Learning Journal Template

Student Name: Aryan Awasthi

Course: Software Project Management

Journal URL: https://github.com/arextron/SPM---Journals.git

Dates Range of activities: 21st September to 3rd October

Date of the journal: 3rd October

Chapter 4

Key Concepts Learned

- **Risk Definition:** Combination of event probability and negative consequence.
- **Risk Categories:** Include technical, legal, organizational, economic, and schedule risks.
- **Risk Assessment Process:**
  - Risk identification, analysis, and prioritization.
  - Conducted at the start of a project and revisited in iterations.
- **Risk Management Strategies:**
  - Acceptance, avoidance, transference, and mitigation.
- **Risk Prioritization:** Focus on risks with high likelihood and impact.
- **Risk Response:** Assign risk items and monitor them throughout the project.

Application in Real Project:

- Applied risk prioritization in an AI project, focusing on critical uncertainties like model accuracy.
- Added buffer time during project estimation to manage estimation risks related to technology challenges.

Peer Interactions.

- Discussed qualitative vs. quantitative risk assessments with classmates, realizing the benefits of combining both approaches.
- Learned from peers about using risk transference (outsourcing) to manage large-scale project risks.

Challenges Faced

- Estimating probability for risks in modern technologies without historical data.
- Identifying trigger points early to avoid delayed risk responses.
- Balancing risk mitigation costs versus the impact of risks.

Personal Development Activities

- Explored JIRA for risk tracking and monitoring.
- Read article on high-risk, high-benefit projects to apply strategies in software development**.**

Goals for the Next Week:

- Focus on improving contingency planning and identifying risk-trigger points.
- Research integrating risk management tools with real-time data analytics.
- Apply risk prioritization to a new project from the beginning.

Chapter 5

Key Concepts Learned:

- **Configuration Management (CM):** Process of controlling and documenting changes to a system.
- **Sources of Change:** Include requirement changes, technology advancements, scheduling constraints, and customer expectations.
- **CM Importance:** Prevents project chaos, ensures product integrity, and reduces rework and lifecycle costs.
- **Four Functions of CM:**
  - Configuration Identification
  - Configuration Control
  - Configuration Status Accounting
  - Configuration Auditing
- **Change Control Policy:** Requires documented, approved, and traceable changes to avoid confusion and errors.

Application in Real Projects:

- Implemented CM tools in projects to track multiple versions and manage change requests efficiently.
- Used status accounting to maintain a detailed history of changes, improving traceability and compliance with project standards

Peer Interactions:

- Discussed the challenges of uncontrolled changes with peers, who shared how version control systems like Git have helped maintain order in their projects.
- Peers highlighted the importance of audits to verify that all changes conform to baseline requirements, which inspired me to conduct more frequent audits for commits.

Challenges Faced:

- Managing scope creep: Dealing with additional customer requirements during later stages of a project.
- Ensuring proper version control: It was sometimes challenging to ensure that all team members were working on the correct version of the software.

Personal development Activities:

- Explored tools like **Git and Jenkins** for better configuration management and continuous integration.
- Read articles on **change management** best practices to improve the handling of change requests in future projects

Goals For the Next week:

- Integrate **automated CM tools** (like Jenkins) for better tracking of code changes and deployment.
- Focus on **reducing scope creep** by applying stricter change control procedures.