ATME COLLEGE OF ENGINEERING

13th KM Stone, Bannur Road, Mysore - 560 028

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

LECTURE NOTES

COURSE: SOFTWARE ENGINEERING & PROJECT MANAGEMENT

COURSE CODE: BCS501

SEMESTER: V

COURSE COORDINATOR: Mr. Chetan G

(ACADEMIC YEAR 2026-27)

MODULE 4

1.1 INTRODUCTION TO SOFTWARE PROJECT MANAGEMENT

1. Software Project Management is an art & Science of planning & leading software Projects from **ideas to reality.**
2. A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product
3. Project management is the discipline of defining and achieving targets while optimizing the new resources (time, money, people, materials, energy, space , etc.) over the course of a project (a set of activities of finite duration).
4. Project management involves the planning, monitoring, and control of people, process, and events that occur during software development.

Everyone manages, but the scope of each person's management activities varies according his or her role in the project.

Software needs to be managed because it is a complex undertaking with a long duration time.

Managers must focus on the fours P's to be successful (people, product, process, and project).

A project plan is a document that defines the four P's in such a way as to ensure a cost effective, high quality software product.

The only way to be sure that a project plan worked correctly is by observing that a high-quality product was delivered on time and under budget.

1.2 WHY IS SOFTWARE PROJECT MANAGEMENT IMPORTANT ?

- Large amounts of money are spent on ICT (information and communication technology) e.g. UK government in 2003-04 spent € 2.3 billions on contracts for ICT and only € 1.4 billion on road building. (1 billion =100 crore).
- Project often fail – Standish Group claim only a third of ICT projects are successful. 82 % were late and 43 % exceeded their budget. Poor project management a major factor in these failures.
- The methodology used by the Standish Group to arrive at their findings has been criticized,

but the general perception of the prevalence of ICT project failure is still clear.

Software Development Life Cycle:

The Software Development life cycle is a methodology that also forms the framework for planning and controlling the creation, testing, and delivery of an information system.

The software development life cycle concept acts as the foundation for multiple different development and delivery methodologies, such as the Hardware development life -cycle and software development life -cycle . While Hardware development life -cycle deal specially with hardware and Software development life -cycle deal with software, a systems development life -cycle differs from each in that it can deal with any combination of hardware and software , as a system can be composed of hardware only , software only, or a combination of both.

Four Project Dimensions:

- People
- Process
- Product
- Technology

The 5 Variables of Project Control

1. Time: amount of time required to complete the project.
2. Cost: calculated from the time variable
3. Quality: The amount of time put into individual tasks determines the overall quality of the project.
4. Scope: Requirements specified for the end result.
5. Risk: Potential points of failure.

Trade -off Triangle

The triangle illustrates the relationship between three primart forces in a project. Time is the available time to deliver the project. Cost represents the amount of money or resources available and quality represents the fit-to-purpose that the project must achieve to be a scuccess.

The normal situation is that one of thse factors is fixed and the other two will vary in inverse proportion to each other. For example , time is often fixed and the quality of the end product will depend on the cost and resources available. Similarly if you are working to a fixed level of quality then the cost of the project will largely be dependable upon the time available(if you have longer you can do it with fewer people).

Software project management is particularly important due to the unique challenges and complexities associated with software development.

1. Complexity Management
2. Requirement Management
3. Time and Budget Control
4. Risk Management
5. Quality Assurance
6. Team Coordination
7. Stakeholder Management
8. Scope Management
9. Process Improvement
10. Resource Allocation

1. **Complexity Management**
   - Software projects often involve intricate systems and interdependencies. Effective management of this complexity ensures that the project remains coherent and manageable.
2. **Requirement Management**
   - Clear and precise requirement management is essential to ensure that the final product meets user needs and expectations. Mismanagement here can lead to scope creep and project failure.
3. **Time and Budget Control**
   - Monitoring and controlling the project timeline and budget is vital. This includes planning, estimating, and adhering to schedules and financial constraints to prevent overruns.
4. **Risk Management**

Identifying, assessing, and mitigating risks can prevent unforeseen issues from derailing a

project. This proactive approach helps in managing uncertainties effectively.

5. **Quality Assurance**
   - Ensuring that the project meets quality standards is crucial for user satisfaction and reducing post-release defects. Continuous testing and validation are key practices.
   -
6. **Team Coordination**
   - Effective communication and coordination among team members are essential for collaboration and timely problem-solving, ensuring that everyone is aligned with project goals.
   -
7. **Stakeholder Management**
   - Engaging and managing stakeholders helps in gaining their support and addressing their concerns, which is critical for project acceptance and success.
8. **Scope Management**
   - Defining and controlling what is included in the project prevents scope creep, ensures that all necessary features are delivered, and avoids unnecessary work.
9. **Process Improvement**
   - Continuously improving processes ensures that the project is using the most efficient methods and practices, leading to better performance and outcomes.
10. **Resource Allocation**
    - Efficient allocation and management of resources (human, financial, and material) ensure that the project has what it needs to succeed without wastage.

Statistics Highlighting the Importance of Efficient Project Management

1. **32% of Projects Fail Due to Poor Management**
   - This statistic underscores the critical impact of project management on the overall success of software projects. Poor management can lead to project failures, highlighting the need for skilled project managers.
2. **68% of Projects Fail to Meet Deadlines, Budgets, and Quality Targets**
   - This indicates that a significant majority of projects struggle with time, budget, and quality control. Effective project management practices in these areas can significantly improve success rates.
3. **97% of Businesses Believe Project Management is Essential for Success**
   - This near-unanimous belief among businesses highlights the recognized value of project management. It underscores that investing in good project management practices is seen as crucial for achieving business objectives.
4. **80% of High-Performing Projects are Led by a Project Manager with Qualifications**
   - This shows a clear correlation between the qualifications of the project manager and the performance of the project. Qualified project managers bring skills and knowledge that drive project success.
5. **On Average, a Large IT Project Runs 45% Over Budget**
   - This statistic points to common budget overruns in large IT projects, emphasizing the need for rigorous budget control and efficient resource management to prevent financial overshooting.

Conclusion

Effective software project management is essential due to the inherent complexities and challenges of software development. The key areas outlined require diligent attention and management to ensure project success. The statistics provided illustrate the high stakes involved and the substantial impact that good project management can have on the success rates of software projects. By focusing on these areas, businesses can significantly improve their chances of delivering successful projects that meet deadlines, stay within budget, and satisfy quality standards.

1.3 WHAT IS A PROJECT:

The definition of a project as being planned assume that to a large extent we can determine how we are going to carry out a task before we start. There may be some projects of an exploratory nature where this might be quite hard. Planning is in essence thinking carefully about something before you do it and even in the case of uncertain projects this is worth doing as long as it is accepted that the resulting plans will have provisional and speculative elements. Other activities, concerning, for example, to routine maintenance, might have been performed so many times that everyone involved knows exactly what needs to be done. In these cases, planning hardly seems necessary, although procedures might need to be documented to ensure consistency and to help newcomers to the job.

Dictionary definitions of 'project' include:

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

Key points above are planning and size of task

Here are some definitions of 'project'. No doubt there are other ones: for example,

'Unique process, consisting of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements, including constraints of time, cost and resources. There is a hazy boundary between the non-routine project and the routine job. The first time you do a routine task, it will be like a project. On the other hand, a project to develop a system similar to previous ones you have developed will have a large element of the routine.

1.3.1 Characteristics of Project:

- Non-routine tasks are involved
- Planning is required
- Specific objectives are to be met or a specified product is to be created
- The project has a pre-determined time span
- Work is carried out for someone other than yourself

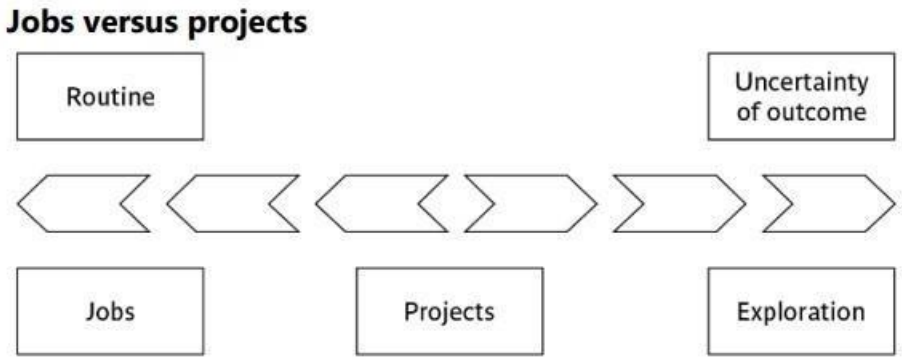

Fig:1.1 Activities most likely to benefit from project management.

