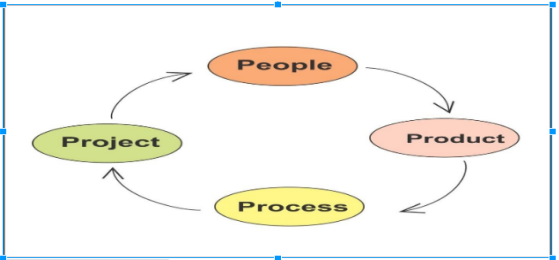
Unit 5

31.PROJECT MANAGEMENT CONCEPTS

1.“Effective software project management focuses on four P’s”. Justify this statement with suitable analogy.

2.Briefly explain the management spectrum in software project management.

31.1 THE MANAGEMENT SPECTRUM



Effective software project management focuses on the four Ps: people, product, process, and project. The order is not arbitrary. The manager who forgets that software engineering work is an intensely human endeavour will never have success in project management. A manager who fails to encourage comprehensive stakeholder communication early in the evolution of a product risks building an elegant solution for the wrong problem. The manager who pays little attention to the process runs the risk of inserting competent technical methods and tools into a vacuum. The manager who embarks without a solid project plan jeopardizes the success of the project.

**31.1.1 The People**

The cultivation of motivated, highly skilled software people has been discussed since the 1960s. In fact, the “people factor” is so important that the Software Engineering Institute has developed a People Capability Maturity Model (People-CMM), in recognition of the fact that “every organization needs to continually improve its ability to attract, develop, motivate, organize, and retain the workforce needed to accomplish its strategic business objectives” [Cur01].

The people capability maturity model defines the following key practice areas for software people: staffing, communication and coordination, work environment, performance management, training, compensation, competency analysis and development, career development, workgroup development, and team/ culture development, and others. Organizations that achieve high levels of People-CMM maturity have a higher likelihood of implementing effective software project management practices. The People-CMM is a companion to the Software Capability Maturity Model– Integration (Chapter 37) that guides organizations in the creation of a mature software process. Issues associated with people management and structure for software projects are considered later in this chapter.

**31.1.2 The Product**

Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered, and technical and management constraints should be identified. Without this information, it is impossible to define reasonable estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.

As a software developer, you and other stakeholders must meet to define product objectives and scope. In many cases, this activity begins as part of the system engineering or business process engineering and continues as the first step in software requirements engineering . Objectives identify the overall goals for the product (from the stakeholders’ points of view) without considering how these goals will be achieved. Scope identifies the primary data, functions, and behaviours that characterize the product, and more important, attempts to bound these characteristics in a quantitative manner. Once the product objectives and scope are understood, alternative solutions are considered. Although very little detail is discussed, the alternatives enable managers and practitioners to select a “best” approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and myriad other factors.

**31.1.3 The Process**

A software process (Chapters 3–5) provides the framework from which a comprehensive plan for software development can be established. A small number of framework activities are applicable to all software projects, regardless of their size or complexity. A number of different task sets—tasks, milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement—overlay the process model. Umbrella activities are independent of any one framework activity and occur throughout the process.

**31.1.4 The Project**

We conduct planned and controlled software projects for one primary reason—it is the only known way to manage complexity. And yet, software teams still struggle. In a study of 250 large software projects between 1998 and 2004, Capers Jones [Jon04] found that “about 25 were deemed successful in that they achieved their schedule, cost, and quality objectives. About 50 had delays or overruns below 35 percent, while about 175 experienced major delays and overruns, or were terminated without completion.” Although the success rate for present-day software projects may have improved somewhat, our project failure rate remains much higher than it should be. 1 To avoid project failure, a software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a commonsense approach for planning, monitoring, and controlling the project. Each of these issues is discussed in Section 31.5 and in the chapters that follow.

**Illustrate the roles of software teams and team leaders in software project management.**

**31.2 PEOPLE**

The effectiveness of a software engineering team relies heavily on the people involved and their interactions within the software process. This process includes various stakeholders, each playing a crucial role in the success of the project. Here's a closer look at these stakeholders and how they contribute:

**31.2.1 The Stakeholders**

**1. Senior Managers**

* **Role:** Senior managers define the overarching business issues and strategic goals that often shape the direction and priorities of software projects.
* **Influence:** Their decisions on funding, resource allocation, and strategic focus can significantly impact the project's scope, schedule, and overall success.
* **Impact Example:** If senior managers prioritize a quick market release, the team may need to focus on delivering a minimum viable product (MVP) quickly, which might impact the depth of initial features but align with business goals.

