SUBJECT: Software Engineering & Project Management (BCS501)

Module 4 – Project Management & Evalution

Syllabus: ***Introduction to Project Management****: Introduction, Project and Importance of Project Management, Contract Management, Activities Covered by Software Project Management, Plans, Methods and Methodologies, Some ways of categorizing Software Projects, Stakeholders, Setting Objectives, Business Case, Project Success and Failure, Management and Management Control, Project, Management life cycle, Traditional versus Modern Project Management Practices.*
Project Evaluation: *Evaluation of Individual projects, Cost–benefit Evaluation Techniques, Risk Evaluation*

Introduction to Software Project Management

Definition: Software Project Management is an art & Science of planning & leading software Projects from ideas to reality**.**

- 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).
- Project management involves the planning, monitoring, and control of people, process, and events that occur during software development.

1.1 WHY IS SOFTWARE PROJECT MANAGEMENT IMPORTANT?

Software project management is essential for ensuring that software projects are completed on time, within budget, and meet quality expectations. Some reasons why it's important are:

1. **Complexity Management**
   - Software projects often involve Complex 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**
2. **68% of Projects Fail to Meet Deadlines, Budgets, and Quality Targets**
3. **97% of Businesses Believe Project Management is Essential for Success**
4. **80% of High-Performing Projects are Led by a Project Manager with Qualifications**
5. **On Average, a Large IT Project Runs 45% Over Budget**

1.2 Software Development Life Cycle:

The Software Development Life Cycle (SDLC) is a structured methodology that outlines the processes for planning, creating, testing, and delivering software systems. It provides a framework to manage and control the development of software and ensures a systematic approach to achieving project goals.

Aspects of Software Development Life Cycle are:

- **Dimensions of a Project**:
    1. **People**: The stakeholders, including developers, testers, and clients.
    2. **Process**: The methodologies and workflows followed during the development process.
    3. **Product**: The final deliverable software or system.
    4. **Technology**: Tools, platforms, and technologies used in the project.
- **Project Control Variables**:
    1. **Time**: The duration required to complete the project.
    2. **Cost**: The budget allocated, influenced by the time and resources required.
    3. **Quality**: The adherence to project requirements and standards.
    4. **Scope**: The specific features and functionalities outlined for the project.
    5. **Risk**: Potential challenges or points of failure during development

Trade -off Triangle

The triangle illustrates the relationship between three primarily forces in a project. I,e Time ,cost and quality.

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 success.

The normal situation is that one of these factors is fixed and the other two will adjust by itself.

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).

1.3 WHAT IS A PROJECT?

Definition: '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.

1.3.1 Characteristics of Project:

1. **Non-routine tasks:** Projects involve tasks that are unique and not repetitive.
2. **Planning is essential**: Projects require thorough planning for successful execution.
3. **Specific objectives**: Projects aim to meet defined goals or create specified products.
4. **Pre-determined time span:** Projects have a fixed timeline with a start and end date.
5. **Work for others:** Projects are often carried out for external stakeholders.
6. **Specialized expertise:** Projects involve multiple areas of expertise.
7. **Phased execution:** Projects progress through various phases (e.g., initiation, planning, execution, closure).
8. **Constrained resources:** Projects operate under limited availability of resources.
9. **Scale and complexity:** Projects are often large or involve intricate tasks.

Differences between Software Projects and Other Projects:

1. Invisibility:
   - Physical Projects: Progress is visible, such as a building under construction.
   - Software Projects: Work like coding or testing is intangible and invisible until specific deliverables are completed.
   - Example: Progress in bridge construction is physically evident, whereas writing code for a payroll system is not.
2. Complexity:
   - Physical Projects: Governed by physical laws, often simpler in functionality.
   - Software Projects: Must account for numerous scenarios, user needs, and integrations, making them more complex.
   - Example: Developing a payroll system involves multiple functionalities like tax calculations and user interfaces, unlike building a bridge.
3. Conformity:
   - Physical Projects: Use consistent materials (e.g., steel, cement) that adhere to predictable properties.
   - Software Projects: Must conform to human requirements, which can change unpredictably.
   - Example: Clients might demand new features during software development, unlike physical projects, where requirements remain largely fixed.
4. Flexibility:
   - Physical Projects: Changes are rare and difficult once construction begins.
   - Software Projects: Require frequent updates or modifications due to organizational or regulatory changes.
   - Example: A payroll system may need frequent updates for new tax rules, while a constructed bridge rarely undergoes design changes.

