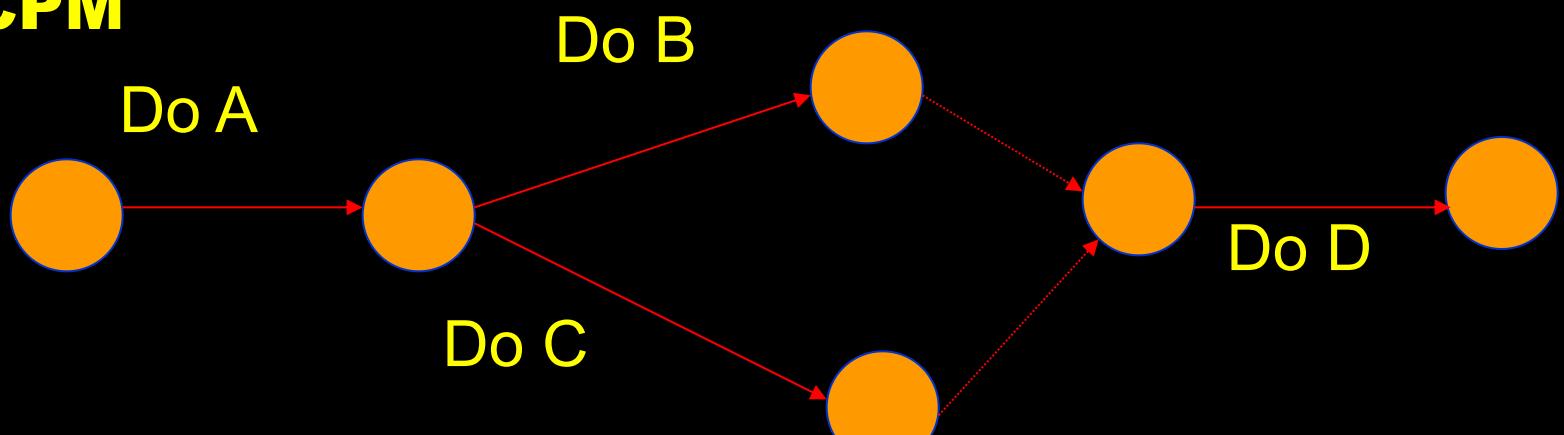


PERT vs CPM

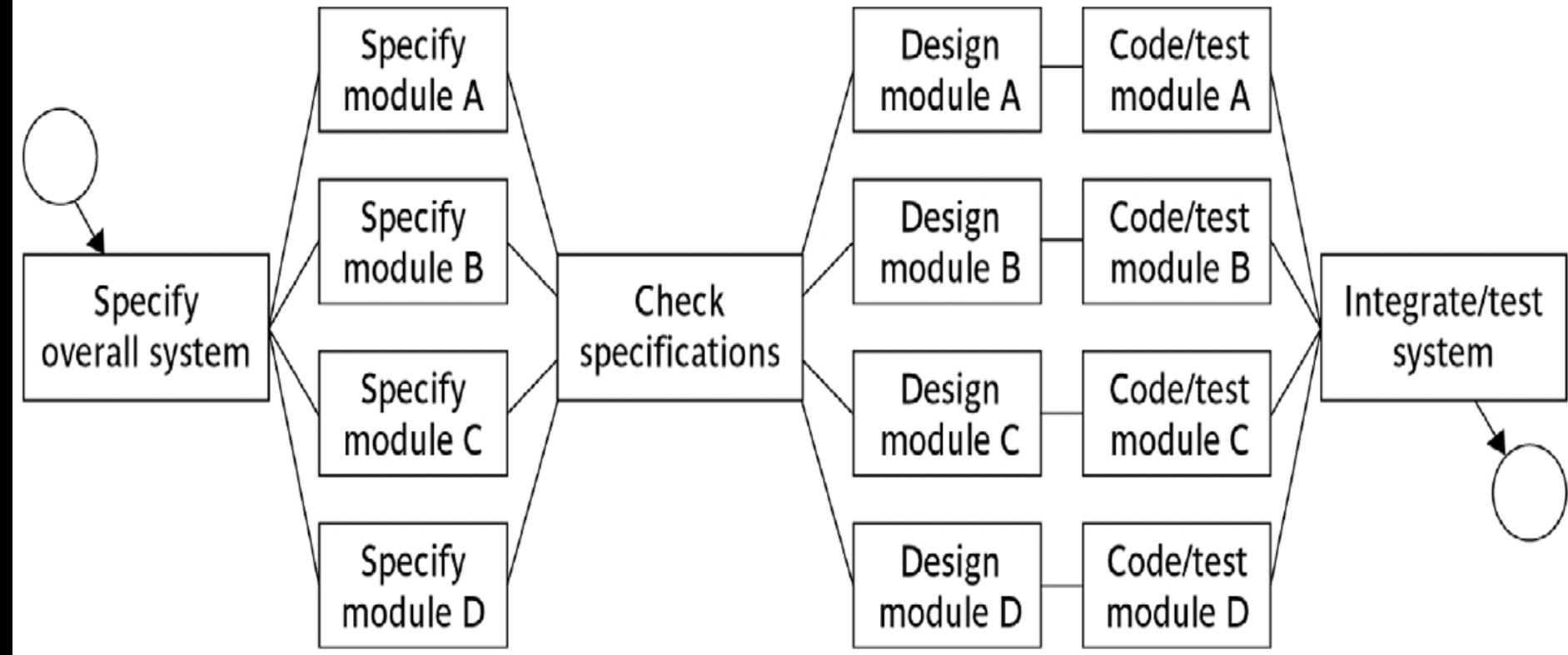
PERT



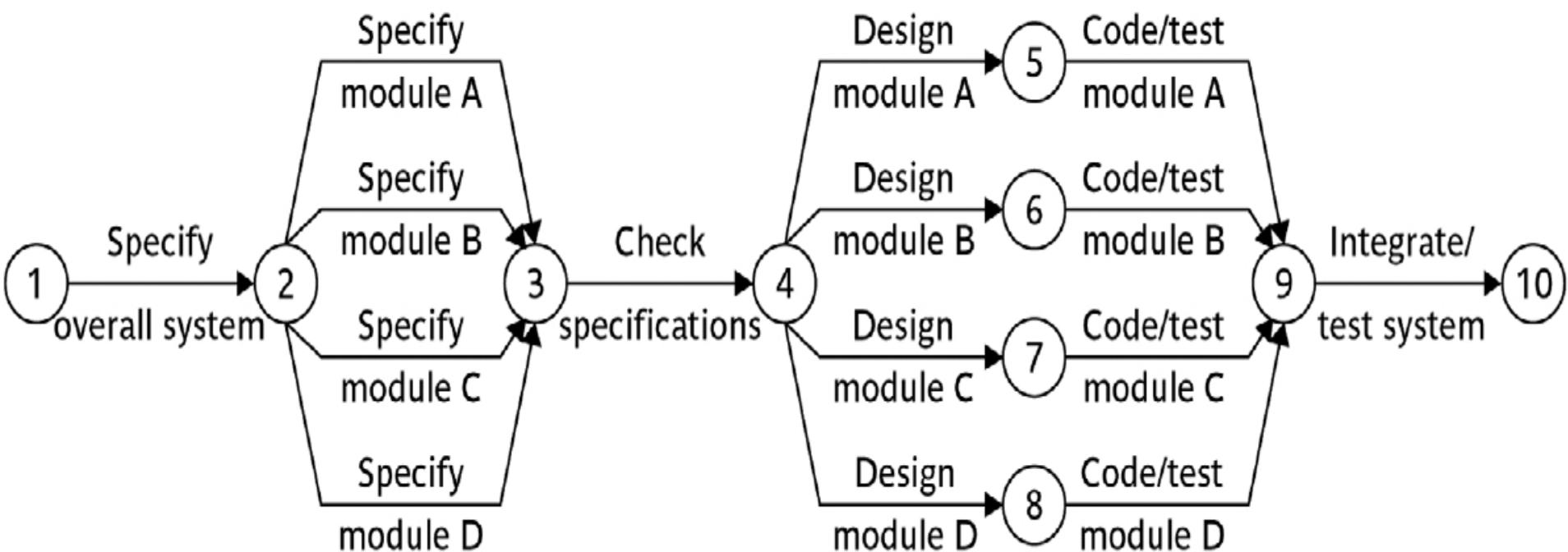
CPM



Annual maintenance contract project – Event based network

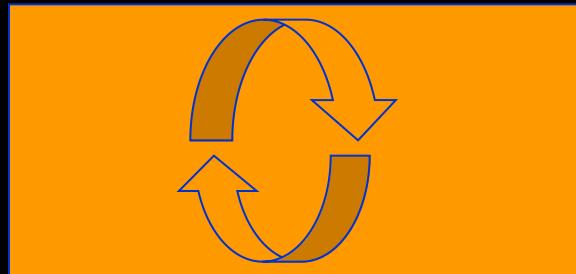


Annual maintenance contract project activity network - CPM



Drawing up a PERT diagram

- No looping back is allowed – deal with iterations by hiding them within single activities



- *milestones* – ‘activities’, such as the start and end of the project, which indicate transition points. They have zero duration.

Lagged activities

- where there is a fixed delay between activities e.g. seven days notice has to be given to users that a new release has been signed off and is to be installed

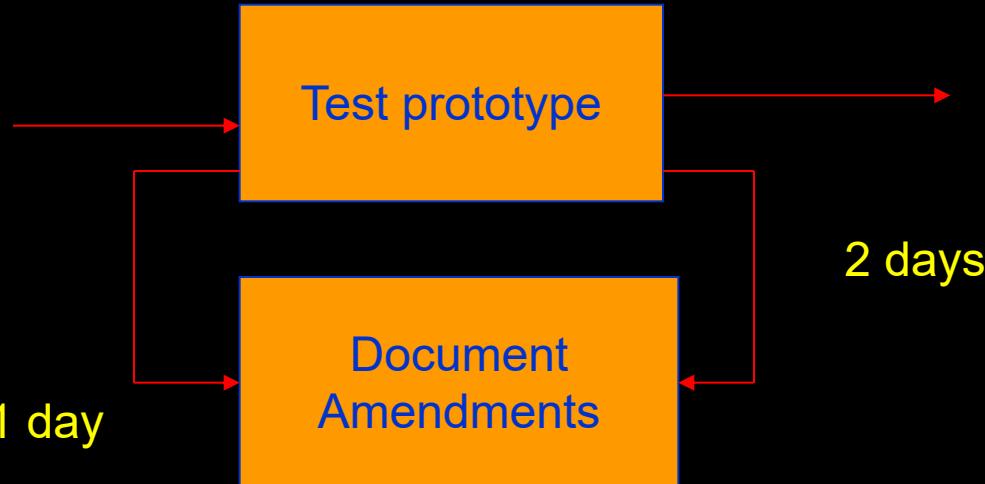


Types of links between activities

- Finish to start

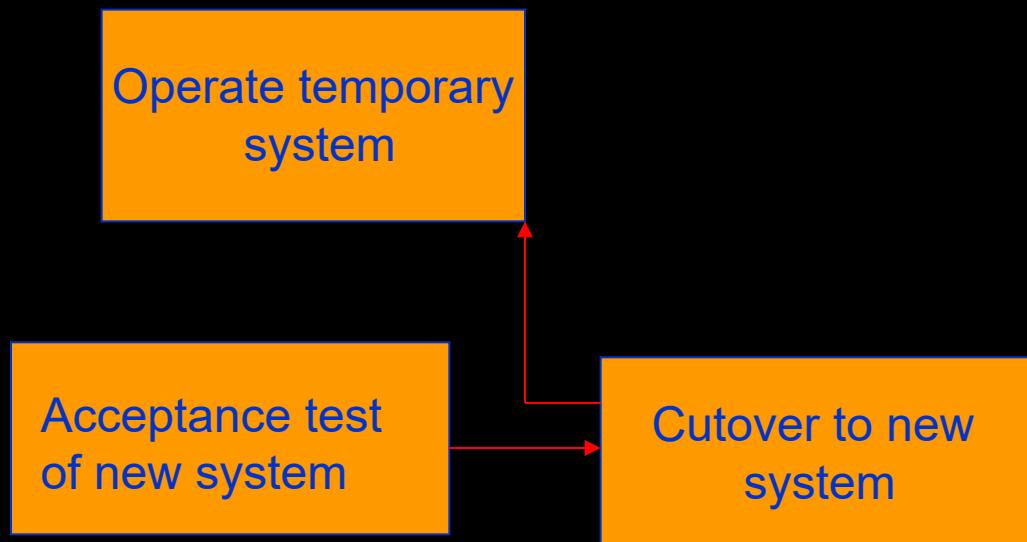


- Start to start/ Finish to finish

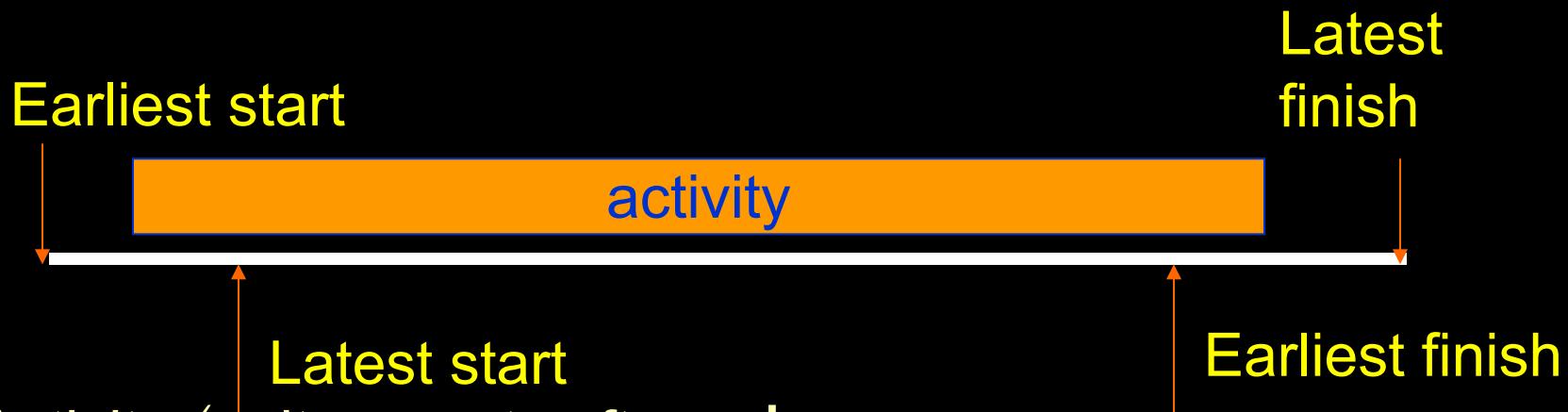


Types of links between activities

- Start to finish



Start and finish times



- Activity ‘write report software’
- Earliest start (ES)
- Earliest finish (EF) = ES + duration
- Latest start(LS) = latest task can be completed without affecting project end Latest start = LF - duration

Example

- earliest start = day 5
- latest finish = day 30
- duration = 10 days
- earliest finish = ?
- latest start = ?

Float = ES – EF – duration

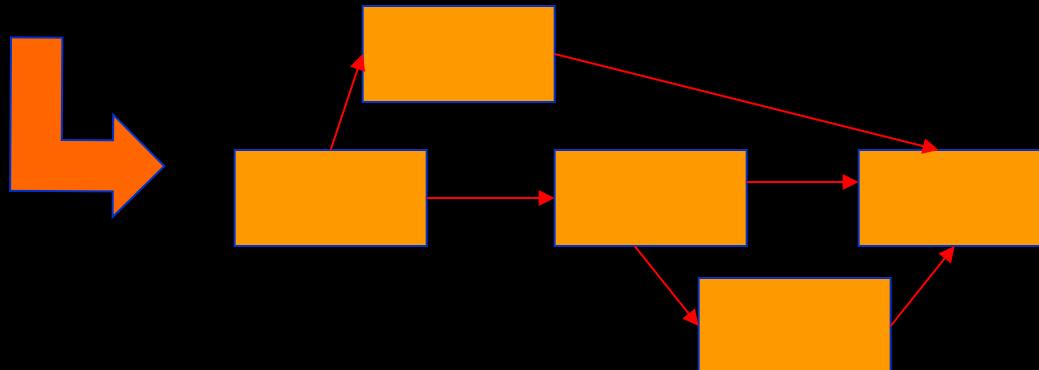
What is it in this case?

‘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 begin at Day 0 i.e. the end of Day 0 i.e. the start of Day 1

T_E t T_E+t

| Earliest start (ES) | Duration | Earliest finish (EF) |
|--------------------------------------|----------|----------------------|
| Activity label, activity description | | |
| Latest Start (LS) | Float | Latest finish (LF) |

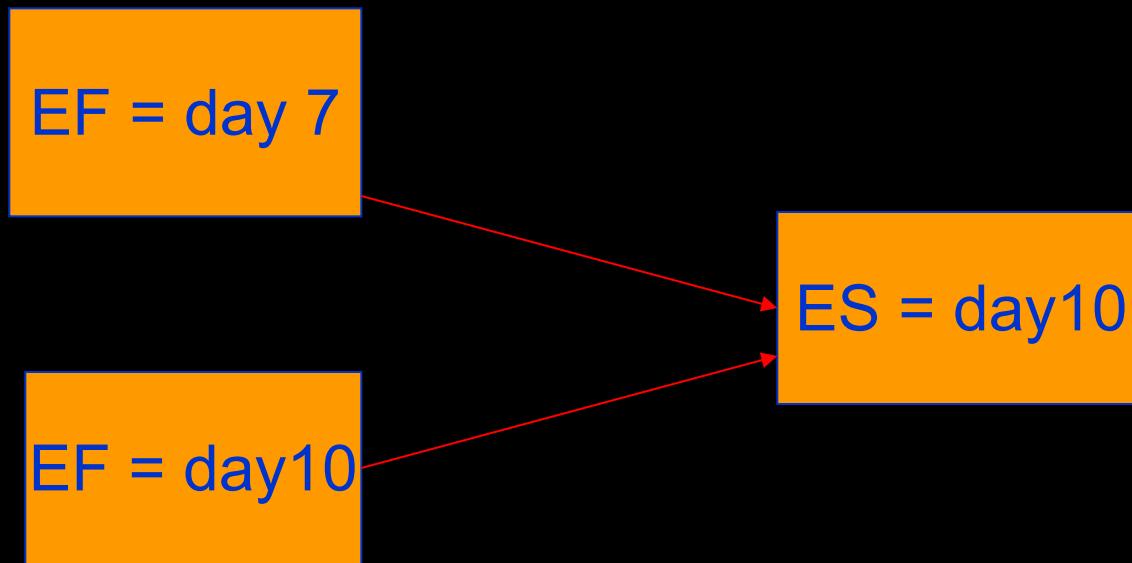
 T_L-t T_L-T_E T_L 

Complete for the previous example

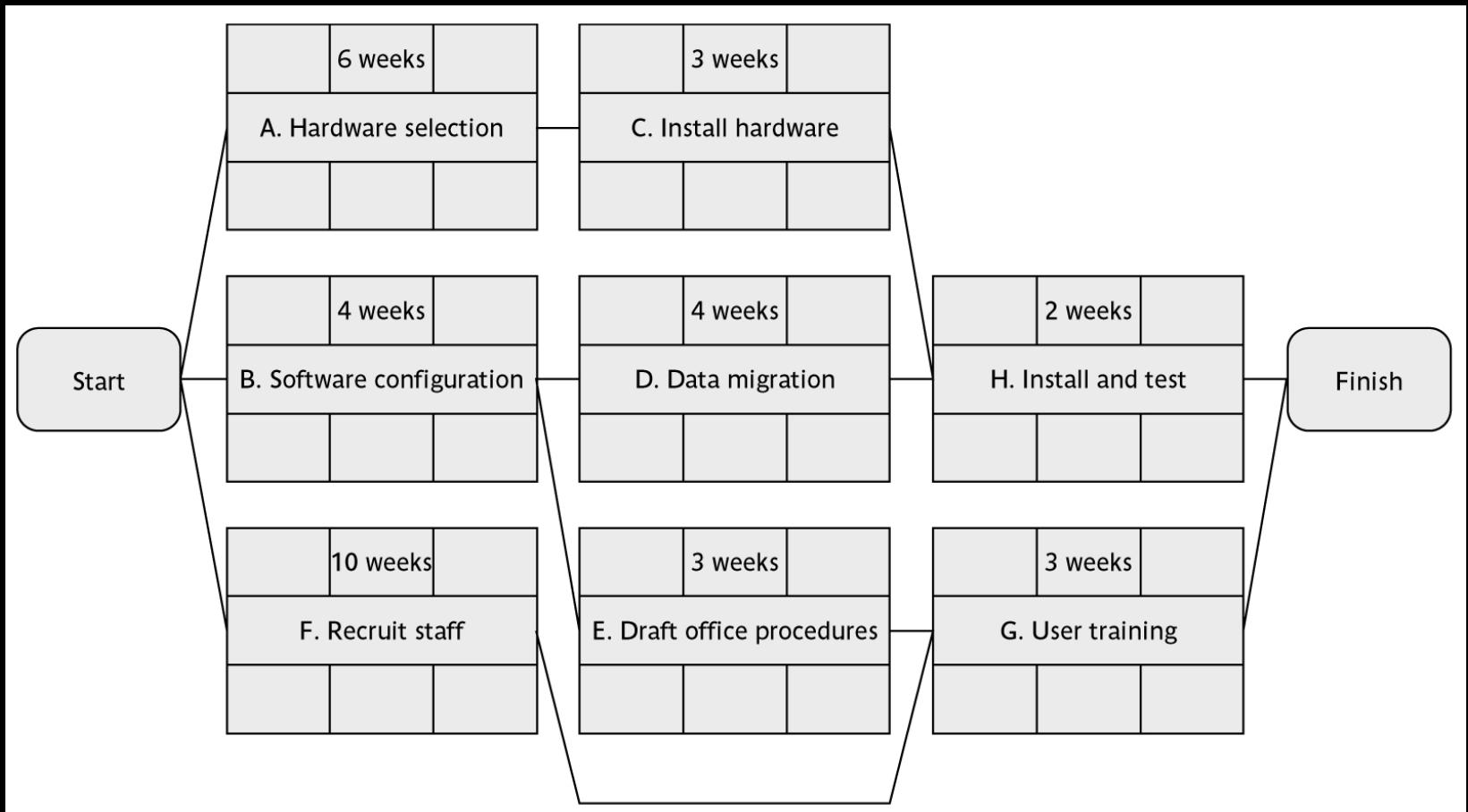
| | | |
|--|--|--|
| | | |
| | | |
| | | |