- Work involves several specialism's
- Work is carried out in several phases
- The resources that are available for use on the project are constrained
- The project is large or complex.

The project that employs 20 developers is likely to be disproportionately more difficult than one with only 20 staff because of the need for additional coordination.

1.3.2 Software Projects versus Other Types of Project:

Many of the techniques of general project management are applicable to software project management. One way of perceiving software project management is as the process of making visible that which is invisible.

Invisibility: When a physical artifact such as a bridge or road is being constructed the progress being made can actually be seen. With software, progress is not immediately visible.
Complexity: Software products contain more complexity than other engineered artifacts.

Conformity: The 'traditional' engineer is usually working with physical systems and physical materials like cement and steel. These physical systems can have some complexity, but are governed by physical laws that are consistent. Software developers have to conform to the requirements of human clients. It is not just that individuals can be inconsistent.

Flexibility: The ease with which software can be changed is usually seen as one of its strengths. However, this means that where the software system interfaces with a physical or organizational system, it is expected that, where necessary, the software will change to accommodate the other components rather than vice versa. This means the software systems are likely to be subject to a high degree of change.

An example for infrastructure project is construction of a flyover. An example for a software project is development of a payroll management system for an organization using Oracle l0g and Oracle Forms 10G.

1.4 CONTRACT MANAGEMENT

- **ln-house projects** are where the users and the developers of new software work for the same organization.
- However, 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.
- Thus, the project manager will not worry about estimating the effort needed to write individual software components as long as the overall project is within budget and on time. On the supplier side, there will need to be project managers who deal with the more technical issues.

- **Contract management** is the process of managing the creation, execution, and analysis of contracts to maximize operational and financial performance and minimize risk.
- It involves various activities from the initial request for a contract, through negotiation, execution, compliance, and renewal. Effective contract management ensures that all parties to a contract fulfill their obligations as efficiently as possible.

<u>Various Stages of Contract Management</u>

1. Request and Creation:

Request: Identifying the need for a contract and gathering the necessary information to draft it.

Creation: Drafting the contract terms and conditions that align with the requirements and objectives of all parties involved.

Example: A software company needs to hire a third-party developer to work on a new project. *The project manager identifies the need for a contract and gathers details about the scope of work, timelines, payment terms, and other specifics.*

2. **Negotiation**:

   Parties involved discuss and negotiate the terms of the contract to reach a mutual agreement. This stage often involves revisions and adjustments.

   Example: The software company and the third-party developer negotiate the terms. The developer might request more time or a higher payment, while the company might request milestones for progress checks. Approval and Execution:
   Approval: Obtaining necessary approvals from stakeholders and legal departments.
   Execution: Signing the contract, making it a legally binding document.

   Example: Once the terms are finalized, the contract is reviewed by both parties' legal teams. After approval, both the software company and the developer sign the contract.

3. **Obligations and Performance:**

   Ensuring that all parties adhere to the terms and conditions agreed upon in the contract. Monitoring performance and compliance.

   Example: The developer starts working on the project, adhering to the deadlines and deliverables specified in the contract. The software company provides the necessary resources and makes payments as per the contract.

4. **Modification and Renewal:**

   Making necessary amendments if any changes occur during the contract period. Reviewing and renewing contracts as needed.

   Example: Midway through the project, the software company requests additional features not covered in the original contract. An amendment is made to include these new features and

adjust the payment terms accordingly. As the project nears completion, the company and

developer may negotiate a renewal for ongoing maintenance.

5. **Closure:**

Completing all contractual obligations, ensuring all parties have met their requirements, and formally closing the contract.

Example: The developer finishes the project, and the software company conducts a final review to ensure all deliverables meet the agreed-upon standards. Once confirmed, the contract is closed, and a final payment is made.

Benefits of Effective Contract Management

Risk Mitigation: Identifies and manages potential risks early in the contract lifecycle.

Improved Compliance: Ensures that all parties comply with legal and regulatory requirements.

Cost Savings: Avoids unnecessary costs and penalties by managing contracts efficiently.

Performance Tracking: Monitors performance against contract terms to ensure objectives are met.

Relationship Management: Maintains positive relationships between contracting parties through clear and consistent communication.

Speed to Market: Accelerates project timelines by leveraging the vendor's expertise and resources.

COMPANY: XYZ Tech

1. **Identifying Needs:** XYZ Tech identifies a need for a mobile app to complement its existing software suite.

2. **Selecting a Vendor**: XYZ Tech shortlists several development firms based in India, known for their expertise in mobile app development.

3. **Defining Requirements**: XYZ Tech provides detailed specifications, including desired

features, user interface design, and performance metrics.

4. **Contract Negotiation**: XYZ Tech negotiates terms, focusing on deliverables, timelines, and confidentiality clauses.

5. **Project Management**: XYZ Tech assigns a project manager to liaise with the vendor, ensuring regular updates and adherence to milestones.

6. **Delivery and Integration**: The vendor delivers the app, which is integrated with XYZ Tech's software suite after thorough testing.

7. **Post-Delivery Support:** The vendor provides ongoing maintenance and support, addressing any post-launch issues. By following these steps and learning from real-world examples, software companies can effectively outsource projects to thirdparty vendors, ensuring successful project completion and maximizing business value.

1.5 ACTIVITIES COVERED BY SOFTWARE PROJECT MANAGEMENT:

The activities covered by Software Project management are diagrammatically illustrated as follows:

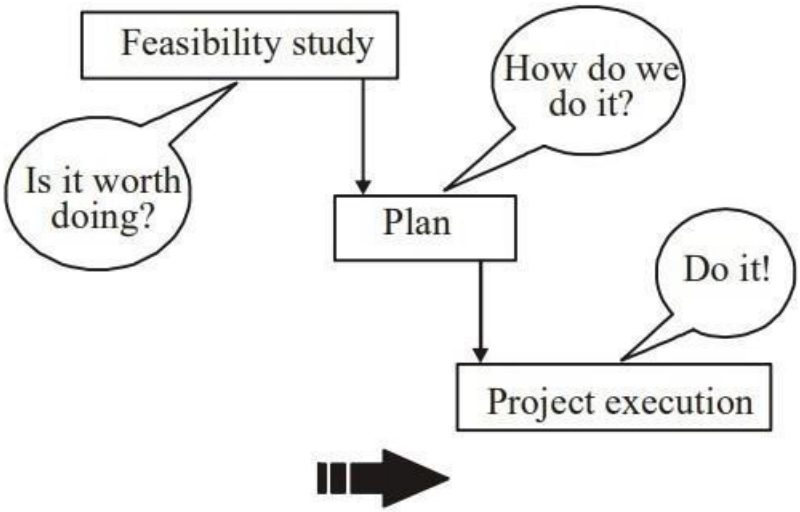

Figure 1.1: The Feasibility Study / Plan / Execution Cycle

1.5.1 The Feasibility Study:

This is an investigation into whether a prospective project is worth starting that it has a valid *business case*. Information is gathered about the requirements of the proposed application. The probable developmental and operational costs, along with the value of the benefits of the new system, are estimated. The study could be part of a strategic planning exercise examining and prioritizing a range of potential software developments.

1.5.2 Planning:

If the feasibility study produces results which indicate that the prospective project appears viable, planning of the project can take place. However, for a large project, we would not do all our detailed planning right at the beginning. We would formulate an outline plan for the whole project and a detailed one for the first stage. More detailed planning of the later stages would be done as they approached. This is because we would have more detailed and accurate information upon which to base our plans nearer to the start of the later stages.

1.5.3 Project Execution:

The project can now be executed. The execution of a project often contains *design* and *implementation* subphases. The same is illustrated in **Figure 1.2** which shows the typical sequence of software development activities recommended in the international standard ISO 12207.

1.5.3.1 Requirements Analysis:

This starts with requirement elicitation or requirement gathering which establishes what the users require of the system that the project is to implement. Some work along these lines will almost certainly have been carried out when the project was evaluated, but now the original information obtained needs to be updated and supplemented.

