Software Project Management

Bachelor of Science Honours in Computing (University of South Africa)

Contents

Chapter 1 - Intro to Software Project Management

Introduction

We see that all projects are about meeting objectives. The project must satisfy real needs.

- We must identify the projects stakeholders and their objectives.

Why is Software Project Management Important?

1. The question of money. There is a lot of money at stake with ICT projects.
2. Most projects are late, and some exceed their budget.

What is a project?

- *Planned Activity*

The emphasis on being planned assumes we can determine how to carry out a task before we start. This is difficult with exploratory projects.

Planning is in essence thinking carefully about something before you do it.

Even with uncertain projects this is worth doing. Other activities, such as routine maintenance, will have been performed so many times that everyone knows exactly what to do.

The activities that benefit most from conventional project management lie somewhere between these two extremes.

The following characteristics distinguish projects:

- Non-routine tasks
- Planning is required
- Specific objectives are to be met or a specific product is to be created
- The project has a predetermined time span
- Work is carried out for someone other than yourself
- Work involves several specialisms
- People are formed into a temporary work group to carry out the task
- Work is carried out in several phases
- The resources that are available for use on the project is constrained
- The project is large or complex

Some argue that projects are especially problematic as they are temporarily sub-organizations. A group of people is brought together to carry out a single task. The existence of this sub-organization cuts across the authority of the existing units.

Software Projects versus other Types of Projects

Many Techniques in general project management also apply to software project management.

Characteristics of Software Projects that make them particularly difficult:

- **`Invisibility`** – When a physical artefact such as a bridge is constructed, the progress can actually be seen. With software, progress is not immediately visible.
- **`Complexity`** – Per dollar, software products contain more complexity than other engineering artefacts.

- **Conformity** – The 'traditional' engineer usually works with physical systems and materials like cement and steel. These physical systems have complexity, but are governed by consistent physical laws. Software developers have to conform to the requirements of human clients.
- **Flexibility** – That software is easy to change is seen as a strength. Where the software system interfaces with a physical or organizational system, it is expected that the software will change to accommodate the other components.

Contract Management and technical Project Management

Increasingly organizations contract out ICT development to outside developers. Here, the client organization will often appoint a 'project manager' to supervise the contract who will delegate many technically oriented decisions to the contractors.

On the supplier side there will be project managers to deal with the more technical issues.

Activates covered by software project management

1. **The Feasibility Study –** This assesses whether a project is worth starting – that it has a valid *business case.* The development and operational costs, and the value of the benefits of the new system, will also have to be estimated.

2. **Planning –** If the feasibility study indicates that the prospective project appears viable, then projects planning can start. We create an outline plan for the whole project, and a detailed one for the first step. Because we will have more detailed and accurate project information after the earlier stages of the project have been completed.

3. **Project Execution** – The execution of a project often contains *design and implementation* sub-phases.
    - Design is making decisions about the form of the *products* to be created.
    - The plan details the *activities* to be carried out to create these products.

    - **Requirement Analysis** – Starts with *requirements elicitation* or requirements gathering which establishes what the potential users and their managers require of the new system.
        - Function – That the system should do something
        - Quality is a measure well the functions must work
        - Training to ensure that operators use the computer system efficiently is a *system requirement,* as opposed to a *software requirement.*
        - There would also be *resource requirements* that relate to application development costs.

    - **Architecture Design** – The components of the new system that fulfil each requirement have to be identified. Existing components may be able to satisfy some requirements, in other cases new components have to be made.

      The design of the *system architecture* is this an input to the *software requirements.* A 2nd architecture design process then takes place that maps the software requirements to *software components.*

- **Detailed Design** – Each software component is made up of a number of software units that can be separately coded and tested. The detailed design of these units is carried out separately.
- **Code and test** – Refers to writing code for each software unit. Initial testing would be carried out at this stage.
- **Integration** – The components are tested together to see if they meet the overall system requirements.
- **Qualification testing** – The system, including the software components, has to be tested carefully to ensure that all the requirements have been fulfilled.
- **Installation** – This is the process of making the new system operational.
- **Acceptance Support** – This is the resolving of problems with the newly installed system, including the correction of any errors, and implementing agreed extensions and improvements.

Plans, Methods and Methodologies

A Plan for an activity must be based on some idea of a *method* of work. While a *method* relates to a type of activity, in general, a *plan* takes that method (and perhaps others) and converts it to real activities.

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

Some ways of categorizing software projects

Compulsory versus Voluntary Users

Use of a system is increasingly voluntary, as in the case of computer games. Here it is difficult to elicit precise requirements from potential users as we could with a business system.

What the game will do will thus depend on mush on the informed ingenuity of the developers, along with techniques such as market surveys, focus groups and prototype evaluation.

Informed Systems versus Embedded Systems

A traditional distinction has been between *information systems* which enable staff to carry out office processes and *embedded systems* which control machines.

Objectives versus Products

Products may be distinguished by whether their aim is to produce a *product* or to meet certain *objectives.*

Many software projects have two stages. First is an objective-driven project, resulting in recommendations. The next stage is a project actually to create the software product.

Stakeholders

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

Stakeholders can be categorized as:

- **Internal to the project team** – This means that they will be under the direct managerial control of the project leader.
- **External to the project team but within the same organization**. The project leader might need the assistance of the users to carry out systems testing.
- **External to both the project team and the organization** – External stakeholders may be customers, or users, who will benefit from the system.

Given the importance of coordinating the efforts of stakeholders, the recommended practice is for a *communication plan* to be created at the start of a project.

Setting Objectives

The owners of the project control the financing and set the objectives. The objectives should define what the project team must achieve pro project success.

Objectives focus on the desired outcomes, they are post conditions.

"The project will be a success if..."

There may be several stakeholders, including users in different business areas, who might have some claim to project ownership. In such a case, a *project authority* needs to be explicitly identified with overall authority over the project.

The authority is often a *project steering committee, project board* or *project management board.*

Sub-Objective Goals

An effective objective for an individual must be something that is within the control of that individual.

The mnemonic SMART is sometimes used:

1. **Specific** – Effective objectives are concrete and well defined.
2. **Measurable** – Ideally there should be *measures of effectiveness* which tell us how successful the project has been.
3. **Achievable** – It must be within the power of the individual or group to achieve
4. **Relevant** – The objective must be relevant to the true purpose of the project
5. **Time constrained** – There should be a defined point in time by which the objective should have been achieved

Measure of Effectiveness

'Mean time between failures (mtbf)' might be used to measure reliability. This is a *performance* measurement and can only be taken once the system is operational.

Managers want to get some idea of the performance of the completed system as it is being constructed. They will therefore seek *predictive* measures.

The business Case

The effort and expense of pushing the project through must be seen to be worthwhile in terms of the benefits that will eventually be felt. A cost-benefit analysis will often be part of the project's feasibility study.

The quantification of benefits will often require the formulation of a *business model* which explains how the new application can generate the claimed benefits.

Any project plan must ensure that the business case is kept intact:

- That development costs are not allowed to rise to a level which threatens to exceed the value of benefits.
- That the features of the system are not reduced to a level where the expected benefits cannot be realized.
- That the delivery date is not delayed so that there is an unacceptable loss of benefits.

Project Success and Failure

We can distinguish between *project objectives* and *business objectives.* The project objectives are the targets that the project team is expected to achieve.

In the case of software projects, they can usually be summarized as delivering:

- The agreed functionality
- To the required level of quality
- On time
- Within budget

We have seen that in business terms it can generally be said that a project is a success if the value of benefits exceeds the costs.

Customer Relationships can also be built over a number of projects.

What is Management?

It has been suggested that management involves the following activities:

- Planning – Deciding what is to be done
- Organization – making arrangements
- Staffing – selecting the right people for the job
- Directing – giving instructions
- Monitoring – checking on progress
- Controlling – taking action to remedy hold-ups
- Innovating – coming up with new solutions
- Representing – liaising with clients, users, developer, suppliers and other stakeholders

Management Control

Management in general involves setting objectives for the system and then monitoring the performance of the system.

This will involve the local managers in *data collection. Data processing* will be needed to transform raw *data* into useful *information.*

Example:
The project manager might examine the 'estimated completion date' for completing data transfer for each branch.

In effect, they are comparing actual performance with one aspect of the overall project objectives. Before going ahead with a solution, the project manager would need to first calculate the impact. This is *modelling* the consequences of a potential solution. Several different proposals could be modelled in this way before one was chosen for *implementation.*

Having implemented the decision, the situation needs to be kept under review by collecting and processing further progress details.

It can be seen that a project is dynamic and will need constant adjustment during the execution of the project. A good plan provides a foundation for a good project, but is nothing without intelligent execution.

Chapter 2 - Project Evaluation and Programme Management

A Business Case

Organizations may have different titles such as *feasibility study* or a *project justification* for what we call the business case. Its objective is to provide a rationale for the project by showing that the benefits of the project outcomes will exceed the costs of development, implementation and operation.

Typically a business case document might contain:

1. **Introduction and background** – Description of the current environment of the proposed project and the problem to be solved.
2. **The proposed Project** – Brief outline
3. **The market** – this is needed when the project is to create a new product or a new service capability. This would contain info like the estimated demand.
4. **Organizational and Operational Infrastructure** – This describes how the structure of the organization will be affected by the implementation of the project. This is most relevant where the project is implementing or modifying an information system.
5. **Benefits** – A financial value should be put on the benefits. This could be related to increased profits, or by making savings.
6. **Outline implementation plan** – In addition to ICT aspects, activities such as marketing, promotion and operational and maintenance infrastructures need to be considered. The responsibilities are allocated for the tasks identified in the outline implementation plan.
7. **Costs** – A schedule of expected costs associated with the planned approach can now be presented.
8. **Financial Case** – there are a number of ways in which the information on income and costs can be analysed.
9. **Risks** – We note here that many estimates of costs and benefits of the project will be speculative at this stage and the section of risk take account of this.

Project Portfolio Management

Portfolio project management provides an overview of all the projects that an organization is undertaking or is considering. It prioritizes the allocation of resources to projects.

The concerns of project portfolio management include:

- Identifying which proposals are worth implementing
- Assessing the amount of risk of failure that a potential project has
- Deciding how to share limited resources
- Being aware of the dependencies between projects
- Ensuring that projects do not duplicate work
- Ensuring that necessary developments have not been inadvertently been missed

The 3 key aspects of portfolio management are:

1. Project Portfolio Definition

An organization should record, in a single repository, details of all current projects. One problem is that projects can be divided into *new product developments (NPD),* where the project deliverable is a product. *Renewal* projects improve the way an organization operates.

2. Project Portfolio Management

More detailed costings of projects can be recorded. The value that managers hope will be generated by each project can also be recorded. Actual performance of projects on these performance indicators can then be tracked.

3. Project Portfolio Optimization

The performance of the portfolio can be tracked by high level managers on a regular basis. A better balance of projects may be achieved. Some projects could potentially be very profitable but could also be risky. Other projects could have modest benefits, but have fewer risks.

Some problems with Project Portfolio Management

An important role of a project portfolio is sharing resources between projects. The official project portfolio may not accurately reflect organizational activity if some are excluded. A formal decision may be made that only projects over a certain level of cost will be recorded.

Evaluation of Individual Projects

Technical assessment

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

Cost-Benefit analysis

Cost-benefit analysis comprises two steps:

1. Identifying all of the costs and benefits of carrying out the project and operating the delivered application
2. Expressing these costs and benefits in common units

Most direct costs are easy to quantify in monetary terms and can be categorized as:

- Development costs
- Setup costs
- Operational costs

Cash Flow Forecasting

Indicates when expenditure and income will take place. We need to spend money during the project's development. We need to know that we can fund this development expenditure either from the company's own resources or by borrowing. A forecast is needed of when expenditure, such as payment of salaries and any income are expected.