1.4 CONTRACT MANAGEMENT

Contract Management is the systematic process of creating, executing, and analyzing contracts to optimize operational and financial outcomes while minimizing risks. It ensures that all parties to a contract meet their obligations effectively and efficiently.

The Contract lifecycle management is as shown below

Various Stages of Contract Management

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.

3. **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.

4. **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.

5. **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.

6. **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

1.5 ACTIVITIES COVERED BY SOFTWARE PROJECT MANAGEMENT:

Project Management encompasses a range of activities that ensure the successful completion of a project, from inception to delivery and beyond. Each activity plays a crucial role in managing resources, timelines, and objectives effectively.
The activities covered by Software Project management are diagrammatically illustrated as follows:

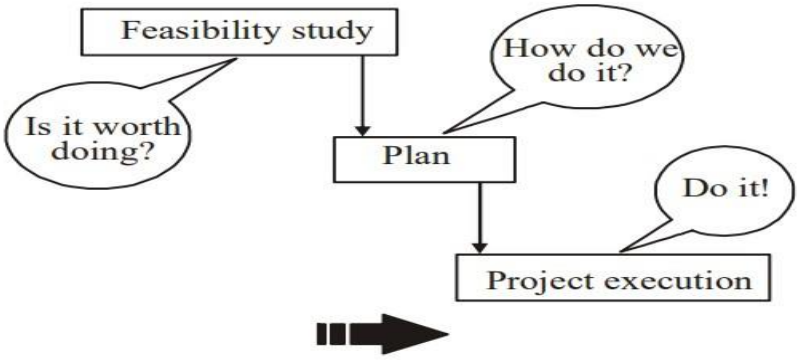


Figure 1.1: The Feasibility Study / Plan / Execution Cycle

1.5.1 The Feasibility Study:

This is initial phase investigates whether a prospective project is worth pursuing by assessing its **business case** and viability. In this phase the following activities are carried out

- Gathering requirements for the proposed system.
- Estimating development and operational costs.
- Analyzing the benefits and risks.
- Supporting strategic planning decisions.
- Decision on whether to proceed with the project or not will be taken.

1.5.2 Planning:

If the project is considered feasible, the planning phase begins to create a clear roadmap for execution. This involves developing an outline plan for the entire project and a detailed plan for the initial stage. As the project progresses, plans for later stages are updated based on newly available information. The result is a structured project plan that serves as a guide for the development process, ensuring organized and efficient progress.

1.5.3 Project Execution:

The execution of a project often contains *design* and *implementation* sub phases. The same is illustrated in **Figure 1.2** which shows the typical sequence of software development activities recommended.

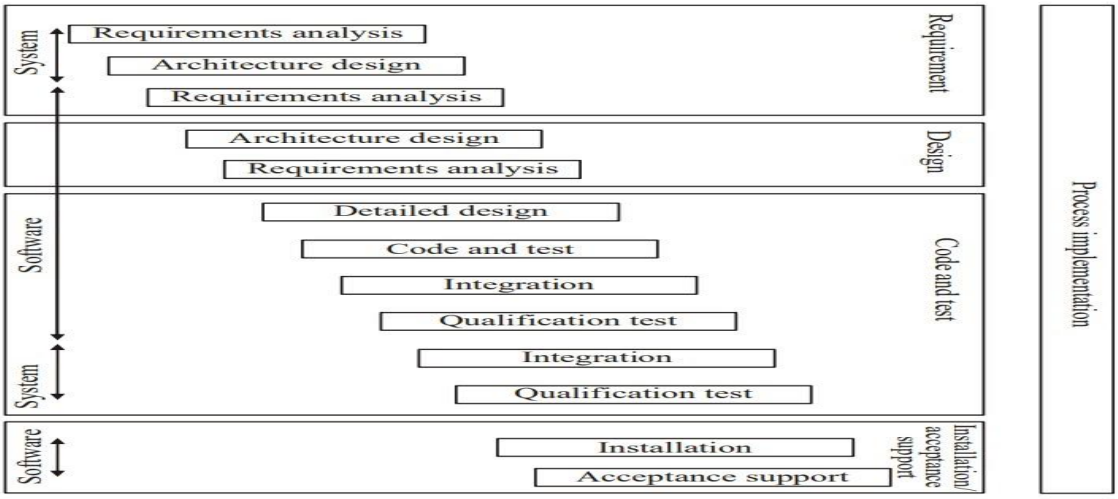


Figure 1.2: Sub Phases in Project Execution

1.5.3.1 Requirements Analysis:

This phase focuses on gathering and refining requirements to understand user needs thoroughly. It builds on the information collected during the evaluation phase, updating and supplementing it to ensure a complete and detailed understanding of the system's requirements. This process is essential to align the project's goals with user expectations and lay a strong foundation for development.