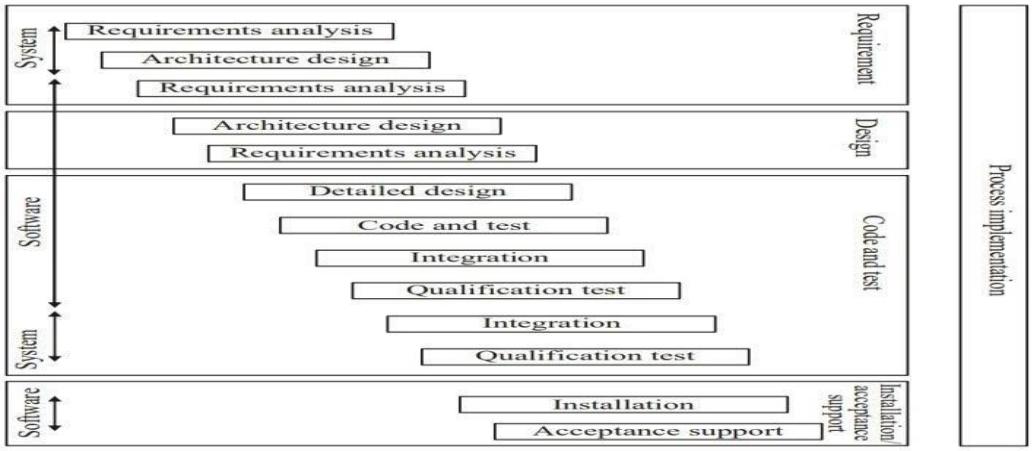

Figure 1.2: Sub Phases in Project Execution

1.5.3.2 Specification:

Detailed documentation of what the proposed system is to do.

1.5.3.3 Design:

A design has to be drawn up which meets the specification. This design will be in two stages. One will be the external or user design concerned with the external appearance of the application. The other produces the physical design which tackles the way that the data and software procedures are to be structured internally.

- **Architecture Design:** This maps the requirements to the components of the system that is to be built. At the system level, decisions will need to be made about which processes in the new system will be carried out by the user and which can be computerized. This design of the system architecture thus forms an input to the development of the software requirements. A second architecture design process then takes place which maps the software requirements to software components.

1.5.3.4 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. Coding:This may refer to writing code in a procedural language or an object-oriented language or could refer to the use of an application-builder. Even where software is not being built from scratch, some modification to the base package could be required to meet the needs of the new application.

1.5.3.5 Testing(Verification and Validation):

Whether software is developed specially for the current application or not, careful testing will be needed to check that the proposed system meets its requirements.

Integration: The individual components are collected together and tested to see if they meet the overall requirements. Integration could be at the level of software where different software components are combined, or at the level of the system as a whole where the software and other components of the system such as the hardware platforms and networks and the user procedures are brought together.

Qualification Testing: The system, including the software components, has to be tested carefully to ensure that all the requirements have been fulfilled.

1.5.3.6 Implementation/ Installation:

Some system development practitioners refer to the whole of the project after design as

‘implementation’ (that is, the implementation of the design) while others insist that the term

refers to the installation of the system after the software has been developed.

1.5.3.7 Acceptance Support:

Once the system has been implemented there is a continuing need for the correction of any errors that may have crept into the system and for extensions and improvements to the system. Maintenance and support activities may be seen as a series of minor software projects.

1.6 PLANS, METHODS AND METHODOLOGIES

A plan for an activity must be based on some idea of a *method of work.* To take a simple example, if you were asked to test some software, even though you do not know anything about the software to be tested, you could assume that you would need to:

- Analyze the requirements for the software
- Devise and write test cases that will check that each requirement has been satisfied
- Create test scripts and expected results for each test case
- Compare the actual results and the expected results and identify discrepancies

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, identifying for each activity:

- Its start and end dates
- Who will carry it out?
- What tools and materials will be used?

'Materials' in this context could include information, for example a requirements document. With complex procedures, several methods may be deployed, in sequence or in parallel. The output from one method might be the input to another. Groups of methods or techniques are often referred to as methodologies.

1.7 SOME WAYS OF CATEGORIZING SOFTWARE PROJECTS

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

- Changes to the characteristics of software projects
- Voluntary systems (such as computer games) versus compulsory systems e.g. the order processing system in an organization
- Information systems versus embedded systems.
- **S**oftware Products verses services
- Product-development versus outsourced.
- Object-driven development.

1.7.1 Changes to the characteristics of software projects

- Over the last few decades, the characteristics of software projects have undergone drastic changes. In earlier days of software development every software was being written from scratch and there was no code reusability. In contrast, at present almost every programming language supports ways of reusing existing code, by customizing and extending existing code, efficiently and dynamically linking library routines and support for frameworks.
- Project durations have now shrunk to only a few months compared to multi-year projects.
- In past, customer participation in software projects was largely restricted to only initial interactions, gathering and specification and taking delivery of the developed software, but at present, customer participation in almost every aspect of a project.

1.7.2 Compulsory versus voluntary users

In workplaces there are systems that staff have to use if they want to do something, such as recording a sale. However, 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 much on the informed ingenuity of the developers, along with techniques such as market surveys, focus groups and prototype evaluation.

1.7.3 Information 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. A stock control system would be an information system. An embedded, or process control, system might control the air conditioning equipment in a building. Some systems may have elements of both where, for example, the stock control system also controls an automated warehouse.

1.7.4 Software Products verses services

All types of software projects can broadly be classified into software product development projects and software services projects. It can be further classified as shown in below Fig.1.7 A software product development concerns developing the software by keeping the requirements to the general customers in mind and developed software is usually sold-off-the helf to a large number of customers. Examples of generic software development are Microsoft's Windows operating system and Oracle Corporatism's Oracle 8i database management software. Domain-specific software targets specific segments of customers(verticals) Example BANCS from TCS. FINACLE from Infosys.

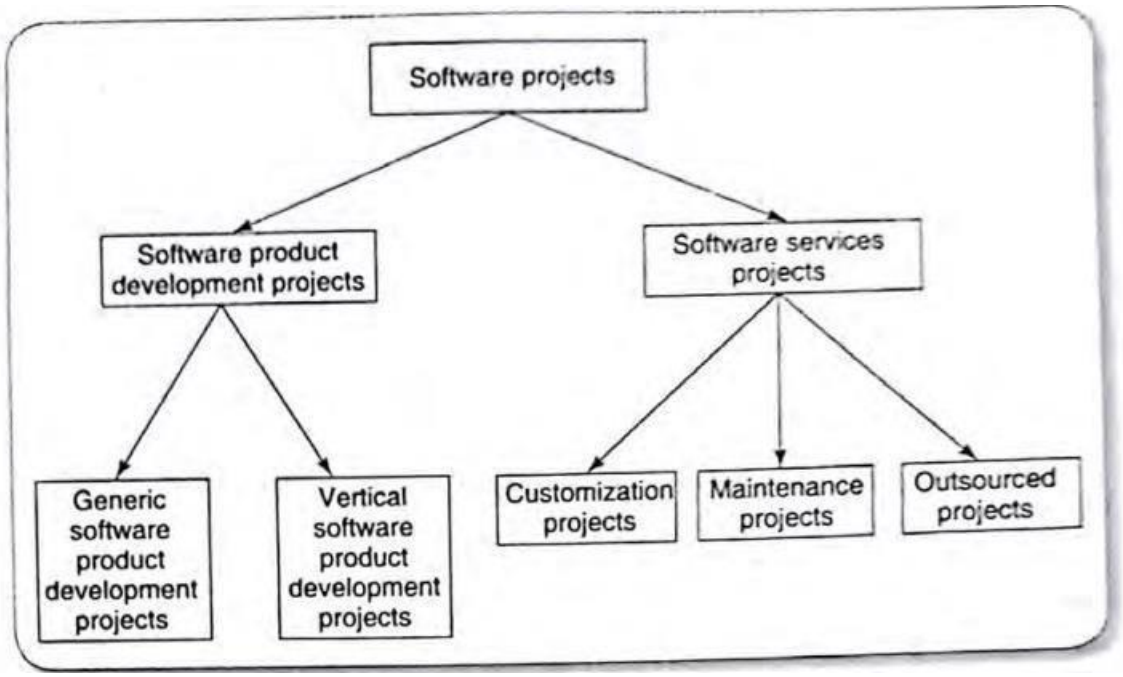

Fig: 1.7 A Classification of software projects

Software services cover a large gamut of software projects such as customization, outsourcing, maintenance, testing and consultancy.