**2. Project (Technical) Managers**

* **Role:** Project managers plan, motivate, organize, and control the practitioners doing the software work.
* **Responsibilities:** They are responsible for scheduling, resource management, risk management, and ensuring that the project meets its objectives.
* **Impact Example:** Effective project management can lead to better team coordination, timely project completion, and efficient handling of obstacles. Poor project management, on the other hand, can result in missed deadlines and project failure.

**3. Practitioners**

* **Role:** Practitioners are the technical professionals who carry out the actual software engineering tasks, including coding, testing, and maintenance.
* **Skills:** They bring the necessary technical expertise and problem-solving skills to develop high-quality software products.
* **Impact Example:** Well-trained and motivated practitioners can innovate and optimize processes, leading to better performance and quality. Conversely, lack of skills or motivation among practitioners can lead to subpar software quality and delays.

**4. Customers**

* **Role:** Customers specify the requirements for the software, providing the essential input needed to guide the development process.
* **Expectations:** They define what the software should do and what features it must include, directly influencing the design and functionality of the product.
* **Impact Example:** Clear and well-documented requirements from customers can lead to a product that meets expectations and adds value. Ambiguous or changing requirements can lead to scope creep and project delays.

**5. End Users**

* **Role:** End users are the individuals who interact with the software once it is deployed.
* **Feedback:** Their feedback is critical for continuous improvement and ensuring the software remains useful and relevant.
* **Impact Example:** Positive end-user feedback can lead to higher adoption rates and customer satisfaction, while negative feedback can highlight areas for improvement and potential issues that need addressing.

**Integration of Stakeholders**

Effective software engineering requires a collaborative effort among all these stakeholders. Here’s how they work together:

* **Communication:** Regular and clear communication among stakeholders ensures that everyone is aligned with the project goals and understands their roles and responsibilities.
* **Feedback Loops:** Continuous feedback from customers and end users helps practitioners make necessary adjustments, leading to a better final product.
* **Decision-Making:** Senior managers provide strategic direction, while project managers handle day-to-day decisions that keep the project on track.
* **Skill Utilization:** Practitioners apply their technical skills to build the software, guided by the requirements specified by customers and influenced by end-user feedback.

**31.2.2 Team Leaders**

Project management, especially in software development, demands a unique blend of technical expertise and people skills. Competent practitioners may not always make effective team leaders due to a lack of necessary leadership qualities. This section explores the characteristics of effective team leaders and project managers through the MOI model of leadership and other key traits.

**MOI Model of Leadership (Jerry Weinberg)**

**1. Motivation**

* **Definition:** The ability to encourage technical people to perform at their best.
* **Application:** Effective leaders inspire their teams through a combination of pushing (direct encouragement) and pulling (creating an environment that naturally motivates).

**2. Organization**

* **Definition:** The ability to shape processes to translate initial concepts into final products.
* **Application:** Leaders must be adept at molding existing workflows or creating new ones that facilitate efficient and effective software development.

**3. Ideas or Innovation**

* **Definition:** The ability to foster creativity within the constraints of a given project.
* **Application:** Encouraging team members to think creatively and innovate, even when working within established boundaries, is crucial for successful project outcomes.

**Key Traits of an Effective Project Manager (Edgemon)**

**1. Problem Solving:**

* **Definition:** The capacity to diagnose technical and organizational issues, structure solutions, and adapt lessons from past projects.
* **Application:** A project manager must be flexible and able to pivot when initial solutions are ineffective, maintaining a problem-solving mindset throughout the project lifecycle.

**2. Managerial Identity:**

* **Definition:** The confidence to assume control when necessary and allow technical experts to exercise their instincts.
* **Application:** Balancing control with delegation is critical. A project manager must know when to step in and when to step back, enabling team members to utilize their expertise effectively.

**3. Achievement:**

* **Definition:** Rewarding initiative and accomplishment to optimize productivity.
* **Application:** Recognizing and rewarding team members' efforts and achievements fosters a culture of productivity and controlled risk-taking, which is essential for innovation and progress.

**4. Influence and Team Building:**

* **Definition:** The ability to read people, understand their needs, and maintain composure under stress.
* **Application:** An effective project manager must be adept at interpreting verbal and nonverbal cues, responding to team members' needs, and remaining calm in high-stress situations to keep the team focused and motivated.