1.5.3.2 Specification:

This phase involves creating detailed documentation that defines the system's intended functionality. It serves as a blueprint for the development process, providing clear guidance to the development team and ensuring that the system aligns with the specified requirements.

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.
- **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.

1.5.3.4 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.

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 Activity Planning in Software Project Management

Activity planning is a critical aspect of software project management. It ensures that the project tasks are well-defined, sequenced, and scheduled to meet the project objectives within constraints like time, cost, and resources. Effective activity planning helps mitigate risks, optimize resource allocation, and improve project tracking.

1. **Identifying Activities**

In this phase the project is breakdown into manageable tasks or activities. It defines tasks in terms of their objectives, deliverables, and dependencies and organizes activities hierarchically.

2. Sequencing Activities

Establish logical relationships between tasks to determine their order of execution then identify dependencies using tools like precedence diagrams or dependency tables.in this the dependency types may be

- Finish-to-Start (Task A must finish before Task B starts).
- Start-to-Start (Task A and Task B must start simultaneously).
- Finish-to-Finish (Task A and Task B must finish together).

3**. Estimating Time and Resources**

Time estimation involves using historical data, expert judgment, or techniques like PERT (Program Evaluation and Review Technique) to estimate the duration required for completing the project.

Resource estimation focuses on determining the type and quantity of resources, such as personnel, hardware, and tools, needed to carry out each activity effectively. Both estimations are critical for setting realistic schedules and ensuring that the necessary resources are available throughout the project.

4. Scheduling Activities

In this phase, activities are organized on a timeline using project scheduling tools like Gantt charts or network diagrams. Start and end dates for tasks are calculated, and techniques are applied to identify critical tasks that directly affect the overall project duration. This helps ensure that the project stays on track and deadlines are met.

5. Setting Milestones

Key checkpoints or deliverables in a project are important stages that show when major tasks or phases are completed. These milestones help track the progress of the project and ensure that everything is going as planned. They allow project managers and team members to review the work done so far, identify any issues, and make necessary adjustments to stay on schedule and meet project goals.

6. **Risk Assessment and Contingency Planning**

In this phase, potential risks that could affect the execution of activities are identified. Contingency plans are then developed to address possible delays or disruptions, ensuring that the project can continue smoothly even if unforeseen issues arise.

7. **Allocating Resources**

In this, resources are assigned to tasks based on their availability and the specific requirements of each activity. It is important to avoid over-allocating resources and ensure their optimal utilization to maintain efficiency and prevent burnout or delays.

8. Monitoring and Updating the Plan

In this phase, progress is continuously tracked and compared with the planned schedule to ensure the project stays on course. If any deviations or changes in scope occur, the plan is adjusted accordingly to address these issues and keep the project on track.

1.7 PLANS, METHODS AND METHODOLOGIES

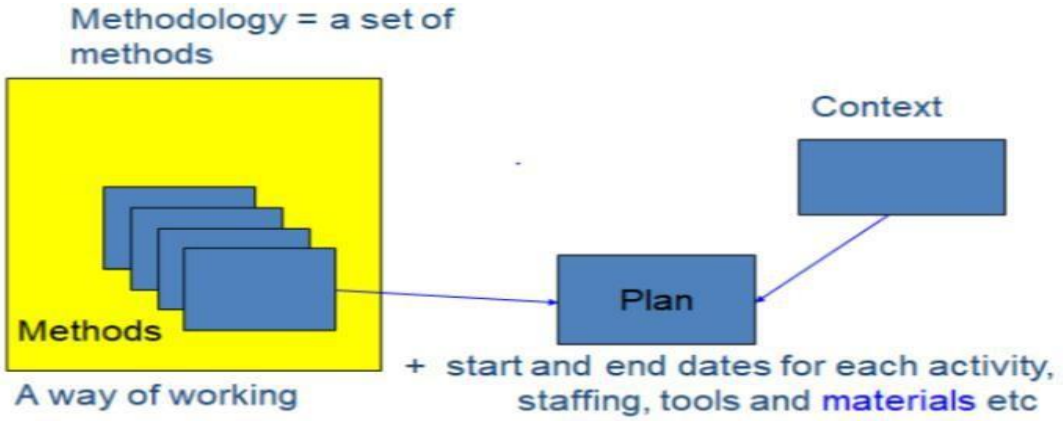
A plan for an activity must be based on some idea of a method of work. To take a simple example, if it was asked to test some software, even though you do not know anything about the software to be tested, you could assume that you would need to:

- 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.