When estimating future cash flows, it is usual to ignore the effects of inflation.

Cost-Benefit Evaluation Techniques

We now look at some methods for comparing projects on the basis of their cash flow forecasts:

1. Net Profit

The net profit is the difference between the total costs and the total income over the life of the project. The simple net profit takes no account off the timing of the cash flows.

2. Payback period

The payback period is the time taken to break even or pay back the initial investment. The project with the shortest payback period will be chosen. It is simple to calculate and is not particularly sensitive to small forecasting errors. It ignores the overall profitability.

3. Return on investment

ROI (also known as the *accounting rate of return (ARR))* provides a way of comparing the net profitability to the investment required.

$$ROI = \frac{average\ annual\ profit}{total\ investment} \times 100$$

It takes no account of the timing of the cash flows. This rate of return bears no relationship to the interest rates offered or charged by banks.

4. Net present Value

The net present value takes into account the profitability of a project and the timing of the cash flows that are produced.

$$Present\ Value = \frac{value\ in\ year\ t}{(1+r)^t}$$

Where r is the discount rate, expressed as a decimal value.

Table 2.2 – p. 31 – NPV discount factors

The main difficulty with NPV for deciding between projects is selecting an appropriate discount rate. The exact discount rate is normally less important than ensuring that the same discount rate is used for all projects.

Alternatively, the discount rate can be thought of as a target rate of return.

5. Internal Rate of Return

NPV might not be directly comparable with earnings from other investments or the cost of borrowing capital. Such cotes are usually quoted as a percentage interest rate.

Internal rate of return (IRR) attempts to provide a profitability measure as a percentage return that is directly comparable with interest rates.

A project with an estimated IRR of 10% would be worthwhile if the capital could be borrowed for less than 10%.

NPV and IRR are not, however, a complete answer to economic project evaluation.

- Total evaluation must also take into account the problems of funding the cash flows.
- While a project's IRR might indicate a profitable project, future earnings from a relatively risky project might be far less reliable than, say, investing in a bank.
- We must also consider any one project within the financial and economic framework of the organization as a whole – if we fund this one, will we also be able to fund other worthy projects.

Risk Evaluation

1. Risk Identification and Ranking

In any project evaluation we need to identify the risks and quantify their effects. One approach is to construct a project risk matrix utilizing a checklist of possible risks and classifying risks according to their relative importance and likelihood. Importance and likelihood need to be separately assessed.

We might be less concerned with something that, although serious, is very unlikely to occur than with something less serious that is almost certain.

Importance and likelihood classified as high (H), medium (M), low (L) or exceedingly unlikely (-).

2. Risk and net present value

Where a project is relatively risky it is common practice to use a higher discount rate to calculate net present value. The premiums, even if arbitrary, provide a consistent method of taking risk into account.

3. Cost-Benefit Analysis

A rather more sophisticated approach to the evaluation of risk is to consider each possible outcome and estimate the probability of its occurring and the corresponding value of the outcome. The value of the project is then obtained by summing the cost or benefit for each possible outcome weighted by its corresponding probability.

This approach is frequently used to evaluate large projects. This technique relies being able to assign probabilities of occurrence to each scenario, which requires extensive research.

4. Risk Profile Analysis

An approach which attempts to overcome some of the objections to cost-benefit averaging is the construction of risk profiles using sensitivity analysis.

This involves varying each of the parameters that affect the project's cost or benefits to ascertain how sensitive the project's profitability is to each factor.

By studying the results of a sensitivity analysis we can identify those factors that are most important to the success of the project.

5. Using Decision Trees

The approaches to risk analysis discussed previously rather assume that we are passive bystanders allowing nature to take its own course – the best we can do is reject risky projects. There are many situations where we can evaluate whether a risk is important and, if it is, decide a suitable course of action.

Such decisions will limit or affect future options and, at any point, it is important to be able to assess how a decision will affect the future profitability of the company.

The analysis of a decision tree consists of evaluating the expected benefit of taking each path from a decision point. The expected value of each path is the sum of the value of each possible outcome, multiplied by its probability of occurrence.

Programme Management

D. C. Ferns defined a programme as '*a group of projects that are managed in a coordinated way to gain benefits that would not be possible were the projects to be managed independently*'.

1. Business Cycle Programmes

The collection of projects that an organization undertakes within a particular planning cycle has already been discussed under the topic of project portfolios.

2. Strategic Programmes

Several projects together can implement a single strategy. Each activity could be treated as a distinct project, but would be coordinated as a programme.

3. Infrastructure Programmes

Organizations can have various departments which carry out distinct, relatively self-contained, activities. These distinct activities will probably require distinct databases and information systems. The central ICT function would have responsibility for setting up and maintaining the ICT infrastructure.

An infrastructure programme could refer to the activities of identifying a common ICT infrastructure and its implementation and maintenance.

4. Research and Development Programmes

A successful portfolio would need to be a mixture of 'safe projects' with relatively low returns and some riskier projects that might fail, but if successful would generate handsome profits which will offset the losses on the failures.

5. Innovative Partnerships

Companies sometimes come together to work collaboratively on new technologies in a 'pre-competitive' phase. Separate projects in different organizations need to be coordinated and this might be done as a programme.

Managing the allocation of resources within programmes

Typically, an ICT department has pools of particular types of expertise, and these might be called upon to participate in a number of concurrent projects. Programme managers will have concerns about the optimal use of specialist staff.

When a project is planned, at the stage of allocating resources, programme management is involved.

Strategic Programme Management

A different form of programme management is where a portfolio of projects all contributes to a common objective.

For example, a customer's experience of the organization might be found to be very variable and inconsistent. The employee who records the customer's requirements is different from the people who actually carry out the work and different again from the clerk who deals with the accounts.

A business objective might be to present a consistent and uniform front to the client. The work to reorganize each individual area could be treated as a separate project, coordinated at a higher level as a programme.

Creating a Programme

The Programme Mandate

This should be a formal document describing:

- The new services or capabilities the programme should deliver
- How the organization will be improved by the use of the new services or capability
- How the programme fits with corporate goals and any other initiatives

At this point a *programme director* ought to be appointed to provide initial leadership for the programme.

The programming brief

A *programme brief* is now produced which outlines the business case for the programme.

- A preliminary *vision statement* which describes the new capacity that the organization seeks
- The *benefits* that the programme should create
- Risks and issues
- Estimated cost, timescale and effort

Vision statement

This stage would justify the setting up of a small team. A **programme manager** with day-to-day responsibility for the programme would be appointed.

This group takes the vision statement from the project brief and refines and expands it. It should describe in detail the new capability that the programme will give the organization.

Blueprint

This should contain:

- Business models outlining the new processes required
- Organizational structure – including the numbers of staff required in the new systems and the skills they will need
- The information systems
- Data and information requirements
- Costs, performance and service level requirements

The blueprint is supported by **benefit profiles** which estimate when the expected benefits will be experienced following implementation of the enhanced capabilities.

The management structure needed to drive this programme forward would also need to be planned and organized.

A preliminary list of projects needed to achieve the programme objectives will be created with estimated timescales. This **programme portfolio** will be presented to the programme sponsors.

A **stakeholder map** identifying the groups of people with an interest in the project and its outcomes and their particular interests could be drawn up. This can be used to write a **communications strategy** and plan showing how the appropriate information flows between stakeholders can be set up and maintained.

At the initial programme planning stage, a preliminary plan can be produced containing:

- Project portfolio
- Cost estimates for each project
- The benefits expected
- Risk identified
- The resources need to manage, support and monitor the programme

This information allows a **financial plan** to be created. This enables higher management to put into place the budget arrangement to meet the expected costs at identified points in time.

Aids to programme management

Dependency Diagrams

There will often be physical and technical dependencies between projects.

`Figure 2.3 – p. 43 – An Example of a dependency Diagram`

Delivery Planning

The creation of a delivery dependency diagram would typically lead to the definition of *tranches* of projects. A tranche is a group of projects that will deliver their products as one step in the programme. The projects in a tranche should combine to provide a coherent new capability or set of benefits for the client.

At this point the planning of individual projects can be considered. This could be initiated by writing of *project briefs,* defining the scope and objectives of each project.

Some Reservations about programme management

- Programme management is not simply a scaled-up project management
- Different forms of programme management may be appropriate for different types of projects.

Benefits management

Benefits management encompasses the identification, optimization and tracking of the expected benefits from a business change in order to ensure they are actually achieved.

To do this, you must:

- Define the expected benefits from the programme
- Analyse the balance between costs and benefits
- Plan how the benefits will be achieved and measured
- Allocate responsibilities for the successful delivery of the benefits

- Monitor the realization of the benefits

Benefits can be of many different types, including:

- **Mandatory Compliance –** Governmental or European legislation might make certain changes mandatory.
- **Quality Of Service –** an insurance company, for example, might want to settle claims by customers more quickly
- **Productivity –** the same or even more work can be done at less cost and staff time
- **More motivated workforce –** this might be because of an improved reward system, or through job enlargement or job enrichment.
- **Internal management benefits –** better decision making
- **Risk reduction**
- **Economy –** The reduction of costs, other than those related to staff.
- **Revenue enhancement/acceleration –** the sooner bills reach customers, the sooner they can pay them
- **Strategic fit –** A change might not directly benefit a particular group within the organization but has to be made in order to obtain some strategic advantage for the organization as a whole.

Quantifying Benefits

We have already seen than benefits can be:

- Quantified and valued – that is, a direct financial benefit is experienced.
- Quantified but not valued – that is, a decrease in customer complaints
- Identified but not quantified – public approval of the organization in the locality where it is based

Some key tests have been suggested in order to sound out whether a putative benefit is likely to be genuine:

- Can you explain in precise terms why this benefit should result from this business change?
- Can you identify the ways in which we will be able to see the consequences of this benefit?
- If the required outcomes do occur, can they be attributed directly to the change, or could other factors explain them?
- Is there any way in which the benefits can be measured?

Chapter 3 - An Overview of Project Planning

Introduction to Step Wise project Planning

The framework described is called the Step Wise method to help to distinguish it from other methods such PRINCE2.

```
Table 3.1 – p. 85 – An outline of Step Wise Planning activities
Figure 3.1 p. 51 – An overview of Step Wise
```

0. Select Project

This is outside the main project planning process.

1. Identify Project Scope and Objectives

The activities in this step ensure that all the parties to the project agree on the objectives and are committed to the success of the project.

1.1 Identify objectives and practical measures of the effectiveness in meeting those objectives
1.2 Establish a project authority
1.3 Stakeholders analysis – identify all stakeholders in the project and their interests
1.4 Modify objectives in the light of stakeholder analysis
1.5 Establish methods of communicating with all parties

2. Identify project Infrastructure

There is usually some kind of existing infrastructure into which the project must fit.

2.1 Identify relationship between the project and strategic planning

There is a technical framework within which the proposed new systems are to fit. Hardware and software standards are needed so that various systems can communicate with each other. These technical strategic decisions should be documented as part of an *enterprise architecture* process.

ICT projects should create software and other components compatible with those created by previous projects and also with the existing hardware and software platforms.

2.2 Identify installation standards and procedures

Change control and *configuration management* standards should be in place to ensure that changes to requirements are implemented in a safe and orderly way.

The procedural standards may lay down the quality checks that need to be done at each point of the project life cycle or these may be documented in separate *quality standards and procedures* manual.

The organization, as part of its monitoring and control policy, may have been a *measurement programme* in place which dictates that certain statistics have to be collected at various stages of a project.

Finally the project manager should be aware of any *project planning and control standards.*

2.3 Identify project team organization

Project leaders might have some control over the way that their project team is to be organized. Often, though, an organizational structure will be dictated to them.

3. Analyse project characteristics

The purpose of this part is to ensure that the appropriate methods are used for the project.