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

1.7.5 Outsourced Projects

While developing a large project, it makes good commercial sense for a company to outsource some parts of its work to other companies.

For example, A company may consider outsourcing as a good option, it if feels that it does not have sufficient expertise to develop some specific parts of the product or if it determines that some parts can be developed cost-effectively another company.

1.7.6 Object-driven development

Projects may be distinguished by whether their aim is to produce or to meet certain objective. any software projects have two stages, First is an object-driven project resulting in recommendations which identify the need for a new software system and next stage is a project actually to create the software product.

1.8 STAKEHOLDERS

These are people who have a stake or interest in the project. It is important that they be identified as early as possible, because you need to set up adequate communication channels with them right from the start. The project leader also has to be aware that not everybody who is involved with a project has the same motivation and objectives. The end-users might, for instance, be concerned about the ease of use of the system while their managers might be interested in the staff savings the new system will allow.

Boehm and Ross proposed a *'Theory W'* of software project management where the manager concentrates on creating the role and format situations where all parties benefit from a project and therefore have an of communication interest in its success. (The 'W' stands for 'win-win'.)

Stakeholders might be internal to the project team, external to the project team but in the same organization, or totally external to the organization.

- ***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:*** For example, the project leader might need the assistance of the information management group in order to add some additional data types to a database or the assistance of the users to carry out systems testing. Here the commitment of the people involved has to be negotiated.

- ***External to both the project team and the organization:*** External stakeholders may be customers (or users) who will benefit from the system that the project implements or contractors who will carry out work for the project. One feature of the relationship with these people is that it is likely to be based on a legally binding contract.

Different types of Stakeholders may have different objectives and one of the jobs of the successful project leader is to recognize these different interests and to be able to reconcile them. It should therefore come as no surprise that the project leader needs to be a good communicator and negotiator.

1.9 SETTING OBJECTIVES

- The objectives should define what the project team must achieve for project success.
- Objectives focus on the desired outcomes of the project rather than the tasks within it-they are the 'post-conditions' of the project.
- Objectives could be set of statements following the opening words ***'the project will be a success if ….***' .
- To have a successful software project, the manager and the project team members must know what will constitute success. This will make them concentrate on what is essential to project success.
- There may be several sets of users of a system and there may be several different groups of specialists involved its development. There is a need for well-defined objectives that are accepted by all these people. Where there is more than one user group, a project authority needs to be identified which has overall authority over what the project is to achieve.
- This authority is often held by a *project steering committee* (or project board or project management board) which has overall responsibility for setting, monitoring and modifying objectives. The project manager still has responsibility for running the project on a day-to-day basis, but has to report to the steering committee at regular intervals. Only the steering committee can authorize changes to the project objectives and resources.

1.9.1 Sub-objectives and Goals:

Setting objectives can guide and motivate individuals and groups of staff. An effective objective for an individual must be something that is within the control of that individual. An objective might be that the software application to be produced must pay for itself by reducing staff costs over two years. As an overall business objective this might be reasonable. For software developers it would be unreasonable as, though they can control development costs, any reduction

in operational staff costs depends not just on them but on the operational management after the application has 'gone live'. What would be appropriate would be to set a goal or sub-objective for the software developers to keep development costs within a certain budget.

Thus, objectives will need be broken down into goals or sub-objectives. Here we say that in order to achieve the objective we must achieve certain goals first. These goals are steps on the way to achieving an objective, just as goals scored in a football match are steps towards the objective of winning the match.

The mnemonic SMART is sometimes used to describe well defined objectives:

- **Specific**: Effective objectives are *concrete and well defined*. Vague aspirations such as 'to improve customer relations' are unsatisfactory. Objectives should be defined in such a way that it is obvious to all whether the project has been successful or not.

- **Measurable**: Ideally there should be measures of effectiveness which tell us how successful the project has been. For example, 'to reduce customer complaints' would be more satisfactory as an objective than 'to improve customer relations'. The measure can, in some cases, be an answer to simple yes/no questions, e.g. 'Can we install the new software by 1 November 2011?'

- **Achievable**: It must be within the power of the individual or group to achieve the objective.

- **Relevant**: The objective must be relevant to the true purpose of the project.

- **Time constrained:** There should be a defined point in time by which the objective should have been achieved.

1.9.2 Measures of effectiveness

Measures of effectiveness provide practical methods of ascertaining whether an objective has been met. 'Mean time between failures' (mtbf) is used to measure reliability. A measure of effectiveness will usually be related to the installed operational system.

1.10 BUSINESS CASE

- Most projects need to have a justification or 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.

- 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. For example:

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

1.11 PROJECT SUCCESS AND FAILURE

- The project plan should be designed to ensure project success preserving the business case for the project.
- Different stakeholders have different interests, some stakeholders in a project might see it as a success while others do not.
- The project objectives are the targets that the project team is expected to achieve—They are summarized as delivering:
    - The agreed functionality
    - To the required level of quality
    - In time
    - Within budget
- A project could meet these targets but the application, once delivered could fail to meet the business case. A computer game could be delivered on time and within budget, but might then not sell.
- In business terms, the project is a success if the value of benefits exceeds the costs.
- A project can be a success on delivery but then be a business failure, On the other hand, a project could be late and over budget, but its deliverables could still, over time, generate benefits that outweigh the initial expenditure.
- The possible gap between project and business concerns can be reduced by having a broader view of projects that includes business issues.
- **Technical learning** will increase costs on the earlier projects, but later projects benefit as the learnt technologies can be deployed more quickly cheaply and accurately.
- **Customer relationships** can also be built up over a number of projects. If a client has trust in a supplier who has done satisfactory work in the past, they are more likely to use that company again.

1.12 MANAGEMENT AND MANAGEMENT CONTROL

1.12.1 MANAGEMENT:

Management involves following activities:

- Planning - deciding what is to be done;
- Organizing - making arrangements;
- Staffing - selecting the right people for the job etc.;
- 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.

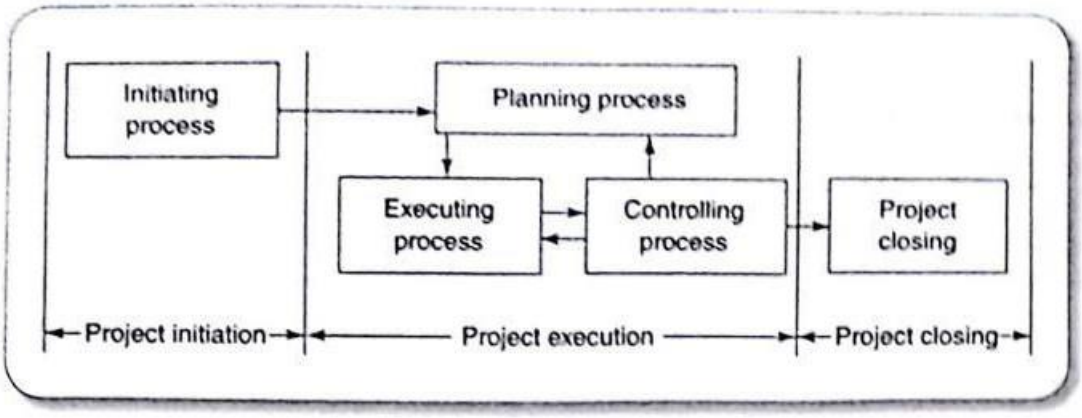

FIGURE 1.5 Principal project management processes

Much of the project manager's time is spent only in three activities , i.e. Project Planning , Monitoring and control. This time period during which these activities are carried out is indicated in Fig 1.5.

It shows that project management is carried out over three well-defined stages or processes irrespective of the methodology used.

In the **Project initiation stage**, an initial plan is made. As a project starts, **the project is monitored and controlled to process as planned**. Initial plan is revised periodically to accommodate additional details and constraints about the project as they become available. Finally, the **project is closed**.

Initial project is undertaken immediately after the feasibility study phase and before starting the

requirement analysis and specification process.

Initial project planning involves estimating several characteristics of a project. Based on these estimates all subsequent project activities are planned.

The **monitoring activity** involves monitoring the progress of the project. **Control activities** are initiated to minimize any significant variation in the plan,

Project Planning is an important responsibility of the project Manager. During project planning, the project manger needs to perform a few well-defined activities that have been outlined below/ Several best practices have been proposed for software project planning activities, PRINCE2 is used extensively in UK and Europe . In USA Project management Institute's 'PMBOK' which refers to their publication "A Gude to the Project Management Body of knowledge, is used.