1.8 SOME WAYS OF CATEGORIZING SOFTWARE PROJECTS

Software projects can be categorized in different ways, and each type requires a different approach due to its unique features and goals. Here are some common ways to categorize them:

Changes in Software Projects: Over time, software projects have changed a lot. In the past, software had to be built from scratch with no code reuse. Today, programming languages allow for reusing code, which makes development faster and more efficient. Projects are also shorter now, and customers are involved throughout the process, not just at the beginning and end.

Compulsory vs. Voluntary Systems: Some systems, like those used for order processing in companies, are required for employees to use. On the other hand, systems like computer games are voluntary, where users choose to use them. For voluntary systems, it's harder to define exact requirements since it depends on creativity and user feedback through surveys and prototypes.

Information Systems vs. Embedded Systems: Information systems help manages office tasks, like tracking stock or processing orders. Embedded systems control physical machines, such as air conditioning. Some systems may combine both, like a stock system that also controls an automated warehouse.

Software Products vs. Software Services: Software projects can focus on creating products or providing services. A software product is made for a wide audience and sold to many customers, like Microsoft Windows or Oracle's database software. Software services include things like customizing, maintaining, or testing software, and focus on meeting specific customer needs rather than selling a product.

Outsourced Projects: Large projects are sometimes split into parts that are outsourced to other companies. Outsourcing is useful when a company lacks expertise in certain areas or when another company can do the work more cost-effectively.

Object-Driven Development: Some projects are divided into two stages. The first stage focuses on understanding the need for a new system, and the second stage is about developing the actual software product.

These categories help determine how a software project should be planned, managed, and executed based on its specific characteristics and goals.

1.8 STAKEHOLDERS

Stakeholders are individuals or groups who have an interest or stake in the successful completion of a software project. Identifying stakeholders early in the project is crucial, as it allows for setting up proper communication channels from the start. Stakeholders may have different motivations and objectives. For instance, end-users may care about the system's ease of use, while managers may focus on how the system reduces staff costs.

According to Boehm and Ross's 'Theory W' of software project management, the goal is to create situations where all parties benefit from the project, ensuring mutual success. The 'W' stands for "win-win."

Stakeholders can be categorized as follows:

1. **Internal to the Project Team**: These stakeholders are directly managed by the project leader. They include team members such as developers, designers, and testers, who work under the project leader's supervision.
2. **External to the Project Team but Within the Organization**: These are individuals or groups within the same organization who may not be part of the project team but provide support. Examples include departments like information management or user groups involved in system testing. The project leader must negotiate and gain commitment from these stakeholders.

3. **External to Both the Project Team and the Organization**: These stakeholders are outside the organization and may include customers, end-users, or contractors who carry out specific tasks for the project. Relationships with these stakeholders are often governed by legally binding contracts.

Each stakeholder group may have different objectives, and it is the project leader's responsibility to understand and reconcile these differing interests. Therefore, effective communication and negotiation skills are essential for the success of the project.

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.

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.

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.

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.

PROJECT SUCCESS AND FAILURE

Project success is achieved when the project meets its objectives by delivering the agreed functionality, maintaining the required quality, and doing so within the constraints of time and budget. Additionally, the project must align with the business case, providing benefits that outweigh its costs.

However, project failure occurs when these objectives are not met, or when the delivered product does not fulfill the intended business needs. For instance, a project may be completed on time and within budget, but if the end product fails to generate anticipated revenue or meet user expectations, it is considered a failure in business terms. Conversely, a project might exceed its budget and timeline but still be successful if it generates long-term benefits that justify the initial setbacks.

Software Project Management is Crucial

Software project management is critical in ensuring project success as it provides a structured approach to planning, executing, and monitoring projects. It ensures that projects align with their business case by balancing technical and business considerations. Effective management addresses the diverse interests of stakeholders, minimizes risks, and keeps the project on track in terms of time, cost, and quality.

Good project management also allows teams to learn from experience, enabling faster and more efficient execution of future projects while building strong customer relationships. This comprehensive focus helps bridge the gap between project objectives and business outcomes, reducing the likelihood of failures.