3.1 Distinguish the project as either objective- or product-driven
3.2 Analyse other project characteristics (including quality-based ones)
3.3 Identify high-level project risks
3.4 Take into account user requirements concerning implementation
3.5 Select development methodology and life-cycle approach
3.6 Review overall resource estimates

4. Identify project products and activities

The more detailed planning of the individual activities now takes place.

4.1 Identify and describe project products (or deliverables)

Identifying all the things the project is to create helps us to ensure that all the activities we need to carry out are accounted for.

Some of these products will be handed over to the client at the end of the project – these are *deliverables.* Other products might not be in the final configuration, but are needed as *intermediate* products used in the process of creating deliverables.

These products will include a large number of *technical* products, such as training material and operating instructions. There will also be products to do with the *management* and the *quality* of the project.

The products form a hierarchy. The main products will have sets of component products which in turn may have sub-component products and so on.

Figure 3.2 – p. 61 – A fragment of a Product Breakdown Structure for a system development task.

This part of the planning process draws heavily on the standards laid down in PRINCE2.

These specify that products at the bottom of the Product breakdown structure (PBS) should be documented by *product descriptions,* which contain:

- name/identity
- purpose
- derivations
- composition
- form
- standards
- quality criteria

Figure 3.3 – p. 62 – PBS for the products need to produce an invitation to tender (ITT)

4.2 Document Generic product flows

Some product will need one or more other products to exist first before they can be created. These relationships can be portrayed in a *product flow diagram(PFD).*

PFDs should not have links between products which loop back iteratively. This is *not* because iterations are not recognized. The form the PFD takes will depend on assumptions and decisions about how the project is to be carried out.

4.3 Recognize Product instances

Where the same generic PFD fragment relates to more than one instance of a particular type of product, an attempt should be made to identify each of those instances.

4.4 Produce ideal activity network

Inorder to generate one product from another there must be one or more activities that carry out the transformation. By identifying these activities, we can create an activity network which shows the tasks that have to be carried out and the order in which they have to be executed.

4.5 Modify the ideal to take into account need for stages checkpoints

This approach to sequencing the activities encourages the formulation of a plan which will minimize the overall duration for the project.

There might, however, be a need to modify this by dividing the project into stages and introducing checkpoint activities. These are activities which draw together the products of preceding activities to check that they are compatible.

There could be some key activities, or *milestones* which represent the completion of important stages of the project of which they would want to take particular note. Checkpoint activities are often useful milestones.

5. Estimate effort for each activity

5.1 Carry out bottom-up estimates

Estimates of the staff effort required, the probable elapsed time and the non-staff resources needed for each activity will need to be produced.

Effort is the amount of work that needs to be done. Elapsed time is the time between the start and end of a task.

The individual activity estimates of effort should be summer to get an overall bottom up estimate which can be reconciled with the previous top-down estimate.

5.2 Revise plan to create controllable activities

The estimates for individual activities could reveal that some are going to take quite a long time. It would be better to break this down into a series of smaller subtasks.

In general, try to make activities about the length of the reporting period used for monitoring and controlling the project.

6. Identify activity Risks

6.1 Identify and quantify activity-based risks

Risks inherent in the overall nature of the project have already been considered in Step 3. We now want to look at each activity in turn and asses the risks to its successful outcome.

A project plan will be based on a huge number of assumptions, and so some way of picking out the risks that are most important is needed. The damage that each risk could cause and the likelihood of it occurring have to be gauged.

6.2 Plan Risk reduction and contingency measures where appropriate

Contingency plans specify action that is to be taken if a risk materializes.

6.3 Adjust overall plans and estimates to take account of risks

We may change our plans, perhaps by adding new activities which reduce risks.

7. Allocate Resources

7.1 Identify and allocate resources

The type of staff needed for each activity is recorded. The staff available for the project is identified and are provisionally allocated to tasks.

7.2 Revise plans and estimates to take into account resource constraints

Some staff may be needed for more than one task at the same time and, in this case, an order priority is established.

8. Review/publicise plan

8.1 Review quality aspects of the project plan

A danger when controlling any project is that an activity can reveal that an earlier activity was not properly completed and need to be reworked. It is important to know that when a task is reported as completed, it really is. Each task should have *quality criteria.* These are quality checks that have to be passed before the activity can be 'signed off' as completed.

8.2 Document plans and obtain agreement

It is important that the plans be carefully documented and that all the parties to the project understand and agree to the commitments required of them in the plan.

9. and 10. Execute plan/lower level planning

Once the project is under way, plans will need to be drawn up in greater details for each activity as it becomes due.

Chapter 4 – Selection of an appropriate project approach

The development of software in house usually means that:

- devs and the users belong to the same organization
- application will slot into a portfolio of existing computer based systems
- the methodologies and technologies are largely dictated by organizational standards and policies, including the existing enterprise architecture

A software supplier could carry out successive development projects for a variety of external customers. They would need to review the methodologies and technologies.

The decision making process has been called *project analysis* (*method engineering, method tailoring, technical planning*).

Build or Buy?

With in-house development the developers and the users are in the same organization. Where the development is *outsourced* they are in different organizations.

These factors will affect the way that a project is organized:

- recruitment of technical staff
- lack of executives qualified to lead the effort

The contracting company will have technical and project expertise not readily available to the client.

Software is still needed. An option which is increasingly taken is to obtain a license to run off-shelf software.

The advantages:

- The supplier of the application can spread the cost of development over a large number of customers and thus the cost per customer should be reduced.
- The software already exists, so
    - It can be examined and perhaps even trailed before acquisition
    - There is no delay while the software is being built
- When lots of people have already used the software, most of the bugs are likely to have been reported and removed.

***The disadvantages*:**

- You have the same app as everyone else, there is no competitive advantage
- Modern off-the-shelf software tends to be very customizable; however, there are limits to this. You may end up having to change your office procedures in order to fit in with the new system.
- You will not own the software code
- Your organization might become too reliant upon it. This may create a considerable barrier to moving to a different application.

Choosing Methodologies and Technologies

In the context of system development and software engineering, the term *methodology* describes a collection of *methods.*

Methods often involve the creation of models. A *model* is a representation of a system which abstracts certain features but ignores others.

Project analysis should select the most appropriate methodologies and technologies for a project. Methodologies include approaches like:

- The Unified Software Development Process (USDP)
- Structures systems analysis and design method (SSADM)
- Human-Centred Design

As well as the products and activities, the chosen methods and technologies will affect:

- Training requirements
- Types of staff
- Development environment
- System maintenance arrangements

1. Identify project as either objective-driven or product-driven

(Chapter 1)

2. Analyse other project characteristics

The following questions can be asked:

- Is a *data-oriented* or *process oriented system to be implemented?*
  Data-oriented systems generally mean information systems that will have a substantial database. Process-oriented systems refer to embedded control systems.
- *Will the software that is to be produced be a general tool or application specific?*
  A general tool would be a spreadsheet or word-processing package. An application-specific package could be like an airline seat reservation system.
- *Are there specific tools available for implementing the particular type of application?*
  - Does it involve concurrent processing?
  - Will the system to be created be knowledge based?
  - Will the system to be produced make heavy use of computer graphics?
- *Is the system to be created safety critical?*
- *Is the system designed primarily to carry out predefined services or to be engaging and entertaining?*
- *What is the nature of the hardware/software environment in which the system will operate?*

3. Identify high-level project risks

One suggestion is that uncertainty can be associated with the *products, processes* or *resources* of a project.

- *Product uncertainty* – How well are the requirements understood? The users themselves could be uncertain about what a proposed information system is to do.
- *Process Uncertainty* – Any change in the way that the systems are developed introduces uncertainty.

- *Resource Uncertainty* – The main area here is likely to be the availability of staff of the right ability and experience.

4. Take into account user requirements concerning implementation

Staff planning a project should try to ensure that unnecessary constraints are not imposed on the way that a project's objectives are met. International conglomerates have found that imposing uniform applications and technologies throughout all their component parts can save time and money.

5. Select general life-cycle approach

- *Control systems* – A real time system will need to be implemented using an appropriate methodology.

- *Information Systems* – An information system will need a methodology that matches the type of environment.

- *Availability of Users* – Where the software is for the general market rather than application and user specific, then a methodology which assume that identifiable users exist who can be quizzed about their needs would have to be thought about with caution.

- *Specialized Techniques* – Expert system shells and logic based programming languages have been invented to expedite the development of *knowledge based systems.* A number of specialized techniques and standard components are available to assist in the development of *graphics-based systems.*

- *Hardware environment* – The environment in which the system is to operate could put constraints on the way it is to be implemented.

- *Safety-critical systems* – Where safety and reliability are essential, this might justify the additional expense of a formal specification.

- *Imprecise Requirements* – uncertainties or a *novel hardware/software platform* mean that a prototyping approach should be considered. If the environment in which the system is to be implemented is a rapidly changing one, then serious consideration would need to be given to incremental delivery.

Structure versus Speed of Delivery

Structured methods consist of sets of steps and rules, which when applied, generate systems products such as use case diagrams. Such methods are more time consuming and expensive than more intuitive approaches. The pay-off is a less error prone and more maintainable system. The balance of costs and benefits is more lie to be justified on a large project.

Because of the additional effort needed and their greater applicability to large and complex projects, these are often called *heavyweight* methods.

One response to this has been *rapid application development (RAD)* which puts the emphasis on quickly producing prototypes of the software for users to evaluate.

RAD adopts tactics such as *Joint Application Development (JAD).* In these workshops, developers and users work together intensively for a limited time and identify and agree fully documented system requirements.

The workshops are conducted away from the normal working environment in *clean rooms.*

Another way of speeding up delivery is delivering less.

The Waterfall model

(a.k.a. *One Shot* or *Once-through* model) There is a sequence of activities working from top to bottom. Sometimes previous steps are revisited, but this is an exception. It creates a natural milestone at the end of each stage. The waterfall model can be expanded in to the **V-Process Model.**

Feasibility Study – User Requirements – Analysis – Systems Design – Program Design – Coding – Testing – Operation

The Spiral model

This can be portrayed as a loop where the system to be implemented is considered in more detail in each sweep. Each sweep terminates with an evaluation before the next iteration.

Software prototypes

A *prototype* is a working model of one or more aspects of the projected system. It is constructed and tested quickly and inexpensively in order to test out assumptions.

1. *Throw-away Prototypes*

The prototype tests out some ideas and is then discarded when the true development of the operational system is commenced. The prototype could be developed using a different software or hardware environment.

2. *Evolutionary Prototypes*

The prototype is developed and modified until it is finally in a state where it can become the operational system.

Some of the reasons for prototyping:

- Learning by doing
- Improved communication
- Reduced need for documentation
- Reduced maintenance costs
- Clarification of partially known requirements
- Demonstration of the consistency and completeness of a specification
- Improved user involvement
- Feature constraint
- Production of expected results

Software prototyping is not without drawbacks:

- Users can misunderstand the role of the prototype
- Lack of project standards possible
- Lack of control
- Additional expense
- Machine efficiency
- Close proximity of developers

Other Ways of categorising prototypes

1. What is being learnt?

The most important reason for prototyping is a need to learn about an area of uncertainty.

If it's a real prototype, they must:

- Specify what they hope to learn from the prototype
- Plan how the prototype is to be evaluated
- Report on what has actually been learnt

Prototypes can be used to find out about new development technique. The nature is of the application might also be uncertain.

2. To what extent is the prototyping to be done?

Prototyping usually simulates only some aspect of the target application:

- *Mock Ups* – as when copies of screens are shown to the users on a terminal, but the screens cannot actually be used.
- *Simulated Interaction* – The user can type in a request to access a record and the system will show the same details and no access is made to a database.
- *Partial working model*:
    - Vertical, some, but not all, features are prototyped fully.
    - Horizontal, all features are prototyped but not in detail.

