Learning Journal 1

Student Name: Santhosh Dayakar

Course: SOEN 6481 Software Project Management

Journal URL:https://github.com/S-15-77/Software-Project-Management-Journal

Dates Range of activities: 08-09-2025 to 15-09-2025

Date of the journal: 18-09-2025

Key Concept Learned:

This week, I explored two foundational concepts in software project management: SMART objectives and effort estimation techniques. SMART objectives (Specific, Measurable, Achievable, Relevant, and Time-bound) are essential during the project initiation phase to define clear and actionable goals. For instance, instead of stating "enhance system speed," a SMART goal would be "reduce page load time by 40% within three weeks." This reminded me of how poorly defined goals in earlier course projects often led to delays, showing me how SMART goals can align teams and improve outcomes.

Effort estimation techniques like estimation by analogy and the Delphi method introduced structured ways to predict project workload. Estimation by analogy compares current tasks to similar past ones, while Delphi involves expert consensus through iterative feedback. These techniques resonated with my experience in capstone planning sessions where initial guesses improved after team discussion. I now understand how formal estimation reduces planning errors and supports better budgeting and scheduling.

Application to Real Project:

The concept of SMART objectives can directly improve the planning process in any collaborative project. For example, in my current course-based team assignment, we initially had a vague goal to "finish the prototype soon." After learning about SMART objectives, I proposed reframing it as "Complete the UI and core functionality of the login and dashboard modules by October 5th." This added clarity and accountability. The benefit was immediate; team members knew what to prioritize, but the challenge was estimating how long each task would realistically take, especially with differing skill levels.

Another area where I applied course learning was during a past internship, where our team often struggled with accurate effort estimation. Had we used the Delphi method, we could have avoided underestimating sprint workloads. For instance, we frequently misjudged API integration times, which led to feature delays. Delphi would have helped us gather diverse input, reduce bias, and align on more realistic estimates. The challenge, however, would be coordinating schedules and ensuring each team member understands the scope well enough to contribute meaningfully.

Finally, the risk management strategies from Chapter 4 gave me new insight into handling project uncertainty. In a personal side project, I recognized that relying on an external API with no uptime guarantee posed a technical risk. Applying the mitigation strategy, I added local caching and fallback responses. While this increased development time slightly, it made the system more resilient and reliable, highlighting the tradeoff between upfront effort and long-term stability.

Peer Interaction and Collaboration:

During a recent discussion with my classmate Visakan, we compared different effort estimation techniques. He shared his experience using the Delphi method during a previous internship and explained how his team reached more accurate estimates through rounds of anonymous input and consensus. Her insights helped me better understand the value of group-based estimation over individual guesswork, especially in reducing bias.

Also, we worked on a practice exercise for function point analysis, where he pointed out an error. His clarification helped me grasp the subtle distinctions between FPA components. These interactions not only improved my technical understanding but also reinforced the importance of peer collaboration in refining assumptions and strengthening accuracy in project planning.

Challenges Faced:

One of the main challenges I faced was understanding how to apply Function Point Analysis (FPA) for effort estimation. Initially, the terminology like "counting boundaries," "unadjusted function point (UFP)," and "value adjustment factor (VAF)" felt overwhelming and abstract. I struggled to see how this method could be practically applied to real software components.

To tackle this, I created a small example based on a login module from a previous project and manually walked through the FPA steps: identifying inputs (like username/password fields), outputs (error messages), and interfaces (like the authentication API). Then, I applied the UFP and VAF formulas from the slides. This hands-on exercise, paired with peer discussion and revisiting lecture examples, helped clarify how effort is calculated systematically rather than intuitively.

Reflecting on this experience, I realized that breaking down complex frameworks into relatable use cases made the learning more meaningful and gave me a tool I can now apply in estimating real projects.

Personal development activities:

To deepen my grasp of the concepts covered, I supplemented the lecture material with independent exploration. I watched a series of video tutorials on Function Point Analysis, which provided real-world walkthroughs of calculating UFP and VAF values. In parallel, I've started reading "Software Project Management" by Ashfaqe Ahmed to better understand how agile practices contrast with traditional models like Waterfall, which we discussed in class. These activities have reinforced my classroom learning and helped me connect theory to practice more effectively.

Goals for the Next Week:

Looking ahead, I aim to:

- Deepen my understanding of risk management strategies by exploring their practical application in real-world scenarios.
- Enhance my effort estimation skills, particularly in applying function point analysis and analogy-based techniques with greater accuracy.
- Investigating the pros and cons of iterative and waterfall models to enhance my perspective.
- Gain hands-on experience with project management tools like Jira to simulate task tracking, sprint planning, and workflow management in a practical context.