Forward pass

- 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 of an activity network



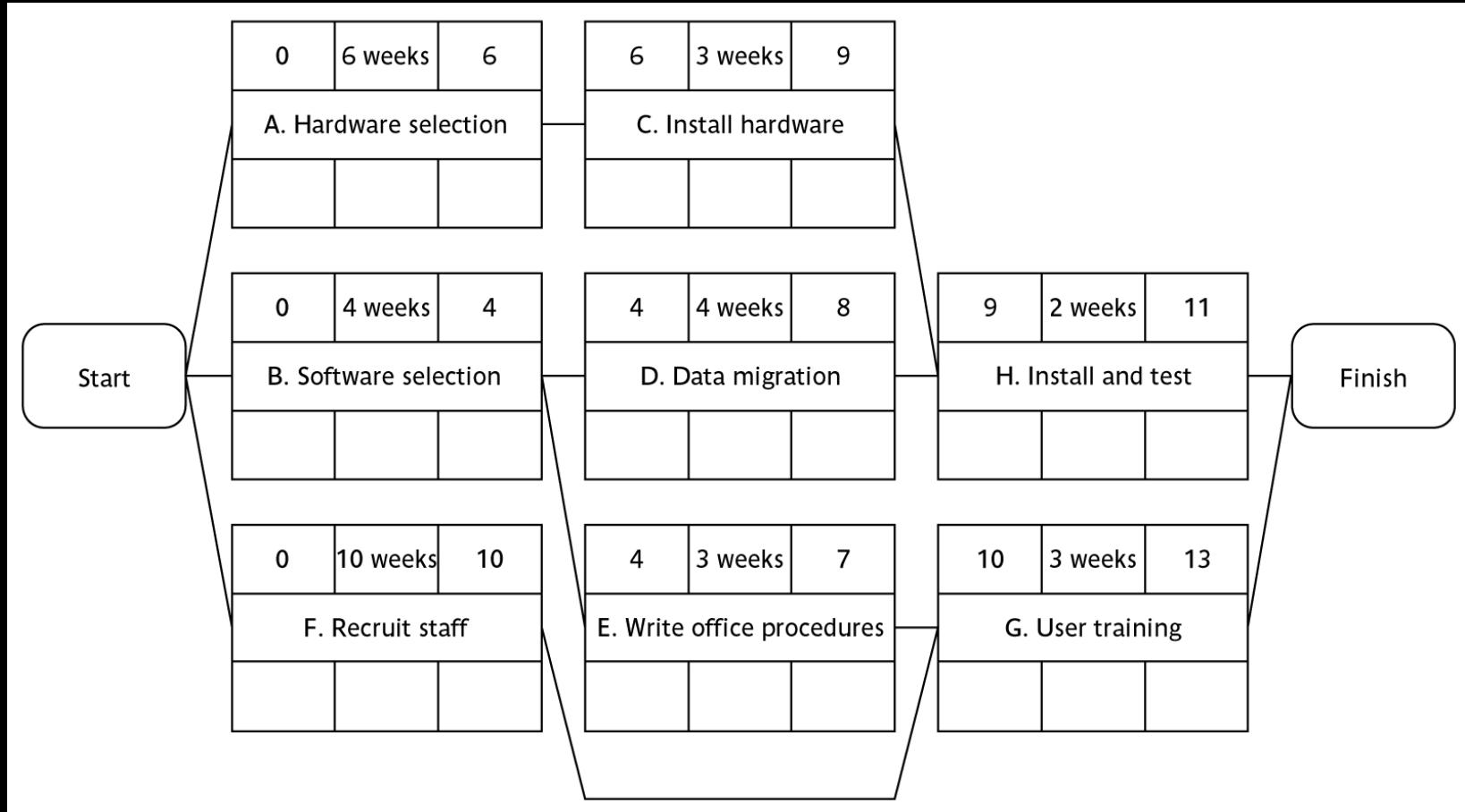
Complete the table

| Activity | ES | duration | EF |
|----------|----|----------|----|
| A | | | |
| B | | | |
| C | | | |
| D | | | |
| E | | | |
| F | | | |
| G | | | |
| H | | | |

Backward pass

- Start from the *last* activity
- Latest finish (LF) for last activity = earliest finish (EF)
- work backwards
- Latest finish for *current* activity = Latest start for the *following*
- More than one following activity - take the *earliest* LS
- Latest start (LS) = LF for activity - duration

Example: LS for all activities?



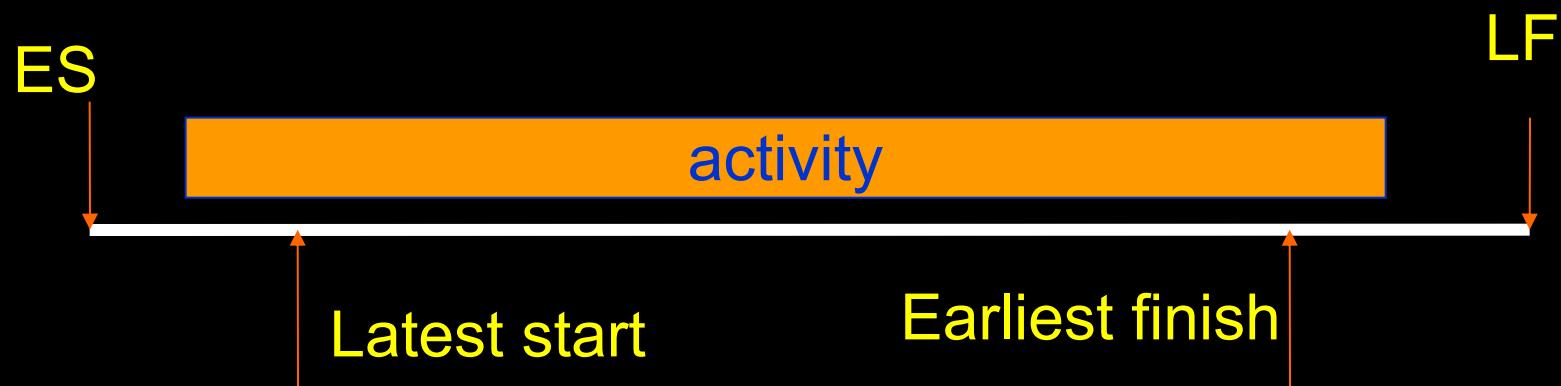
Complete the table

| Activity | ES | Dur | EF | LS | LF |
|----------|----|-----|----|----|----|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| F | | | | | |
| G | | | | | |
| H | | | | | |

Float



Float = Latest finish -
Earliest start -
Duration



Complete the table

| Activity | ES | Dur | EF | LS | LF | Float |
|----------|----|-----|----|----|----|-------|
| A | | | | | | |
| B | | | | | | |
| C | | | | | | |
| D | | | | | | |
| E | | | | | | |
| F | | | | | | |
| G | | | | | | |

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

Free and interfering float

| | | |
|---|----|---|
| 0 | 7w | 7 |
| A | | |
| 2 | 2 | 9 |

B can be up to 3 days late
and not affect any
other activity = **free float**

| | | |
|---|----|---|
| 0 | 4w | 4 |
| B | | |
| 5 | 5 | 9 |

| | | |
|---|----|----|
| 7 | 1w | 8 |
| D | | |
| 9 | 2 | 10 |

| | | |
|----|----|----|
| 10 | 2w | 12 |
| E | | |
| 10 | 0 | 12 |

| | | |
|---|-----|----|
| 0 | 10w | 10 |
| A | | |
| 0 | 0 | 10 |

B can be a further 2 days late – affects
D but not the project end date =
interfering float

Learning Resource On Software Project Management

Chapter-7 Risk Management

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

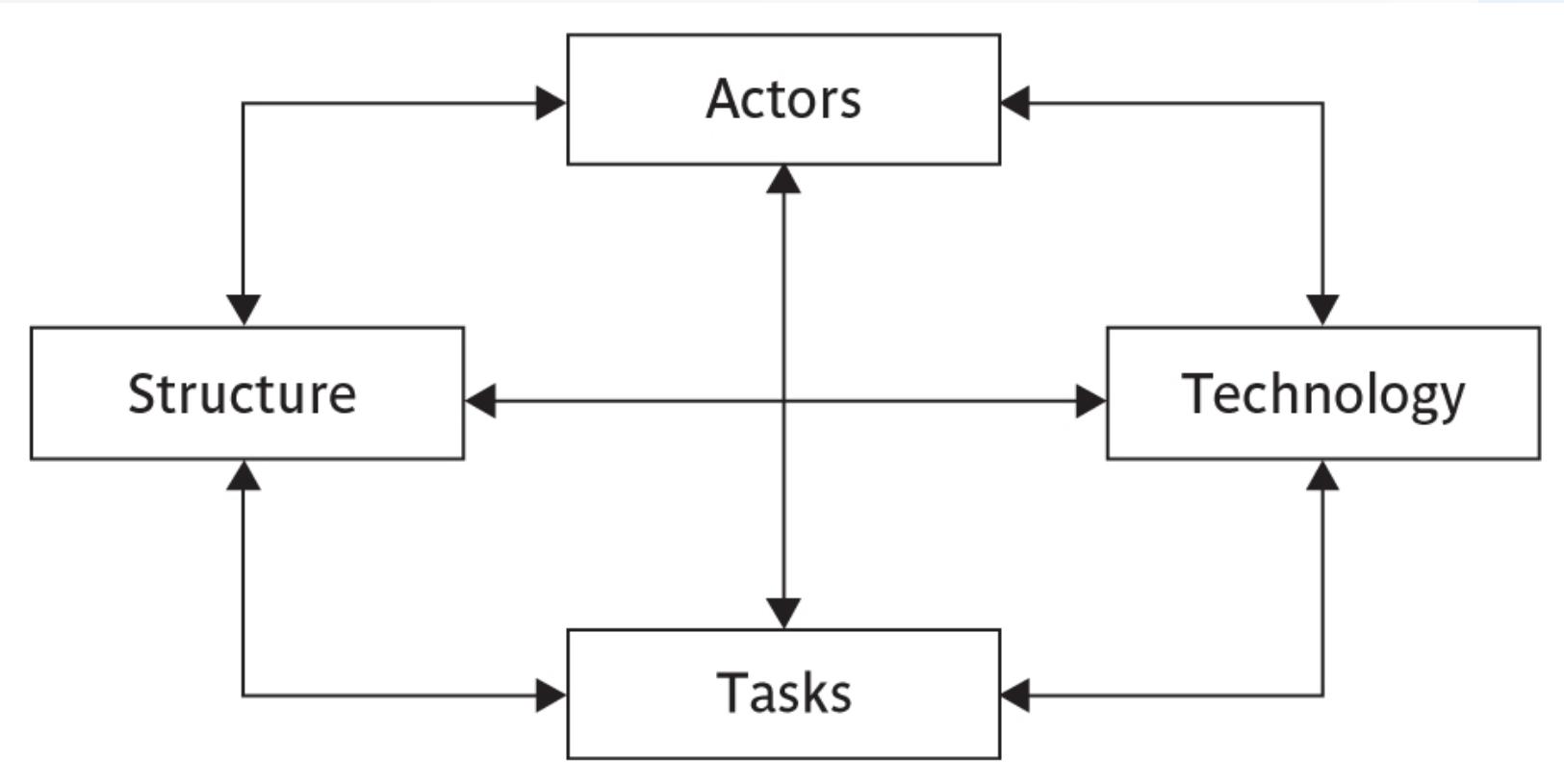
Some definitions of risk

'the chance of exposure to the adverse consequences of future events' PRINCE2

'an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives' PM-BOK

- Risks relate to **possible future** problems, not current ones
- They involve a possible cause and its effect(s) e.g. developer leaves > task delayed

Categories of risk



A framework for dealing with risk

The planning for risk includes these steps:

- Risk identification – what risks might there be?
- Risk analysis and prioritization – which are the most serious risks?
- Risk planning – what are we going to do about them?
- 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

| <i>Risk</i> | <i>Risk reduction techniques</i> |
|---|--|
| Personnel shortfalls | Staffing with top talent; job matching; teambuilding; 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 |
| 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 - continued

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

Risk prioritization

Risk exposure (RE)

= (potential damage) x (probability of occurrence)

Ideally

Potential damage: a money value e.g. a flood would cause Rs. 0.5 millions of damage

Probability 0.00 (absolutely no chance) to 1.00 (absolutely certain) e.g. 0.01 (one in hundred chance)

$$RE = Rs. 0.5m \times 0.01 = Rs. 5,000$$

Crudely analogous to the amount needed for an insurance premium

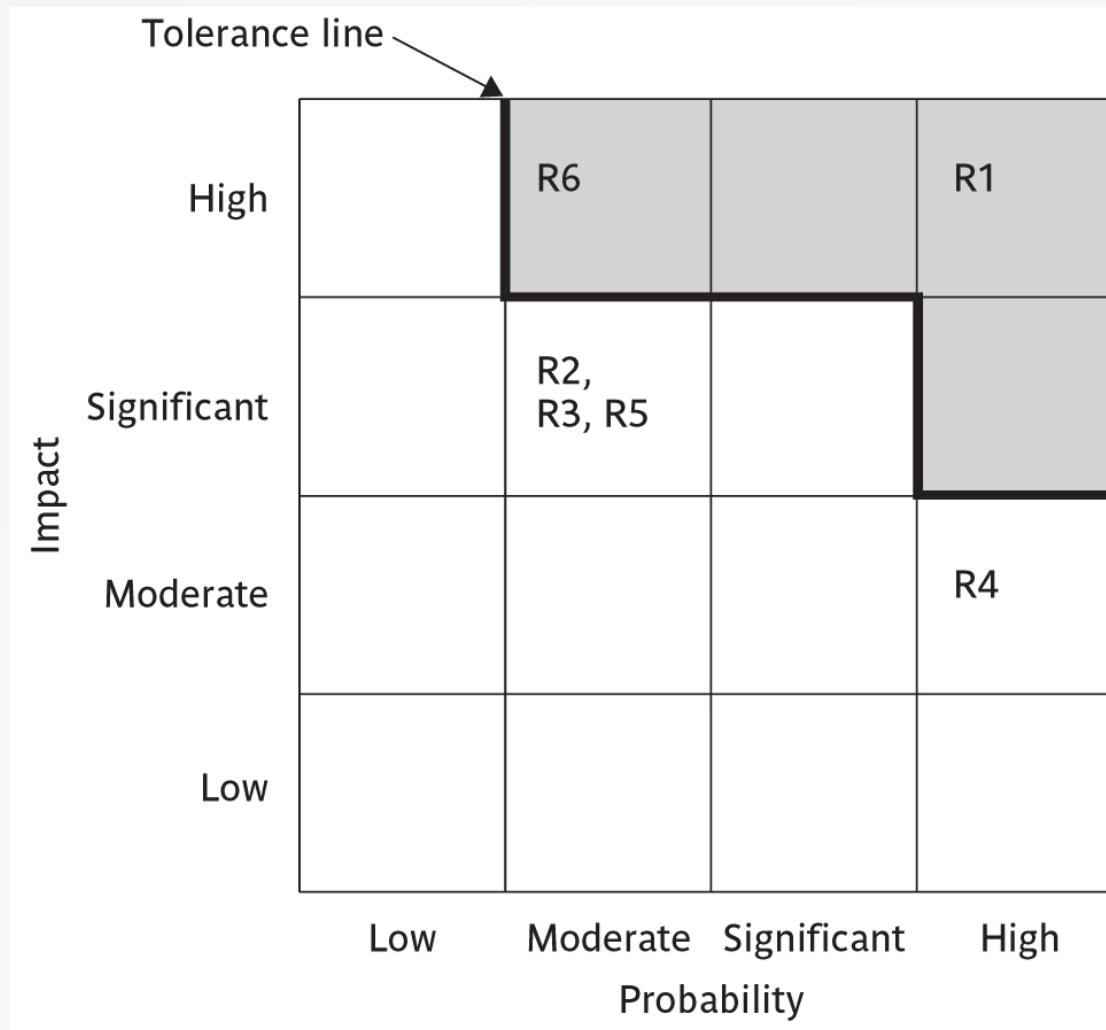
Risk probability: qualitative descriptors

| <i>Probability level</i> | <i>Range</i> |
|--------------------------|--------------------------------------|
| 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

| <i>Impact level</i> | <i>Range</i> |
|---------------------|---|
| 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

Risk reduction leverage (RRL)

Risk reduction leverage =

$$(RE_{\text{before}} - RE_{\text{after}}) / (\text{cost of risk reduction})$$

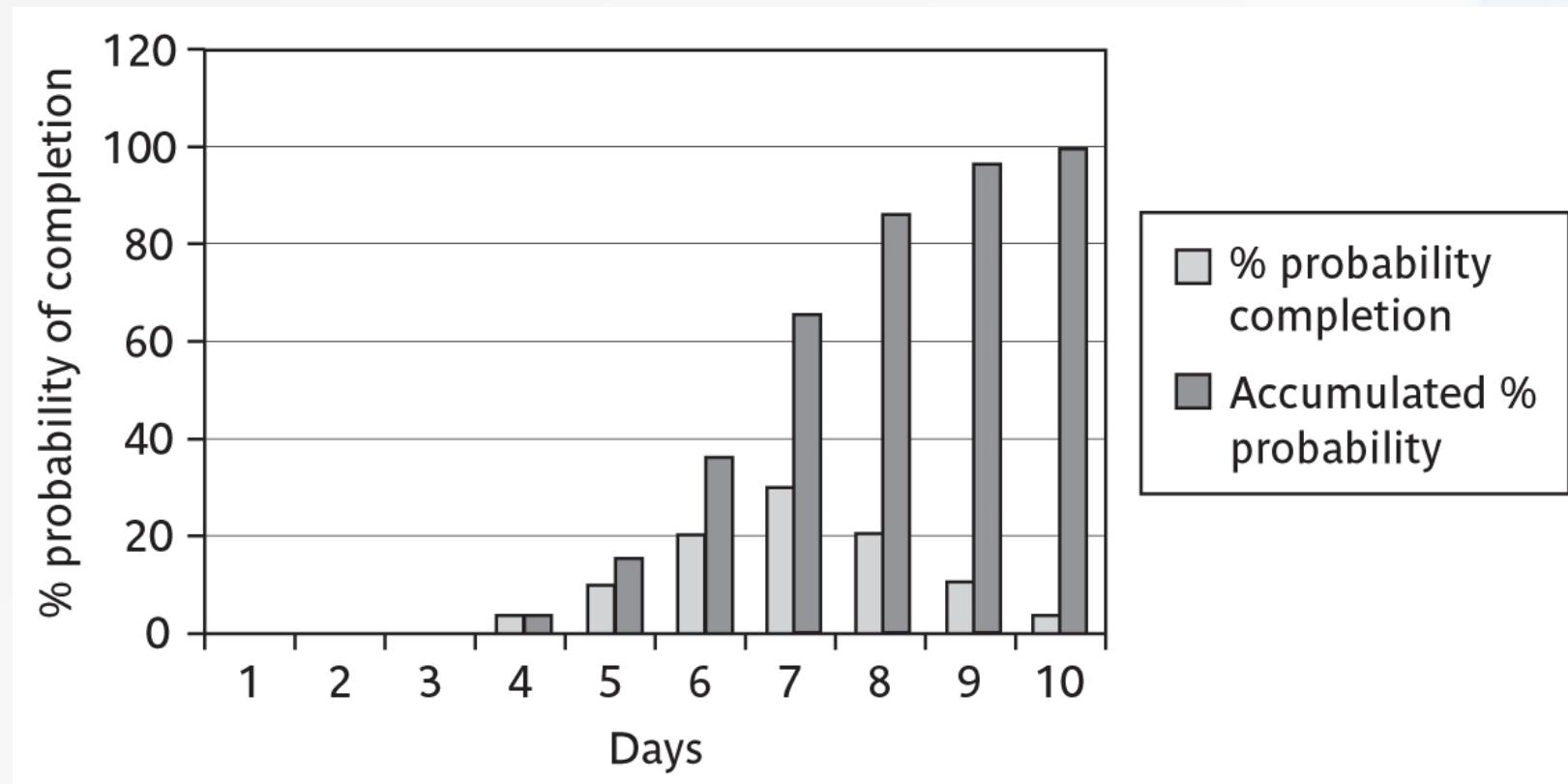
RE_{before} is risk exposure before risk reduction e.g. 1% chance of a fire causing £200k damage

RE_{after} is risk exposure after risk reduction e.g. fire alarm costing Rs. 500 reduces probability of fire damage to 0.5%

$$RRL = (1\% \text{ of Rs. } 200k) - (0.5\% \text{ of Rs. } 200k) / \text{Rs. } 500 = 2$$

$RRL > 1.00$ therefore worth doing

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 | a | m | b | t_e | s |
|------|----|----|----|-------|---|
| A | 10 | 12 | 16 | ? | ? |
| B | 8 | 10 | 14 | ? | ? |
| C | 20 | 24 | 38 | ? | ? |

A chain of activities

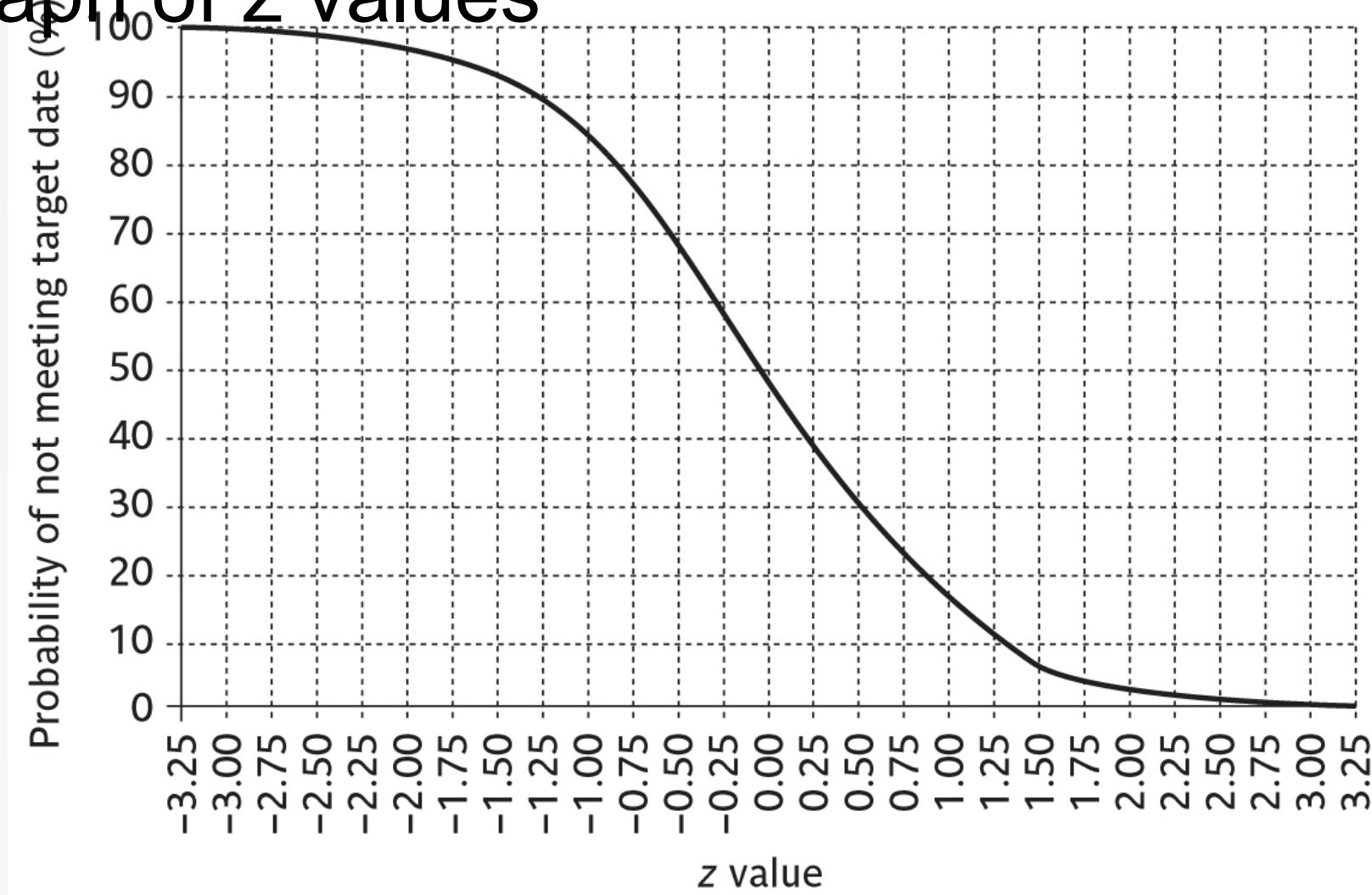
- What would be the expected duration of the chain A + B + C?
- Answer: $12.66 + 10.33 + 25.66$ i.e. 48.65