3. What to prototype

- *The Human-Computer Interface* – Prototyping tends to be confined to the nature of the operator interaction. Here the physical vehicle for the prototype should be as similar as possible to the operational system.
- *The functionality of the system* – Here the precise way the system should function internally is not known.

4. Controlling Changes during prototyping

One approach has been to categorize changes as belonging to 1 of 3 types:

- ***Cosmetic*** (Often about 35% of changes)
  These are simple changes to the layout of the screens or reports, they are:
    a) Implemented
    b) Recorded
- ***Local*** (Often about 60% of changes)
  These change the way that a screen or report is processed but do not affect other parts of the system, they are:
    a) Implemented
    b) Recorded
    c) Backed up
    d) Inspected retrospectively
- ***Global*** (about 5% of changes)
  These are changes that affect more than one part of the processing.

Incremental Delivery

This approach breaks the application down into small components which are then implemented and delivered in sequence.

Time-Boxing is often associated with an incremental approach. Here the scope of the deliverables for an increment is rigidly constrained y an agreed deadline.

Advantages

- Feedback from early increments improves later stages
- The possibility of changes in requirements is reduced because of the shorter time span between the design of a components and its delivery
- Users get benefits earlier than with a conventional approach
- Early delivery of some useful components improves cash flow.
- Smaller sub-projects are easier to control and manager.
- *Gold-plating,* the requesting of features that are unnecessary and not in fact used, is lees as users know that if a feature is not in the current increment then it can be included in the next.
- The project can be temporarily abandoned if more urgent work emerges.
- Job satisfaction is increased fir developers who see their labours bearing fruit.

Disadvantages

- Later increments might require modifications to earlier increments. This is known as *software breakage.*
- Software devs may be more productive working on one large system than on a series of smaller ones.
- Conceptual integrity sometimes suffers because there is little motivation to deal with scalability, extensibility, portability or reusability beyond what any vague requirements might imply.

The incremental Delivery Plan

The elements of the incremental plan are the:

1. **System Objectives**

   Project planners ideally want well-defined objectives but as much freedom as possible about how these are to be met. These overall objectives can be expanded into more specific functional goals and quality goals.

   Functional Goals Include:
   - Objectives it is intended to achieve
   - Jobs the system is to do
   - Computer/non-computer functions to achieve them

   Measurable quality characteristics:
   - Reliability
   - Response
   - Security

2. **Open Technology Plan**

   If the system is to be able to cope with new components being continually added, then it needs to be extendible, portable and maintainable.

   As a minimum this will require the use of:
   - A standard high level language

- A standard operating system
- Small modules
- Variable parameters
- Standard database management system

3. **Incremental Plan**
   Having defined the overall objectives and an open technology plan, the next stage is to plan the increments, using the following guidelines:
   - Steps typically should consist of 1-5% of the total project
   - Non computer steps should, ideally, not exceed one month and should not, at worst, take more than three months.
   - Each increment should deliver some benefit to the user
   - Some increments will be physically dependent on others
   - In other cases value to cost ratios may be used to decide priorities

Table 4.1 – p. 92 – Ranking by value-to-cost ratio

Agile Methods

Agile methods are designed to overcome the disadvantages we have noticed with heavyweight implementation methodologies.

There are various agile approaches, including:

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

The various methods shared 4 core components:

1. Individuals and interaction processes and tools
2. Working together over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Atern/Dynamic Systems Development Method

8 Core Atern principles have been enunciated:

1. ***Focus on Business Need*** – Every decision in the development process should be taken with a view to best satisfying business needs.
2. ***Deliver on Time*** – Time boxing is applied. Every deadline will see the delivery of valuable products, even if some are less valuable ones are held over.
3. ***Collaborate*** – A one-team culture should be promoted, where user representatives are integrated into the delivery team.
4. ***Never Compromise Quality*** – Realistic quality targets are set early in the project. A process of continuously testing developing products starting as soon as possible is adopted.
5. ***Develop Iteratively*** – The prototyping approach is described in Section 4.8.

6. ***Build incrementally from firm foundation*** – the incremental delivery approach as described in section 4.10.
7. ***Communicate continuously*** – in the case of users this could be done via workshops and the demonstration of prototyping.
8. ***Demonstrate Control*** – Atern methodology has a range of plans and reports that can be used to communicate project intentions and outcomes to project sponsors and other management stakeholders.
9. ***Feasibility/foundation*** – Among the activities undertaken here is derivation of a business case and general outlines of the proposed architecture of the system to be developed.
10. ***Exploration Cycle*** – this investigates the business requirements. These requirements are translated into a viable design for the application.
11. ***Engineering Cycle*** – This takes the design generated and converts it into usable components of the final system that will be used operationally.
12. ***Deployment*** – This gets the application created in the engineering cycle into actual operational use.

The relative importance of requirements can be categorized using the ***'MoSCoW'*** *classification:*

- *Must Have* – that is, essential features
- *Should have* – these would probably be mandatory is you were using a conventional development approach, but the system can operate without them
- *Could have* – these requirements can be delayed with some inconvenience
- *Won't have* – these features are wanted, but their delay to a later increment is readily accepted

Extreme Programming

4 Core values are presented as the foundation of XP:

1. **`Communication and Feedback`** - The best method is face to face.
2. **`Simplicity`** - The simplest design that implements the users' requirements should always be adopted.
3. **`Responsibility`** - The developers are the one who are ultimately responsible for the quality of the software.
4. **`Courage`** - This is the courage to throw away any work in which you have already invested a lot of effort, and to start with a fresh design if that is what is called for.

Among the core practices of XP are the following:

`The Planning Exercise`
Previously we talked about increments, XP refers to these are *releases.* The planning game is the process whereby the features to be incorporated in the next release are negotiated. Each of the features is documented in a short textual description called a *story.*

`Simple Design`
This is the practical implementation of the value of simplicity that was described above.

`Small Releases`
The time between released of functionality to users should be as short as possible.

Metaphor
The system to be built will be software code that reflects things that exist and happen in the real world. The terms used to describe the corresponding software elements should, as far as possible, reflect real-world terminology.

Testing
Testing is done at the same time as coding. The test inputs and expected results should be scripted so that the testing can be done using automated testing tools. These test cases can then be accumulated so that they can be used for regression testing to ensure that later developments do not insert errors into existing working code.

Two types of testing are needed:
Unit testing which focusses on the code a developer has just written and *function testing* which is user-oriented and checks the correctness of a particular feature.

Refactoring
The solution to spaghetti-like code is to have the courage to resist the temptation to make changes that affect as little of the code as possible and be prepared to rewrite whole sections of code if this will keep the code structured.

Pair programming
All software code is written by pairs of developers, once actually doing the typing and the other observing, discussing and making comments and suggestions about what the other is doing. At intervals the developers can switch roles.

The ideal is that you are constantly changing partners so that you get to know about a wide range of features that are under development.

Collective Ownership
The team as a whole take collective responsibility for the code in the system.

Continuous integration
As changes are made to software units, integration tests are run regularly to ensure that all the components work together correctly.

Forty Hour Weeks
Working excessive hours can lead to ill-health and be generally counterproductive.

On-Site Customers
Fast and effective communication with the users is achieved by having a user domain expert on-site with the developers.

Coding Standards
If code is genuinely to be shared, then there must be common, accepted, coding standards to support the understanding and ease of modification of the code.

Limitations of XP:
- There must be easy access to users. This may be difficult where developers and users belong to different organizations.
- Development staff need to be physically located in the same office
- As users find out how the system will work only by being presented with working versions of the code, there may be communication problems if the application does not have a visual interface.

- For work to be sequenced into small iterations of work, it must be possible to break the system functionality into relatively small and self-contained components.
- Large, complex systems may initially need significant architectural effort. This might preclude the use of XP.
- There is a reliance on high quality developers which makes software development vulnerable if staff turnover is significant.
- It might be difficult for maintenance staff without documentation to identify which bits of code to modify to implement a change in requirements.
- Having a repository of comprehensive and accurate test data and expected results may not be as helpful as might be expected if the rationale for particular test cases is not documented.

Chapter 5 – Software Effort Estimation

A successful project is delivered *on time, within budget and with the required quality.* A project manager has to produce estimates of *effort,* which affect costs, and of activity *durations,* which affect delivery time.

Some difficulty arises from the complexity and invisibility of software. The intensely human activities which make up system development cannot be treated in a purely mechanistic way.

Other difficulties include:

- *Subjective Nature of Estimating* – Some people tend to underestimate the difficulty of small tasks and over-estimate large ones.
- *Political Implications* – Different groups within an organization have different objectives. To avoid these political influences, one suggestion is that estimates be produced by a specialist estimating group, independent of the users and the project team.
- *Changing Technology* – Where technologies change rapidly, it is difficult to use the experience of previous projects on new ones.
- *Lack of homogeneity of project experience* – Even where technologies have not changed, knowledge about typical task durations may not be easily transferred from one project to another because of other differences between projects.

Where are estimates done?

Estimates are carried out at various stages of a software project for a variety of reasons.

- *Strategic planning* – Project portfolio management involves estimating the costs and benefits of new applications in order to allocate priorities.
- *Feasibility Study* – This confirms that the benefits of the potential system will justify the costs.
- *System Specification* – The effort needed to implement different design proposals will need to be estimated.
- *Evaluation of Suppliers' proposals* – The costs of bids could also be compared to in-house development.
- *Project Planning* – More estimates of smaller work components will be made.

Figure 5.1 – p. 106 – Software estimation takes place in steps 3 and 5 in particular.

Problems with over-and under-estimates

Over-estimate may cause the project to take longer than it would otherwise. This can be explained by the application of two laws:

1. Parkinson's Law - '*Work expands to fill the time available'*, that is, given an easy target staff will work less hard.
2. Brook's Law – As the project team grows in size, so will the effort that has to go into management, coordination and communication. '*putting more people on a late job makes it later'.*

The danger with under-estimated is the effect of quality. This may be seen as a manifestation of Weinberg's zeroth law of reliability: '*if a system does not have to be reliable, it can meet any other objective'.*

Research has found that motivation and morale are enhanced where targets are achievable.

An estimate is not really a prediction, it's a management goal.

The Basis for Software Estimating

The need for historical Data

Most estimation methods need information about past projects. However, care is needed because of possible differences in factors such as programming language and the experience of staff. If past project data is lacking, externally maintained datasets can be accessed. One well known international database is that maintained by the International Software Benchmark Standards Group (ISBSG).

Measure of Work

The time to complete software will depend on the individual capability and experience of staff not yet identified. Implementation time and costs will also depend on the particular technologies selected.

The usual practice is therefore to start by expressing work size independently of effort using a measure such as source line of code (SLOC) or thousands of lines of code (KLOC).

Software Estimation techniques

Barry Boehm identified the main ways of deriving estimates of software development effort as:

- *Algorithmic models* – which use 'effort drivers' representing characteristics of the target system and the implementation environment to predict effort.
- *Expert judgement* – based on the advice of knowledgeable staff
- *Analogy* – where a similar, completed, project is identified and its actual effort is used as the basis of the estimate
- *Parkinson* – where the staff effort available to do the project becomes the 'estimate'
- *Price to win* – where the 'estimate' is a figure that seems sufficiently low to win a contract
- *Top down* – where an overall estimate for the whole project is broken down int the effort required for component tasks
- *Bottom up* – where the component tasks are identified and sized and these individual estimates are aggregated

Bottom Up Estimating

The estimator breaks the project into its component tasks. Each task is decomposed into its component subtasks and these in turn could be further analysed.

Although the top-down analysis is an essential precursor to bottom-up estimating, it is really a separate process – that of producing a work-breakdown schedule (WBS). The bottom-up part comes in adding up the calculated effort for each activity to get an overall estimate.

This approach is better at the later more detailed stages of project planning.