MANAGEMENT AND MANAGEMENT CONTROL

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.

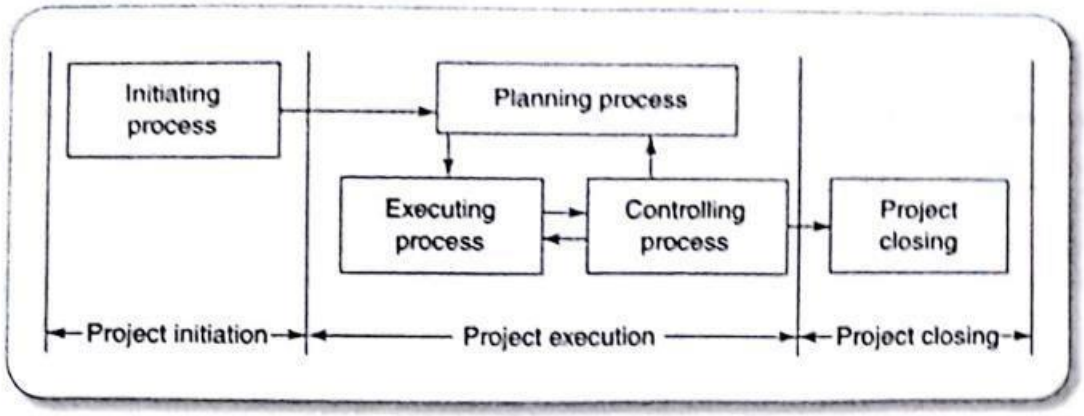


FIGURE 1.5 Principal project management processes

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.

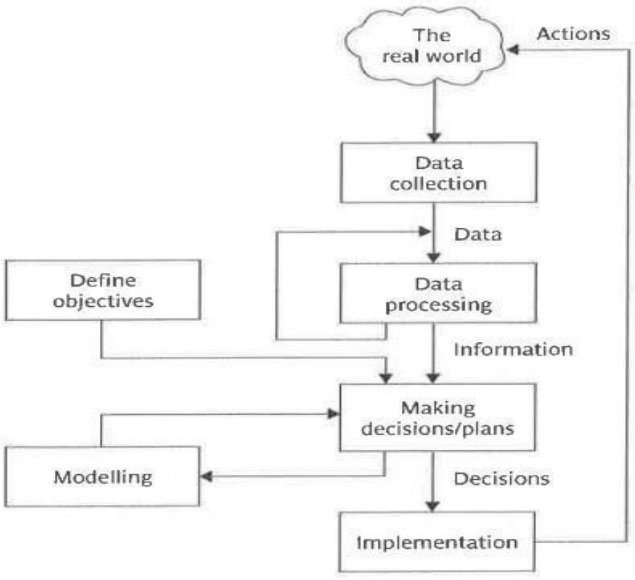


Fig: The Project control cycle

- 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.

1.9 PROJECT MANAGEMENT LIFE CYCLE

The Project Management Life Cycle outlines the sequential stages a project undergoes from initiation to closure, ensuring systematic and efficient execution. Unlike the Software Development Life Cycle (SDLC), which focuses solely on software creation, the project management life cycle begins before software development starts and spans the entire project duration.

The main phases of the project management life cycle are: project initiation, planning, execution, monitoring, controlling and closing.