- **Estimation**: The following project attributes are estimated.
- **Cost**: How much is it going to cost to complete the project.
- **Duration**: How long is it going to take to complete the project.
- **Effort**: How much effort would be necessary for completing the project?

The effectiveness of all activities such as scheduling and staffing are planned at later stage.

- **Scheduling**: Based on estimations of effort and duration, the schedules for manpower and other resources are developed.
- **Staffing**: Staff organization and staffing plans are made.
- **Risk Management**: This activity includes risk identification, analysis, and abatement planning.
- **Miscellaneous Plans**: This includes making several other plans such as quality assurance plan, configuration management plan etc.

While carrying out project monitoring and control activities, a project manager may sometimes find it necessary to change the plan to cope with specific situations and make the plan more accurate as more project data becomes available.

1.12.2 MANAGEMENT CONTROL

Management involves setting objectives for a system and monitoring the performance of the system.

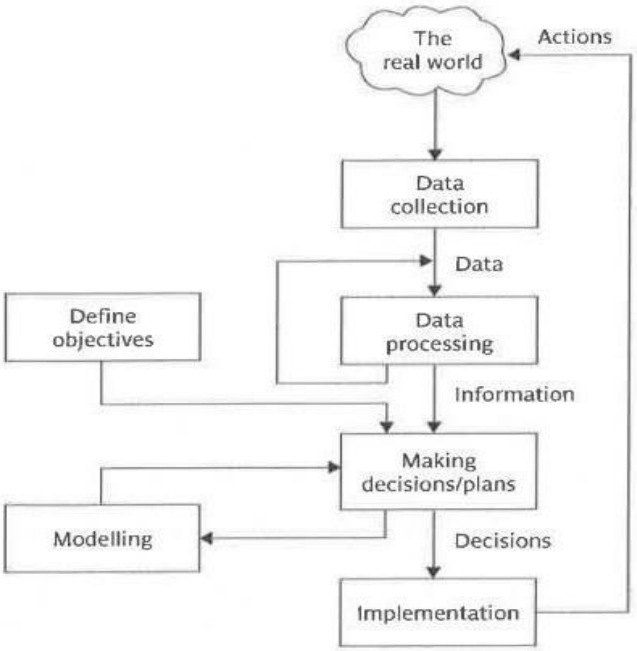

Fig: The Project control cycle

- In the above Fig, local mangers involve in data collection. Bare details such as "location X has processed 2000 documents" may not be useful to higher management.
- Data processing is required to transform this raw data into useful information. This might be in such forms as "Percentage of records Processed", average documents per day per person", and estimated completion date".
- In this example , the project management might examine the "estimated completion date" for completing data transfer for each branch. They are comparing actual performance with overall project objectives.
- They might find that one or two branches will fail to complete the transfer of details in time.
- It can be seen that a project plan 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.

1.13 PROJECT MANAGEMENT LIFE CYCLE

Software development life cycle denotes (SDLC) the stages through which a software is developed. In contrast to SDLC, the project management life cycle typically starts well before the software development activities start and continues for the entire duration of SDLC. (Fig 1.7)

In Project Management process, the project manager carries out project initiation, planning, execution, monitoring, controlling and closing.

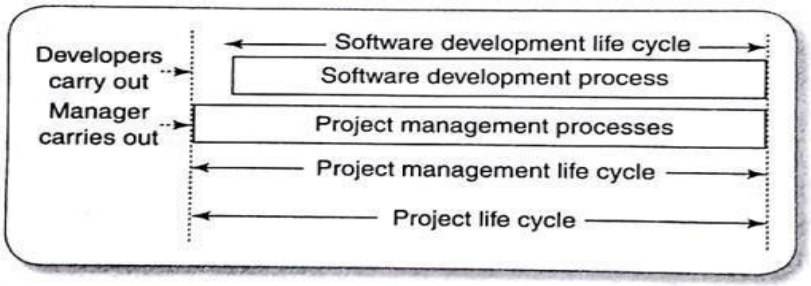

FIGURE 1.7 Project management life cycle versus software development life cycle

The different phases of the project management life cycle are shown in Fig: 1.8.

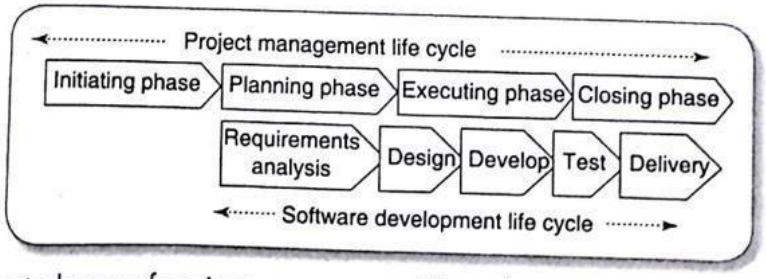

FIGURE 1.8 Different phases of project management life cycle and software development life cycle

1. **Project Initiation**: The project initiation phase starts with project **concept development**. During concept development the different characteristics of the software to be developed are thoroughly understood, which includes, the scope of the project, the project constraints, the cost that would be incurred and the benefits that would accrue. Based on this understanding, a **feasibility study** is undertaken to determine the project would be financially and technically feasible.

   Based on feasibility study, the **business case** is developed. Once the top management agrees to the business case, the **project manager** is appointed, the **project charter** is written and finally **project team** is formed. This sets the ground for the manager to start the **project planning phase.**

2. **W5HH Principle:** Barry Boehm, summarized the questions that need to be asked and answered in order to have an understanding of these project characteristics.

- Why is the software being built?
- What will be done?
- When will it be done?
- Who is responsible for a function?

- Where are they organizationally located?
- How will the job be done technically and managerially?
- How much of these each resource is needed.

3. Project Bidding: Once the top management is convinced by the business case, the **project charter** is developed. For some categories of projects, it may be necessary to have formal bidding process to select suitable vendor based on some cost-performance criteria. The different types of bidding techniques are:

- **Request for quotation(RFQ) :** An organization advertises an RFQ if it has good understanding of the project and the possible solutions.
- **Request for Proposal(RFP) :** An organization had reasonable understanding of the problem to be solved, however, it does not have good grasp of the solution aspects. i.e. may not have sufficient knowledge about different features to be implemented. The purpose of RFP is to get an understanding of the alternative solutions possible that can be deployed and not vendor selection. Based on the RFP process, the requesting organization can form a clear idea of the project solutions required, based on which it can form a statement work (SOW) for requesting RFQ for the vendors.
- **Request for Information (RFI):** An organization soliciting bids may publish an RFI. Based on the vendor response to the RFI, the organization can assess the competencies of the vendors and shortlist the vendors who can bid for the work.

4. Project Planning: An importance of the project initiation phase is the project charter. During the project planning the project manger carries out several processes and creates the following documents:

- **Project plan:** This document identifies the project the project tasks and a schedule for the project tasks that assigns project resources and time frames to the tasks.
- **Resource Plan:** It lists the resources , manpower and equipment that would be required to execute the project.
- **Functional Plan:** It documents the plan for manpower, equipment and other costs.
- **Quality Plan:** Plan of quality targets and control plans are included in this document.
- **Risk Plan:** This document lists the identification of the potential risks, their prioritization and a plan for the actions that would be taken to contain the different risks.

5. **Project Execution:** In this phase the tasks are executed as per the project plan developed during the planning phase. Quality of the deliverables is ensured through execution of proper processes. Once all the deliverables are produced and accepted by the customer, the project execution phase completes and the project closure phase starts.

6. **Project Closure:** Project closure involves completing the release of all the required deliverables to the customer along with the necessary documentation. All the Project resources are released and supply agreements with the vendors are terminated and all the pending payments are completed. Finally, a postimplementation review is undertaken to analyze the project performance and to list the lessons for use in future projects.

1.14 TRADITIONAL VERSUS MODERN MANAGEMENT PRACTICES

Over the last two decades, the basic approach taken by the software industry to develop software has undergone a radical change.

Software is not developed from scratch any more, Software development projects are based on either tailoring some existing product or reusing certain pre-built libraries both will maximize code reuse and compression of project durations.

Other goals include facilitating and accommodating client feedback and client feedbacks and customer participation in project development work and incremental delivery of the product with evolving functionality.