A Procedural code-oriented Approach
Here we describe how a bottom up approach can be used at the level of software components:

a. Envisage the number and type of software modules in the final system
   Most information systems are built from a small set of system operations.

b. Estimate the SLOC of each identified module
   One way to judge the number of instructions likely to be in a program is to draw up a program structure diagram and to visualize how many instructions would be needed to implement each identified procedure.

c. Estimate the work content, taking into account complexity and technical difficulty
   The practice is to multiply the SLOC by a factor for complexity and technical difficulty. This factor will depend largely on the subjective judgement of the estimator.

d. Calculate the work-days effort
   Historical data can be used to provide ratios to convert weighted SLOC to effort.

The Top-Down Approach and Parametric Models

This approach is normally associated with *parametric* (or *algorithmic*) models.

Project effort relates mainly to variables associated with characteristics of the final system. A parametric model will normally have one or more formulae in the form:

$$effort = (system\ size) \times (productivity\ rate)$$

A model to forecast software development effort therefore has two key components.

1. A method of assessing the amount of work needed.
2. Assessment of the rate of work at which the task can be done.

KLOC is a *size driver* indicating the amount of work to be done, while developer experience is a productivity driver influencing the productivity driver influencing the productivity or work rate.

$$Productivity = effort/size$$
$$Effort = constant_1 + (size \times constant_2)$$

Expert Judgement

This is asking or an estimate of task effort from someone who is knowledgeable about either the application of the development environment.

Some have suggested that expert judgement is simply a matter of guessing, but experts tend to use a combination of an informal analogy approach where similar projects from the past are identified, supplemented by bottom-up estimating.

Estimating by analogy

This is also called case base reasoning. This estimator identifies complete projects (source cases) with similar characteristics to the new project (the target case).

1. The effort recorded for the matching source case is then used as a base estimate for the target.
2. The estimator then identifies differences between the target and the source and adjusts the base estimate to produce an estimate for the new project.

A problem is identifying the similarities and differences between applications where you have a large number of past projects to analyse.

One attempt to automate this selection process is the ANGEL software tool. This identifies the source case that is nearest the target by measuring the Euclidean distance between the cases.

$$Distance = sqrt((targetParameter_1 - sourceParameter_1)^2 + \ldots (targetParamter_n - sourceParamter_n)^2)$$

Albrecht Function Point Analysis (IFPUG)

Allan Albrecht developed the idea of function points (FPs).
The basis of the function point analysis is that information systems comprise five major components or '*external user types',* that are the benefit to the users:

1. *External Input Types* are input transactions which update internal computer files
2. *External Output Types* are transactions where data is output to the user.
3. *External Inquiry Types* – are transactions initiated by the user which provide information but do not update internal files.
4. *Logical Internal File Types* are the standing files used by the system. The term file refers to a group of data items that is usually accessed together. It may be made up of one or more record.
5. *External Interface file types* allow for output and input that may pass to and from other computer applications.

The analyst identifies each instance of each external user type in the application. Each component is then classified as having either high, average or low complexity. The counts of each external user type in each complexity band are multiplied by specified weights to get FP scores which are summed to obtain an overall FP count.

Function Points Mark II

As with Albrecht, the information processing size is initially measured in unadjusted function points, (UFPs) to which a technical complexity adjustment can then be applied (TCA).

Figure 5.2 – p. 117 – Model of a transaction

For each transaction, the UFPs are calculated;

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

W_i, W_3 and W_o are weightings derived by asking developers the proportions of effort spent in previous projects developing, the code dealing with inputs, accessing and modifying stored data and processing outputs.

The proportions of effort are then normalized into ratios which add up to 2.5.

COSMIC Full Function Points (FFP)

This method should be seen as an extension to the IFPUG method for real-time system. The original work has been taken forward by the formation of the Common Software Measurement Consortium (COSMIC).

COSMIC decomposes the system architecture into a hierarchy of software *layers.* The software component to be sized can receive requests for services from layers above and can request services from those below.

At the same time there could be separate software components at the same level that engage in *peer-to-peer communication.* This identifies the boundary of the software component to be assessed and this the points at which it receives input and transmits outputs.

Inputs and outputs are aggregated into *data groups,* where each group bring together data items that relate to the same object of interest.

Data Groups can be moved about in 4 ways:

1. *Entries (E)* which are effected by sub processes that move the data group into the software components in question from a 'user' outside its boundary. This could be from another layer or another separate software component in the same layer via peer-to-peer communication.
2. *Exits (X)* which are effected by sub processes that move the data group from the software component to a 'user' outside its boundary.
3. *Reads(R)* which are data movements that move data groups from persistent storage (such as a database) into the software component;
4. *Writes (W)* which are data movements that transfer data groups from the software component into persistent storage.

The overall FFP count is derived by simply adding up the counts for each of the 4 types of data movement. The resulting units are *Cfsu* (COSMIC functional size units).

COCOMO (COnstructive COst MOdel) II: A Parametric Productivity Model

The basic model was built around the equation

$$(effort) = c(size)^k$$

Where $effort$ was measured in pm or the number of 'person-months' consisting off units of 152 working hours, $size$ was measured in $kdsi$, thousands of delivered source code instructions, and c and k were constants.

The 1st step was to derive an estimate of the system size in terms of $kdsi$. The constants c and k depend on whether the system could be classified as 'organic', 'semi-detached' or 'embedded'.

- *Organic Mode* – This would typically be the case when relatively small teams developed software in a highly familiar in-house environment and when the system being developed was small and the interface requirements flexible.

- *Embedded Mode* – This meant that the product being developed had to operate within very tight constraints and changes to the system were very costly.

- *Semi-detached mode* – this combined elements of the organic and the embedded modes or had characteristics that came between the two.

System type	**c**	**k**
Organic	2.4	1.05
Semi-Detached	3.0	1.12
Embedded	3.6	1.20

The exponent value k when it is greater than 1 means that larger projects are seen as requiring disproportionately more effort than smaller ones.

Estimates are required at different stages in the system life cycle and COCOMO II has been designed to accommodate this by having models for 3 different stages.

1. Application Composition – Here the external features of the system that the users will experience are designed.
2. Early Design – Here the fundamental software structures are designed.
3. Post Architecture – here the software structures undergo final construction, modification and tuning to create a system that will perform as required

To estimate the effort for *application composition,* the counting of *object point* is recommended by the developers of COCOMO II. This follows the function point approach of counting externally apparent features of the software.

At the *early design* stage, FPs are recommended as the way of gauging a basic system size. An FP count may be converted to a LOC equivalent by implying the FPs by a factor for the programming language that is to be used.

The following model can be used to calculate an estimate of person-months:

$$pm = A(size)^{sf} \times (em_1) \times (em_2) \ldots \times (em_n)$$

Where pm is the effort in 'person-months', A is a constant, $size$ is measured in $kdsi$, and sf is an exponent factor

The scale factor is derived thus:

$$sf = B + 0.01 \times \Sigma(exponent\ driver\ ratings)$$

Where B is a constant.

The fact that these factors are used to calculate the exponent implies that the lack of these qualities increases the effort required disproportionately more on larger projects.

- *Precedentedness (PREC)* – This quality is the degree to which there are precedents or similar past cases for the current project. The greater the novelty of the new system, the more uncertainty there is and the higher the value given to the exponent driver.
- *Development Flexibility (FLEX)* – This reflects the number of different ways there are of meeting the requirements. The less flexibility there is the higher the value of the exponent driver.
- *Architecture/risk resolution (RSL)* – This reflects the degree of uncertainty about the requirements. It may be liable to change then a high value would be given to this exponent driver.
- *Team cohesion (TEAM)* – This reflects the degree to which there is a large dispersed team as opposed to there being a small tightly knit team.
- *Process Maturity (PMAT)* – The more structured and organized the way the software is produced, the lower the uncertainty and the lower the rating will be for this exponent driver.

In the COCOMO II model the $effort\ multipliers\ (em)$ adjust the estimate to take account or productivity factors, but do not involve economics or diseconomies of scale.

Conclusion – p. 125
Further Exercises – p. 126

Chapter 6 - Activity Planning

A detailed plan for the project must also include a schedule indicating the start and completion times for each activity. This will enable 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
- Produce a detailed schedule showing which staff carry out each activity
- Produce a detailed plan against which actual achievement may be measured
- Produce a timed cash flow forecast
- Replan the project during its life to correct drift from target

The Objectives of Activity Planning

1. Feasibility Assessment – is the project possible within required time scales and resource constraints?
2. Resource allocation – What are the most effective ways of allocating resources to the project.
3. Detailed Costing – How much will the project cost and when is that expenditure likely to take place?
4. Motivation – Providing targets and being seen to monitor achievement against targets is an effective way of motivating staff, particularly where they have been involved in setting those targets in the first place.
5. Coordination – The project plan provides an effective vehicle for communication and coordination among teams.

When to Plan

Planning is an ongoing process of refinement each iteration becoming more detailed and more accurate than the last. Over successive iterations, the emphasis and purpose of planning will shift.

During project start up, the main purpose of planning will be to estimate timescales and the risks of not achieving target completion dates or keeping within budget.

Throughout the project monitoring and planning must continue to correct any drift that might prevent meeting time or cost targets.

Project Schedules

Before work commences on a project, the project plan must be developed to the level of showing dates when each activity should start and finish and when and how much of each resource will be required.

The 1st step is to decide what activities need to be carried out and in what order they are to be done. From this we construct an *activity plan.* The ideal activity plan will then be the subject of an activity risk analysis.

The 3rd step is *resource allocation.* The expected availability of resources might place constraints on when certain activities can be carried out.

The final step is *schedule production.* Once resources have been allocated to each activity, we will be in a position to draw up a project schedule.

Project and Activities

Defining Activities

- A project is composed of a number of interrelated activities
- A project may start when at least one of its activities is ready to start
- A project will be completed when all of the activities it encompasses have been completed.
- An activity must have a clearly defined start and a clearly defined end-point.
- If an activity requires a resource then that resource requirement must be forecastable
- The duration of the activity must be forecastable
- Some activities might require that others are completed before they can begin

Identifying Activities
There are 3 approaches:

1. **The Activity Based Approach**

This approach consists of creating a list of all the activities that the project is thought to involve. When listing activities, it might be useful to subdivide the project into the main life-cycle stages and consider each of these separately.

A much favoured way to list activities is to make a *Work Breakdown Structure (WBS).* This involves identifying the main tasks required to complete the project and then breaking each down into a set off lower-level tasks.

Figure 6.2 – p. 134 – A fragment of an activity based Work breakdown structure

Advantages:

- It is more likely to result in a task catalogue that is complete and is composed of bob-overlapping activities.
- The WBS also represents a structure that may be refined as the project proceeds.

2. **The product-based approach**

(Chapter 3)

3. **The Hybrid Approach**

A WBS may be based upon the project's products. The degree to which the structuring is product based or activity based might be influenced by the nature of the project and the particular development method adopted.

Figure 6.3 – p. 136 – Product Breakdown Structure
Figure 6.4 – p. 137 – Structuring of Activities
Figure 6.5 – p. 137 – A hybrid Work Breakdown Structure

Sequencing and Scheduling Activities

Figure 6.6 – p. 138 – A project Plan as a Bar Chart

When drawing up the chart we have done two things: we have sequenced the tasks and scheduled them.

In small projects, this combined Sequencing-scheduling approach might be quite suitable, particularly where we wish to allocate individuals to particular tasks at an early planning stage.

Network Planning Models

These project scheduling techniques model the project's activities and their relationships as a network. In the network, time flows from left to right. The two best known methods are:

1. CPM – Critical Path Method
2. PERT – Program Evaluation Technique

Both of these use an activity-on-arrow approach to visualizing the project as a network where activities are drawn as arrows joining circles, or nodes, which represent the possible start and/or completion of an activity or set of activities.

Figure 6.7 – p. 140 – The IOE annual maintenance contracts project activity network fragment with a checkpoint activity added.

