Student Name: Dhruvil Rajabhai Shah
Course: SOEN 6841
Journal URL: Learning Journals
Dates Rage of activities: 06/02/2025 - 20/02/2025
Date of the journal: 21/02/2025

Key Concepts Learned

Over the past two weeks, I explored six key chapters covering fundamental topics such as:

- **Software Process Models**: Waterfall is linear and rigid, Agile is flexible with iterative development, and Spiral combines iterative development with risk management.
- **Agile Methodologies**: Scrum involves fixed sprints for iterative progress, while Kanban focuses on continuous flow and process optimization.
- **Requirements Engineering**: Distinguishes between functional (what the system should do) and non-functional requirements (how it should perform), and emphasizes stakeholder analysis and validation.
- **System Modeling**: UML diagrams (use case, sequence, and class) visualize system interactions and structure.
- **Software Architecture**: Layered architecture separates concerns, while microservices improve scalability but add complexity.
- **Design Patterns**: Patterns like Singleton, Factory, and Observer improve modularity and code reusability.

Application in Real Projects

The knowledge gained has been directly applicable to my **software project management tasks**, where I:

- **Used project management software (JIRA and Microsoft Project)** to break down tasks and track progress.
- **Implemented Agile methodologies**, following Scrum practices like sprint planning, backlog refinement, and daily stand-ups to improve workflow efficiency.
- **Developed a risk assessment matrix**, identifying risks such as scope creep and unrealistic deadlines, and implemented mitigation strategies inspired by large-scale projects like NASA's software development model.
- **Applied cost estimation models**, such as COCOMO, to estimate project effort and budget, similar to how major IT firms plan software releases.

Key Takeaways:

- Agile frameworks enabled **efficient project tracking and adaptability** in dynamic project environments.
- **Risk analysis techniques helped anticipate challenges** and create contingency plans to ensure smooth project execution.
- Understanding **stakeholder management improved communication** and requirement clarity within the project.

Peer Interactions and Collaboration

Engaging in discussions with peers helped clarify complex topics, particularly in requirements gathering and architecture decisions. Constructive feedback on my project pitch improved its clarity and technical depth. Additionally, we formed a **study group** to prepare for exams, solving past papers and discussing challenging topics, which significantly strengthened our conceptual understanding and problem-solving abilities.

Challenges Faced

- **Configuration Management Implementation:** Managing software version control across different development stages seemed complex. I explored **Git and CI/CD pipelines** to understand how companies handle large-scale deployments efficiently.
- **Understanding Architectural Trade-offs:** Choosing between monolithic and microservices architectures required an in-depth analysis of scalability and maintainability. I resolved this by studying industry case studies.
- **Managing Evolving Project Requirements:** Changing requirements made it difficult to keep documentation and task planning up to date. To address this, I adopted an iterative approach by refining scope frequently and maintaining a well-organized backlog.
- **Ensuring Accurate Cost Estimation:** Initial estimates varied significantly due to project uncertainties. I refined estimates using historical data and comparative analysis with similar past projects.
- **Balancing Exam Preparation and Project Work:** Managing multiple deadlines was challenging. I implemented a structured schedule, setting fixed study hours and project work sessions to stay on track.

Time Management

This week, I dedicated significant time to studying and revising the topics covered in lectures in preparation for the upcoming exam. I allocated approximately 12 hours to practice examples of cost estimating models, focusing on developing a deeper understanding of their application in real-world scenarios. Additionally, I spent around 8 hours on comprehending the integration and implementation of Continuous Integration/Continuous Deployment (CI/CD) pipelines, including their real-world application in software projects. A further 6 hours were dedicated to understanding how version control systems, such as Git, function in actual projects, with a focus on branching strategies, commit management, and collaborative workflows.

Personal Development Activities

- Practiced **exam-style project case studies**, focusing on analysing past failed projects and identifying key areas where poor management led to issues.
- Created a **mock project plan** from scratch to simulate a real-world project planning experience, which helped reinforce my learning.
- Collaborated with peers to **solve project scheduling and cost estimation exercises**, helping each other understand complex calculations and methodologies.
- I also started reading “Software Engineering: A Practitioner’s Approach” to deepen my knowledge of project management methodologies.

Goals for the Next Week

- Complete improved writing of the project management plan by incorporating suggestions from peers and fine-tuning the risk assessment strategies.
- Complete a refined version of the project management plan, integrating feedback from peers and refining risk assessment strategies for the project part.
- Create a project timeline using the Critical Path Method (CPM) to optimize scheduling and resource allocation.
- Enhance exam preparation by revising cost estimation models and practicing scenario-based questions.

References

- Lecture slides (Chapters 1-6) on software project management fundamentals.
- H. Kerzner (2009) Project Management: A Systems Approach to Planning, Scheduling, and Controlling, Wiley, Hoboken, NJ.