- What would be the standard deviation for A + B+ C?
- Answer: square root of $(1^2 + 1^2 + 3^2)$ i.e. 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
$$z = (T - t_e)/s$$
- In this example $z = (52 - 48.33)/3.32$ i.e. 1.01
- Look up in table of z values – see next overhead

Graph of z values

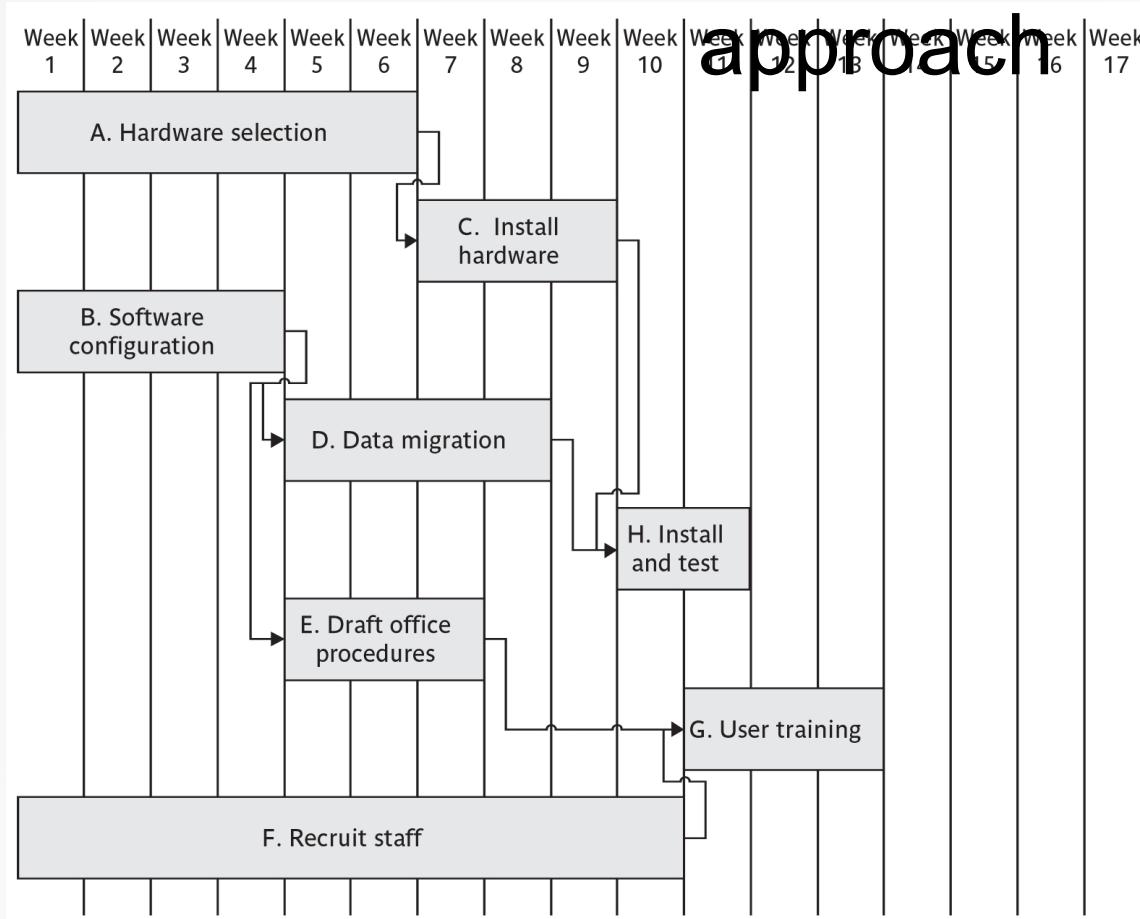


Monte Carlo Simulation

- An alternative to PERT.
- A class of general analysis techniques:
 - Valuable to solve any problem that is complex, nonlinear, or involves more than just a couple of uncertain parameters.
- Monte Carlo simulations involve repeated random sampling to compute the results.
- Gives more realistic results as compared to manual approaches.

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 approach

One answer to this:

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

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 |
| E | 3 | 4 | 1 |
| F | 10 | 15 | 5 |
| G | 3 | 4 | 1 |
| H | 2 | 2.5 | 0.5 |

TABLE 7.8 Most likely and comfort zone estimates (days)

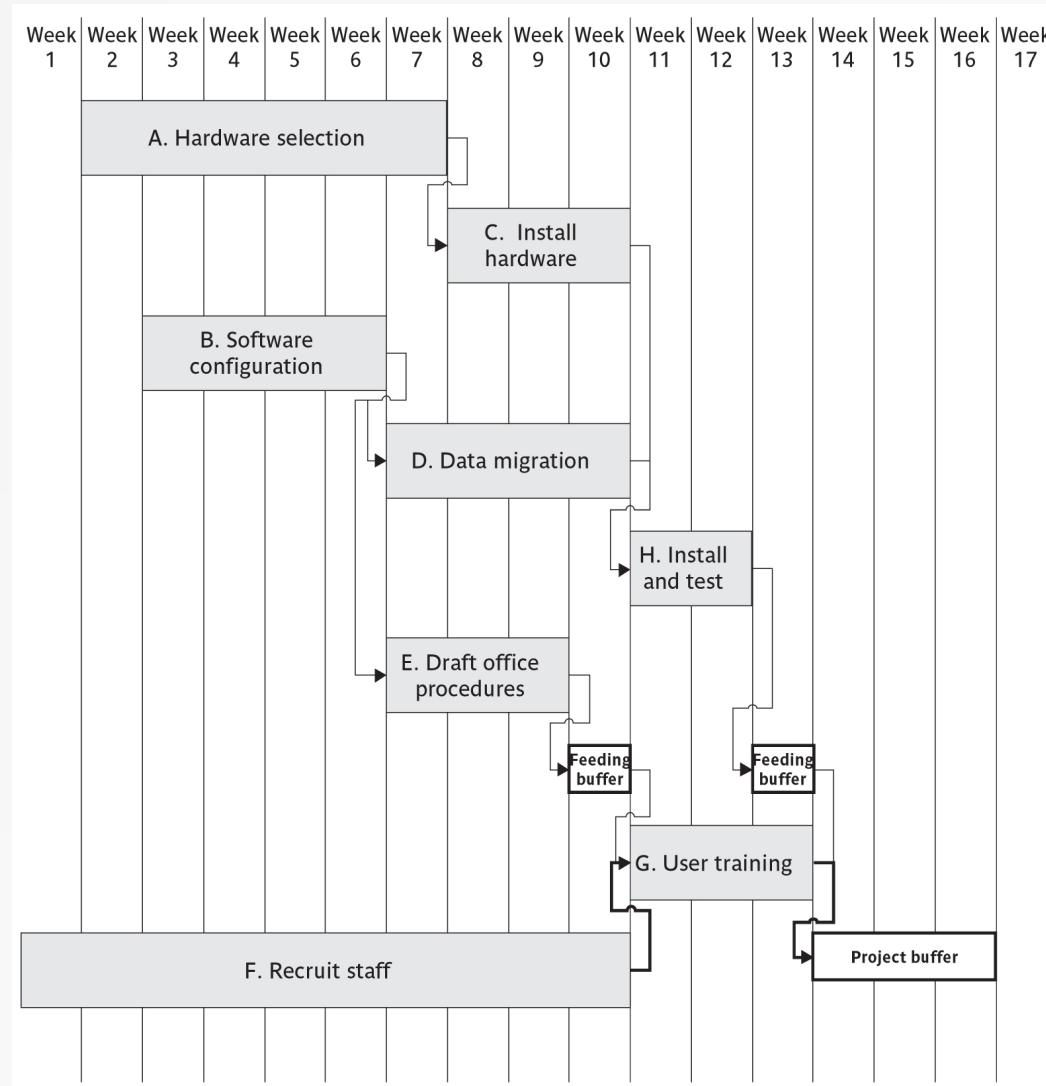
Critical chain - continued

3. Identify the critical chain – same a critical path but resource constraints also taken into account
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 -continued

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

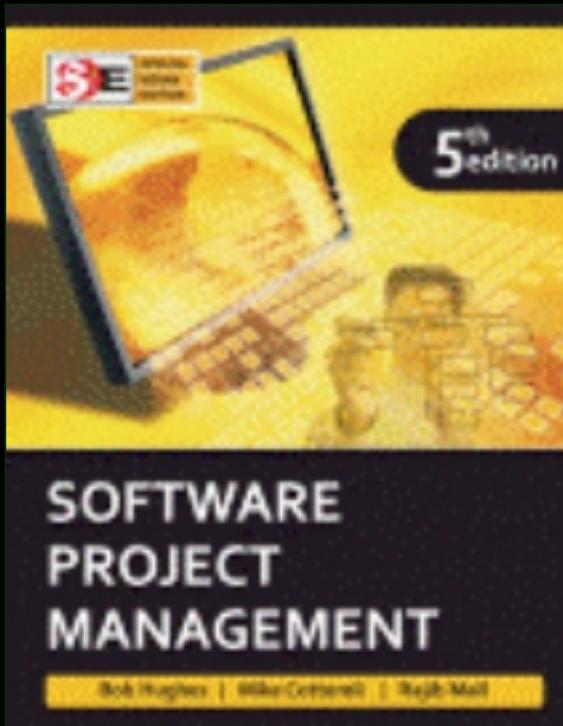
Executing the critical chain-based plan

Buffers are divided into three zones:

- **Green**: the first 33%. No action required
- **Amber** : the next 33%. Plan is formulated
- **Red** : last 33%. Plan is executed.

Software Project Management

Fifth Edition



Chapter 8

Resource allocation

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

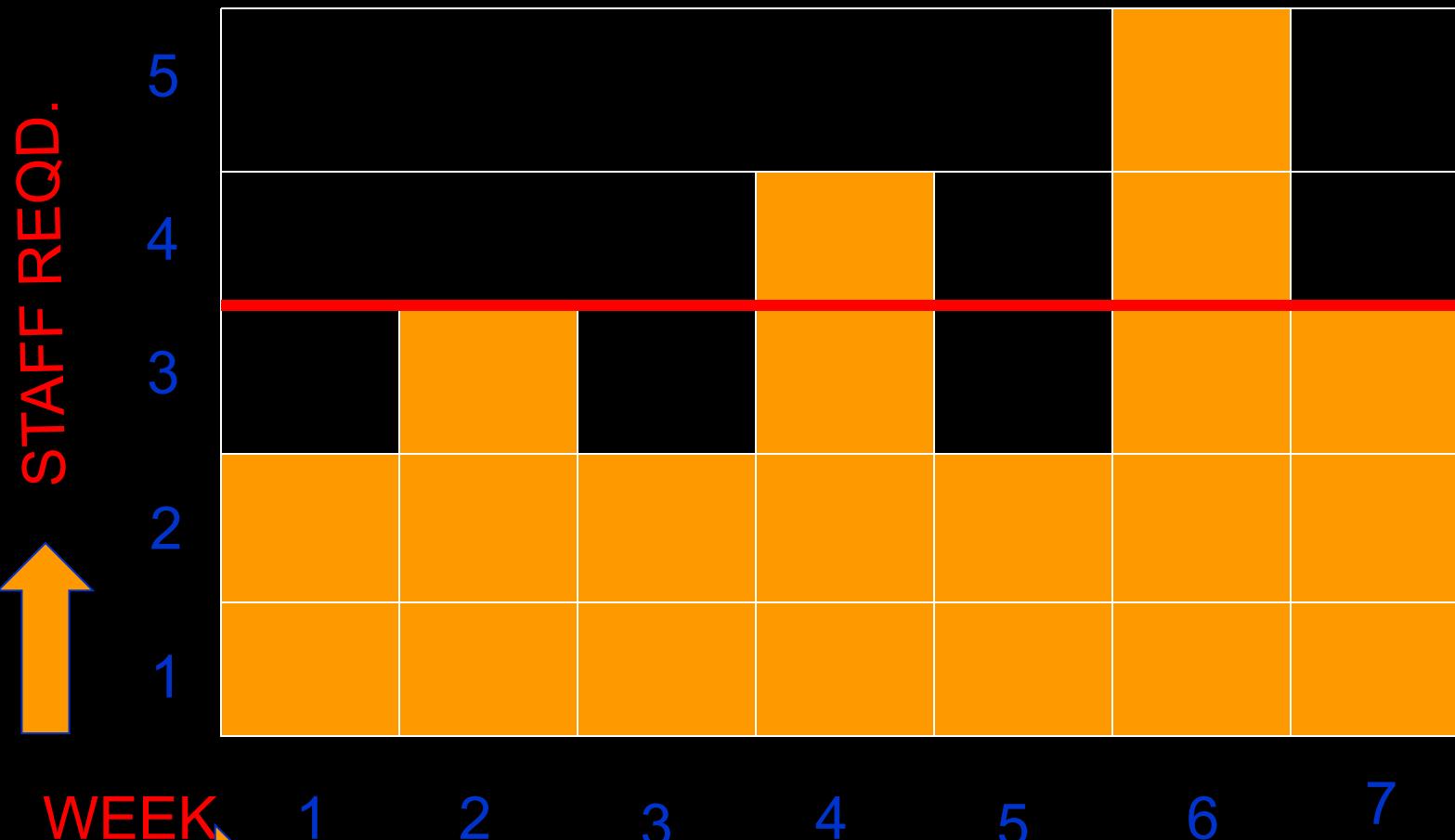
Resources

- These include
 - ◆ labour
 - ◆ equipment (e.g. workstations)
 - ◆ materials
 - ◆ space
 - ◆ services
- Time: elapsed time can often be reduced by adding more staff
- Money: used to buy the other resources

Resource allocation

- Identify the resources needed for each activity and create a *resource requirement list*
- Identify *resource types* - individuals are interchangeable within the group (e.g. ‘VB programmers’ as opposed to ‘software developers’)
- Allocate resource types to activities and examine the *resource histogram*

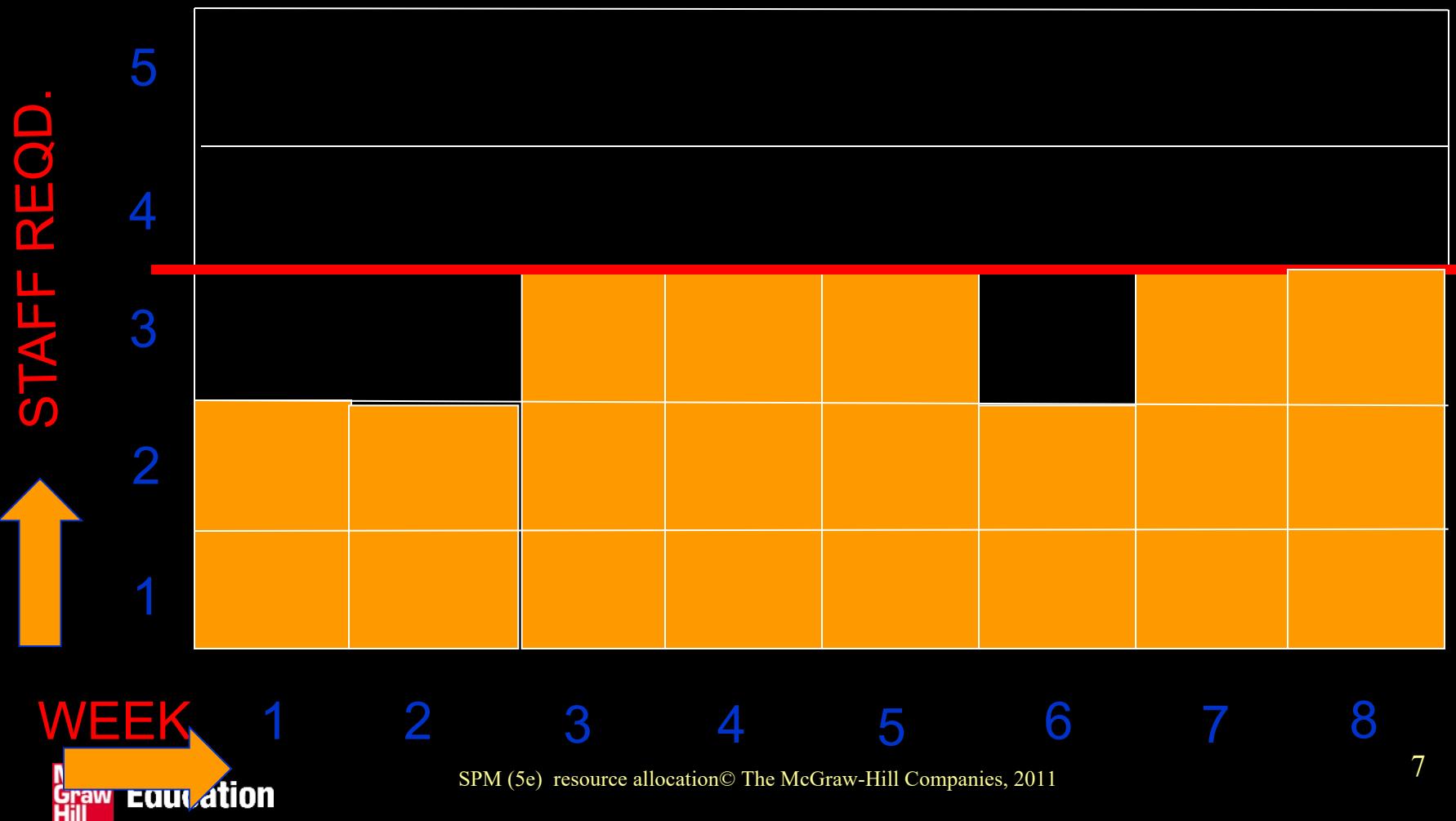
Resource histogram: systems analysts



Resource smoothing

- It is usually difficult to get specialist staff who will work odd days to fill in gaps – need for staff to learn about application etc
- 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*
 - ◆ moving resource from a non-critical activity
 - ◆ 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
- *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:

- Shortest critical activities
- Other critical activities
- Shortest non-critical activities
- Non-critical activities with least float
- Non-critical activities

Resource usage

- need to maximise %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 – recall *critical chains*
- 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

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

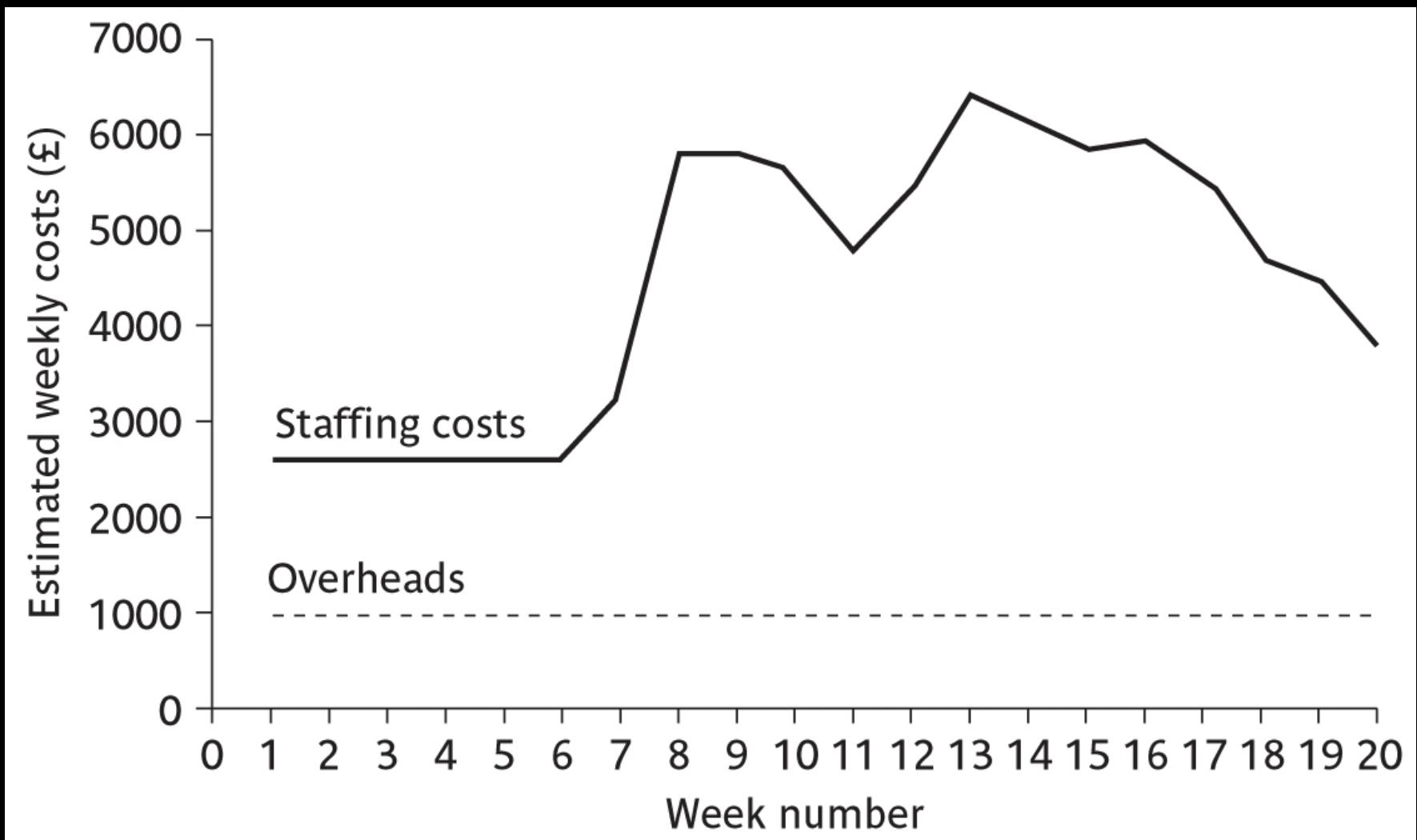
Cost schedules

Cost schedules can now be produced:

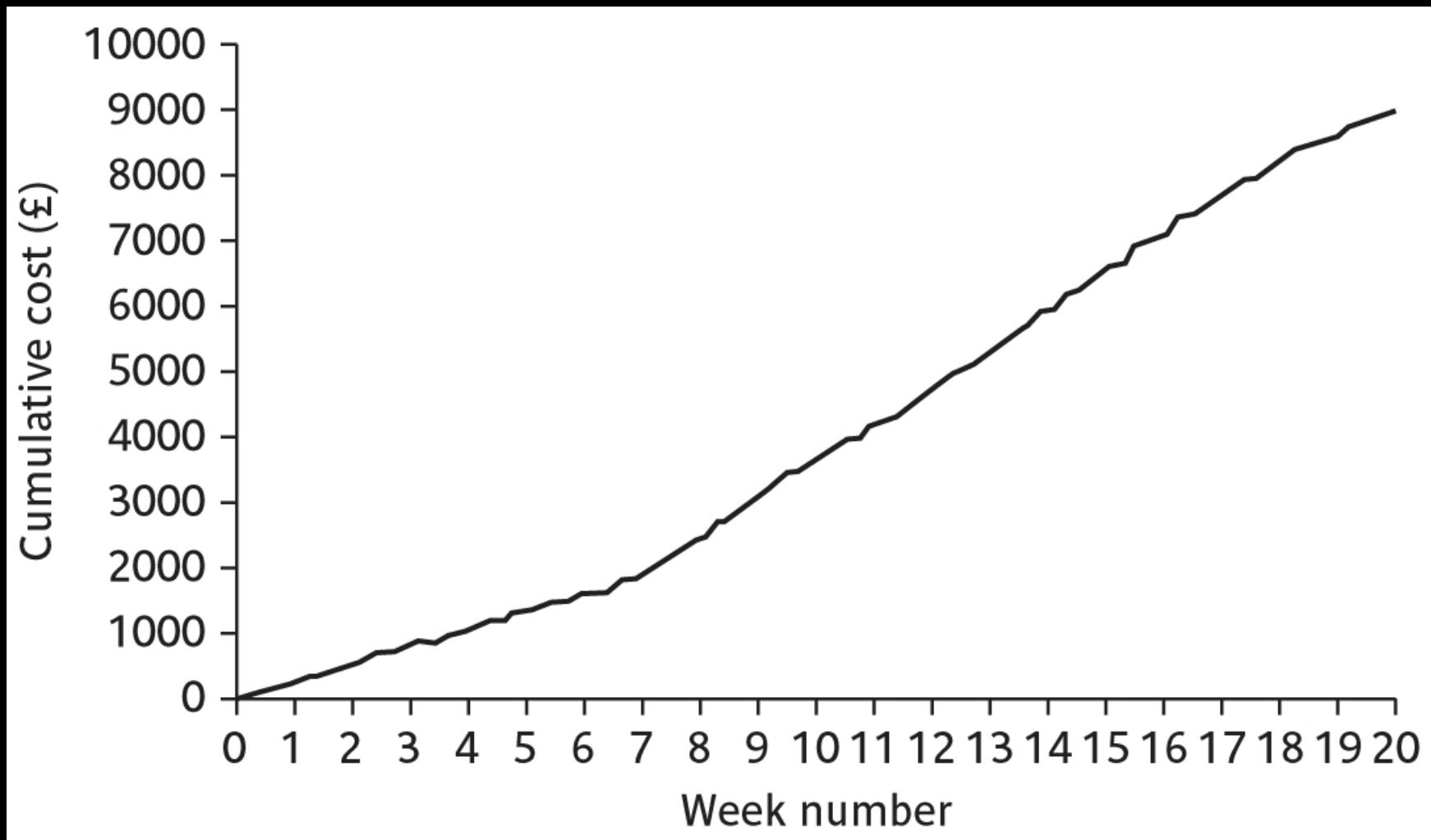
Costs include:

- Staff costs
- Overheads
- Usage charges

Cost profile

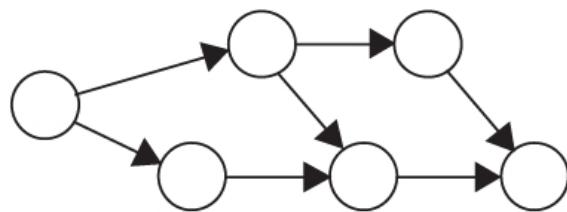


Accumulative costs



Balancing concerns

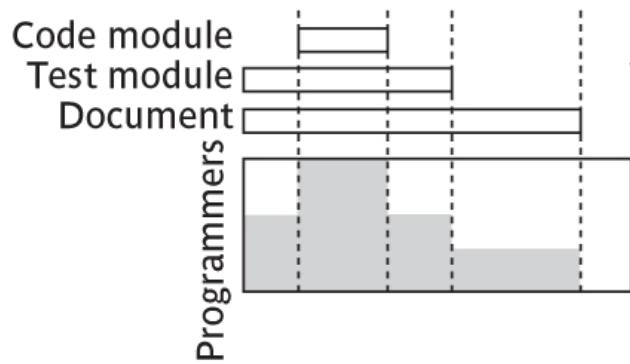
Activity plan



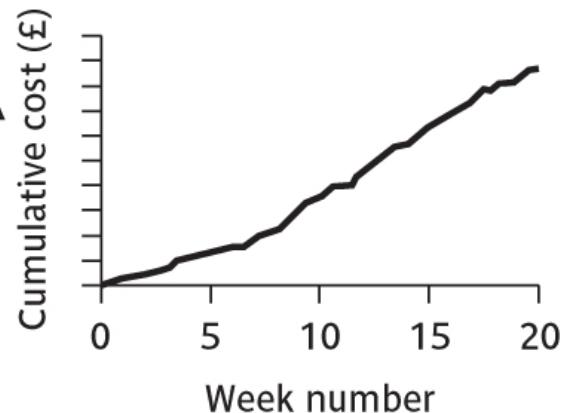
Risk assessment



Resource allocation

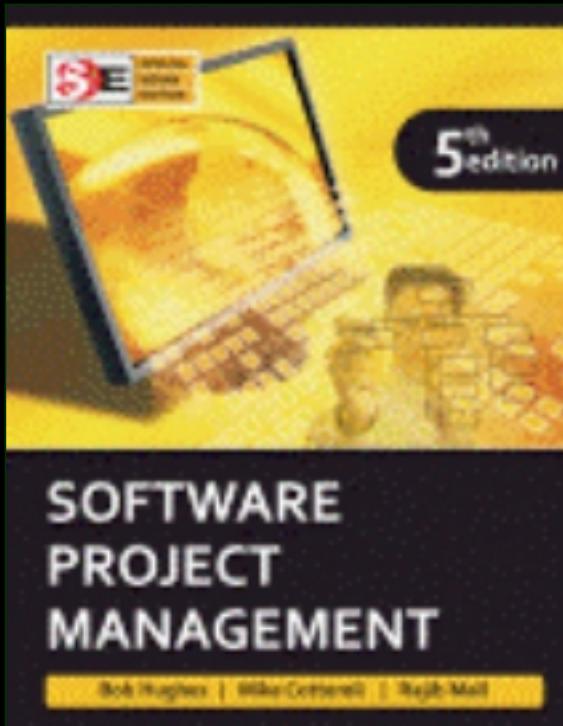


Cost schedule



Software Project Management

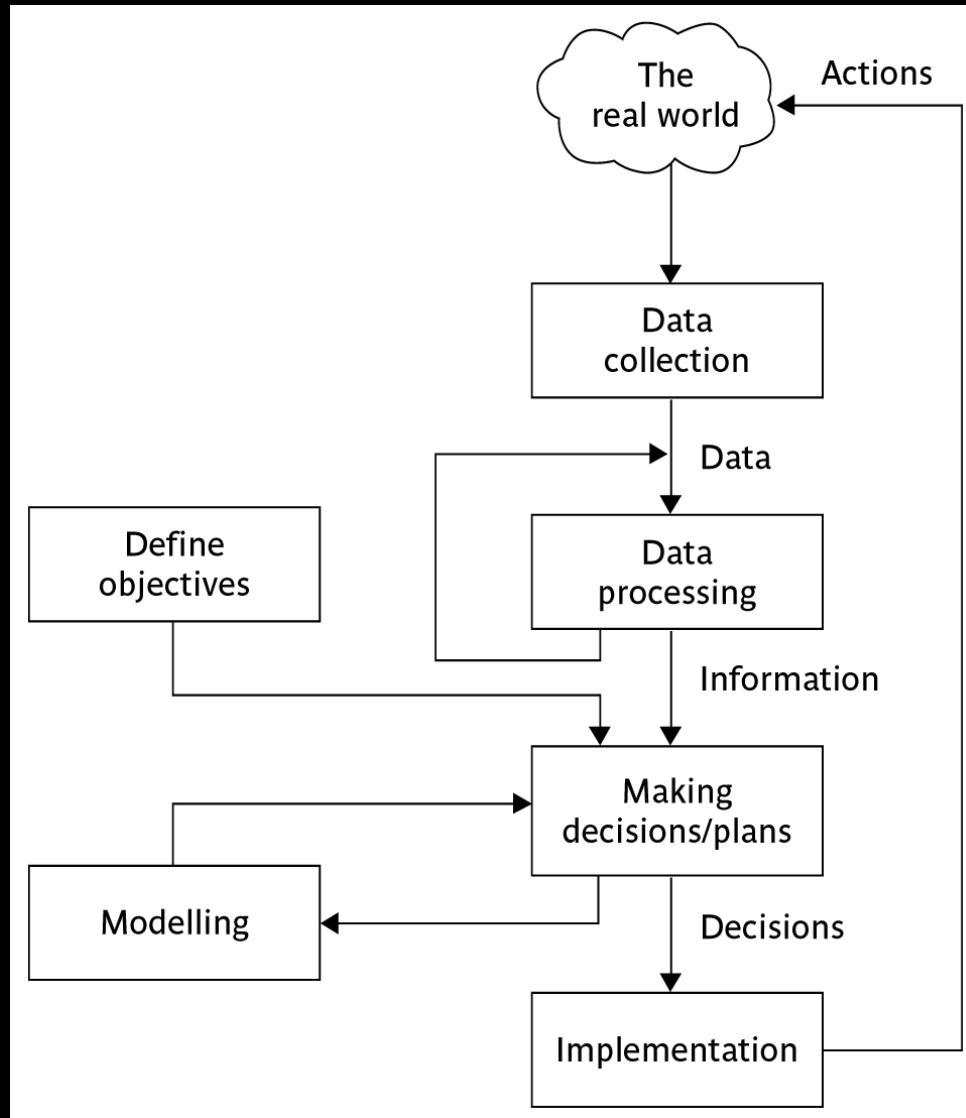
Fifth Edition



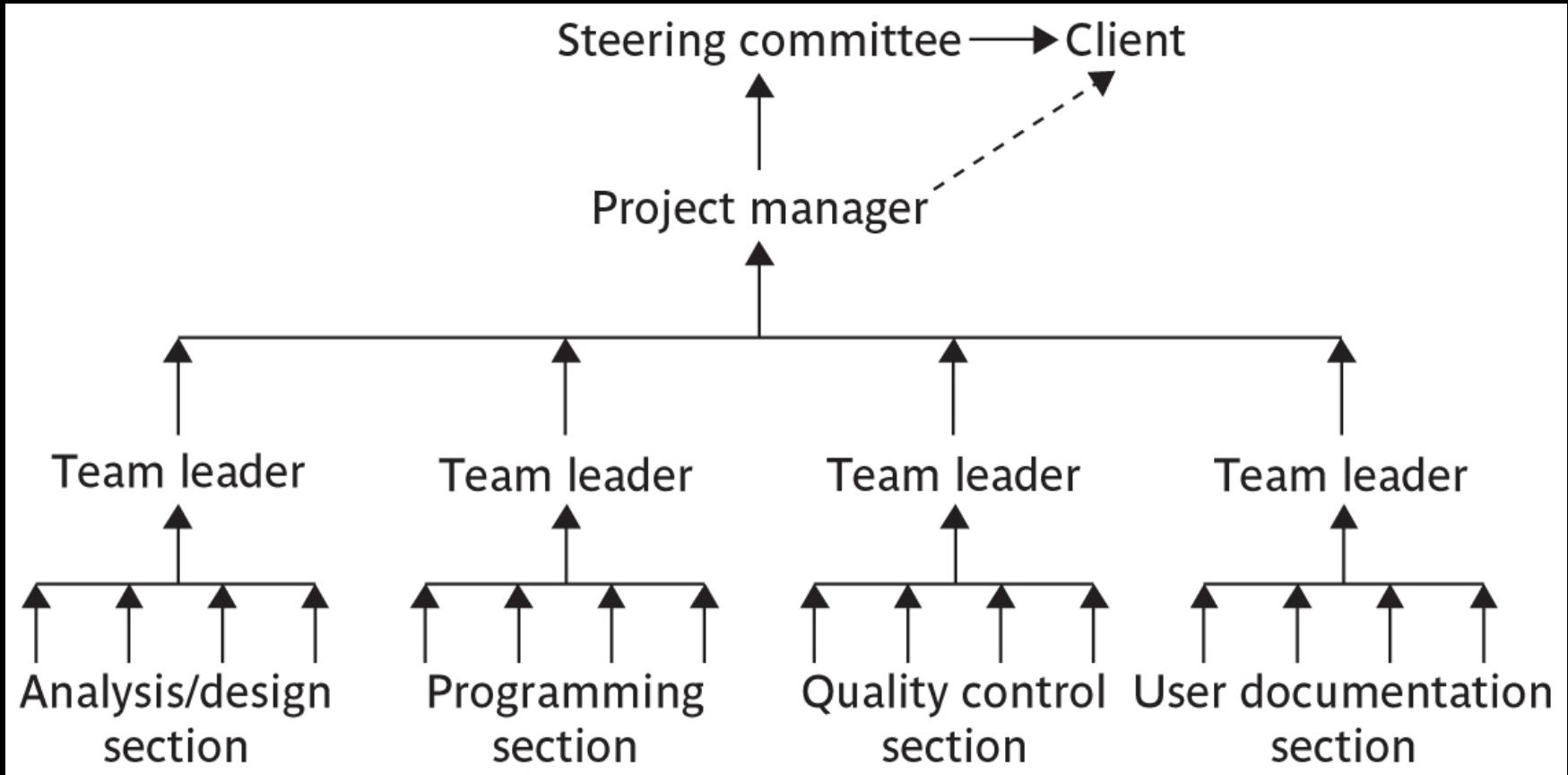
Chapter 9

Monitoring and control

The control cycle



Responsibilities



Assessing progress



Checkpoints – predetermined times when progress is checked

- ↳ Event driven: check takes place when a particular event has been achieved
- ↳ Time driven: date of the check is pre-determined

Frequency of reporting

The higher the management level then generally the longer the gaps between checkpoints

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

Red/Amber/Green 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

Review

- Review of work products is an important mechanism for monitoring the progress of a project and ensuring the quality of the work products.
- Testing is an effective defect removal mechanism.
 - ◆ However, testing is applicable to only executable code.
 - ◆ Review is applicable to all work products.

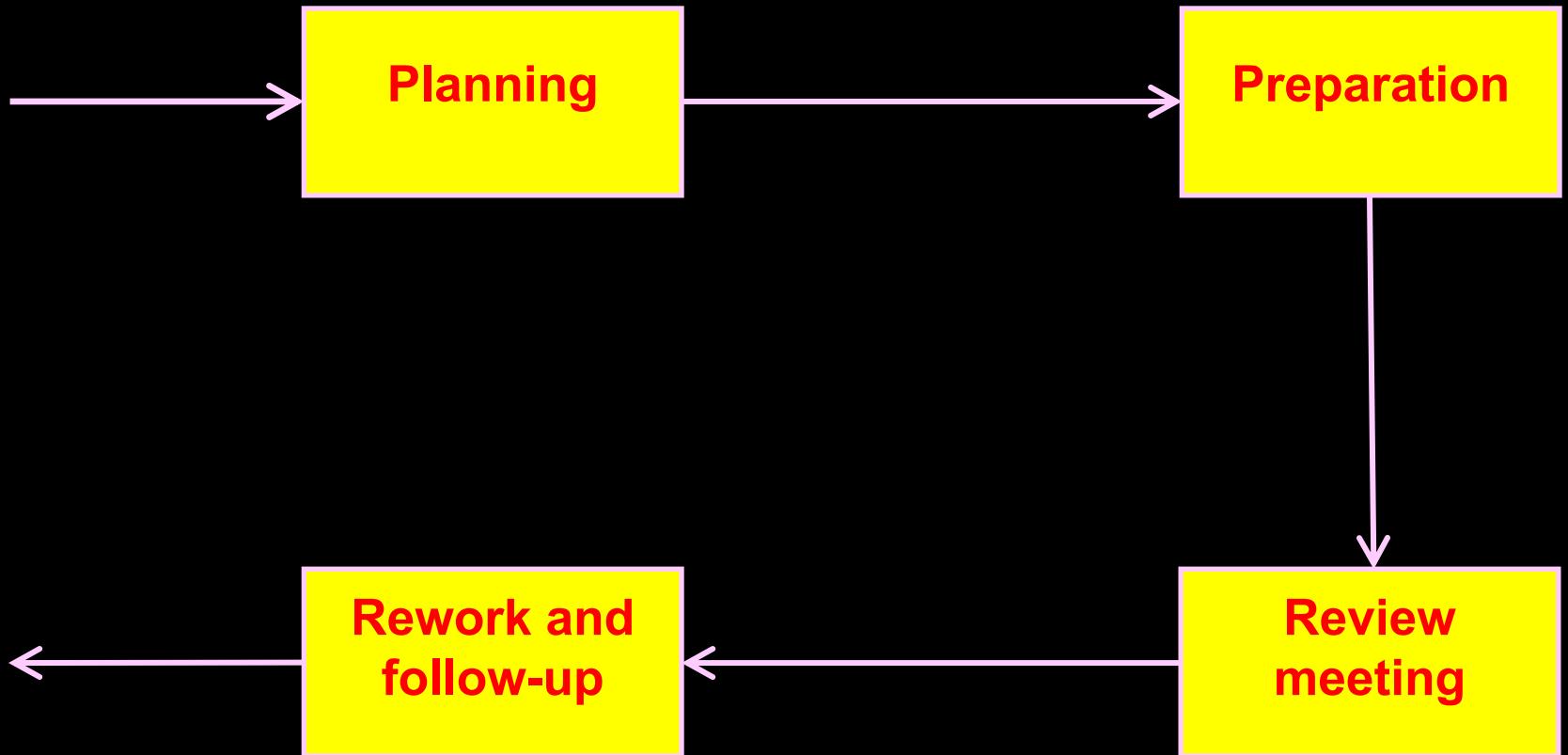
Utility of Review

- A cost-effective defect removal mechanism.
- Review usually helps to identify any deviation from standards.
- Reviewers suggest ways to improve the work product
- a review meeting often provides learning opportunities to not only the author of a work product, but also the other participants of the review meeting.
- The review participants gain a good understanding of the work product under review, making it easier for them to interface or use the work product in their work.

Review Roles

- Moderator:
 - ◆ Schedules and convenes meetings, distributes review materials, leads and moderates review sessions.
- Recorder:
 - ◆ Records the defects found and the time and effort data.
- Reviewers.

Review Process



Project Termination Review

- Project termination reviews provide important opportunities to learn from past mistakes as well as successes.
- Project termination need not necessarily mean project failure or premature abandonment.
 - ↳ A project may be terminated on successful completion

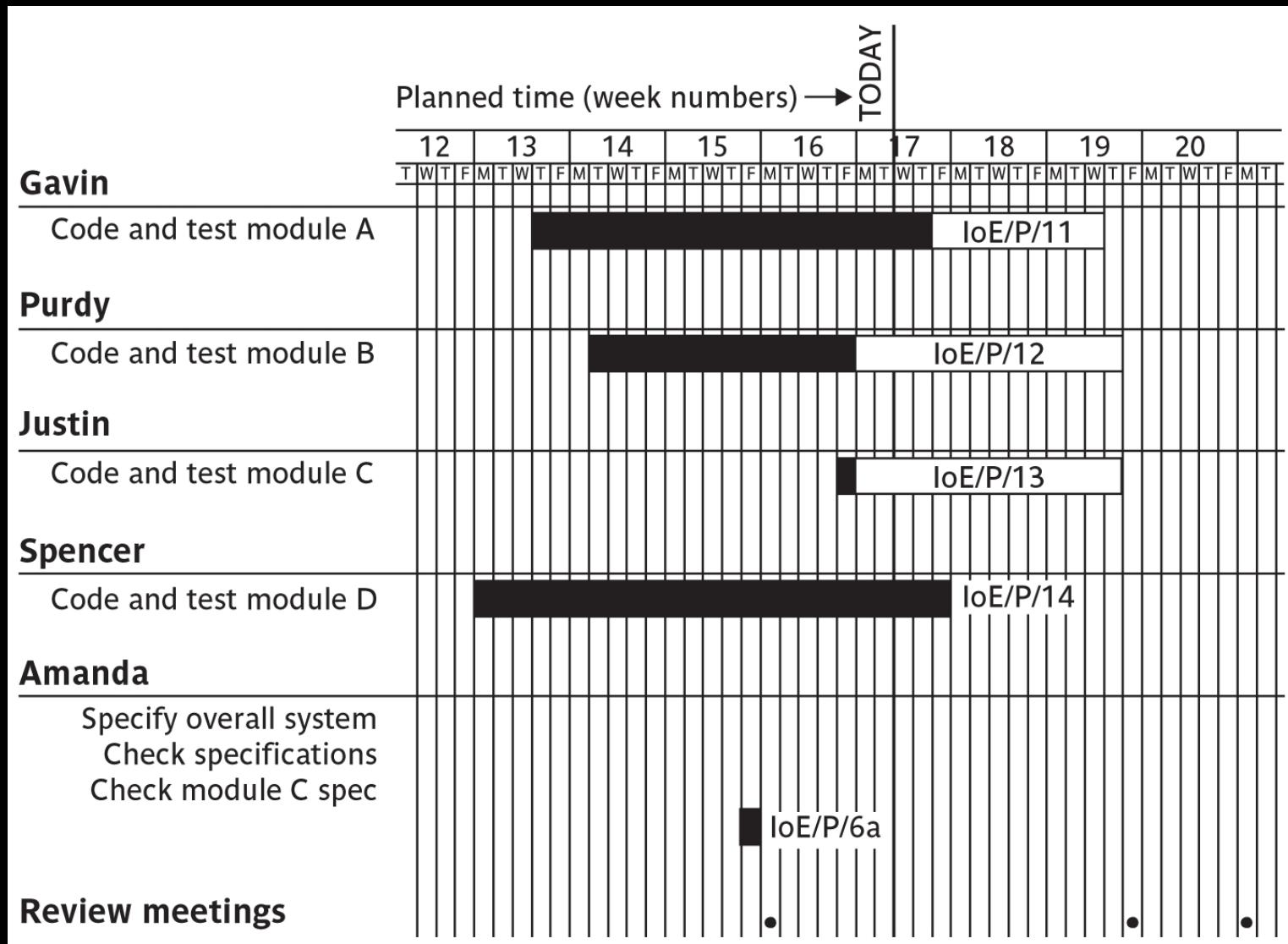
Reasons for Project Termination

- Project is completed successfully handed over to the customer.
- Incomplete requirements
- Lack of resources
- Some key technologies used in the project have become obsolete during project execution
- Economics of the project has changed, for example because many competing product may have become available in the market.