Figure 6.8 – p. 140 – The IOE annual maintenance contracts project activity network fragment represented as a CPM network.

Formulating a network Model

The 1st stage in creating a network model is to represent the activities and their interrelationships as a graph. In an activity-on-node we do this by representing activities as nodes (boxes) and the lines between nodes represent dependencies.

Constructing precedence networks
Consider some rules:

1. **A project network should have only 1 start node** – It is possible but undesirable to do so as it is a potential source of confusion.
2. **A project network should have only 1 end node** – The end node designates the completion of the project and a project may finish only once.
3. **A node has duration** – A node represents an activity and, in general, activities take time to execute.
4. **Precedents are the immediate preceding activities** – Some activities cannot start until some activities have finished.
5. **Time moves from left to right** – If at all possible, networks are drawn so that time moves from left to right.
6. **A Network may not contain loops** – A loop is an error in that it repeats a situation that cannot occur in practice.

Figure 6.9 – p. 141 – Fragment of a precedence network

7. A network should not contain dangles – A dangling activity such as 'Write user manual' in figure 6.11 should not exist as it is likely to lead to errors in subsequent analysis.

Figure 6.10 – p. 142 – A loop represents an impossible sequence
Figure 6.11 – p. 142 – A dangle
Figure 6.12 – p. 143 – Resolving the dangle

Representing lagged activities
We might come across situations where we wish to undertake two activities in parallel so long as there is a lag between the two.

Where activities can occur in parallel with a time lag between them, we represent the lag with a duration on the inking arrow as shown in Figure 6.13.

Figure 6.13 – p. 143 – Indicating Lags

This indicates that documenting amendments can start one day after the start of prototype testing and will be completed two days after prototype testing is completed.

Hammock Activities
Hammock activities are activities which, in themselves, have zero duration, but are assumed to start at the same time as the first 'hammocked' activity and to end at the same time as the last one.

They are normally used for representing overhead costs or other resources that will be incurred at a constant rate over the duration of a set of activities.

Labelling conventions

Earliest Start	Duration	Earliest Finish
Activity Label, Activity Description		
Latest Start	Float	Latest Finish

Adding the time dimension

The critical path approach is concerned with two primary objectives:

1. Planning the project in such a way that it is completed as quickly as possible
2. Identifying those activities where a delay in their execution is likely to affect he overall end date of the project or later activities' start dates.

The method requires that for each activity we have an estimate of its duration. The network is then analysed by carrying out a *forward pass,* to calculate the earliest dates at which activities may commence and the project be completed, and a *backwards pass* to calculate the latest dates for activities and the *critical path.*

Table 6.1 – p. 145 – An example project specifications with estimated activity durations and precedence requirements

The Forward Pass

The forward pass and the calculation of earliest start dates are carried out according to the following reasoning:

1. Activities A, B and F may start immediately, so the earliest date for their start is zero.

2. Activity A will take 6 weeks, so the earliest it can finish is week 6.
3. Activity B will take 4 weeks, so the earliest it can finish is week 4.
4. Activity F will take 10 weeks, so the earliest it can finish is week 10.
5. Activity C can start as soon as A has finished so its earliest start date is week 6. It will take 3 weeks so the earliest it can finish is week 9.
6. Activities D and E can start as soon as B is complete so the earliest they can each start is week 4. Activity D, which will take 4 weeks, can therefore finish by week 8 and activity E, which will take 3 weeks, can therefore finish by week 7.
7. Activity G cannot start until both E and F have been completed. It cannot therefore start until week 10 – the later of weeks 7 (for activity E) and 10 (for activity F). It takes 3 weeks and finishes in week 13.
8. Similarly, activity H cannot start until week 9 – the later of the two earliest finish dates for the preceding activities C and D.
9. The project will be complete when both activities H and G have been completed. Thus the earliest project completion date will be the later of weeks 11 and 13 – that is, week 13.

Figure 6.15 – p. 147 – The network after the forward pass

The backward pass

The second stage in the analysis of a critical path network is to carry out a backward pass to calculate the latest date at which each activity may be started and finished without delaying the end date of the project.

Figure 6.16 – p. 148 – The network after the backward pass

The latest activity dates are calculated as follows:

- The latest completion date for activities G and H is assumed to be week 13.
- Activity H must therefore start a week 11 at the latest (13 – 2) and the latest start date for activity G is week 10 (13 – 3).
- The latest completion date for activities C and D is the latest date at which activity H must start – that is, week 11. They therefore have latest start dates of week 8 (11 – 3) and week 7 (11 – 4) respectively.
- Activities E and F must be completed by week 10 so their earliest start dates are weeks 7 (10 – 3) and 0 (10 – 10) respectively.
- Activity B must be completed by week 7 (the latest start date for both activities D and E) so its latest start week 3 (7 – 4).
- Activity A must be completed by week 8 (the latest start date for activity C) so its latest start is week 2 (8 – 6)
- The latest start date for the project start is the earliest of the latest start dates for activities A, B and F. This is week zero. This is, of course, not very surprising since it tells us that if the project does not start on time it won't finish on time.

Identifying the critical path

There will be at least one path through the network that defines the duration of the project. This is known as the *critical path.* Any delay on any activity on this critical path will delay the completion of the project.

The difference between an activity's earliest start date and its latest start date is known as the activity's *float* – it is a measure of how much the start or completion of an activity may be delayed without affecting the end date of the project.

The significance of the critical path is two-fold:

- In managing the project, we must pay particular attention to monitoring activities on the critical path so that the effects of any delay or resource unavailability are detected and corrected at the earliest opportunity.
- In planning the project, it is the critical path that we must shorten if we are to reduce the overall duration of the project.

The *activity span* is the difference between the earliest start date end the latest finish date and is a measure of the maximum time allowable for the activity.

Figure 6.17 – p. 149 – The critical path

Activity Float

There are a number of other measures of activity float, including the following:

- **`Free Float`** – the time by which an activity may be delayed without affecting any subsequent activity. It is calculated as the difference between the earliest completion date for the activity and the earliest start date of the succeeding activity. This might be considered a more satisfactory measure of float for publicising to the staff involved in undertaking the activities.
- **`Interfering float`** – the difference between total float and free float. This is quite commonly used, particularly in association with the free float. Once the free float has been used, the interfering float tells us by how much the activity may be delayed without delaying the project end date – even though it will delay the start of subsequent activities.

Shortening the project duration

To shorten the overall duration of a project we would normally consider attempting to reduce activity durations. In many cases, this can be done by applying more resources to the task – working overtime or procuring additional staff.

The critical path indicates where we must look to save time – if we are trying to bring forward the end date of the project, there is clearly no point in attempting to shorten non-critical activities.

Generally, time savings are to be found by increasing the amount of parallelism in the network and the removal of bottlenecks.

Identifying critical activities

It is common practice to identify *near-critical* paths – those whose lengths are within, ay, 10% - 20% of the duration of the critical path or those with a total float of less than, say, 10% of the project's uncompleted duration.

Activity-on-arrow Networks

In activity-on-arrow networks activities are represented by links (or arrows) and the nodes represent events of activities (or groups of activities) starting or finishing.

Activity-on-arrow network rules and conventions

1. A project network may have only one start node
2. A project network may have only one end node
3. A link has duration
4. Nodes have no duration
5. Time moves from left to right
6. Nodes are numbered sequentially
7. A network may not contain loops
8. A network may not contain dangles

Figure 6.18 – p. 152 – An activity-on-arrow network
Figure 6.19 – p. 152 – Fragment of a CPM network
Figure 6.20 – p. 153 – A loop represents an impossible sequence
Figure 6.21 – p. 153 – a dangle

Using Dummy Activities

When two paths within a network have a common event although they are , in other respects, independent, a logical error might occur.

For Example:
Suppose that it is necessary to specify a certain piece of hardware before placing an order for it, and before coding the software. Before coding the software it is also necessary to specify the appropriate data structures, although clearly we do not need to wait for this to be done before the hardware is ordered.

Figure 6.23 – p. 155 – The paths with a common node

We can resolve this problem by separating the two independent paths and introduce a dummy activity to link the completion of 'specify hardware' to the start of the activity 'code software'.

Figure 6.24 – p. 155 – Two paths linked by a dummy activity
Figure 6.25 – p. 156 – Another use of a dummy activity

Representing lagged activities

We need to represent these with pairs of dummy activities. Where these activities are lagged because a stage on one activity must be completed before the other may proceed, it is likely to be better to show each stage as a separate activity.

Activity Labelling

One of the more common conventions for labelling nodes, and the one adopted here, is to divide the node circle into quadrants and use those quadrants to show the event number, the latest and earliest dates, and the event slack.

Figure 6.27 – p. 158 – A CPM network after the forward pass
Figure 6.28 – p. 159 – The CPM network after the backwards pass
Figure 6.29 – p. 159 – The critical path

Chapter 7 – Risk Management

Risk

PM-BOK (Project Management Body of Knowledge) defines risk as "*an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives."* PRINCE2 defines risk as "*the chance of exposure to the adverse consequences of future events."*

The key elements of risk:

- It relates to the future
- It involves **cause** and **effect**

Figure 7.1 – p. 164 – Risk planning is carried out primarily in Steps 3 and 6.

Categories of Risk

Kalle Lyytinen and his colleagues have proposed a *sociotechnical* model of risk.

- Structure describes the management structures and systems, including those affecting planning and control.
- Tasks relates to the work planned.

Figure 7.2 – p. 165 – The Lyyntinen-Mathiassen-Ropponen risk framework

A Framework for dealing with risk

Planning for risk includes these steps:

I. Risk Identification
II. Risk analysis and Prioritization
III. Risk Planning
IV. Risk monitoring

Risk Identification

The two main approaches:

- Checklists
- Brainstorming

Checklists are simply lists of the risks that have been found to occur regularly in software development projects.

Table 7.1 – p. 167 – Software project risks and strategies for risk reduction

Brainstorming

Representatives of the main stakeholders should be brought together once some kind of preliminary plan has been drafted. They then identify, using their individual knowledge of different parts of the project, the problems that might occur.

Risk Assessment

A way is needed of distinguishing the damaging and likely risks. This can be done by estimating the *risk exposure* for each risk using the formula:

$$Risk\ Exposure = (Potential\ Damage) \times (Probabilityof\ Occurence)$$

The calculation of risk exposure assumes that the amount of damage sustained will always be the same. It is usually the case that there could be varying amounts of damage. With some risks, there could be not only damage but also gains.

Figure 7.3 – p. 169 – Probability Chart

In Figure 7.3 the 'loss' is measures in days rather than money. In this context, days, or some other unit of personal effort, are often used as a *surrogate* for financial loss.

Most managers resist very precise estimates of loss or of the probability of something occurring, as such figures are usually guesses.

Boehm suggests that planners focus attention on the 10 risks with the highest risk exposure scores. For smaller projects, the focus could be on a smaller number of risks.

Another approach is to use qualitative descriptions of the possible impact and the likelihood of each risk. Consistency between assessors is facilitated by associating each qualitative description with a range of values.

Table 7.4 – p. 170 – Quantitative descriptions of impact on cost and associated range values

In Table 7.4 the potential amount of damage has been categorized in terms of its impact on *project costs.* Other tables could show the impact of risks on *project duration* or on the *quality of the project deliverables.*

Where the potential damage and likelihood of a risk are defined by qualitative descriptors, the risk exposure cannot be calculated by multiplying the two factors together. In this case, the risk exposure is indicated by the position of the risk in a matrix.

Figure 7.4 – A probability impact matrix

Some of the cells in the top right of the matrix have been zone off by a *tolerance line.* Risks that appear within this zone have a degree of seriousness that calls for particular attention.

Risk Planning

Having identified the major risks and allocated priorities, the task is to decide how to deal with them. The choices discussed will be:

- **Acceptance**

This is the do-nothing option.