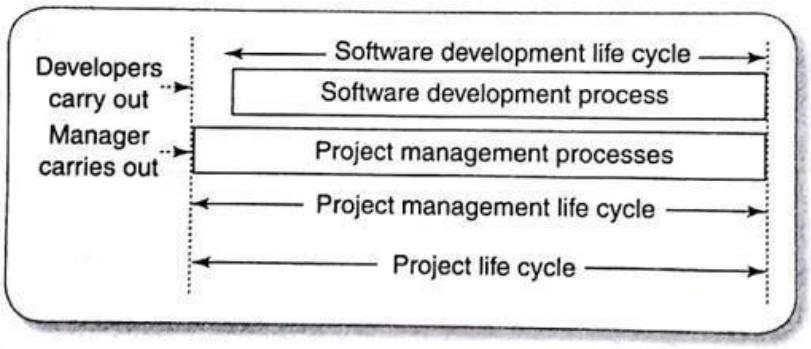


FIGURE 1.7 Project management life cycle versus software development life cycle

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

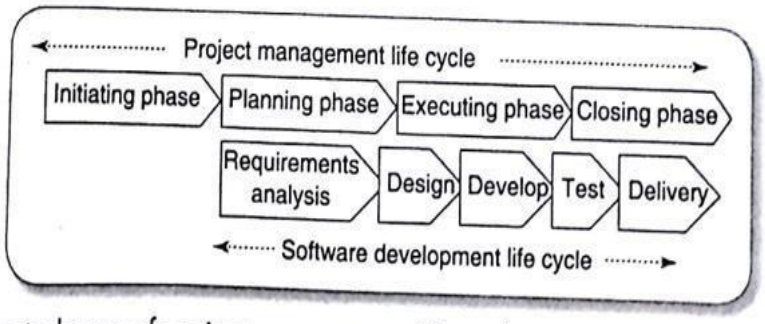


FIGURE 1.8 Different phases of project management life cycle and software development life cycle

Project Initiation

The project initiation phase focuses on defining the project scope, constraints, costs, and benefits. This includes conducting a **feasibility study** to determine whether the project is financially and technically viable. Once deemed feasible, a **business case** is prepared and approved by top management. Subsequently:

- A project manager is appointed.
- The project charter is developed.
- The project team is formed.

W5HH Principle: Barry Boehm proposed a set of key questions to guide project initiation:

- *Why* is the software being built?
- *What* will be done?
- *When* will it be completed?
- *Who* is responsible for specific functions?
- *Where* are the team members located?
- *How* will the work be done technically and managerially?
- *How much* resources are required?

2. Project Bidding

For specific projects, a formal bidding process may be required to select suitable vendors. Bidding techniques include:

- **Request for Quotation (RFQ)**: Used when the organization understands both the problem and the solution, seeking competitive pricing.
- **Request for Proposal (RFP)**: Used when the organization understands the problem but not the solution, aiming to gather potential solutions before finalizing requirements.
- **Request for Information (RFI)**: Used to assess vendor competencies and shortlist suitable candidates for bidding.

3. Project Planning

This phase involves creating comprehensive plans to guide project execution. The project manager prepares several key documents, including:

1. **Project Plan**: Defines tasks, schedules, resource assignments, and timeframes.
2. **Resource Plan**: Lists required manpower, tools, and equipment.
3. **Functional Plan**: Details manpower, equipment, and associated costs.
4. **Quality Plan**: Sets quality targets and control processes.
5. **Risk Plan**: Identifies potential risks, prioritizes them, and outlines mitigation strategies.

4. Project Execution:

In this phase, the project team executes tasks as outlined in the project plan. Key deliverables are produced and tested for quality assurance. Once the deliverables are completed and accepted by the customer, the project moves to the closure phase.

5. Project Closure

Project closure ensures the formal completion of the project. Activities include:

- Delivering all required outputs to the customer along with documentation.
- Releasing project resources and terminating vendor agreements.
- Finalizing pending payments.
- Conducting a **post-implementation review** to evaluate project performance and document lessons learned for future projects.

The project management life cycle provides a structured approach to managing projects, ensuring clarity, efficiency, and alignment with business goals. Each phase plays a critical role in steering the project towards successful completion.

1.10 TRADITIONAL VERSUS MODERN MANAGEMENT PRACTICES

Over the last two decades, the approach to software development has shifted significantly. Modern project management practices have evolved to address the complexities of shorter timelines, increased customer involvement, and rapid delivery cycles. Below are the key differences between traditional and modern project management practices:

1. **Planning and Delivery Approach**
   - **Traditional:** Projects were simpler, more predictable, and planned in detail before execution. Monitoring focused on adherence to this plan.
   - **Modern:** Emphasizes incremental delivery with evolving functionalities. Rapid application development and deployment are key strategies. Extreme project management, focusing on flexibility and stakeholder management, is commonly adopted.
2. **Quality Management**
   - **Traditional:** Quality assessment was primarily at the final stages of the project.
   - **Modern:** Increased customer awareness requires continuous quality tracking of all intermediate artifacts throughout the project lifecycle.
3. **Change Management**
   - **Traditional:** Changes to requirements were rarely entertained once they were signed off.
   - **Modern:** Actively solicits and incorporates customer feedback during the development process. Incremental delivery models and version control ensure effective change management, also known as configuration management.
4. **Requirement Management**
   - **Traditional:** Requirements were identified upfront, signed off, and frozen before development began.
   - **Modern:** Requirements often evolve during the development cycle. Modern practices involve systematic processes for documenting, analyzing, tracing, prioritizing, and communicating changes to stakeholders.
5. **Release Management**
   - **Traditional:** Software was typically delivered as a single release after development was complete.
   - **Modern**: Advocates for frequent and regular releases, starting with core functionalities and gradually adding more. Effective release management ensures timely and prioritized delivery.