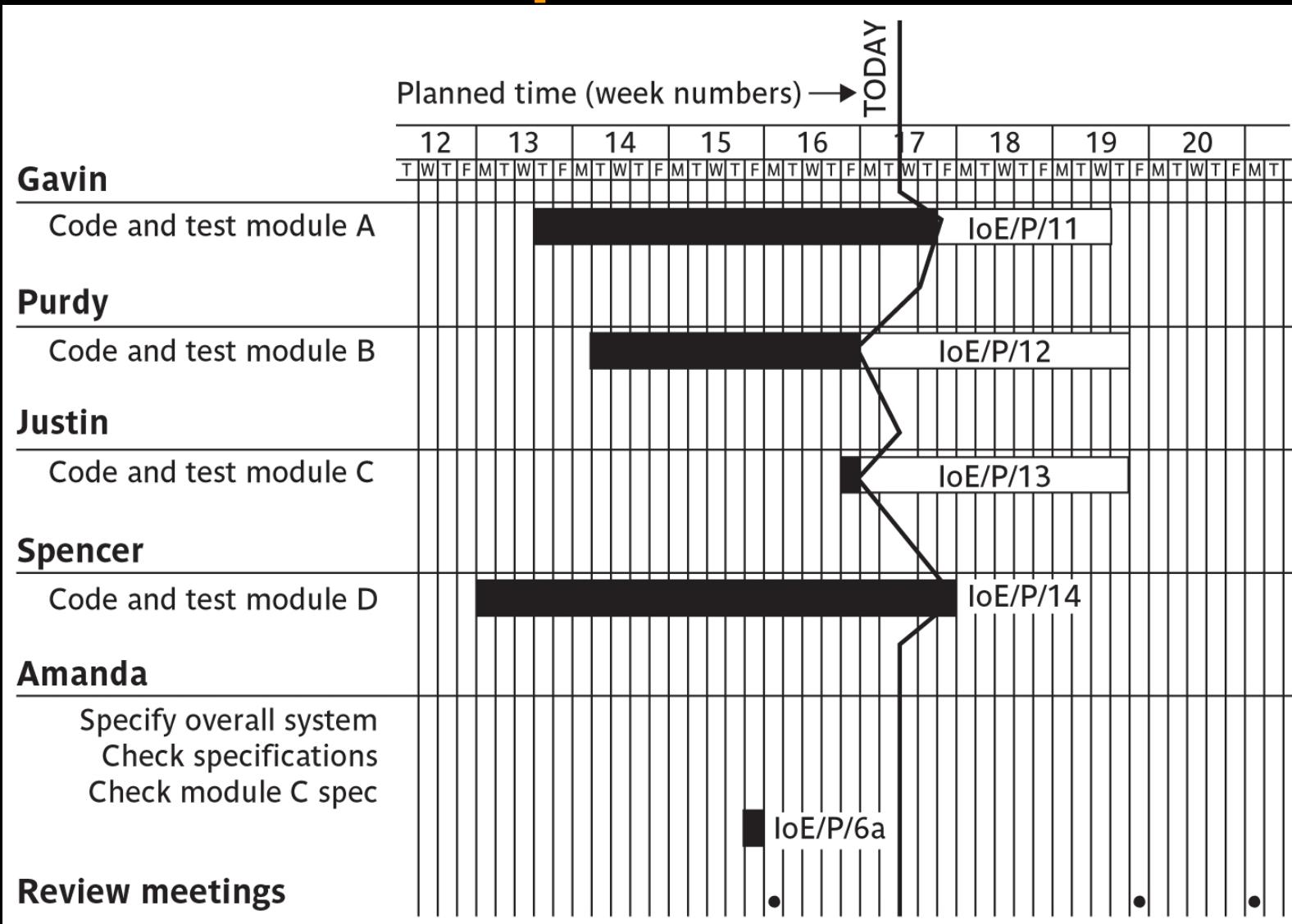
Project Termination Process

- Project survey
- Collection of objective information
- Debriefing meeting
- Final project review
- Result publication

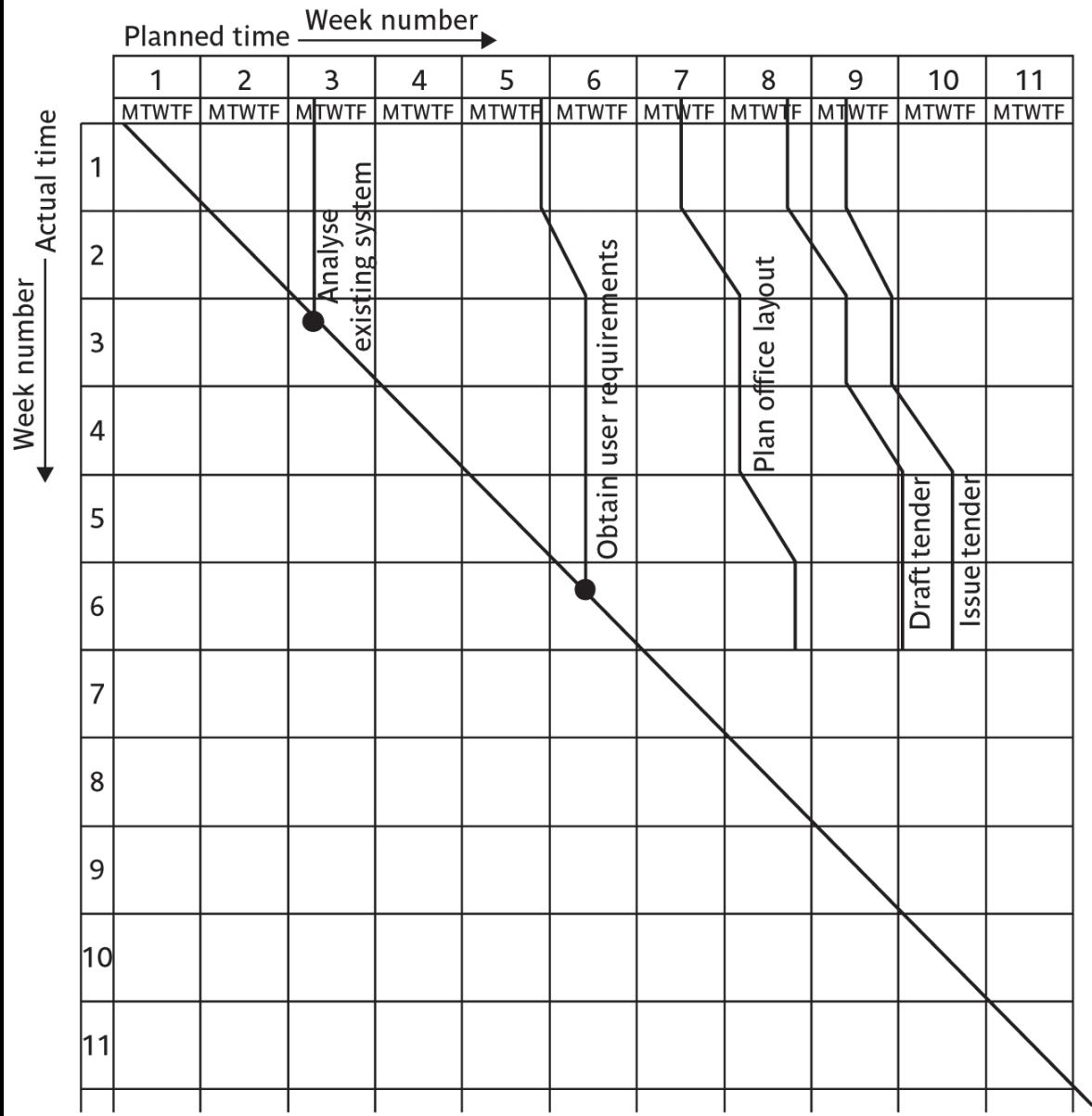
Gantt charts



Slip charts



The timeline



Cost monitoring

- A project could be late because the staff originally committed have not been deployed
- In this case the project will be *behind time* but *under budget*
- A project could be on time but only because additional resources have been added and so be *over budget*
- Need to monitor both achievements and costs

Earned value analysis

- *Planned value (PV)* or *Budgeted cost of work scheduled (BCWS)* – original estimate of the effort/cost to complete a task (compare with idea of a ‘price’)
- *Earned value (EV)* or *Budgeted cost of work performed (BCWP)* – total of PVs for the work completed at this time

Accounting conventions

- 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

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 = EV-PV i.e. $13-19 = -6$
- Schedule performance indicator (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
- Actual cost = 7 days
- Cost variance (CV) = EV-AC i.e. $13-7 = 6$ days
- Cost performance indicator = $13/7 = 1.86$
- Positive CV 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)

Time variance

- Time variance (TV) – difference between time when specified EV should have been reached and time it actually was
- For example say an EV of £19000 was supposed to have been reached on 1st April and it was actually reached on 1st July then $TV = -3$ months

Earned value chart with revised forecasts

Activity Assessment Sheet

Staff Justin

Ref: IoE/P/13

Activity: Code and test module C

| Week number | 13 | 14 | 15 | 16 | 17 | 18 | |
|----------------------------|----|----|----|----|----|----|----------|
| Activity summary | G | A | A | R | | | |
| Component | | | | | | | Comments |
| Screen handling procedures | G | A | A | G | | | |
| File update procedures | G | G | R | A | | | |
| Housekeeping procedures | G | G | G | A | | | |
| Compilation | G | G | G | R | | | |
| Test data runs | G | G | G | A | | | |
| Program documentation | G | G | A | R | | | |

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 e.g.
 - ◆ Work overtime
 - ◆ Re-allocate staff from less pressing work
 - ◆ Buy in more staff
- Reconsider activity dependencies
 - ◆ Over-lap the activities so that the start of one activity does not have to wait for completion of another
 - ◆ Split activities

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 - continued

- 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

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

Typical change control process

One or more users might perceive the need for a change

User management decide that the change is valid and worthwhile and pass it to development management

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 contd.

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

Software Configuration Management (SCM)

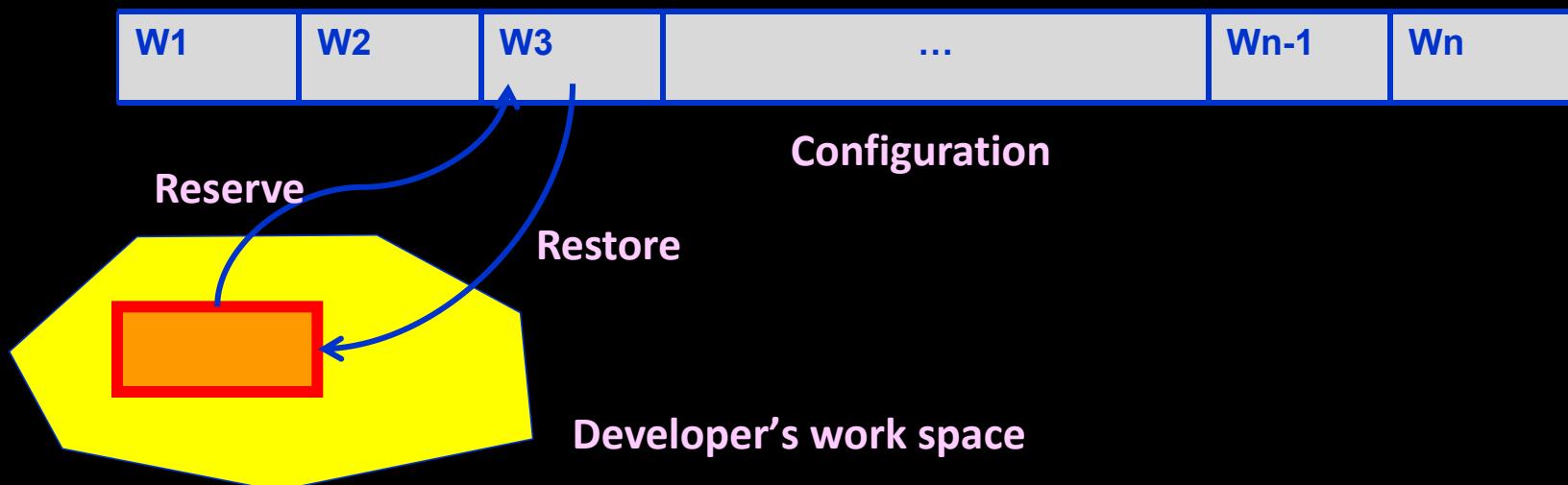
- SCM is concerned with tracking and controlling changes to a software.
- Development and maintenance environment:
 - ◆ Various work products associated with the software continually change.
 - ◆ Unless a proper configuration management system is deployed, several problems can appear.

Why Use SCM?

- **Problems associated with concurrent access**
- **Undoing Changes**
- **System accounting**
- **Handling variants**
- **Accurate determination project status**
- **Preventing unauthorized access to the work products**

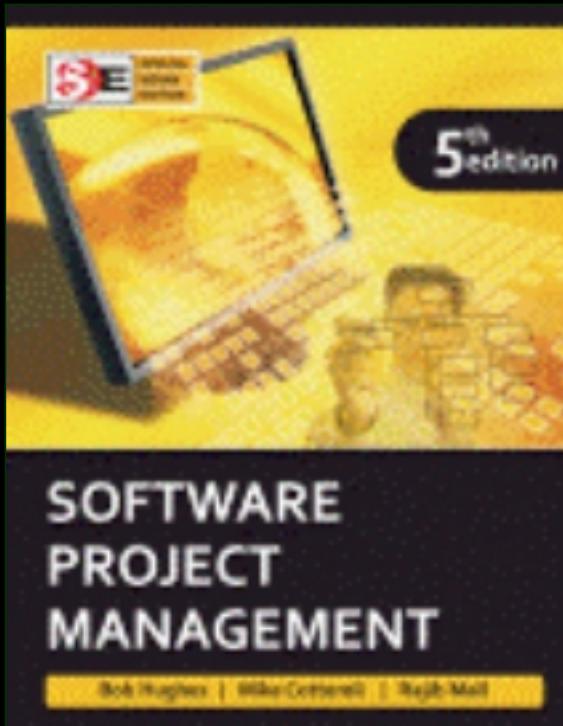
Configuration Control

- Two main operations:
 - ↳ Reserve
 - ↳ Restore



Software Project Management

Fifth Edition



Chapter 10

Contract management

Acquiring software from external supplier

This could be:

- a *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

- fixed price contracts
- time and materials contracts
- fixed price per delivered unit

Note difference between goods and services

Often licence to use software is bought rather than the software itself

Fixed price contracts

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

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

| <i>FP count</i> | <i>Design cost/FP</i> | <i>implementation cost/FP</i> | <i>total cost/FP</i> |
|-----------------|-----------------------|-------------------------------|----------------------|
| to 2,000 | \$242 | \$725 | \$967 |
| 2,001- 2,500 | \$255 | \$764 | \$1,019 |
| 2,501- 3,000 | \$265 | \$793 | \$1,058 |
| 3,001- 3,500 | \$274 | \$820 | \$1,094 |
| 3,501- 4,000 | \$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

Fixed price/unit

Disadvantages

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

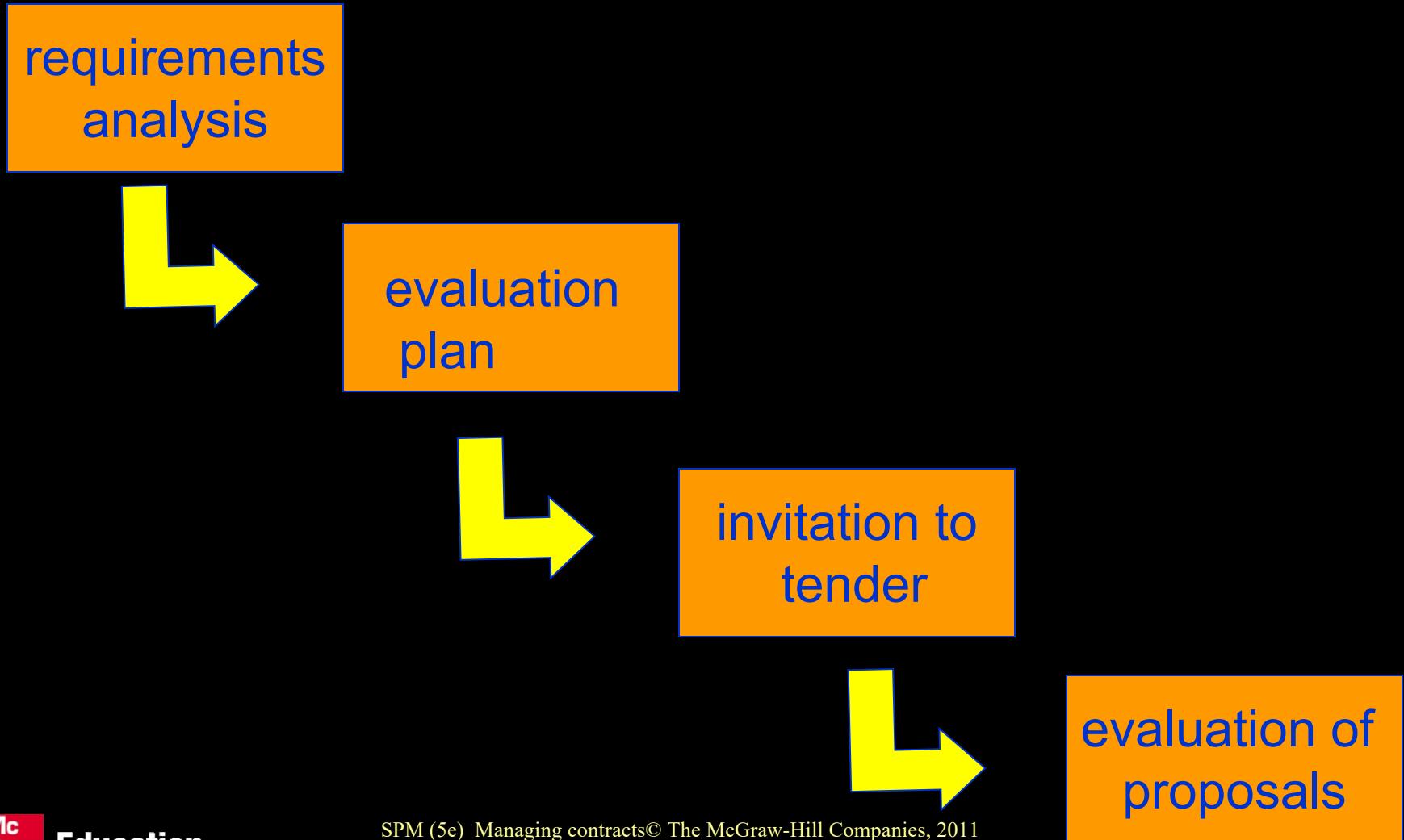
The tendering process

- 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, World Trade Organization, requirements)

The tendering process

- Restricted tendering process
 - ◆ bids only from those specifically invited
 - ◆ can reduce suppliers being considered at any stage
- Negotiated procedure
 - ◆ negotiate with one supplier e.g. for extensions to software already supplied

Stages in contract placement



Requirements document: sections

- introduction
- description of existing system and current environment
- future strategy or plans
- system requirements -
 - ◆ mandatory/desirable features
- deadlines
- additional information required from bidders

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

Evaluation plan

- How are proposals to be evaluated?
- Methods could include:
 - ◆ reading proposals
 - ◆ interviews
 - ◆ demonstrations
 - ◆ site visits
 - ◆ practical tests

Evaluation plan -contd.

- 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

Memoranda of agreement (MoA)

- Customer asks for technical proposals
- Technical proposals are examined and discussed
- Agreed technical solution in MoA
- Tenders are then requested from suppliers based in MoA
- Tenders judged on price
- Fee could be paid for technical proposals by customer

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 - continued

- Ownership of software
 - ◆ Can client sell software to others?
 - ◆ Can supplier sell software to others? Could specify that customer has 'exclusive use'
 - ◆ Does supplier retain the copyright?
 - ◆ Where supplier retains source code, may be a problem if supplier goes out of business; to circumvent a copy of code could be deposited with an **escrow** service

Contract checklist - continued

- Environment – for example, where equipment is to be installed, who is responsible for various aspects of site preparation e.g. electricity supply?
- Customer commitments – for example providing access, supplying information
- Standards to be met

Contract management

Some terms of contract will relate to management of contract, for example,

- Progress reporting
- Decision points – could be linked to release of payments to the contractor
- Variations to the contract, i.e. how are changes to requirements dealt with?
- Acceptance criteria

How would you evaluate the following?

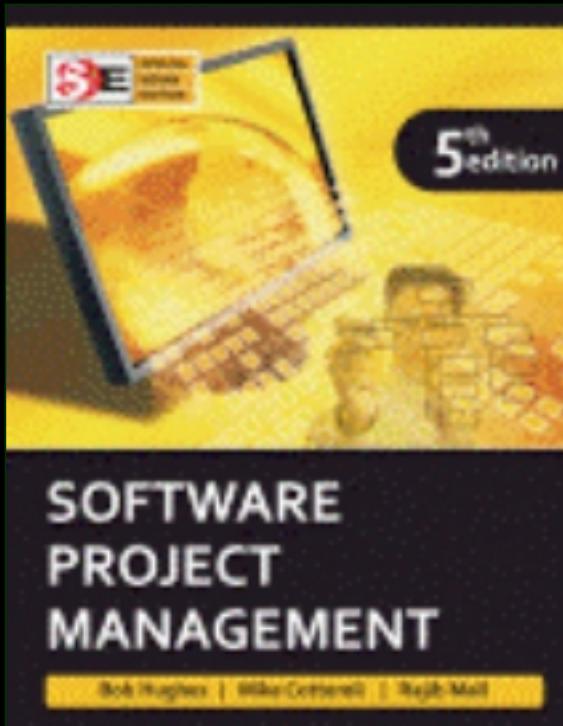
- usability of an existing package
- usability of an application yet to be built
- maintenance costs of hardware
- time taken to respond to requests for software support
- training

Contract management

- Contracts should include agreement about how customer/supplier relationship is to be managed e.g.
 - ◆ *decision points* - could be linked to payment
 - ◆ *quality reviews*
 - ◆ *changes to requirements*

Software Project Management

Fifth Edition



Chapter 11

**Managing people in
software
environments**

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 piece work
- 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
- 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.