- **Avoidance**

Some activities may be so prone to accident that it is best to avoid them altogether.

- **Reduction and mitigation**

Here we decide to go ahead with a course of action despite the risks, but take precautions that reduce the probability of the risk.

Risk Reduction attempts to reduce the likelihood of the risk occurring.

Risk mitigation is action taken to ensure the impact of the risk is lessened when it occurs.

- **Transfer**

In this case risk is transferred to another person or organization.

Risk Management

- **Contingency**

This is a planned action to be carried out if the particular risk materializes. The preventative measures mentioned above will usually incur some cost regardless of the risk materializing or not. The cost of a contingency measure will only be incurred if the risk actually materializes.

- **Deciding on the risk actions**

Whatever the countermeasures that are considered, they must be cost-effective. On those occasions where a risk exposure value can be calculated as a financial value using the $(value\ of\ damage) \times (probability\ of\ occurence)$ formula, the cost-effectiveness of a risk reduction action can be assessed by calculating the *risk reduction leverage (RRL).*

$$Risk\ Reduction\ Leverage = \frac{RE_{before} - RE_{After}}{Cost\ of\ risk\ Reduction}$$

RE_{before} is risk exposure, as discussed before. RE_{after} is the risk exposure after taking the risk reduction action.

- **Creating and Maintaining the Risk Register**

When the project planners have picked out and examined what appear to be the most threatening risks to the project, they need to record their findings in a *risk register*.

Figure 7.5 – p. 175 – Risk Register Page

Evaluating Risks to the Schedule

Previously we showed a probability chart – Figure 7.3. This illustrated the point that a forecast of the time needed to do a job is most realistically presented as a graph of likelihood of a range of figures, with the most likely duration as the peak end the chances of the job taking longer or shorter shown as curves sloping down on either side.

A drawback to the application of methods like PERT, is that in practice there is a tendency for developers to work to the schedule even if a task could be completed more quickly.

Applying the PERT technique

Using PERT to evaluate the effects of uncertainty

The method is very similar to the CPM technique but, instead of using a single estimate for the duration of each task, PERT requires three estimates:

- Most Likely time: the time we would expect the task to take under normal circumstances
- Optimistic Time: the shortest time in which we could expect to complete the activity.

- Pessimistic Time: the worst possible time

PERT then combines these three estimates to form a single expected duration, t_e using the formula:

$$t_e = \frac{a + 4m + b}{6}$$

Using expected durations

The expected durations are used to carry out a forward pass through a network, using the same method as the CPM technique. In this case, the calculated event dates are not the earliest possible dates but the dates by which we expect to achieve those events.

Figure 7.6 – p. 178 – The PERT network after the forward pass

Unlike the CPM approach, the PERT method does not indicate the earliest date by which we could complete the project but the expected date. An advantage of this approach is that it places an emphasis on the uncertainty of the real world.

Activity Standard Deviations

A quantitative measure of the degree of uncertainty of an activity duration estimate may be obtained by calculating the standard deviation s of an activity time, using the formula

$$s = \frac{b - a}{6}$$

The activity standard deviation is proportional to the difference between the optimistic and pessimistic estimates, and can be used as a ranking measure of the degree of uncertainty or risk for each activity.

The likelihood of meeting targets

The main advantage of the PERT technique is that it provides a method for estimating the probability of meeting or missing target dates.

The PERT technique uses the following 3-step method for calculating the probability of meeting ore missing a target date:

- Calculate the standard deviation of each project event;
- Calculate the z value for each event that has a target date;
- Convert z values to a probabilities;

Calculating the Standard deviation of each project event

Standard deviations for the project events can be calculated by carrying out a forward pass using the activity standard deviations in a manner similar to that used with expected durations.

There is however, one small difference – to add two standard deviations we must add their squares and then find the square root of the sum.

Figure 7.7 – p. 179 – The PERT network with 3 target dates and calculated event standard deviations.

Exercise 7.7 – p. 180

The standard deviation for event 3 depends solely on that of activity B. The standard deviation for event 3 is therefore 0.33.

For event 5 there are two possible paths, $B + E$ or F. The total standard deviation for path $B + E$ is $\sqrt{(0.33^2 + 0.5^2)} = 0.6$ and that for path F is 1.17; the standard deviation for event 5 is therefore the greater of the two, 1.17.

Verify that the standard deviations for each of the other events in the project are as shown in Figure 7.7.

Calculating the Z values

It is calculated using the formula:

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

where t_e is the expected date and T is the target date.

Converting z values to probabilities

A z value may be converted to the probability of not meeting the target date using the graph in Figure 7.8.

Figure 7.8 – p. 181 – The probability of obtaining a value within z standard deviations of the mean for a normal distribution.

The Advantages of PERT

- We can use the technique to calculate the standard deviation for each task and use this to rank them according to their degree of risk.
- If we can use the expected times and standard deviations for forward passes through the network we can estimate the probability of meeting any set target.

Critical Chain Concepts

This chapter has stressed the idea that the forecast for the duration of an activity cannot in reality be a single number, but must be a range of durations. However, we would want to pick one value in that range which would be the *target.*

The duration chosen as the target might be the one that seems *most likely.*

One approach which attempts to solve some of these problems is the application of the *critical chain* concept.

Figure 7.11 – p. 184 – Grantt Chart – 'traditional' planning approach

Deriving 'most likely' activity durations

The target date generated by critical chain planning is one where it is estimated that there is a 50% chance of success – his approximates to the expected time identified in the PERT risk method.

A good approach would be to ass developers to supply two estimates. One of these would be a 'most likely' estimate and the other would include a safety margin or comfort zone. From now on we are going to assume that this is what has happened.

Using latest start dates for activities

Working backward from the target completion date, each activity is scheduled to start as late as possible. This should reduce the chance of staff being pulled off the project on to other work. It is also argued, that most developers would tend to start work on the task at the latest start time anyway.

However, if one is late, the whole project is late.

Inserting project and feeder buffers

To cope with activity overruns, a **project buffer** is inserted at the end of the project before the target completion date. One way to calculate this buffer is as the equivalent of 50% of the sum of the lengths of the 'comfort zones' that have been removed from the *critical chain.*

A **resource dependency** is where one activity has to wait for a resource which is being used by another activity to become available.

An alternative proposal is to sum the squares of the comfort zones and then take the square root of the total. This is based on the idea that each comfort zone is the equivalent to the standard deviation of the activity.

Buffers are also inserted into the project schedule where a subsidiary chain of activities feeds into the critical chain. These *feeding buffers* could once again be set at 50% of the length of the 'comfort zone' removed from the subsidiary or *feeding chain.*

Project Execution

When the project is executed, the following principles are followed:

- No chain of tasks should be started earlier than scheduled, but once it has been started, it should be finished as soon as possible – this invokes the *relay race principle,* where developers should be ready to start their tasks as soon as the previous, dependent, tasks are completed.
- Buffers are divided into three zone:
    - Green – where no action is required
    - Amber – where an action plan is formulated if the project completion dates moves into this zone
    - Red – where the action plan above is executed if the project penetrates this zone

Conclusion – p. 188
Further Exercises – p. 188

Chapter 8 – Resource Allocation

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

- An *activity schedule* indicating the planed start and completion dates for each activity
- A *resource schedule* showing the dates on which each resource will be required and the level of that requirement
- A *cost schedule* showing the planned cumulative expenditure incurred by the use of resources over time

The Nature of Resources

In general, resources will fall into one of 7 categories:

1. **Labour** – the main items in this category will be members of the development project team such as the project manager, systems analyst and software developers.
2. **Equipment** – Obvious items include workstations and other computing and office equipment.
3. **Materials** – Materials are the items consumed, rather than equipment that is used.
4. **Space** – For projects that are undertaken with existing staff, space is normally readily available. If staff is recruited, more space will be required.
5. **Services** – some projects will require procurement for specialist services, development of a wide area distributed system, for example, requires scheduling of telecommunications services.
6. **Time** – Time is the resource that is being offset against other primary resources.
7. **Money** – Money is a secondary resource, it is used to buy resources and will be consumed as other resources are used.

Identifying resource requirements

The 1st step in producing a resource allocation plan is to list the resources that will be required along with the expected level of demand. This will normally be done by considering each activity in turn and identifying the resources required.

Scheduling resources

Having produced the resource requirement list, the next step is to map this on the activity plan to assess the distribution of resources required over the duration of the project. This is best done by representing the activity plan as a bar chart and using this to produce a resource histogram for each resource.

Figure 8.3 – p. 197 – Part of Amanda's bar chart and resource histogram for analyst/designers

Changing the level of resources on a project over time, particularly personnel, generally adds to the cost of a project. Recruiting staff has costs and, even where staff are transferred internally, time will be needed for familiarization with the new project environment.

Figure 8.4 – p. 198 – A resource histogram showing demand for staff before and after smoothing

By delaying the start of some of the activities, it has been possible to smooth the histogram and reduce the maximum level of demand for the resource.

In practice, resources have to be allocated to a project on an activity-by activity basis and finding the 'best' allocation can be time consuming and difficult. It is therefore helpful to prioritize activities so that resources can be allocated to competing activities in some rational order.

Of the various ways of prioritizing activities, two are described below:

1. **Total Float Priority** – Activities are ordered according to their total float, those with the smallest total float having the highest priority.
2. **Ordered List Priority** – With this method, activities that can proceed at the same time are ordered according to a set of simple criteria.

For Example:

Burman's priority list takes into account activity durations as well as total float:

1. Shortest Critical Activity
2. Critical Activities
3. Shortest non critical activity
4. Non critical activity with least float
5. Non critical activities

Creating Critical paths

Scheduling resources can create new critical paths. Delaying the start of an activity because of lack of resources will cause that activity to become critical if this used up its float. Delaying one activity can delay the availability of a resource required for a later activity.

Figure 8.6 – p. 202 – Amanda's project scheduled to require three analyst/designers

Being Specific

When allocating labourers to activities in a large building project we need not distinguish among individuals – there are likely to be many labourers and they may be treated as equals so far as skills and productivity are concerned.

This is seldom the case with software projects. Because of the nature of software development, skill and experience play a significant part in determining the time taken and, potentially, the quality of the final project.

In allocating individuals to tasks, a number of factors need to be taken into account:

1. **Availability** – We need to know whether a particular individual will be available when required.
2. **Critically** – allocation of more experienced personnel to activities on the critical path often helps in shortening project durations or at least reduces the risk of overrun.
3. **Risk** – Identifying those activities posing the greatest risk and knowing factors influencing them, helps to allocate staff.
4. **Training** – It will benefit the organization if positive steps are taken to allocate junior staff to appropriate non critical activities where there will be sufficient slack for them to train and develop skills.
5. **Team Building** – The selection of individuals must also take account of the final shape of the project team and the way they will work together.

Publishing the Resource Schedule

Although good as planning tools, they are not the best way of publishing and communicating project schedules. For this we need some form of work plan. Work plans are commonly published as either lists or charts.

Figure 8.7 – p. 205 – Amanda's work Schedule

Cost Schedules

It is now time to produce a detailed cost schedule showing weekly or monthly costs over the life of the project.

Calculating cost is straightforward where the organization has standard cost figures for staff and other resources. Where this is not the case, then the project manager will have to calculate the costs.

In general, costs are categorized as follows:

1. **Staff costs** – These will include staff salaries as well as the other direct costs of employment such as the employer's contribution to social security funds, pension scheme contributions, holiday pay and sickness benefit.
2. **Overheads** – Overheads represent expenditure that an organization incurs, which cannot be directly related to individual projects or jobs, including space rental, interest charges and the costs of service departments.
3. **Usage Charges** – In some organizations, projects are charged directly for use of resources such as computer time.