Some Important difference between modern management practices and traditional practices are:

- **Planning Incremental Delivery:** Earlier, projects were simpler and therefore more predictable than the present-day projects. In those days, projects were planned with sufficient detail much before the actual project execution started. After the project initiation, monitoring and control activities were carried out to ensure that the project execution proceeded as per plan, Now, the projects are required to be completed over a much shorter duration, and rapid application development and deployment are considered key strategies.
  Instead of making a long-term project completion plan, the project manger now plans all incremental deliveries with evolving functionalities. This type of project management is often called **extreme project management.** Extreme project management is highly flexible approach that concentrates on human side of project management(e.g. managing project stakeholders).

- **Quality Management:** Customer awareness about product quality has increased

significantly. The key responsibility of a project manager now includes assessment of project progress and tracking the quality of all intermediate artifacts.

- **Change Management:** Earlier, when the requirements were signed off by the customer, any changes to the requirements were rarely entertained. Customer suggestions are now actively solicited and incorporated throughout the development process. To facilitate customer feedback, **incremental delivery models** are popularly being used. Product development is being carried out through a series of product versions implementing increasingly greater functionalities. The Project manager plays a key role in product base lining and version control. This has made change management a crucial responsibility of the project manager. Change Management is also known as configuration management.

- **Requirement Management:** In older development methodologies , the requirements had to be identified upfront and these were 'signed off' by the customer and frozen before the development could start. At present , in most projects, the requirements change frequently during the development cycle. **Requirement management** has therefore become a systematic process of controlling changes, documenting , analyzing, tracing, prioritizing requirements and then communicating the changes to the relevant stakeholders.

- **Release Management:** Release management concerns planning, prioritizing and controlling the different releases of a software. Modern development processes such as Agile development processes advocate frequent and regular releases of the software to be made to the customer during the software development. Starting with the release of basic or core functionalities of the software, more complete functionalities are made available to the customer every couple of weeks . Hence effective Release Management has become important.

- **Risk Management:** In modern software development practices. Effective risk management is considered very important to the success of a project. A risk is any negative situation that may arise as the project progresses and may threaten the success of the

  project. Risk Management involves identification of risks, assessment of the impacts of various risks, prioritization of the risks and preparation of risk-containment plans.

- **Scope Management:** Modern software development encourages customer to come up with change requests. While accepting the requests, three critical project parameters: scope , schedule and project cost are interdependent and related.

2.4 Evaluation of Individual Projects

We will now look more closely at how the feasibility of an individual project can be evaluated.

Technical assessment

Technical assessment of a proposed system consists of evaluating whether the required functionality can be achieved with current affordable technologies. Organizational policy, aimed at providing a consistent hardware/software infrastructure, is likely to limit the technical solutions considered. The costs of the technology adopted must be taken into account in the cost–benefit analysis.

Cost–benefit analysis

Even where the estimated benefits will exceed the estimated costs, it is often necessary to decide if the proposed project is the best of several options. Not all projects can be undertaken at any one time and, in any case, the most valuable projects should get most resources.

Any project aiming at a return on investment must, as a minimum, provide a greater benefit than putting that investment in, say, a bank.

Cost–benefit analysis comprises two steps:

- *Identifying all of the costs and benefits of carrying out the project and operating the delivered application* These include the development costs, the operating costs, and the benefits expected from the new system. Where the proposed system is a replacement, these estimates should reflect the change in costs and benefits due to the new system. A new sales order processing system, for example, could only claim to benefit an organization by the increase in sales due to the use of the new system.
- *Expressing these costs and benefits in common units* We must express each cost and benefit – and the *net benefit* which is the difference between the two – in money.

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

The different types of benefits will be discussed in greater detail in the context of benefits to management later in this chapter.

- *Development costs*, including development staff costs.
- *Setup costs*, consisting of the costs of putting the system into place, mainly of any new hardware but also including the costs of file conversion, recruitment and staff training.
- *Operational costs* relating to operating the system after installation.

Exercise 2.1

Brightmouth College is considering the replacement of the existing payroll service, operated by a third party, with a tailored, off-the-shelf computer-based system. List some of the costs it might consider under the headings of:

- Development costs
- Setup costs
- Operational costs

List some of the benefits under the headings:

- Quantified and valued benefits
- Quantified but not valued
- Identified but not easily valued

For each cost or benefit, explain how, in principle, it might be measured in monetary terms.

Cash flow forecasting

Typically products generate a negative cash flow during their development followed by a positive cash flow over their operating life. There might be decommissioning costs at the end of a product's life.

As important as estimating the overall costs and benefits of a project is producing a cash flow forecast which indicates when expenditure and income will take place (Figure 2.1).

We need to spend money, such as staff wages, during a project's development. Such expenditure cannot wait until income is received (either from using software developed in-house use or from selling it). 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 the payment of salaries, and any income are to be expected.

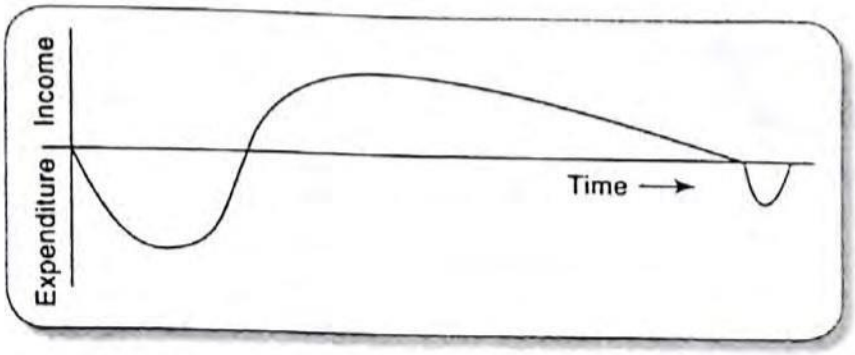

FIGURE 2.1 Typical product life cycle cash flow

Accurate cash flow forecasting is difficult, as it is done early in the project's life cycle (at least before any significant expenditure is committed) and many items to be estimated (particularly the benefits of using software) might be some years in the future.

When estimating future cash flows, it is usual to ignore the effects of inflation. Forecasts of inflation rates tend to be uncertain. Moreover, if expenditure is increased due to inflation it is likely that income will increase proportionately.

The difficulty and importance of cash flow forecasting is evidenced by the number of companies that suffer bankruptcy because, although they are developing profitable products or services, they cannot sustain an unplanned negative cash flow.

2.5 Cost–benefit Evaluation Techniques

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

Table 2.1 illustrates cash flow forecasts for four projects. In each case, it is assumed that the cash flows take place at the end of each year. For short-term projects or where there are significant seasonal cash flow patterns, quarterly, or even monthly, cash flow forecasts could be appropriate.

TABLE 2.1 Four project cash flow projections – figures are end of year totals (£)

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

Exercise 2.2

Consider the project cash flow estimates for four projects at JOE shown in Table 2.1. Negative values represent expenditure and positive values income.

Rank the four projects in order of financial desirability and make a note of your reasons for ranking them in that way before reading further.

Net profit → 2

The net profit of a project is the difference between the total costs and the total income over the life of the project. Project 2 in Table 2.1 shows the greatest net profit but this is at the expense of a large investment. Indeed, if we had £1 m to invest, we might undertake all of the other three projects and obtain an even greater net profit. Note also that all projects contain an element of risk and we might not be prepared to risk £1 m. We shall look at the effects of risk and investment later in this chapter.

Cash flows take place at the end of each year. The year 0 represents the initial investment made at the start of the project.

Moreover, the simple net profit takes no account of the timing of the cash flows. Projects 1 and 3 each have a net profit of £50,000 and therefore, according to this selection criterion, would be equally preferable. The bulk of the income occurs late in the life of project 1, whereas project 3 returns a steady income throughout its life. Having to wait for a return has the disadvantage that the investment must be funded for longer. Add to that the fact that, other things being equal, estimates in the more distant future are less reliable than short-term estimates and we can see that the two projects are not equally preferable.

Payback period → 3*, 4

The *payback period* is the time taken to break even or pay back the initial investment. Normally, the project with the shortest payback period will be chosen on the basis that an organization will wish to minimize the time that a project is 'in debt'.

Exercise 2.3

Consider the four project cash flows given in Table 2.1 and calculate the payback period for each of them.

The advantage of the payback period is that it is simple to calculate and is not particularly sensitive to small forecasting errors. Its disadvantage as a selection technique is that it ignores the overall profitability of the project – in fact, it totally ignores any income (or expenditure) once the project has broken even. Thus the fact that projects 2 and 4 are, overall, more profitable than project 3 is ignored.

