Learning Journal 2

Student Name: Shruti Hiteshbhai Pavasiya

Course: Software Project Management (SOEN 6841)

Dates Range of activities: 23rd September 2024 to 4th October 2024

Date of the journal: 5th October 2024

Chapter 4: Risk Management

Section	Content
Key Concepts Learned	- **Risk Identification**: Understanding internal vs. external risks and how they affect project scope, budget, quality, and time. - **Types of Risks**: Budget risks, resource unavailability, quality issues, time/schedule risks, and technology risks. - **Risk Mitigation Strategies**: Approaches to reduce risks, including proactive risk management and contingency planning. - **Risk Analysis**: Assessing the probability and impact of risks and prioritizing them in a **risk matrix**. - **Dynamic Nature of Risks**: Risks evolve throughout a project, requiring regular reassessment.
Application in Real Projects	- **Risk Prioritization**: Creating a **risk matrix** for real-world projects by ranking risks based on their probability and impact. - **Mitigation Strategies**: Apply strategies like **buffer creation** for schedule and budget overruns. - **Contingency Plans**: Using pre-planned responses to handle risks such as **resource unavailability** or **scope creep**. - **Agile Risk Management**: Adopt iterative models to mitigate large risks by breaking projects into smaller, manageable tasks.
Peer Interactions	- **Real-world Failures**: Discussed peer experiences where **poor risk management** led to project failures (e.g., tech obsolescence, budget overrun). - **Sharing Strategies**: Exchanged ideas on how to handle **external risks** like economic recessions and vendor issues. - **Debating Approaches**: Talked about the benefits of **agile vs. waterfall models** in handling risks, especially scope creep and quality assurance.
Challenges Faced	- **Risk Quantification**: Difficulty in accurately measuring **probability** and **impact** of risks in real life situations and project. - **Multiple Risk Prioritization**: Struggled with which risk should be prioritized that affect both **budget** and **schedule** simultaneously. - **Contingency Planning**: Thought about how to find the right balance between **contingency buffers** without over-allocating resources. - **Data Reliability**: Encountered issues with data quality and reliability, which made it harder to make informed decisions about risk management.
Personal Development Activities	- **Case Study Review**: Read studies comparing **risk management in agile and waterfall models**. - **Risk Tools**: Explored **Monte Carlo simulations** for quantitative risk

Section	Content
	analysis. - **Webinars**: Attended a session on using **RiskyProject** to simulate and manage risks in large projects. - **Collaboration Exercises**: Engaged in team exercises to practice collective risk assessment and mitigation strategies, improving overall team dynamics and response strategies.
Goals for the Next Week	- **Deepen Knowledge in Risk Evaluation**: Learn advanced risk management frameworks and quantitative techniques. - **Simulate Risk**: Use tools like **RiskyProject** to test various risk scenarios in project planning. - **Explore Agile Risk Handling**: Focus on using agile methodologies to handle risks in iterative project environments.

Chapter 5: Configuration Management

Section	Content
Key Concepts Learned	- **Configuration Management**: Ensures proper version control, document tracking, and artifact storage across distributed teams. - **Version Control**: Managing multiple software versions using tools like **Git** or **Subversion**. - **Centralized Systems**: Importance of centralized configuration management for handling distributed teams. - **Continuous Integration**: Frequent code check-ins, automated smoke tests to maintain build integrity. - **Branching and Merging**: Techniques to handle different versions of the software across projects.
Application in Real Projects	- **Version Control**: Implementing tools like **Git** to manage multiple versions of software across distributed teams. - **Continuous Integration**: Using tools like **Jenkins** to automate build processes and ensure smooth integration of new code. - **Role-Based Access**: Establishing permissions to control access to sensitive documents and ensuring the right team members have editing privileges. - **Branching Techniques**: Applying branching strategies to manage various versions of a project, minimizing conflicts and simplifying merges.
Peer Interactions	- **Shared Experience**: Discussed challenges of managing large projects across multiple locations without a **centralized configuration system**. - **Best Practices**: Exchanged ideas on **branching techniques** for handling different versions and managing frequent updates in real-time. - **Automation**: Talked about how using **automated smoke tests** ensures build stability and reduces manual effort in checking code compatibility.
Challenges Faced	- **Branching Strategies**: Struggled with understanding how to implement **advanced branching and merging** techniques in real-world scenarios. - **Automated Testing**: Difficulty in setting up and understanding **automated smoke tests** for continuous integration. - **Permission Management**: Managing role-based access to secure files without compromising team collaboration was challenging.

Section	Content
Personal Development Activities	- **Tool Exploration**: Practiced using **Git** for advanced branching and learned how to set up **continuous integration** using Jenkins. - **Meetups**: Attended a local meetup on **configuration management tools**, where I learned how to implement a centralized system in large projects. - **Security in CM Systems**: Researched **security measures** in configuration management systems, learning about **access control**, **data encryption**, and **audit logging** to protect sensitive project information. - **Continuous Integration Setup**: Practiced setting up **continuous integration**, which enhanced my understanding of automated build processes.
Goals for the Next Week	- **Master Branching**: Focus on learning advanced branching strategies, including **feature branching** and **trunk-based development**. - **Explore Advanced CM Tools**: Test more advanced configuration tools such as **Puppet** or **Chef** to better handle large project environments. - **Audit Facilities**: Learn how to implement **audit trails** for tracking changes in configurations, ensuring traceability and compliance in software projects.