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

Create a job specification.

Content includes types of task to be carried out.

Create a job holder profile

Describes the characteristics of the person who could do the job

Obtain applicants

Identify the media that potential job holders are likely to consult. Elicit CVs

A selection process - continued

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

Hygiene or maintenance factors – make you dissatisfied if they are not right e.g. pay, working conditions

Motivators – make you feel the job is worthwhile e.g. a sense of achievement

Vroom

Vroom and colleagues identified three influences on motivation

Expectancy – the belief that working harder leads to better performance

Instrumentality – the belief that better performance will be rewarded

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.

Health and safety - continued

- Top management must be committed to health and safety (H&S) policy
- Delegation of responsibilities relating to H&S should be clear
- Job descriptions should include H&S related responsibilities
- Need to ensure those given H&S responsibilities should understand and accept them

Health and safety - continued

- There should be a designated safety officer
- Staff, particularly knowledgeable technical specialists, should consulted about safety
- There should be an adequate H&S budget

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

Organizational ethics

There are some who argue that ethical 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

Exercise

Identify some of the possible objections and criticisms that can be made of the stockholder business ethics model described above.

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.

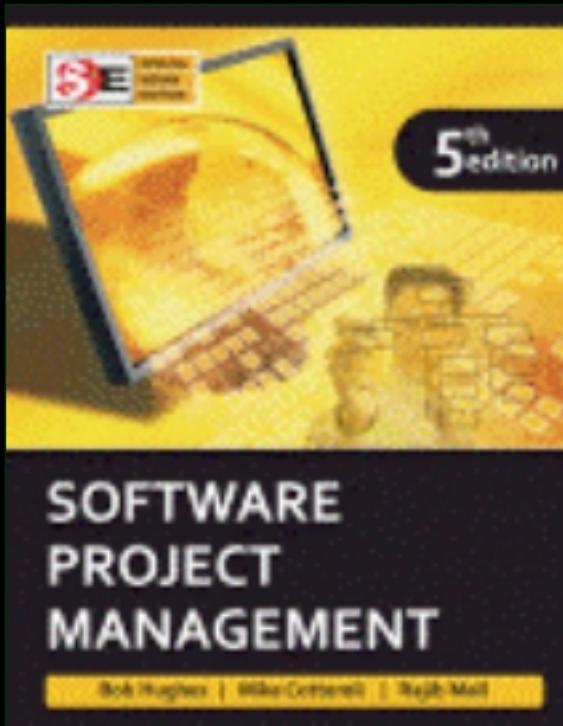
<<http://www.bcs.org/upload/pdf/cop.pdf>>

<<http://www.ieee.org/web/aboutus/ethics>>

<http://www.apm.org/about/se_code>

Software Project Management

Fifth Edition



Chapter 12

Working in teams

Becoming a team

Five basic stages of development:

- Forming
- Storming
- Norming
- Performing
- Adjourning

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 ‘Apollo’ teams – almost invariably did badly
- Identified the need for a balance of skills and management roles in a successful team

Management team roles

- The co-ordinator – 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

Belbin management roles - continued

- 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

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 - continued

- *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

- 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

Enlist co-operation of experts

Moderator presents experts with problem

Experts send in their recommendations to the moderator

Recommendations are collated and circulated to all experts

Experts comment on ideas of others and modify their own recommendation if so moved

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’

Organization and Team Structures

- Two important issues that are critical to the effective functioning of every organization are:
 - ◆ **Department structure:** How is a department organized into teams?
 - ◆ **Team structure:** How are the individual project teams structured?

Department Structure

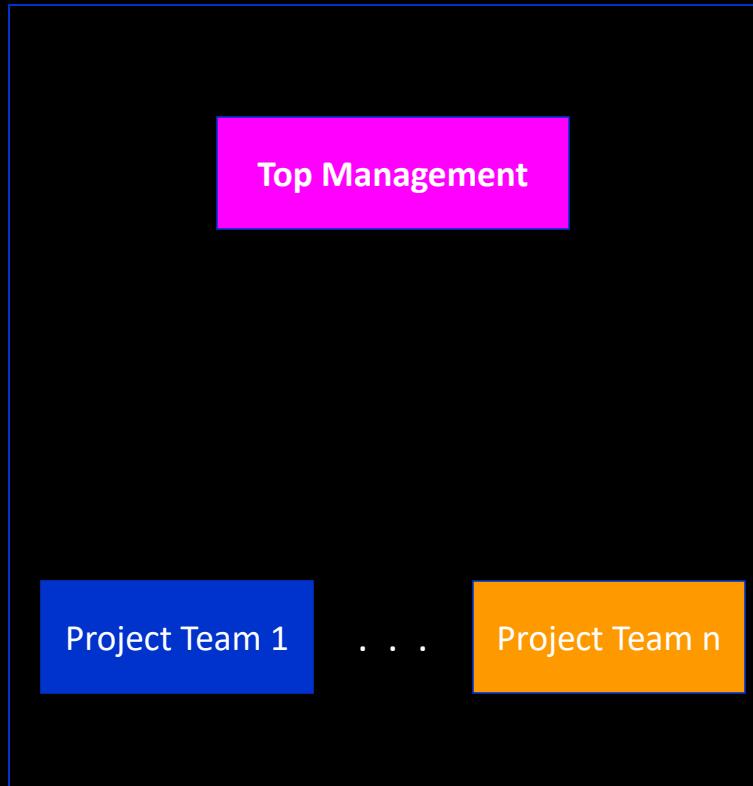
- **Functional format:**

- ◆ Each functional group comprises of developers having expertise in some specific task or functional area.

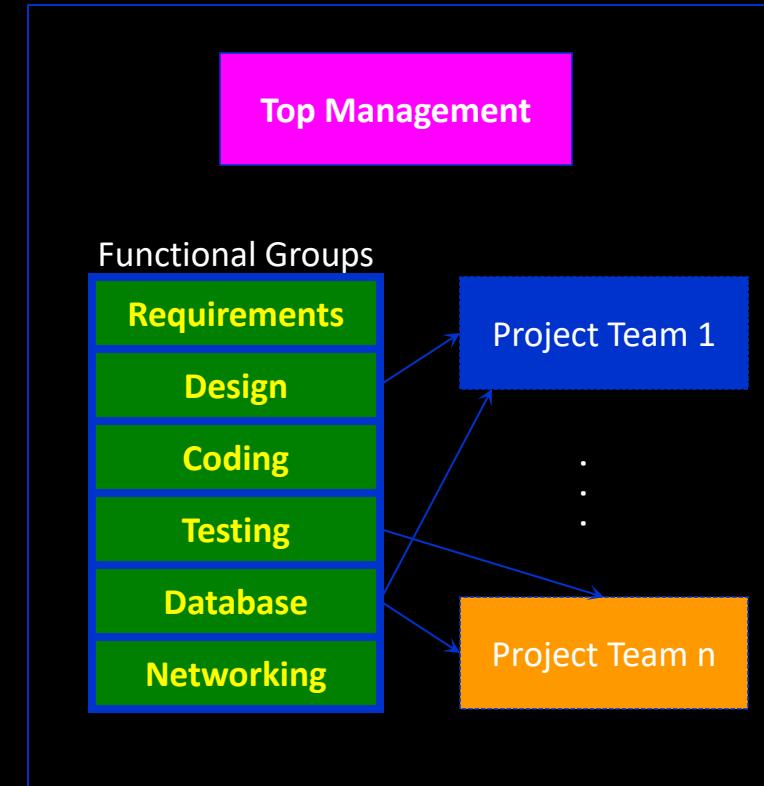
- **Project format:**

- ◆ The same team carries out all the project activities.

Functional and Project Formats



(a) Project Organization



(b) Functional Organization

Functional versus project formats

- Ease of staffing
- Production of good quality documents
- Job specialization
- Efficient handling of the problems associated with manpower turnover
- Career planning

Matrix Format

- The pool of functional specialists are assigned to different projects as needed.

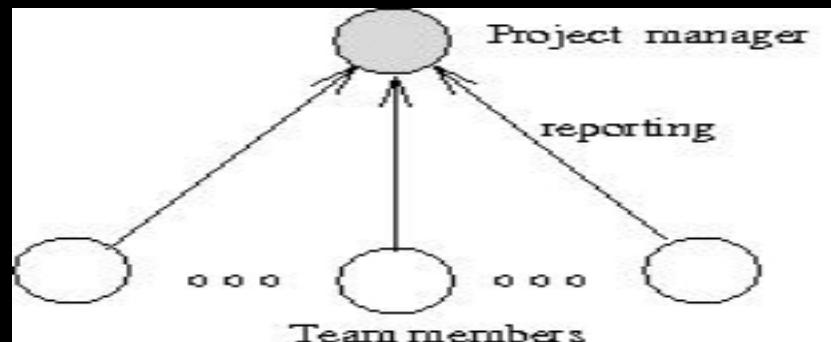
| | | Project | | | |
|------------------|-------------------|-------------------|-------------------|----|----------------------|
| Functional group | | #1 | #2 | #3 | |
| #1 | 2 | 0 | 3 | | Functional manager 1 |
| #2 | 0 | 5 | 3 | | Functional manager 2 |
| #3 | 0 | 4 | 2 | | Functional manager 3 |
| #4 | 1 | 4 | 0 | | Functional manager 4 |
| #5 | 0 | 4 | 6 | | Functional manager 5 |
| | Project manager 1 | Project manager 2 | Project manager 3 | | |

Team Structure

- We consider only three team structures:
 - ◆ Democratic,
 - ◆ Chief programmer,
 - ◆ Mixed team

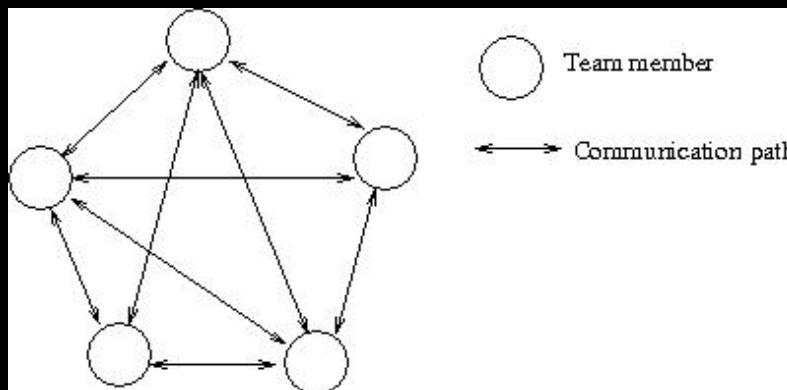
Chief programmer teams

- 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 programmer role



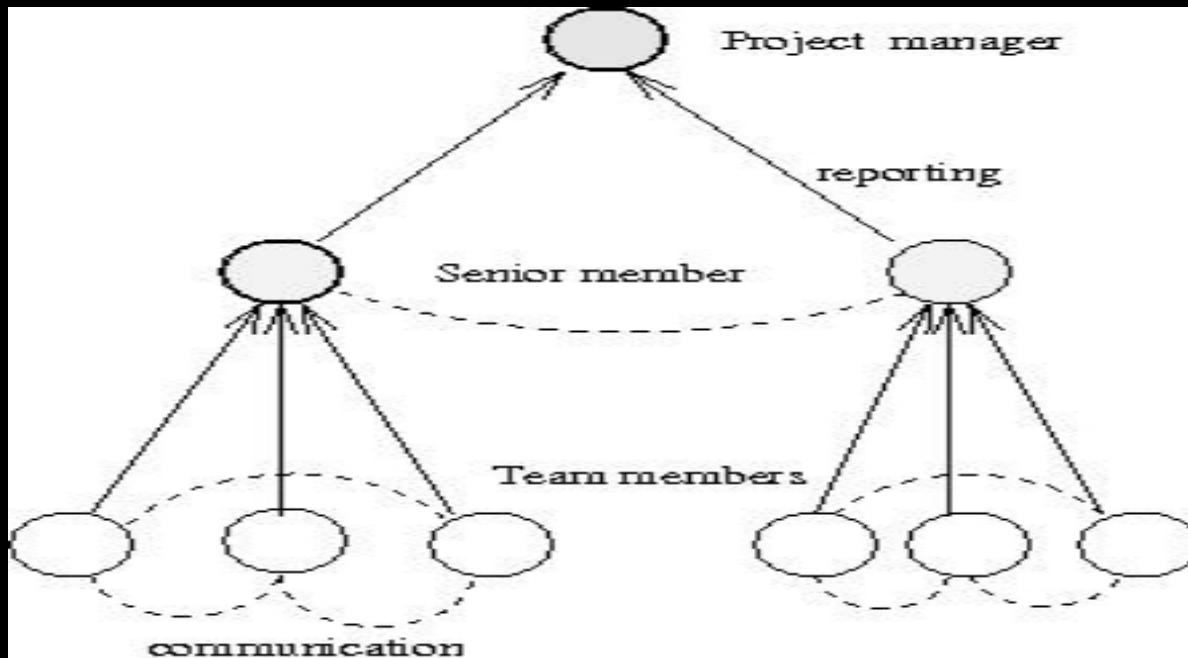
Democratic Team

- Does not enforce any formal team hierarchy.
- Decisions are taken based on discussions,
 - any member is free to discuss with any other member
- Since a lot of debate and discussions among the team members takes place,
 - for large team sizes significant overhead is incurred



Mixed Control Team Structure

- Incorporates both hierarchical reporting and democratic set up.



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
- ‘Sprints’ increments of typically one to four weeks
- Daily ‘scrum’ – daily stand-up meetings of about 15 minutes

Scrum - continued

- 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

Co-ordination of dependencies

- The previous discussion on team heedfulness focused (mainly) in communication inside the team
- What sort of communications are needed between teams and other units
- Co-ordination theory has identified the following types of coordination:
 - ↳ *Shared resources.* e.g. where several projects need the services of scarce technical experts for certain parts of the project.
 - ↳ *Producer-customer ('right time') relationships.* A project activity may depend on a product being delivered first.
 - ↳ *Task-subtask dependencies.* In order to complete a task a sequence of subtasks have to be carried out.

Coordination of dependencies - continued

- ◆ *Accessibility ('right place') dependencies.* This type of dependency is of more relevance to activities that require movement over a large geographical area, but arranging the delivery and installation of IT equipment might be identified as such.
- ◆ *Usability ('right thing') dependencies.* Broader concern than the design of user interfaces: relates to the general question of *fitness for purpose*, e.g. the satisfaction of business requirements.
- ◆ *Fit requirements.* This is ensuring that different system components work together effectively.

Why ‘virtual projects’?

The physical needs of software developers (according to an IBM report):

- 100 square feet of floor space
- 30 square feet of work surface
- Dividers at least 6 feet high to muffle noise
- 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
- Co-ordination 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 times | 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

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

- 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 co-ordination 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
 - Document 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.
- *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 authority

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

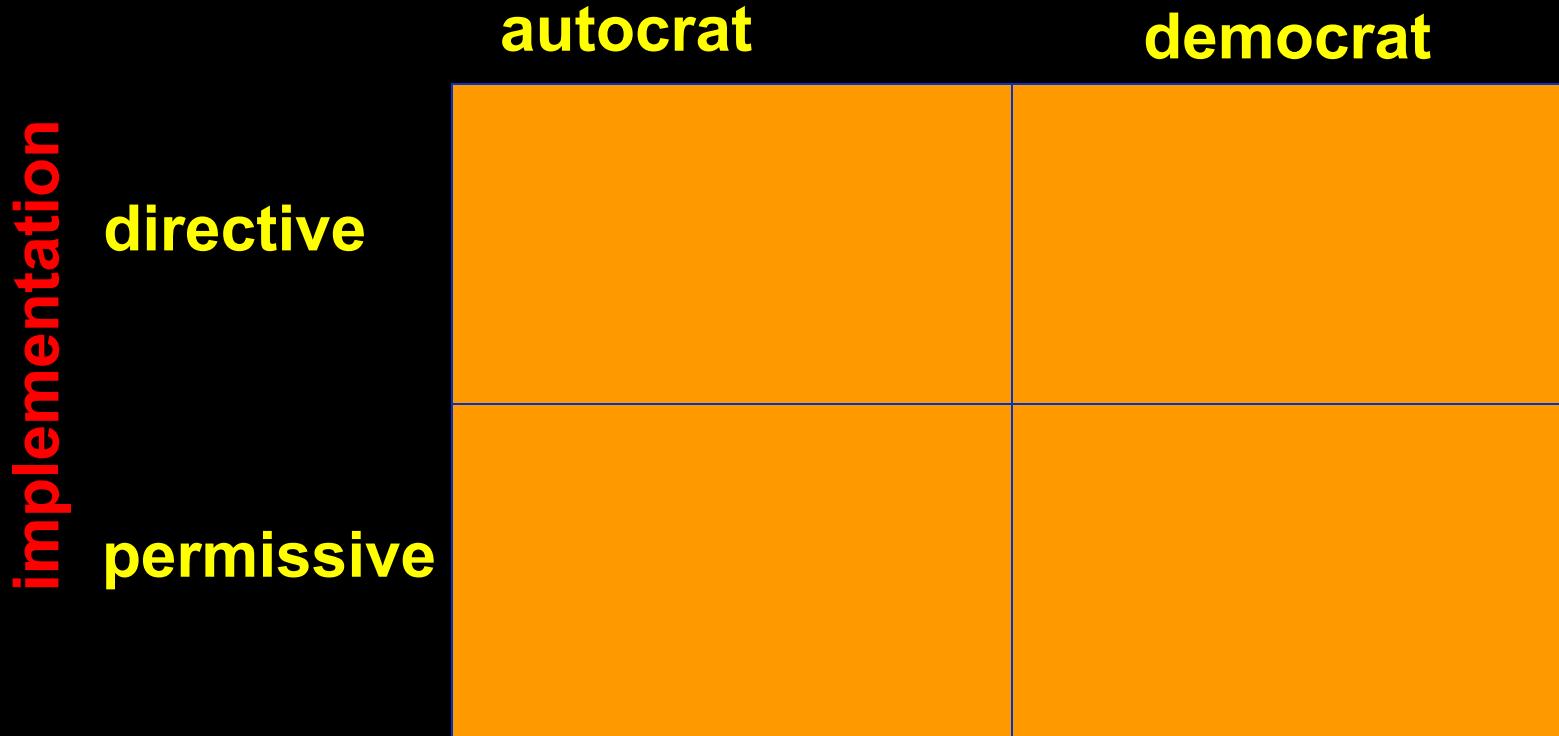
Leadership: types of power

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

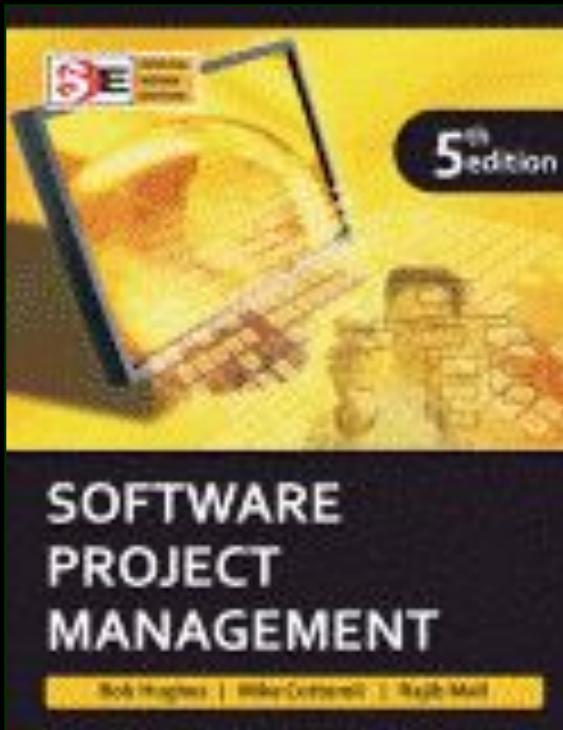


Leadership styles

- 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)

Software Project Management

Fifth Edition



Chapter 13.1

Software product quality

The importance of software quality

Increasing criticality of software

The intangibility of software

Project control concerns:

- errors accumulate with each stage

- errors become more expensive to remove the later they are found

- it is difficult to control the error removal process (e.g. testing)

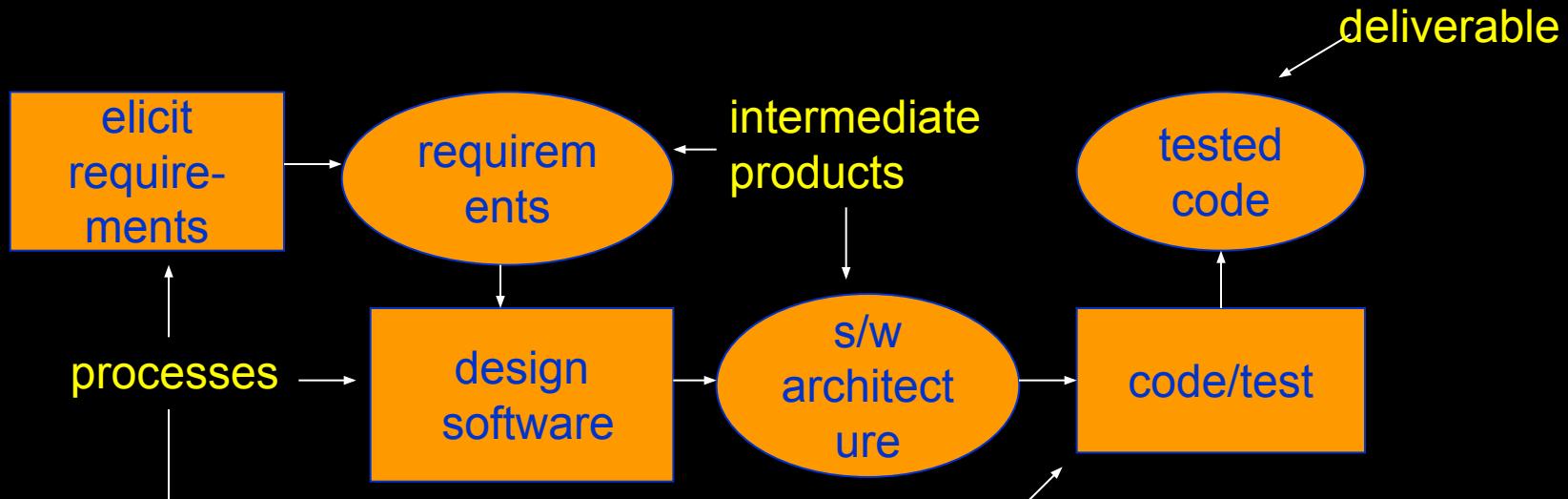
Quality specifications

Where there is a specific need for a quality, produce a quality specification

- Definition/description of the quality
- Scale: the unit of measurement
- Test: practical test of extent of quality
- Minimally acceptable: lowest acceptable value, if compensated for by higher quality level elsewhere
- Target range: desirable value
- Now: value that currently applies

ISO standards: development life cycles

A development life cycle (like ISO 12207) indicates the sequence of *processes* that will produce the software *deliverable* and the *intermediate products* that will pass between the processes.