Return on investment → 4

The *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. There are some variations on the formula used to calculate the return on investment but a straightforward common version is:

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

Exercise 2.4

Calculating the ROI for project 1, the net profit is £50,000 and the total investment is £100,000. The return on investment is therefore calculated as

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

$$= \frac{50{,}000/5}{100{,}000} \times 100 = 10\%$$

1. 10%
2. 2%
3. 10%
4. 12.5%

Calculate the ROI for each of the other projects shown in Table 2.1 and decide which, on the basis of this criterion, is the most worthwhile.

The return on investment provides a simple, easy-to-calculate measure of return on capital. Unfortunately, it suffers from two severe disadvantages. Like the net profitability, it takes no account of the timing of the cash flows. More importantly, this rate of return bears no relationship to the interest rates offered or charged by banks (or any other normal interest rate) since it takes no account of the timing of the cash flows or of the compounding of interest. It is therefore, potentially, very misleading.

Net present value

The calculation of *net present value* is a project evaluation technique that takes into account the profitability of a project and the timing of the cash flows that are produced. This is based on the view that receiving £100 today is better than having to wait until next year to receive it. We could, for example, invest the £100 in a bank today and have £100 plus the interest in a year's time. If we say that the *present value* of £100 in a year's time is £91, we mean that £100 in a year's time is the equivalent of £91 now.

Net present value (NPV) and internal rate of return (IRR) are collectively known as discounted cash flow (DCF) techniques.

The equivalence of £91 now and £100 in a year's time means we are discounting the future income by approximately 10%. If we received £91 now and invested it for a year at an annual interest rate of 10%, it would be worth £100 in a year's time. The annual rate by which we discount future earnings is known as the *discount rate* – 10% in the above example.

Note that this example uses approximate figures.

Similarly, £100 received in two years' time would have a present value of approximately £83 – in other words, £83 invested at an interest rate of 10% would yield approximately £100 in two years' time.

A rate of 10% may be unrealistic but is used here for ease of calculation.

The present value of any future cash flow may be obtained by applying the following formula

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

where r is the discount rate, expressed as a decimal value, and t is the number of years into the future that the cash flow occurs.

$1yr = \frac{1}{(1+0.10)} = 0.9091$ $\quad 2yr = \frac{1}{(1+0.10)^2} = 0.826$

More extensive or detailed tables may be constructed using the formula *discount factor* $\frac{1}{(1+r)^t}$ for various values of r (the discount rate) and t (the number of years from now).

Alternatively, and rather more easily, the present value of a cash flow may be calculated by multiplying the cash flow by the appropriate discount factor. A small table of discount factors is given in Table 2.2.

The NPV for a project is obtained by discounting each cash flow (both negative and positive) and summing the discounted values. It is normally assumed that any initial investment takes place immediately (indicated as year 0) and is not discounted. Later cash flows are normally assumed to take place at the end of each year and are discounted by the appropriate amount.

TABLE 2.2 NPV discount factors

	Discount rate (%)					
Year	**5**	**6**	**8**	**10**	**12**	**15**
1	0.9524	0.9434	0.9259	0.9091	0.8929	0.8696
2	0.9070	0.8900	0.8573	0.8264	0.7972	0.7561
3	0.8638	0.8396	0.7938	0.7513	0.7118	0.6575
4	0.8227	0.7921	0.7350	0.6830	0.6355	0.5718
5	0.7835	0.7473	0.6806	0.6209	0.5674	0.4972
6	0.7462	0.7050	0.6302	0.5645	0.5066	0.4323
7	0.7107	0.6651	0.5835	0.5132	0.4523	0.3759
8	0.6768	0.6274	0.5403	0.4665	0.4039	0.3269
9	0.6446	0.5919	0.5002	0.4241	0.3606	0.2843
10	0.6139	0.5584	0.4632	0.3855	0.3220	0.2472
15	0.4810	0.4173	0.3152	0.2394	0.1827	0.1229
20	0.3769	0.3118	0.2145	0.1486	0.1037	0.0611
25	0.2953	0.2330	0.1460	0.0923	0.0588	0.0304

Exercise 2.5

Assuming a 10% discount rate, the NPV for project 1 (Table 2.1) would be calculated as in Table 2.3. The net present value for project 1, using a 10% discount rate, is therefore £618. Using a 10% discount rate, calculate the net present values for projects 2, 3 and 4 and decide which, on the basis of this, is the most beneficial to pursue.

Scanned by CamScanner

TABLE 2.3 Applying the discount factors to project 1

Year	Project 1 cash flow (£)		Discount factor @ 10%	Discounted cash flow (£)	
0	–100,000	- 100,000	1.0000	–100,000	-100000
1	10,000	30000	0.9091	9,091	27273
2	10,000	30000	0.8264	8,264	24792
3	10,000	30000	0.7513	7,513	22539
4	20,000	30000	0.6830	13,660	20490
5	100,000	30000	0.6209	62,090	18627
Net Profit	£50,000		NPV: £618		13721

It is interesting to note that the net present values for projects 1 and 3 are significantly different – even though they both yield the same net profit and both have the same return on investment. The difference in NPV reflects the fact that, with project 1, we must wait longer for the bulk of the income.

The main difficulty with NPV for deciding between projects is selecting an appropriate discount rate. Some organizations have a standard rate but, where this is not the case, then the discount rate should be chosen to reflect available interest rates (borrowing costs where the project must be funded from loans) plus some premium to reflect the fact that software projects are normally more risky than lending money to a bank. The exact discount rate is normally less important than ensuring that the same discount rate is used for all projects being compared. However, it is important to check that the ranking of projects is not sensitive to small changes in the discount rate – have a look at the following exercise.

Exercise 2.6

Calculate the net present value for each of the projects A, B and C shown in Table 2.4 using each of the discount rates 8%, 10% and 12%.

For each of the discount rates, decide which is the best project. What can you conclude from these results?

TABLE 2.4 Three estimated project cash flows

Year	Project A (£)	Project B (£)	Project C (£)
0	–8,000	–8,000	–10,000
1	4,000	1,000	2,000
2	4,000	2,000	2,000

(Contd)

3	2,000	4,000	6,000
4	1,000	3,000	2,000
5	500	9,000	2,000
6	500	−6,000	2,000
Net Profit	4,000	5,000	6,000

Alternatively, the discount rate can be thought of as a target rate of return. If, for example, we set a target rate of return of 15% we would reject any project that did not display a positive net present value using a 15% discount rate. Any project that displayed a positive NPV would be considered for selection – perhaps by using an additional set of criteria where candidate projects were competing for resources.

Internal rate of return

One disadvantage of NPV as a measure of profitability is that, although it may be used to compare projects, it might not be directly comparable with earnings from other investments or the costs of borrowing capital. Such costs are usually quoted as a percentage interest rate. The internal rate of return (IRR) attempts to provide a profitability measure as a percentage return that is directly comparable with interest rates. Thus, a project that showed an estimated IRR of 10% would be worthwhile if the capital could be borrowed for less than 10% or if the capital could not be invested elsewhere for a return greater than 10%.

The IRR is calculated as that percentage discount rate that would produce an NPV of zero. It is most easily calculated using a spreadsheet or other computer program that provides functions for calculating the IRR. Microsoft Excel, for example, provides IRR functions which, provided with an initial guess or seed value (which may be zero), will search for and return an IRR.

One deficiency of the IRR is that it does not indicate the absolute size of the return. A project with an NPV of £100,000 and an IRR of 15% can be more attractive than one with an NPV of £10,000 and an IRR of 18% – the return on capital is lower but the net benefits greater.

Another objection to the internal rate of return is that, under certain conditions, it is possible to find more than one rate that will produce a zero NPV. However, if there are multiple solutions, it is always appropriate to take the lowest value and ignore the others.

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

- A total evaluation must also take into account the problems of funding the cash flows – will we, for example, be able to repay the interest on any borrowed money at the appropriate time?
- While a project's IRR might indicate a profitable project, future earnings from a relatively risky project might be far less reliable than earnings from, say, investing with a bank. We might undertake a more detailed risk analysis as described below.
- 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?

1 → 10.17%
2 → 3.08%
3 → 15.24%

Exercise 2.7

Check if the projects A, B, and C shown in Table 2.4 are worth taking up when the rate of interest on borrowed capital is 15%.