6. **Risk Management**
    - **Traditional:** Risk management was often limited in scope and focused on obvious risks.
    - **Modern:** Involves identifying, assessing, prioritizing, and preparing containment plans for risks that might threaten project success. Effective risk management is critical to modern projects.
7. **Scope Management**
    - **Traditional**: Scope was predefined and rigid, with minimal changes allowed.
    - **Modern:** Encourages accepting customer change requests. Scope adjustments are carefully managed in relation to the project's schedule, cost, and quality.

PROJECT EVALUTION

Evaluation of Individual Projects

The evaluation of individual software projects involves determining their feasibility, economic viability, and strategic fit within an organization. Hughes outlines key areas of focus:

1. Feasibility Assessment

- **Technical Feasibility**: Evaluate if the necessary technology and expertise are available to complete the project successfully.
    - Example: Assessing whether the required programming languages or frameworks are supported by the organization's skill set.
- **Operational Feasibility**: Analyze if the project aligns with the operational objectives and can be effectively integrated into the existing workflow.
- **Legal and Ethical Feasibility**: Ensure that the project complies with regulations, intellectual property laws, and ethical standards.

2. Cost and Benefit Analysis

- This involves comparing the financial and non-financial costs of the project against its potential benefits.
    - **Net Present Value (NPV)**: To assess the financial viability of the project over time.
    - **Cost-Benefit Ratio**: To determine if the benefits outweigh the costs.
    - **Break-even Analysis**: To find the point at which the project will start generating profit.

3. Risk Assessment

- Projects are evaluated for potential risks that could impede their success. This includes:
    - **Technical Risks**: Will the technology perform as expected?
    - **Project Risks**: Can the project be completed within the constraints of time, budget, and resources?
    - **Market Risks**: Will the software meet user demands and expectations?

4. Strategic Alignment

- Projects must align with the broader business or organizational goals. Hughes emphasizes evaluating whether the project contributes to:
  - Increasing operational efficiency.
  - Enhancing customer satisfaction.
  - Generating revenue or other strategic advantages.

5. **Evaluation Tools**

- **Financial Metrics**: Use of NPV, IRR, and payback period to evaluate financial viability.
- **SWOT Analysis**: Identification of strengths, weaknesses, opportunities, and threats for the project.
- **Prototyping**: Developing a prototype to evaluate the feasibility and scope.

6. **Key Performance Indicators (KPIs)**

- Hughes highlights the importance of defining measurable outcomes to determine project success.
  - Example KPIs: Delivery timelines, cost efficiency, user adoption rates, and software reliability

Cost–Benefit Evaluation Techniques

Cost-Benefit Evaluation Techniques are vital for determining the feasibility of a project by comparing its expected costs against the anticipated benefits. This process is essential in making informed decisions, especially in the context of software development.

Importance of Cost–Benefit Analysis

- Evaluates whether a project should proceed based on its potential value versus its investment.
- Ensures resources are allocated to projects that deliver the highest return or strategic value.

1. **Net profit :** is a financial metric that represents the financial gain achieved from a project. It is calculated as the difference between the total revenue generated by the project and the total costs incurred during its execution. In simpler terms, it reflects the monetary value that a project contributes to the organization after accounting for all expenses.

$$\text{Net Profit} = \text{Total Revenue (or Benefits)} - \text{Total Costs}$$

2. **Payback Period:** is a financial metric used to determine the amount of time required to recover the initial investment of a project through its cash inflows. It provides a straightforward way to assess the financial feasibility of a project by measuring how quickly the initial cost can be recouped.

   One of the main advantages of the payback period is its simplicity and ease of computation, making it a popular choice for quick evaluations and comparisons among projects. However, it has notable limitations. It ignores the time value of money, meaning it does not account for the

fact that money received sooner is more valuable than money received later. Additionally, it does not consider any cash flows that occur beyond the payback period, potentially overlooking the long-term profitability of a project.

3. **Net Present Value (NPV):** is a financial metric that calculates the present value of a project's expected cash inflows and outflows over its life, considering the time value of money. This means that NPV takes into account the fact that money today is worth more than the same amount in the future due to factors such as inflation, interest rates, and opportunity cost.