ISO standards

ISO 9126 Software product quality

Attributes of software product quality

External qualities i.e. apparent to the user of the deliverable

Internal qualities i.e. apparent to the developers of the deliverables and the intermediate products

ISO 14598 Procedures to carry out the assessment of the product qualities defined in ISO 9126

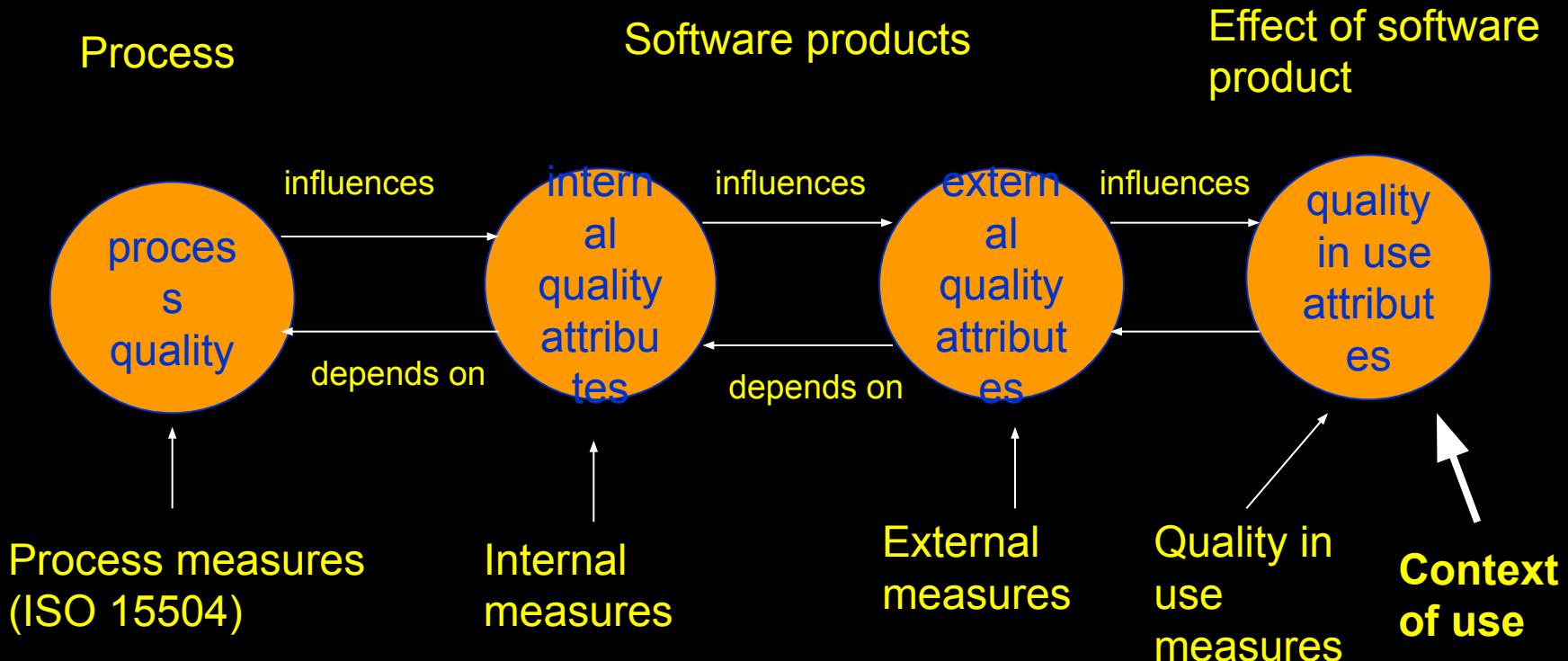
Types of quality assessment

During software development, to assist developers to build software with the required qualities

During software acquisition to allow a customer to compare and select the best quality product

Independent evaluation by assessors rating a software product for a particular community of users

ISO 9126 software product quality



Quality in use

Effectiveness – ability to achieve user goals with accuracy and completeness

Productivity – avoids excessive use of resources in achieving user goals

Safety – within reasonable levels of risk of harm to people, business, software, property, environment etc,

Satisfaction – happy users!

‘users’ include those maintain software as well as those who operate it.

ISO 9126 software qualities

functionality

does it satisfy user needs?

reliability

can the software maintain its level of performance?

usability

how easy is it to use?

efficiency

relates to the physical resources used during execution

Maintainability

relates to the effort needed to make changes to the software

portability

how easy can it be moved to a new environment?

Sub-characteristics of Functionality

Suitability

Accuracy

Interoperability

ability of software to interact with other software components

Functionality compliance

degree to which software adheres to application-related standards or legal requirements e.g audit

Security

control of access to the system

Sub-characteristics of Reliability

Maturity

frequency of failure due to faults - the more the software has been used, the more faults will have been removed

Fault-tolerance

Recoverability

note that this is distinguished from 'security' - see above

Reliability compliance

– complies with standards relating to reliability

Sub-characteristics of Usability

Understandability

easy to understand?

Learnability

easy to learn?

Operability

easy to use?

Attractiveness – this is a recent addition

Usability compliance

compliance with relevant standards

Sub-characteristics of Efficiency

Time behaviour

e.g. response time

Resource utilization

e.g. memory usage

Efficiency compliance

compliance with relevant standards

Sub-characteristics of Maintainability

“Analysability”

ease with which the cause of a failure can be found

Changeability

how easy is software to change?

Stability

low risk of modification having unexpected effects

“Testability”

Maintainability conformance

Sub-characteristics of portability

Adaptability

“Installability”

Co-existence

Capability of co-existing with other independent software products

“Replaceability”

factors giving ‘upwards’ compatibility - ‘downwards’ compatibility is excluded

Portability conformance

Adherence to standards that support portability

Using ISO 9126 quality standards (development mode)

Judge the importance of each quality for the application
for example, safety critical systems - *reliability* very important
real-time systems - *efficiency* important

Select relevant external measurements within ISO 9126 framework for these qualities, for example
mean-time between failures for reliability
response-time for efficiency

Using ISO 9126 quality standards

map measurement onto ratings scale to show degree of user satisfaction – for example response time

| response (secs) | rating |
|-----------------|----------------------|
| <2 | Exceeds requirement |
| 2-5 | Target range |
| 6-10 | Minimally acceptable |
| >10 | Unacceptable |

Using ISO 9126 quality standards

Identify the relevant internal measurements and the intermediate products in which they would appear
e.g. at software design stage the estimated execution time for a transaction could be calculated

Using ISO9126 approach for application software selection

Rather than map engineering measurement to qualitative rating, map it to a score

Rate the importance of each quality in the range 1-5

Multiply quality and importance scores – see next slide

Weighted quality scores

| | | Product A | | Product B | |
|------------------|--------------------------|----------------------|---------------------------|----------------------|---------------------------|
| Product quality | Importance rating (a) | Quality score (b) | Weighted score (a x b) | Quality score (c) | Weighted score (a x c) |
| usability | 3 | 1 | 3 | 3 | 9 |
| efficiency | 4 | 2 | 8 | 2 | 8 |
| maintain-ability | 2 | 3 | 6 | 1 | 2 |
| Overall totals | | | 17 | | 19 |

How do we achieve product quality?

the problem: quality attributes tend to *retrospectively* measurable

need to be able to examine processes by which product is created beforehand

the production process is a network of sub-processes
output from one process forms the input to the next
errors can enter the process at any stage

Correction of errors

Errors are more expensive to correct at later stages
need to rework more stages
later stages are more detailed and less able to absorb change

Barry Boehm

Error typically 10 times more expensive to correct at coding stage than at requirements stage
100 times more expensive at maintenance stage

For each activity, *define*:

Entry requirements

these have to be in place before an activity can be started

example: ‘a comprehensive set of test data and expected results be prepared and independently reviewed against the system requirement before program testing can commence’

For each activity, **define**

Implementation requirements

these define how the process is to be conducted
example ‘whenever an error is found and corrected, *all* test runs must be completed, including those previously successfully passed’

For each activity, *define*

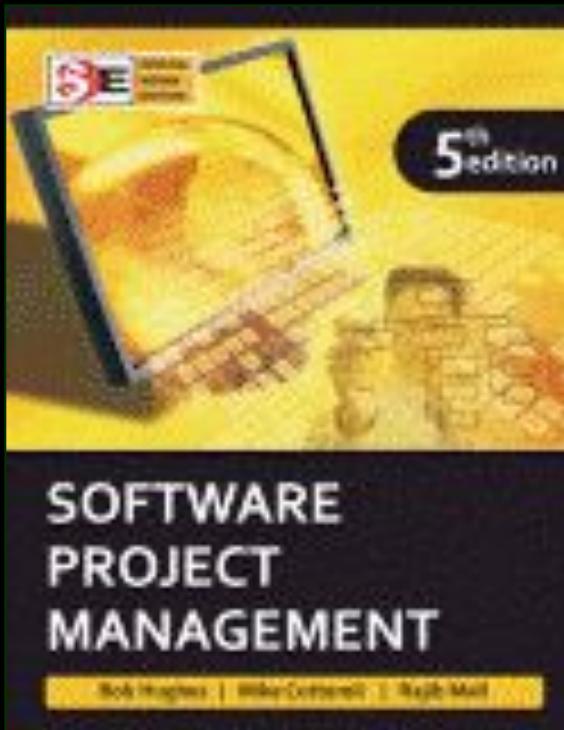
Exit requirements

an activity will not be completed until these requirements have been met

example: ‘the testing phase is finished only when all tests have been run in succession with no outstanding errors’

Software Project Management

Fifth Edition



Chapter 13.2

Software process quality

BS EN ISO 9001:2000 and quality management systems

ISO 9001 is one of a family of standards that specify the characteristics of a good quality management system (QMS)

Can be applied to the creation of any type of product or service, not just IT and software

Does NOT set universal product/service standards
DOES specify the way in which standards are established and monitored

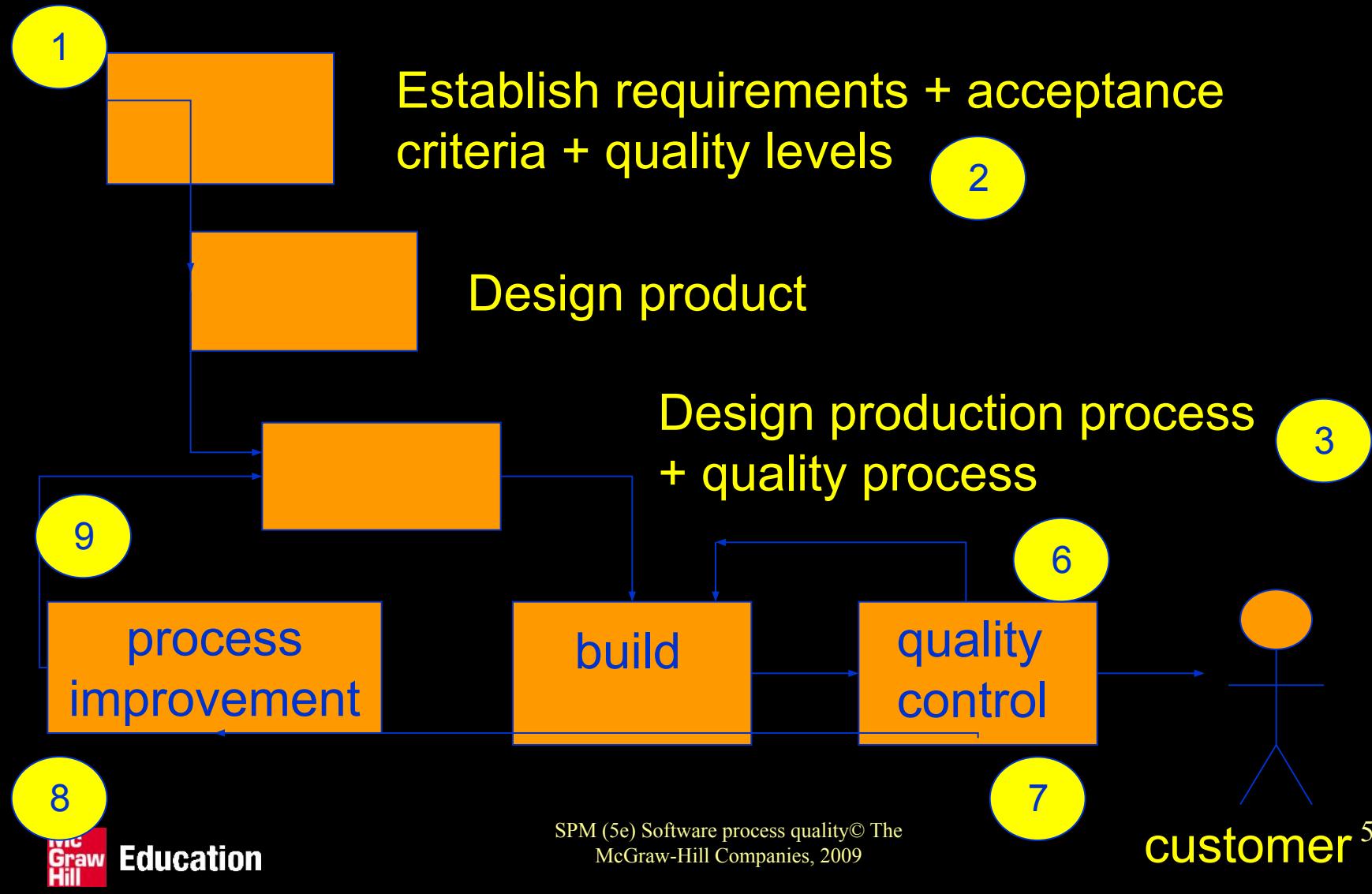
ISO 9001:2000 principles

1. Understanding the requirements of the customer
2. Leadership to provide unity of purpose and direction to achieve quality
3. Involvement of staff at all levels
4. Focus on the individual which create intermediate and deliverable products and services

ISO 9001:2000 principles

5. Focus on interrelation of processes that deliver products and services
6. Continuous process improvement
7. Decision-making based on factual evidence
8. Mutually beneficial relationships with suppliers

ISO 9001:2000 cycle



The need to improve

can everything be improved at one?

no, must tackle the most important things first

‘poor companies are poor at changing’

some later improvements build on earlier ones

but there are problems

improvement takes up time and money

‘improvement’ may simply be more bureaucracy!

Capability maturity model (CMM)

created by Software Engineering Institute, Carnegie Mellon University

CMM developed by SEI for US government to help procurement

Watts S. Humphrey 'Managing the software process'
Addison Wesley

Assessment is by questionnaire and interview

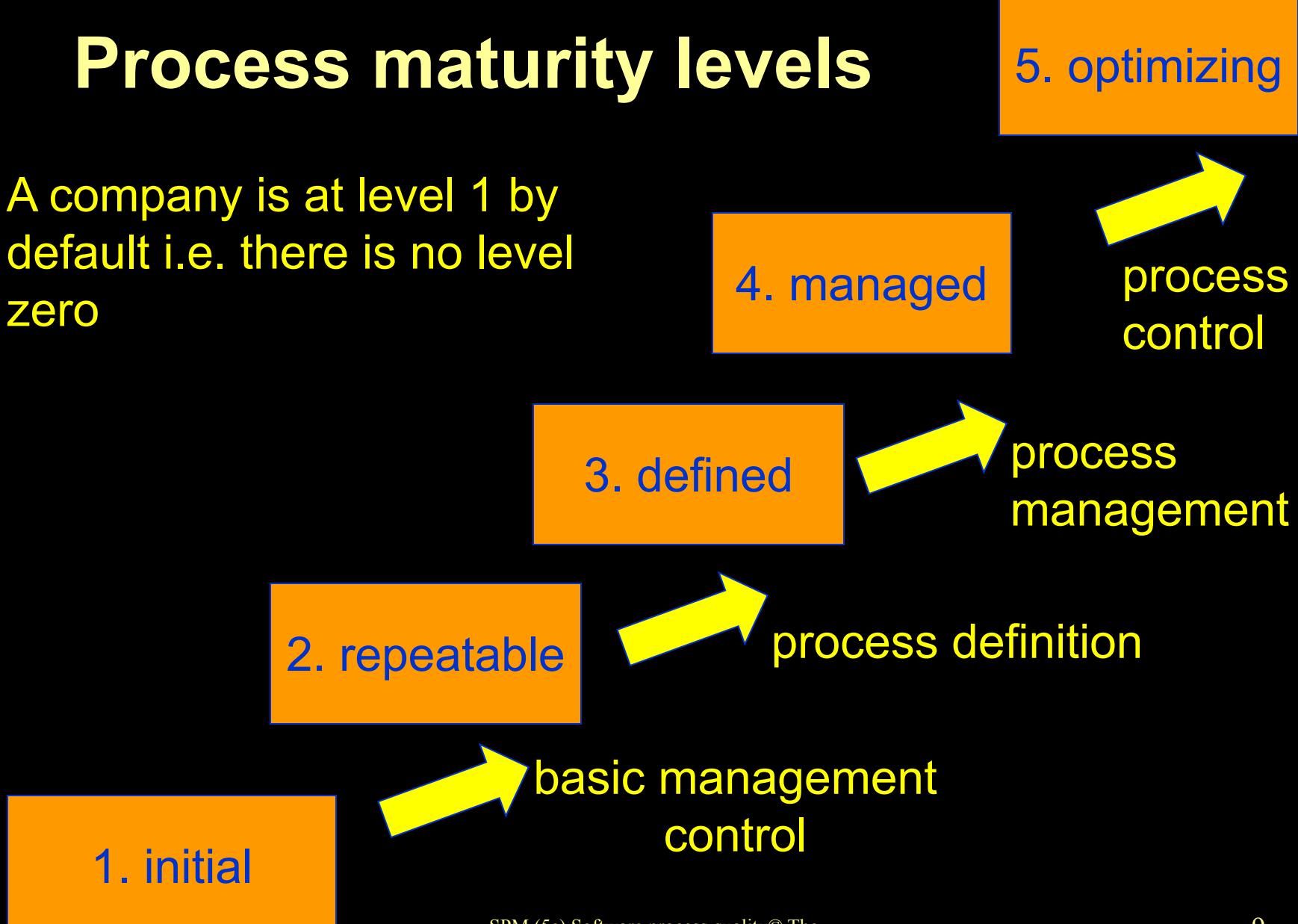
Capability maturity model 2

Different versions have been developed for different environments e.g. software engineering

New version CMMI tries to set up a generic model which can be populated differently for different environments

Process maturity levels

A company is at level 1 by default i.e. there is no level zero



Key process areas

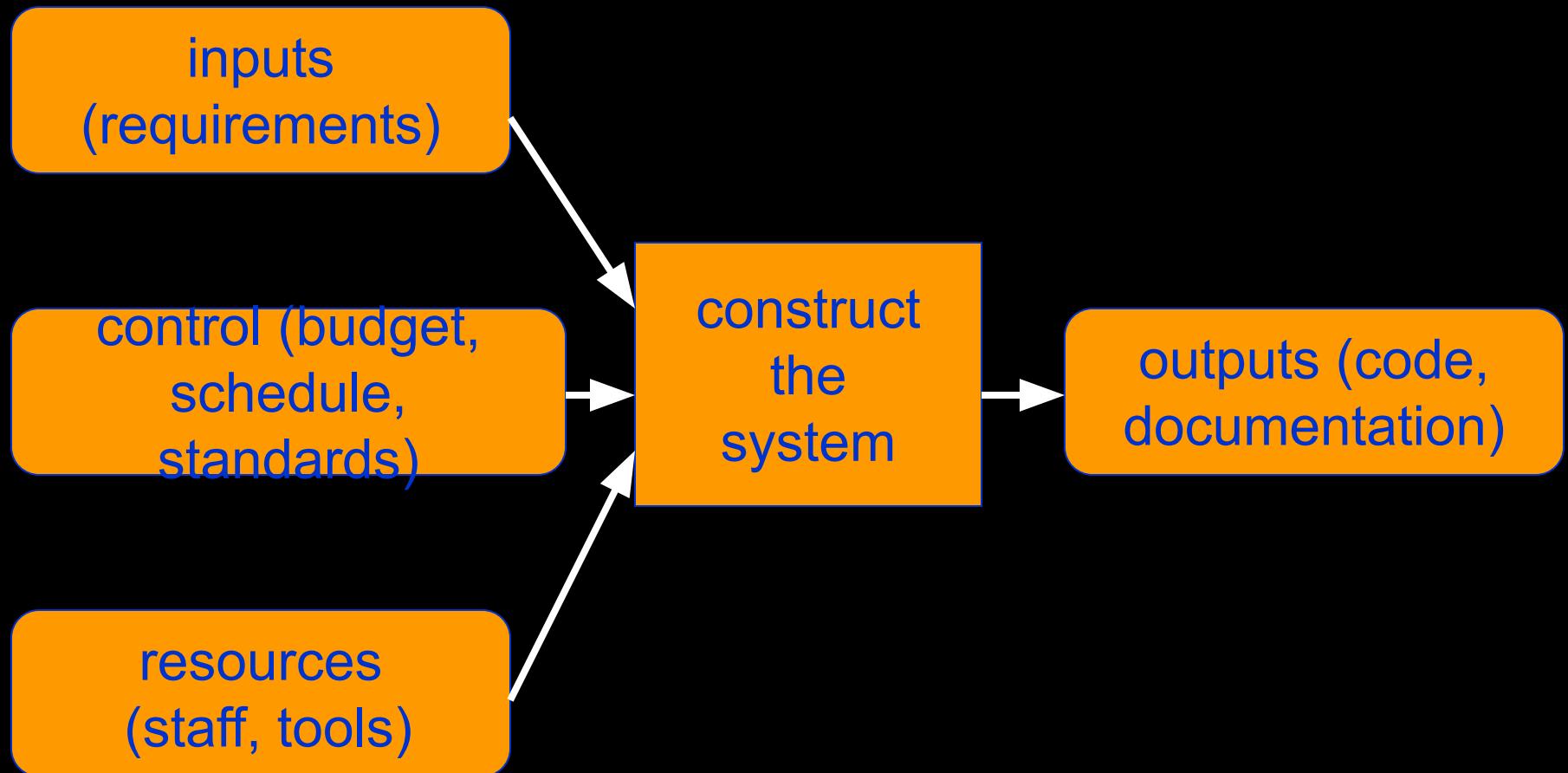
The KPAs of a level indicate the areas that an organization at the lower maturity level needs to focus to reach this level.

KPAs provide a way for an organization to gradually improve its quality of over several stages.

KPAs for each stage has been carefully designed such that one stage enhances the capability already built up.

Trying to focus on some higher level KPAs without achieving the lower level KPAs would be counterproductive.

