Student Name: Anubhav Mahajan (40267770)

Course: SOEN 6841

Journal URL: https://github.com/anubhavm101/SPM-Learning-Journal

Dates Rage of activities: 02-Nov-2024 to 09-Nov-2024

Date of the journal: 09-Nov-2024

1. Key Concepts Learned (12 points)

Key Concepts Learned:

Project Closure: Project closure is a structured process for formally ending a project. It guarantees that all deliverables are finished, version control is established, and metrics data is archived. Capturing and assessing lessons learnt is an important aspect of this process, since it helps enhance future project results. Documenting difficulties, answers, and significant learnings provides insights into prospective changes and promotes continuous progress in project management techniques.

Software Lifecycle Management: This chapter explores key software development life cycle models:

- **Waterfall Model**: A linear, sequential strategy that moves through phases such as requirement collecting, design, and construction without revisiting prior ones. It is suitable for projects with stable requirements (e.g., huge enterprise systems), but risks inefficiency if changes occur later.
- **Iterative Models (e.g., SCRUM, Extreme Programming)**: These models provide flexibility by iterating between phases, allowing teams to respond swiftly to changing requirements. They are useful in rapidly growing industries like mobile applications and social media.

Quality Assurance in the Lifecycle: Both waterfall and iterative approaches use quality gates at each phase. However, iterative models frequently incorporate concurrent engineering processes, allowing for real-time tweaks and making them very adaptive to new technology.

2. Application in Real Projects:

- **Project Closure and Lessons Learned**: Document critical learnings, difficulties, and solutions, especially in iterative projects with frequent modifications. Archiving metrics and source control data enables teams to revisit and improve judgments based on past experiences. This applies to current projects such as modernizing ERP systems or developing apps for upcoming technology.

- **Waterfall vs. Iterative Models**: The waterfall model promotes stability in large-scale corporate software projects, specifically those involving numerous departments. Iterative models, on the contrary, are appropriate for startups that value rapid deployment and market trend adaptability. A hybrid strategy can be advantageous by combining waterfall concepts at the highest levels with iterative methods at the team level.

3. Peer Interactions/Collaboration:

I and my project mates worked on the **Intelligent Tutoring System (ITS)** project, focusing on **project planning, budgeting, risk assessment, and feasibility**. These discussions provided insights into the practical application of the learned concepts. Specifically, we worked on project scope, set budgets for the ITS, evaluated risks, and tested the feasibility of different aspects of the project. A peer shared their experience of managing different projects, assigning budgets and doing risk analysis, reinforcing the importance of proper planning in project execution. The collaboration helped me understand more fully how iterative models could be used in software projects, particularly in cases when adaptability to evolving needs and unanticipated hazards depends on flexibility.

4. Challenges Faced:

Determining the best application for each model, especially in complex projects with nuanced requirements, was a significant difficulty. The waterfall paradigm, while straightforward, is less adaptable and may struggle in fast-changing contexts. I discovered that balancing structure and adaptability is critical to success. Gaining more understanding into managing hybrid models—which integrate waterfall and iterative practices—would be advantageous, especially for teams with diverse skill sets.

5. Personal Development Activities:

To gain a better understanding, I looked at case studies of firms that had successfully implemented SCRUM for managing iterative project models. This clarified the importance of iteration in software lifecycles. In addition, I researched our ITS project's specialized project management requirements, such as budgeting, risk assessment, and managing iterative improvements. I examined effective practices for guaranteeing feasibility while allowing for flexibility through iterative models. This was useful for understanding how to use adaptive project management strategies in real-world software projects. These activities are consistent with my goal of improving my theoretical and practical knowledge of software project management, particularly in balancing iterative development and formal planning.

6. Goals for the Next Week:

- Study **hybrid project management approaches**, focusing on integrating the **structure of the waterfall model** with the flexibility of iterative methods.
- Practice drafting documents for our ITS project to improve my understanding of project closure processes and how to assess risks and outcomes.
- Work on making a Gantt chart for my APP project so I can finish phase 2 of my project tubelytics.