The Scheduling Sequence

In practice successful resource allocation often necessitates revisions to the activity plan, which, in turn, will affect our risk assessment. Similarly, the cost schedule might indicate the need or desirability to reallocate resources or revise activity plans.

The interplay between the plans and schedules is complex, any change to any one will affect each of the others.

Successful project scheduling is largely dependent upon the skill and experience of the project manager in juggling the many factors involved.

Conclusion – p. 209
Further Exercises – p. 209

Chapter 9 – Monitoring and Control

Creating the Framework

Exercising control over a project and ensuring that targets are met is a matter of regular monitoring – finding out what is happening and comparing it with targets.

Figure 9.1 – p. 213 – The project control cycle

In practice we are normally concerned with 4 types of shortfall – delays in meeting targets dates, shortfalls in quality, inadequate functionality, and costs going over target.

Responsibility

The overall responsibility for ensuring satisfactory progress on a project is often the role of the *project steering committee.* Day-to-day responsibility will rest with the project manager and, in all but the smallest of projects; aspects of this can be delegated to team leaders.

Figure 9.2 – p. 214 – Project Reporting Structures

Reporting may be oral or written, formal or informal, and regular or ad hoc. Informal communication is necessary and important, but any such informal reporting of project progress must be complemented by formal reporting procedures.

Assessing Progress

Some information used to assess project progress will be collected routinely, while other information will be triggered by specific events. Wherever possible, this information should be objective and tangible.

Setting Checkpoints

It is essential to set a series of checkpoints in the initial activity plan.

Checkpoints may be:

- Regular
- Tied to specific events

Taking snapshots

The frequency of progress reports will depend upon the size and degree of risk of the project. In general, the higher the level, the less frequent and less detailed the reporting needs to be.

At the individual developers, strong arguments exist for formal weekly collection of information. This ensures that information is provided while memories are still relatively fresh and provides a mechanism for individuals to review and reflect upon their progress.

Major, or project-level progress reviews will generally take place at particular points during the life of a project – commonly known as *review points* or *control points.*

Collecting the Data

Partial Completion Reporting

Many organizations use standard accounting systems with weekly timesheets to charge staff time to individual jobs. The staff time booked to a project indicates the work carried out and the charges to the project. It does not tell the project manager what has been produced or whatever tasks are on schedule.

It is common to adapt or enhance existing accounting data collection systems to meet the needs of project control.

Figure 9.3 – p. 217 – A weekly timesheet and progress review form

Red/amber/green (RAG) reporting

One popular way of overcoming the objections to partial completion reporting is to avoid asking for estimated completion dates, but to ask instead for the team members' estimates of the likelihood of meeting the planned target date.

One way of doing this is the traffic-light method. This consists of the following steps:

- Identify the key elements for assessment in a piece off work (1st level)
- Break these key elements into constituent elements (2nd level)
- Assess each of the 2nd level elements on the scale ***green*** from 'on target', ***amber*** for 'not on target but recoverable' and ***red*** for 'not on target and recoverable only with difficulty'
- Review all the 2nd level assessments to arrive at 1st level assessments
- Review 1st and 2nd level assessments to produce an overall assessment

Visualizing Progress

The Gantt Chart

This is essentially an activity bar chart indicating scheduled activity dates and durations, frequently augmented with activity floats.

Reported progress is recorded on the chart and a 'today cursor' provides an immediate visual indication of which activities are ahead or behind schedule.

Figure 9.5 – *p. 219* – Part of Amanda's Gantt chart with the 'today cursor' in week 17

The Slip Chart

A slip chart is a very similar alternative favoured by some project managers who believe it provides a more striking visual indication of those activities that are not progressing to schedule.

Figure 9.6 – *p. 220* – The slip chart emphasizes the relative position of each activity

The Timeline

One disadvantage of charts shown so far is that they do not show clearly the slippage of the project completion date through the life of the project. Analysing and understanding trends in the project so far allows us to predict the future progress of the project.

The timeline chart is a method of recording and displaying the way in which targets have changed throughout the duration of the project.

Figure 9.7 – *p. 221* – Brigitte's timeline chart at the end of week six

This timeline chart is useful both during the execution of a project and as part of the post-implementation review. Analysis of the timeline chart, and the reasons for the changes, can indicate

failures in the estimation process or other errors that might, with that knowledge, be avoided in the future.

Cost Monitoring

A cumulative expenditure chart provides a simple method of comparing actual and planned expenditure.

Figure 9.8 – *p. 222* – Tracking cumulative expenditure

Cost charts become more useful if we add projected future costs calculated by adding the estimated costs of uncompleted work to the costs already incurred.

Figure 9.9 – p. 223 – The cumulative expenditure chart can also show revised estimates of cost and completion date.

Earned Value Analysis

Earned value analysis is based on assigning a 'value' to each task or work package based on the original expenditure forecasts.

The assigned value is the original budgeted cost for the item and is known as the *planned value (PV)* or *budgeted cost of work scheduled (BCWS).*

A task that has not started is assigned an earned value of zero and when it has been completed, it, and hence the project, is credited with the original planned value of the task.

The total value credited to a project at any point is known as the *earned value (EV)* or budgeted cost of work performed (BCWP) and this can be represented as a money value, and amount of staff or as a percentage of the PV.

Where the tasks have been started but are not yet complete, some consistent method of assigning an earned value must be applied. Common methods in software projects are:

- ***The 0/100 technique:*** where a task is assigned a value of 0 until such time that it is completed when it is given a value of 100% of the budget value;
- ***The 50/50 technique:*** where a task is assigned a value of 50% of its value as soon as it is started and then given a value of 100% once it is complete
- ***The 75/25 technique:*** where the task is assigned 75% on starting and 25% on completion
- ***The milestone technique:*** where a task is given a value based on the achievement of milestones that have been assigned values as part of the original budget plan
- ***Percentage complete:*** in some cases there may have been a way of objectively measuring the amount of work completed

The baseline budget

The 1st stage in setting up an earned value analysis is to create the *baseline budget.* This is based on the project plan and shows the forecast growth in earned value through time. In project such as software development, it is common to measure earned value in person-hours or workdays.

Table 9.2 – p. 225 – Amanda's baseline budget calculation

Figure 9.10 – p. 225 – Amanda's baseline budget

Monitoring earned value

Having created the baseline budget, the next task is to monitor earned value as the project progresses. This is done by monitoring the completion of tasks. As well as recording the EV, the actual cost of each task can be collected as *actual cost (AC).* This is also known as the *actual cost of work performed (ACWP).*

Figure 9.11 – p. 226 – Amanda's earned value analysis at week 12

Schedule Variance (SV)

The SV is measured in cost terms as EV – PV and indicates the degree to which the value of completed work differs from that planned.

Figure 9.12 – p. 227 – An earned value tracking chart

Time Variance (TV)

This is the difference between the time when the achievement of the current earned value was planned to occur and the time now.

Cost Variance (CV)

This is calculated as EV – AC and indicates the difference between the earned value or budgeted cost and the actual cost of completed work.

Performance Ratios

Two ratios are commonly tracked:
The cost performance index (CPI = EV/AC) and the schedule performance index (SPI = EV/PV).

The two ratios can be thought of as a 'value-for-money' indices. A value greater than one indicates that work is being completed better than planned, whereas a value of less than one means that work is costing more than and/or proceeding more slowly than planned.

CPI can be used to produce a revised cost estimate for the project (Or *estimate at completion – EAC).* EAC is calculated as BAC/CPI where BAC (budget at completion) is the current projected budget for the project.

Similarly, the current SPI can be used to project the possible duration of the project in x months – in earned value terminology this is the *schedule at completion (SAC).* A time estimate completion (TEAC) can be calculated as SAC/SPI.

Figure 9.13 – p. 228 – An earned value chart with revised forecasts

Prioritizing Monitoring

In this section we list the priorities we might apply in deciding levels of monitoring:

- ***Critical Path Activities*** – Any delay in an activity on the critical path will cause a delay in the completion date for the project.

- ***Activities with no free float*** – a delay in any activity with no free float will delay at least some subsequent activities even though, if the delay is less than the total float, it might not delay the project completion date.

- ***Activities with less than a specified float*** – If any activity has very little float it might use up its float before the regular activity monitoring brings the problem to the project manager's attention.

- ***High-Risk activities*** – A set of high risk activities should have been identified as part of the initial risk profiling exercise.

- ***Activities using critical resources*** – Activities can be critical because they are very expensive (as in the case of specialized contract programmers).

Getting the project back to target

There are two main strategies to consider when drawing up plans to bring the project back on target – shortening the critical path or altering the activity precedence requirements.

Shortening the Critical Path

There are several ways in which this might be done:

1. Adding resources
2. Increase use of current resources
3. Reallocate staff to critical activities
4. Reduce scope
5. Reduce quality

Reconsider the precedence requirements

The next step is to consider the constraints by which some activities have to be deferred pending completion of others.

One way to overcome precedence constraints is to subdivide an activity into a components that can start immediately and one that is still constrained as before. If we decide to alter the precedence requirements in such a way, it is clearly important to be aware that quality might be compromised to make a considered decision to compromise quality where needed.

Maintaining the business case

In making decisions about the management of the project, the main concern of the project sponsor, that is, the stakeholder who is putting up the money for the project, is whether the business case for the project has been preserved. The value of the benefits of a project must be greater than its cost for the project to be viable.

If costs increase, then this reduces the value of the benefits.

Exception Planning

The project manager will normally be allowed to change the detail of a plan as long as the agreed project outcomes are produced on time and within budget.

In some cases, an operational change may affect other stakeholders. In such a case the project manager would need to gain the acceptance of these stakeholders.

Some changes to the plan might have an impact on the delivery date, project scope or costs. These could affect the business case. Here the project manager would need to gain the approval of the business sponsors.

One approach is to require the project manager to write an *exception report* that explains the reasons for the deviation from the existing plan.

Change control

At some point what is assumed to be the final version will be created. This is *base-lined,* effectively frozen. Baselined products are the foundation for the development of further products. Thus any changes to the baselined products could have knock-on effects on other parts of the products of a project.

Change control procedures

1. One or more users might perceive a need for a modification to a system and ask for a change request to be passed to the development staff.

2. The user management would consider the change request and, if they approve it, pass it to the development management. It is important that there is a single authorized channel for *request for change (RFCs)* between the client community and the management of the developers.

3. There would be one person within the development area who would receive and process RFCs. They would delegate a member of staff to look at the request and to report on the practicality and cost of carrying out the change.

4. The development representative would report back to the user management on the findings and the user management would decide whether, in view of the cost quoted, they wish to go ahead.

5. There would be some individual or group who represented the major stakeholders, both users and developers and also the project sponsor, who would have the authority to prioritize the RFCs for action. Ultimately this would be the Project Board or equivalent.

6. Once an RFC has been approved for action, one or more developers are authorized to take copies of the master products that are to be modified. This would need to be done through the configuration librarian

7. The copies are modified. In the case of software components this would involve modifying the code and recompiling and testing it.

8. When the development of new versions of the product has been completed the user management will be notified and copies of the software will be released for user acceptance.

9. When the user is satisfied that the products are adequate they will authorize their operational release. The master copies of configuration items will be replaced.

Changes in scope of a system

A common occurrence with IS development projects is for the size of the system gradually to increase. One cause of this is changes to requirements that are requested by users.

The scope of a project needs to be carefully monitored and controlled. One way is to estimate the system size in terms of SLOC or function points at key milestones.

Configuration Librarian's Role
Control changes and documentation ought to be the responsibility of someone who may variously be named the configuration librarian, the configuration manager or the project librarian.

Duties include:

- The identification of all items that are subject to change control
- The establishment and maintenance of a central repository of the master copies of all project documentation and software products
- The setting up and running of a formal set of procedures to deal with changes
- The maintenance of record of who has access to which library items and the status of each library item

Conclusion – p. 235
Further exercises – p. 235