**Problem-Solving Management Style**

Weinberg suggests that successful project leaders should adopt a problem-solving management style. This involves:

* **Understanding the Problem:** Focusing on diagnosing and comprehending the core issues to be solved.
* **Managing the Flow of Ideas:** Encouraging open communication and idea sharing within the team.
* **Emphasizing Quality:** Communicating the importance of quality through words and actions, ensuring that it is never compromised.

**31.2.3 The Software Team**

Effective organization of software teams is crucial for the success of software projects. This organization is influenced by several factors, and there are various paradigms to structure teams. Here’s a detailed look at these considerations and paradigms:

**Factors to Consider in Team Structure**

According to Mantei, the structure of a software engineering team should take into account the following factors:

1. **Difficulty of the Problem:** The complexity of the software problem to be solved.
2. **Size of the Program:** Measured in lines of code or function points.
3. **Team Lifetime:** The duration for which the team will remain together.
4. **Modularity of the Problem:** The extent to which the problem can be divided into smaller, manageable parts.
5. **Quality and Reliability Requirements:** The level of quality and reliability expected from the software.
6. **Delivery Date Rigidity:** The strictness of the project deadlines.
7. **Sociability Needs:** The level of communication and interaction required among team members.

**Organizational Paradigms for Software Teams**

Constantine suggests four organizational paradigms for structuring software engineering teams:

1. **Closed Paradigm:**
   * **Structure:** Traditional hierarchy of authority.
   * **Suitability:** Effective for projects similar to past efforts, but may stifle innovation.
   * **Example:** A team producing routine software updates might benefit from this structure.
2. **Random Paradigm:**
   * **Structure:** Loose organization relying on individual initiative.
   * **Suitability:** Excels in projects requiring innovation and technological breakthroughs, but may struggle with structured, orderly performance.
   * **Example:** A research and development team exploring new technologies might thrive under this paradigm.
3. **Open Paradigm:**
   * **Structure:** Balances control and innovation with collaborative work, heavy communication, and consensus-based decision-making.
   * **Suitability:** Well-suited for solving complex problems, though potentially less efficient.
   * **Example:** A team working on a multifaceted, innovative software solution might find this structure beneficial.
4. **Synchronous Paradigm:**
   * **Structure:** Organizes team members to work on different parts of the problem with minimal communication.
   * **Suitability:** Works well when the problem is naturally compartmentalized.
   * **Example:** A team developing different modules of a large software system independently.

**Creating a High-Performance Team**

For a software team to be effective, it must exhibit cohesiveness, often described as a "jelled" team. According to DeMarco and Lister, a jelled team is a group of people so strongly knit that the whole is greater than the sum of its parts. Characteristics of jelled teams include:

* **Common Goals:** Sharing a unified definition of success.
* **Team Spirit:** Exhibiting a strong team spirit and sense of eliteness.
* **Self-Motivation:** Functioning with minimal management and high motivation.
* **Increased Productivity:** Significantly more productive and motivated than average teams.

**Team Toxicity**

Jackman identifies five factors that contribute to a toxic team environment:

1. **Frenzied Work Atmosphere:** High-pressure environments that lead to burnout.
2. **High Frustration:** Frustration causing friction among team members.
3. **Fragmented Processes:** Poorly coordinated software processes.
4. **Unclear Roles:** Lack of clear role definitions on the team.
5. **Continuous Failure Exposure:** Frequent exposure to failure, leading to demotivation.

**Avoiding Team Toxicity**

Project managers can help avoid these toxic factors by:

* **Providing Necessary Information:** Ensuring the team has all the required information.
* **Defining Goals Clearly:** Keeping major goals and objectives stable unless changes are absolutely necessary.
* **Empowering Decision Making:** Giving the team responsibility for decision making.
* **Choosing Appropriate Processes:** Understanding the product and people, and allowing the team to select the process model.
* **Establishing Accountability:** Implementing mechanisms for accountability, such as technical reviews.
* **Using Team-Based Feedback:** Establishing techniques for feedback and problem solving.

**Recognizing Human Differences**

Teams often struggle due to differing human traits. Understanding and managing these differences is essential for creating cohesive teams:

* **Extroverts vs. Introverts:** Different approaches to communication and interaction.
* **Intuitive vs. Logical Information Processing:** Different methods of gathering and organizing information.
* **Decision-Making Styles:** Varying preferences for making decisions based on logic or intuition.
* **Work Style Preferences:** Different attitudes toward schedules and task closure.
* **Stress Responses:** Varied responses to deadlines and stress.

**32.PROCESS AND PROJECT METRICS**

Describe any three-software metrics used for software measurement.