A repeatable model



Repeatable model KPAs

To move to this level concentrate on:

Configuration management

Quality assurance

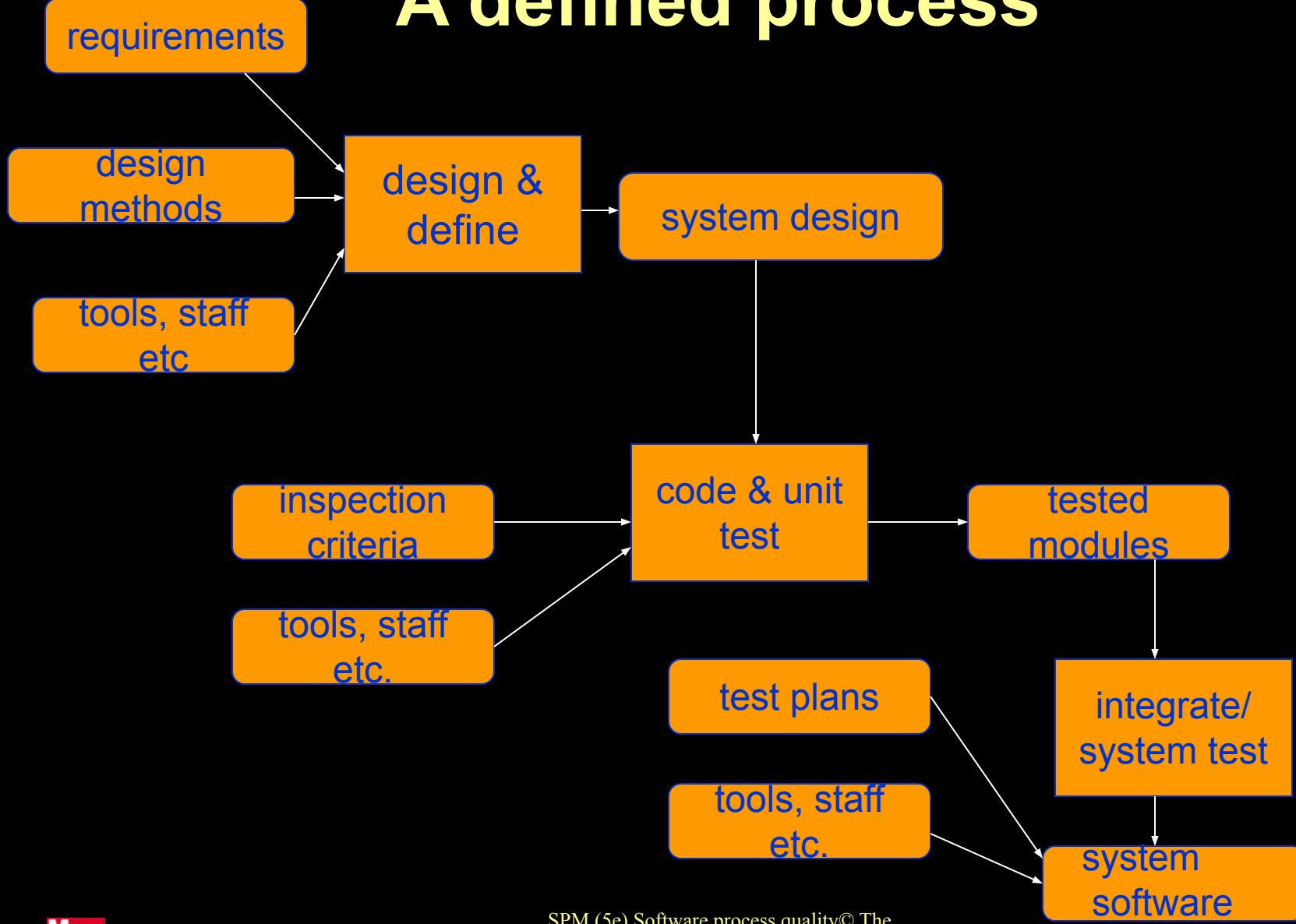
Sub-contract management

Project planning

Project tracking and oversight

Measurement and analysis

A defined process

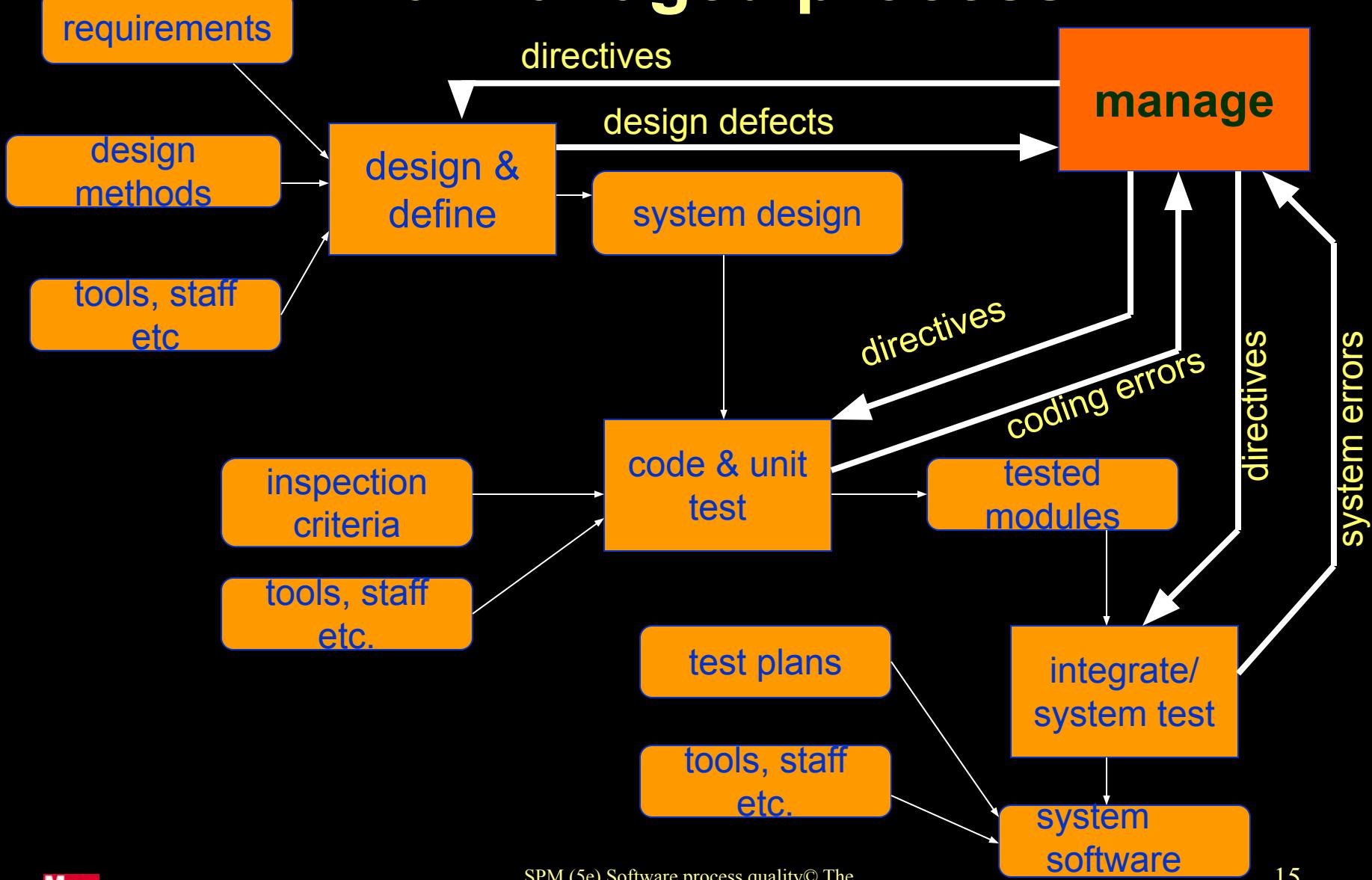


Repeatable to defined KPAs

Concentrate on

- Requirements development and technical solution
- Verification and validation
- Product integration
- Risk management
- Organizational training
- Organizational process focus (function)
- Decision analysis and resolution
- Process definition
- Integrated project management

a managed process



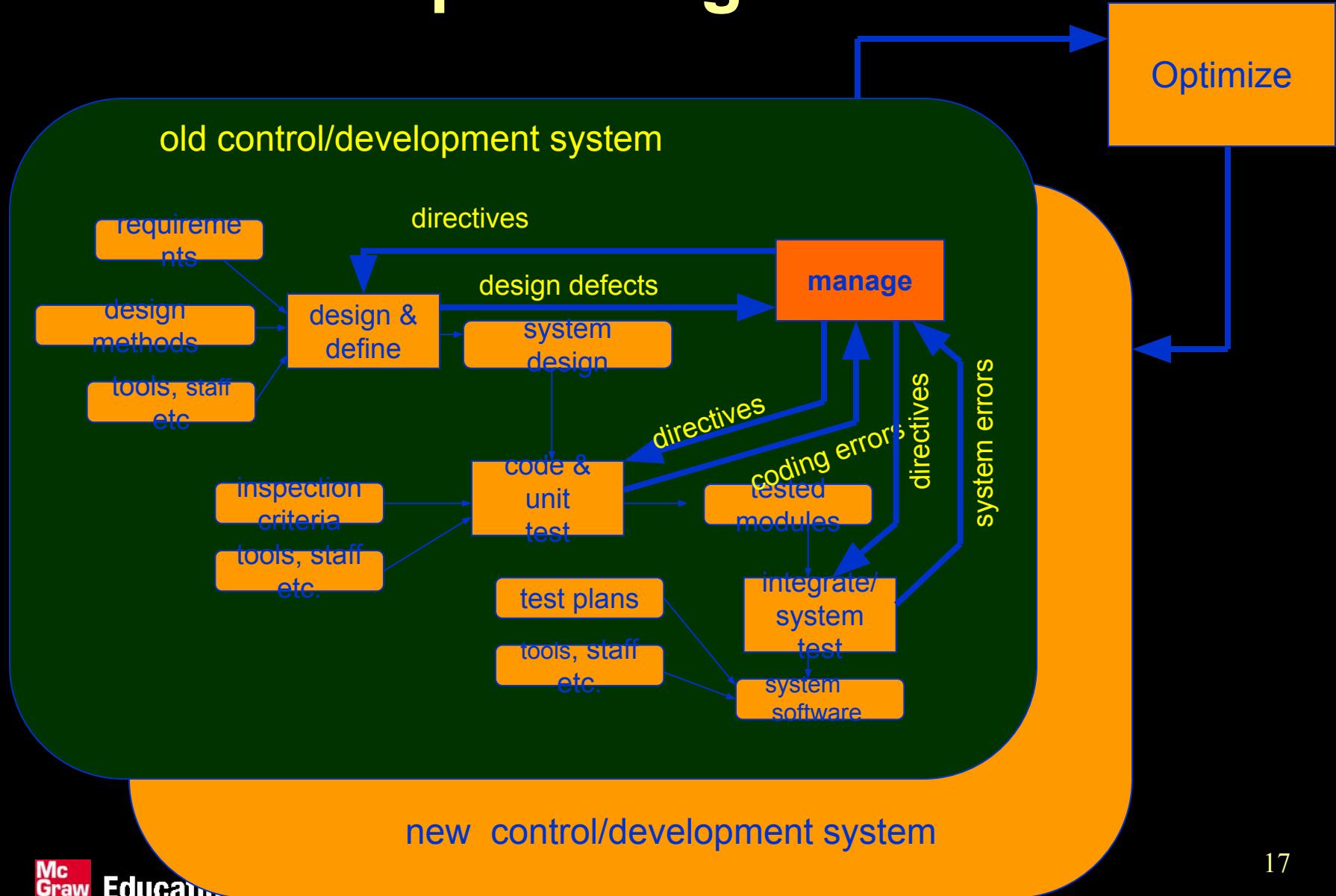
Defined to managed KPAs

Concentrate on:

Organizational process performance

Quantitative project management

Optimizing



Managing to optimizing: KPAs

concentrate on:

Causal analysis and resolution

Organizational innovation and deployment

CMMI (Capability Maturity Model Integration)

CMMI is the successor of the Capability Maturity Model (CMM).

After CMMI was first released in 1990:

- It became popular in many other domains

- Human Resource Management (HRM).: people management (PCMM)

- software acquisition (SA-CMM)

- systems engineering (SE-CMM)

Some questions about CMMI

suitable only for large organizations?

e.g. need for special quality assurance and process improvement groups

defining processes may not be easy with new technology

how can we plan when we've not used the development method before?

higher CMM levels easier with maintenance environments?

can you jump levels? (HP level 5 in India)

ISO/IEC 15504 IT process assessment

To provide guidance on the assessment of software development processes

Process Reference Model: Needs a defined set of processes that represent good practice to be the benchmark

ISO 12207 is the default reference model

Could use others in specific environments

ISO 15504 performance attributes

| CMMI level | ISO 15504 |
|------------|--|
| | 0. incomplete |
| initial | 1.1. process performance – achieves defined outcome |
| repeatable | 2.1 process management – it is planned and monitored |
| | 2.2 work product management – control of work products |

ISO 15504 performance attributes - contd

| CMMI | ISO 15504 |
|------------|---------------------------|
| Defined | 3.1. Process definition |
| | 3.2. Process deployment |
| Managed | 4.1. Process measurement |
| | 4.2. Process control |
| Optimizing | 5.1. Process innovation |
| | 5.2. Process optimization |

Process Reference Model

A defined standard approach to development

Reflects recognized good practice

A benchmark against which the processes to be assessed can be judged

ISO 12207 is the default model

ISO 15504 Process Assessment

For each process in the relevant Process Reference Model

For each set of attribute level criteria

Assess whether:

N: not achieved 0-15%

P: partially achieved >15%-50%

L: largely achieved >50%-85%

F: fully achieved >85%

ISO 15504 performance indicators

This is just an example of how indicators for each level
might be identified

1. Performance

Descriptions of maximum and minimum expected input values exist

2.1 Performance management

A plan of how expected input variable ranges are to be obtained exists which is up to date

ISO 15504 performance indicators 2

2.2 Work product management

There are minutes of a meeting where the input requirements document was reviewed and corrections were mandated

3.1 Process definition

A written procedure for input requirements gathering exists

3.2 Process deployment

A control document exists that is signed as each part of the procedure is completed

ISO 15504 performance indicators 2

4.1. Process measurement

Collected measurement data can be collected e.g.
number of changes resulting from review

4.2. Process control

Memos relating to management actions taken in the
light of the above

ISO 15504 performance indicators 3

5.1 Process innovation

Existence of some kind of 'lessons learnt' report at the end of project

5.2. Process optimization

Existence of documents assessing the feasibility of suggested process improvements and which show consultation with relevant stakeholders

Techniques to improve quality -Inspections

when a piece of work is completed, copies are distributed to co-workers

time is spent individually going through the work noting defects

a meeting is held where the work is then discussed
a list of defects requiring re-work is produced

Inspections - advantages of approach

an effective way of removing superficial errors from a piece of software

motivates the software developer to produce better structured and self-descriptive code

spreads good programming practice

enhances team-spirit

the main problem maintaining the commitment of participants

‘Clean-room’ software development

Ideas associated with Harlan Mills at IBM

Three separate teams:

Specification team – documents user requirements and usage profiles (how much use each function will have)

Development team – develops code but does not test it. Uses mathematical verification techniques

Certification team – tests code. Statistical model used to decide when to stop

Formal methods

Use of mathematical notations such as VDM and Z to produce unambiguous specifications

Can prove correctness of software mathematically (cf. geometric proofs of Pythagoras' theorem)

Newer approach use Object Constraint Language (OCL) to add detail to UML models

Aspiration is to be able to generate applications directly from UML+OCL without manual coding – Model Driven Architectures (MDA)

Verification versus Validation

Verification is the process of determining whether the output of one phase of software development conforms to that of its previous phase;

whereas validation is the process of determining whether a fully developed software conforms to its requirements specification.

Verification is carried out during the development process to check if the development activities are being carried out correctly,

whereas validation is carried out towards the end of the development process to check if the right product as required by the customer has been developed.

Testing: the V-process model

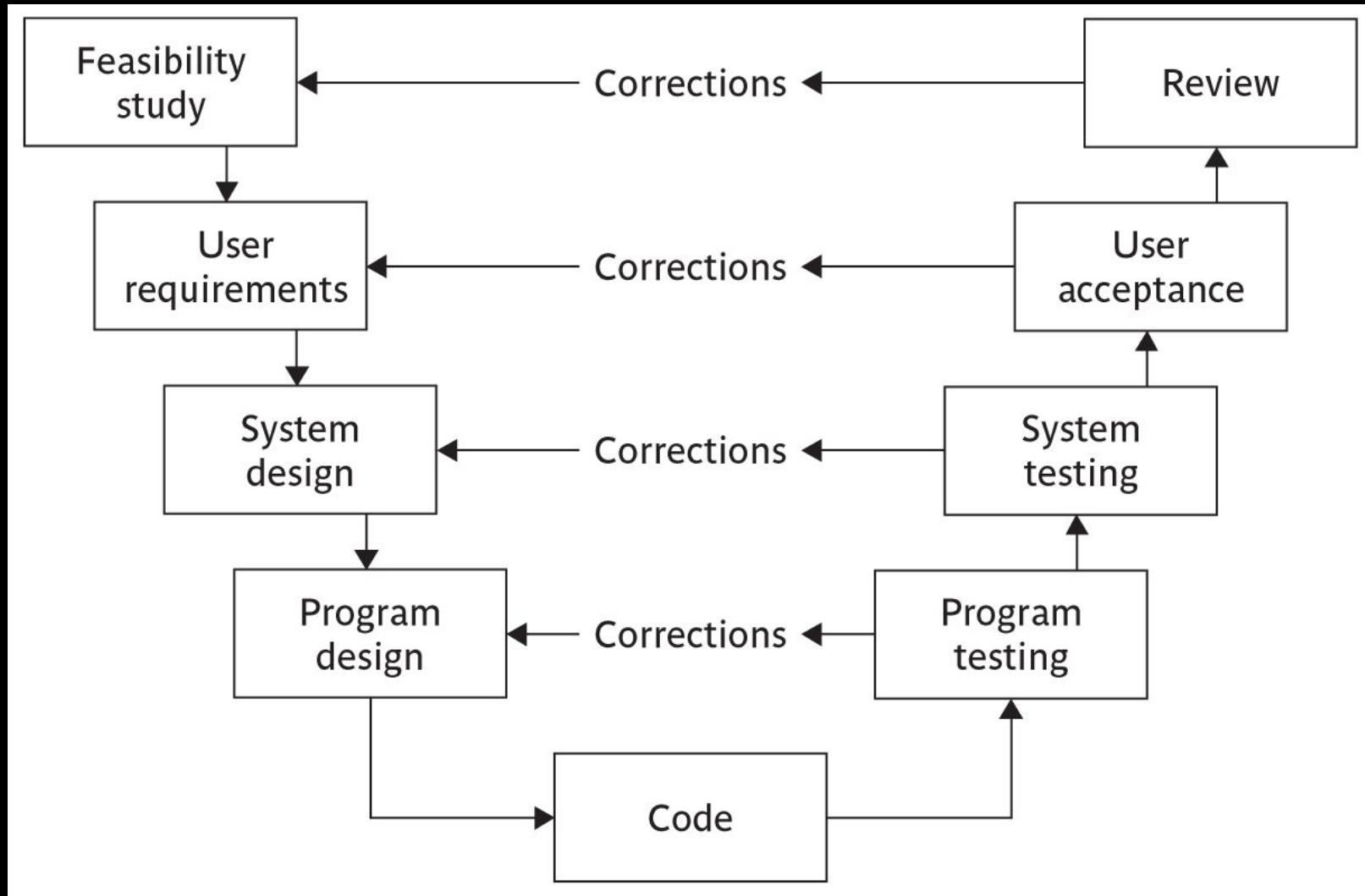
This shown diagrammatically on the next slide

It is an extension of the waterfall approach

For each development stage there is a testing stage

The testing associated with different stages serves different purposes e.g. system testing tests that components work together correctly, user acceptance testing that users can use system to carry out their work

Testing: the V-process model



Black box versus glass box test

Glass box testing

The tester is aware of the internal structure of the code; can test each path; can assess percentage test coverage of the tests e.g. proportion of code that has been executed

Black box testing

The tester is not aware of internal structure; concerned with degree to which it meets user requirements

Levels of testing

Unit testing

Integration testing

System testing

Testing activities

Test planning

Test suite design

Test case execution and result checking

Test reporting:

Debugging:

Error correction:

Defect retesting

Regression testing

Test closure:

Test plans

Specify test environment

In many cases, especially with software that controls equipment, a special test system will need to be set up

Usage profile

failures in operational system more likely in the more heavily used components

Faults in less used parts can lie hidden for a long time

Testing heavily used components more thoroughly tends to reduce number of operational failures

Management of testing

The tester executes test cases and may as a result find discrepancies between actual results and expected results – **issues**

Issue resolution – could be:

a mistake by tester

a fault – needs correction

a fault – may decide not to correct: **off-specification**

a change – software works as specified, but specification wrong:
submit to change control

Decision to stop testing

The problem: impossible to know there are no more errors in code

Need to estimate how many errors are likely to be left

Bug seeding – insert (or leave) known bugs in code

Estimate of bugs left =

$$\text{Estimate of bugs left} = \frac{\text{total errors found}}{\text{seeded errors found}} \times (\text{total seeded errors})$$

Alternative method of error estimation

Have two independent testers, A and B

N_1 = valid errors found by A

N_2 = valid errors found by B

N_{12} = number of cases where same error found by A and B

Estimate = $(N_1 \times N_2) / N_{12}$

Example: A finds 30 errors, B finds 20 errors. 15 are common to A and B. How many errors are there likely to be?

Test automation

Other than reducing human effort and time in this otherwise time and effort-intensive work,

Test automation also significantly improves the thoroughness of testing.

A large number of tools are at present available both in the public domain as well as from commercial sources.

Types of Testing Tools

Capture and playback

Automated test script

Random input test

Model-based test

Software reliability

The reliability of a software product essentially denotes its *trustworthiness* or *dependability*.

Reliability of a software product usually keeps on improving with time during the testing and operational phases as defects are identified and repaired.

A reliability growth model (RGM) models how the reliability of a software product improves as failures are reported and bugs are corrected.

An RGM can be used to determine when during the testing phase a given reliability level will be attained, so that testing can be stopped

Quality plans

quality standards and procedures should be documented in an organization's *quality manual* for each separate project, the quality needs should be assessed
select the level of quality assurance needed for the project and document in a *quality plan*

typical contents of a quality plan

scope of plan

references to other documents

quality management, including organization, tasks,
and responsibilities

documentation to be produced

standards, practices and conventions

reviews and audits

more contents of a quality plan

testing

problem reporting and corrective action

tools, techniques, and methodologies

code, media and supplier control

records collection, maintenance and retention

training

risk management