2.6 Risk Evaluation

Every project involves risk. We have already noted that *project* risks, which prevent the project from being completed successfully, are different from the *business* risk that the delivered products are not profitable. Project risks will be discussed in Chapter 7. Here we focus on business risk.

Risk identification and ranking

In any project evaluation we should 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. Table 2.5 illustrates a basic project risk matrix listing some of the business risks for a project, with their importance and likelihood classified as high (H), medium (M), low (L) or exceedingly unlikely (—). So that projects may be compared, the list of risks must be the same for each project assessed. It is likely, in reality, that it would be longer than shown and more precise.

The project risk matrix may be used as a way of evaluating projects (those with high risks being less favoured) or as a means of identifying and ranking the risks for a specific project.

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

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

Risk and net present value

Where a project is relatively risky, it is a common practice to use a higher discount rate to calculate net present value. This risk premium might, for example, be an additional 2% for a reasonably safe project or 5% for a fairly risky one. Projects may be categorized as high, medium, or low risk using a scoring method and risk premiums designated for each category. The premiums, even if arbitrary, provide a consistent method of taking risk into account.

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. Rather than a single cash flow forecast for a project, we will then have a set of cash flow forecasts, each with an associated probability of occurring. The value of the project is then obtained by summing the cost or benefit for each possible outcome weighted by its corresponding probability. Exercise 2.8 illustrates how this may be done.

Exercise 2.8

BuyRight, a software house, is considering developing a payroll application for use in academic institutions and is currently engaged in a cost–benefit analysis. Study of the market has shown that, if BuyRight can target it efficiently and no competing products become available, it will obtain a high level of sales generating an annual income of £800,000. It estimates that there is a 1 in 10 chances of this happening. However, a competitor might launch a competing application before its own launch date and then sales might generate only £100,000 per year. It estimates that there is a 30% chance of this happening. The most likely outcome, it believes, is somewhere in between these two extremes – it will gain a market lead by launching before any competing product becomes available and achieve an annual income of £650,000. BuyRight has therefore calculated its expected sales income as in Table 2.6.

Table 2.6 BuyRight's income forecasts

Sales	Annual sales income (£) i	Probability p	Expected value (£) $i \times p$
High	800,000	0.1	80,000
Medium	650,000	0.6	390,000
Low	100,000	0.3	30,000
Expected Income			500,000

Development costs are estimated at £750,000. Sales levels are expected to be constant for at least four years. Annual costs of marketing and product maintenance are estimated at £200,000, irrespective of the market share. Would you advise going ahead with the project?

This approach is frequently used to evaluate large projects such as the building of motorways, where variables such as traffic volumes, and hence the total benefit of shorter journey times, are uncertain. The technique, of course, relies on being able to assign probabilities of occurrence to each scenario, which requires extensive research.

When used to evaluate a single major project, the cost–benefit approach, by 'averaging out' the negative and positive outcomes of the different scenarios, does not take full account of 'worst-case scenarios'. Because

of this, it is more appropriate for the evaluation of a portfolio of projects where overall profitability is the primary concern, more successful projects can offset the impact of less successful ones.

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. We might, for example, vary one of our original estimates by plus or minus 5% and recalculate the expected costs and benefits for the project. By repeating this exercise for each of our estimates in turn we can evaluate the sensitivity of the project 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. We then need to decide whether we can exercise greater control over them or otherwise mitigate their effects. If neither is the case, then we must live with the risk or abandon the project.

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 to reject over-risky projects or choose those with the best risk profile. There are many situations, however, 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 project.

As an example, say a successful company is considering when to replace its sales order processing system. The decision largely rests upon the rate at which its business expands – if its market share significantly increases (which it believes will happen if rumours of a competitor's imminent bankruptcy are fulfilled) the existing system might need to be replaced within two years. Not replacing the system in time could be an expensive option as it could lead to lost revenue if it cannot cope with increased sales. Replacing the system immediately will, however, be expensive as it will mean deferring other projects already scheduled.

It is calculated that extending the existing system will have an NPV of £75,000, although if the market expands significantly, this will be turned into a loss with an NPV of –£1 00,000 due to lost revenue. If the market does expand, replacing the system now has an NPV of £250,000 due to the benefits of being able to handle increased sales and other benefits such as improved management information. If sales do not increase, however, the benefits will be severely reduced and the project will suffer a loss with an NPV of –£50,000.

The company estimate the likelihood of the market increasing significantly at 20% – and, hence, the probability that it will not increase at 80%. This scenario can be represented as a tree structure as shown in Figure 2.2.

The analysis of a decision tree consists of evaluating the expected benefit of taking each path from a decision point (denoted by D in Figure 2.2). The expected value of each path is the sum of the value of each possible outcome multiplied by its probability of occurrence. The expected value of extending the system is therefore £40,000 (75,000 × 0.8 – 100,000 × 0.2) and the expected value of replacing the system £10,000 (250,000 × 0.2 – 50,000 × 0.8). The company should therefore choose the option of extending the existing system.

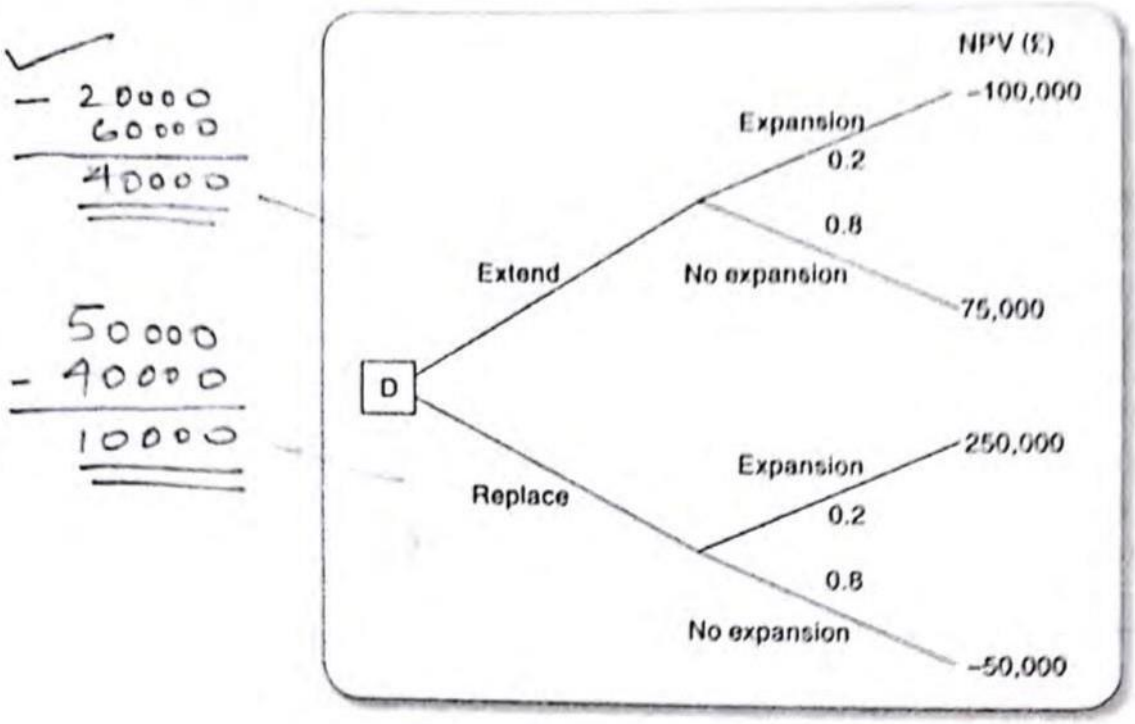

FIGURE 2.2 A decision tree

2.7 Programme Management

Ferns' paper appeared in the *International Journal of Project Management* August 1991.

It should now have been made clear that there will be an element of risk with any single project. Even where projects produce real financial benefits, the precise size of those benefits will often be uncertain at the start of the project. This makes it important for organizations to take a broad view of all its projects to ensure that while some projects may disappoint, organizational developments overall will generate substantial benefits.

We introduced project portfolios in Section 2.3. We will now examine how careful management of programmes of projects can provide benefits. 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*'.

Programmes can exist in different forms, as can be seen below.

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. We have seen that many organizations have a fixed budget for ICT development. Decisions have to be made about which projects to implement within that budget within the accounting period, which often coincides with the financial year.

Strategic programmes

Several projects together can implement a single strategy. For example, the merging of two organizations' computer systems could require several projects each dealing with a particular application area. Each activity could be treated as a distinct project, but would be coordinated as a programme.