NPV Works on :

1. **Cash Inflows**: These are the revenues or benefits the project is expected to generate over time.
2. **Cash Outflows**: These include all costs associated with the project, such as development, operation, and maintenance costs.
3. **Discount Rate**: The rate used to adjust future cash flows to their present value. This rate reflects the opportunity cost of capital (the return expected from other investments of similar risk).
4. **Time Period**: The length of time over which the project will generate cash flows.

Formula

$$NPV = \sum_{t=1}^{n} \frac{C_t}{(1+r)^t} - C_0$$

Where

- C_t: Cash inflow in year ttt
- C_0: Initial investment (cash outflow at the beginning)
- r: Discount rate
- n: Number of years or periods.

If Positive NPV: Project is financially viable.
Negative NPV: Project may not be worth pursuing.

4. Internal Rate of Return (IRR): Is a critical financial metric used to evaluate the profitability of projects. It is defined as the discount rate at which the Net Present Value (NPV) of all cash flows associated with a project equals zero. Essentially, IRR represents the break-even cost of capital for a project.

The primary purpose of IRR is to compare projects by determining their profitability rates. It is particularly useful for ranking multiple investment opportunities or assessing the feasibility of a single project.

If

IRR > Cost of Capital: Project is viable.
IRR < Cost of Capital: Project is not viable.

5. **Cost-Benefit Ratio (CBR):** is the ratio of **total benefits** to **total costs**. A higher ratio indicates that the project provides more value for each unit of cost, making it more favorable from a financial perspective.

$$CBR = \frac{\text{Total Benefits}}{\text{Total Costs}}$$

Where:

- **Total Benefits**: The overall financial or strategic gains from the project (can include revenue, savings, or intangible benefits).
- **Total Costs**: The sum of all expenses incurred in the project (including development, operational, and maintenance costs).

If CBR > 1: Project is beneficial and if CBR < 1: Costs outweigh benefits.

Risk Evaluation

Risk evaluation is a critical part of project management, particularly for software projects, due to their complexity and uncertainty. The process involves identifying, analyzing, and prioritizing risks to mitigate their impact on project success.

Importance of Risk Evaluation

- Software projects often face challenges like technical complexity, time constraints, and changing requirements.
- Risk evaluation ensures potential problems are anticipated and planned for, minimizing disruptions.

Risk Management Process

The Risk management process includes:

1. **Risk Identification:** Identifying potential risks is an important part of managing any project. In software projects, there are several risks that can affect success.

   Incomplete requirements are a common risk, where unclear or changing project needs can lead to mistakes or features that don't meet expectations. **Unrealistic schedules** can also be a problem, where setting overly short deadlines might force the team to rush, resulting in poor quality or missed deadlines. **Resource unavailability** is another risk, which happens when key team members are unavailable, or when the team lacks the skills or tools needed to complete the project. These risks can cause delays or failure, so it's important to identify them early and find ways to manage them.

2. **Risk Analysis**: To manage risks in a project, it's important to **identify** and **measure** how likely a risk is to happen and how much it could affect the project.

   Techniques

   1. **Qualitative Analysis**:
      - This approach involves describing risks based on how likely they are to happen and how much damage they could cause. One common method is using a **risk matrix**, where risks are rated on two scales: one for the **likelihood** (e.g., low, medium, high) and one for the **impact** (e.g., low, moderate, high). This helps prioritize which risks need the most attention.
   2. **Quantitative Analysis**:
      - This method uses numbers and data to estimate the exact impact of a risk on things like budget or timeline. For example, if there's a 30% chance of a risk costing $10,000, the analysis helps calculate how much this risk could affect the overall project in dollar terms.

By using both qualitative and quantitative methods, project managers can better understand the risks, decide which ones to focus on, and plan how to reduce their impact on the project.

3. **Risk Prioritization**: To manage risks effectively, it's important to **rank** them based on how likely they are to happen and how serious their impact would be. Tools like the **Risk Priority Number (RPN)** or **Risk Matrices** help with this ranking.
4. **Risk Mitigation Planning**:
   - Develop strategies to address risks.
     - **Avoidance**: Alter project plans to eliminate risks.
     - **Reduction**: Minimize the likelihood or impact of risks.
     - **Transfer**: Outsource or insure to shift responsibility.
     - **Acceptance**: Acknowledge and monitor low-priority risks.
5. **Monitoring and Control**:
   - Continuously track risks throughout the project lifecycle.
   - Update the risk plan based on changing circumstances.