Learning Journal - Final

Student Name: Dakshina Ravishankar
Course: SOEN 6841- Software Project Management
Journal URL: https://github.com/dashi1601/SOEN6841LJ_40267664/

Final Reflections:
Overall Course Impact:

Software project management has four stages: initiation, planning, monitoring and control and closure. Each phase depends on the pervious phase and they continuously evolve as project evolves. Initiation involves setting up an initial estimated baseline for the project. Initial estimation of cost, schedule and effort is calculated. For software product development, market analysis is done to get an understanding of why and how a particular software needs to be developed to earn profits. At this stage, artefacts produced are used for getting internal customer's approval. On the other hand, for projects that will be outsourced, the initial estimations, especially, cost and effort is used to get a bid/approval. At this stage, requirements may or may not be clear or detailed.

The next phase is planning which begins when we have detailed requirements. It is more like a very low-level version of initiation phase but also involves planning various other aspects of software project. It involves supplier management, communication management, cost estimation, schedule estimation, effort estimation, resource management, tool management, issue management, etc. COCOMO (Basic, Intermediate, II) and function point analysis are common mathematical models used for effort estimation. More the quality and complexity, more is the effort required. Critical path method is used to find the duration of the project by finding the longest path of tasks that have to be completed for the project to finish. Work breakdown structure involves breaking down the work to pseudo tasks (Designing, Implementation, Testing) which is further broken down into individual tasks with set start and end time. End of pseudo task is a milestone. There are two types of planning, top-down and bottom-up planning. Bottom-up is for software application projects where requirements are clear but duration and start and end date is not determined or not very important. On the other hand, top-down is for software product development projects where time is very important as it can determine whether the software can compete in the market. Planning phase is very crucial because any underestimation or overestimation at this phase can severely impact the next phase, monitoring and control.

Monitoring entails ensuring whether the project is going as planned. Frequent, efficient and accurate monitoring is important and is achieved by measuring software processes, products, cost, schedule, work done, quality etc. Deviations from plan are determined by Earned Value Analysis that measures how much work should be actually done for the budget consumed. This is used to determine whether the project is actually progressing in the right direction. Control is the process of removing the deviations.

Risk management is the process of identifying possible events that impact cost, schedule and quality that can occur during the project execution. It involves risk identification, risk assessment and risk prioritisation. Assessment entails assigning impact and likelihood to each identified to further assign priority. Each risk must have a mitigation plan to ensure the likelihood of risk is reduced. Contingency plan involves steps to be executed when the risk occurs.

Project closure involves steps that are executed when the project is coming to an end. It includes assessing whether everything in project went according to plan, archiving project data, lessons learnt and releasing unused resources. Each artifact produced is assessed and evaluated to improve future software projects. Lessons learnt facilitate better ways of doing a task, areas to improve on, figure out what went wrong, better negotiation with customers and what techniques worked better than others. It also helps assess risks that were handled better than others and what contingency or mitigation plan worked the best for a risk.

All aspects of software project management phases are affected by software lifecycle model (Waterfall, iterative). Software lifecycle refers to the sequence of process executed while developing a software. Software lifecycle models include waterfall, iterative, incremental, etc. Each model is based on its unique principal thus, requiring careful analysis before choosing a software lifecycle model for a software project. Software lifecycle metrics are measurement strategies used to measure software process and product attributes. Essentially, they are used to measure the work products. It also involves finding relationship between different work products, degree of association and impact, etc.

Requirement engineering is the process refining, verifying and validating requirements after initial requirements are gathered from the customer. There are two types of software requirements, functional and non functional (quality attributes). Requirements are finetuned overtime by following requirement development steps like interacting with customers and stakeholders frequently.

Application in Professional Life:

The knowledge gained can be applied to manage software projects. For example, in leading a software development project, I can use the project charter to align stakeholders and apply Function Point Analysis for accurate size and cost estimates. However, developing a comprehensive charter demands a thorough understanding of the project which may be unclear at the beginning. Understanding project closure will facilitate effective release of unused resources and documentation of lessons learned. Lessons learned is important for continuous learning and enables us to take better decisions in future projects to improve as a software project manager. In a software project, resource planning and budgeting involve purchasing software licenses and allocating developers to different modules. Monitoring can track features implemented or LOC. Lessons learnt can be assessing how many person-months a feature took as opposed to the initial estimate.

By applying the skills and knowledge from this course, I can improve my effectiveness in project closure process and requirement development. Additionally, it is important to have an understanding as to what type of lifecycle model will be suitable for a software project. These practices will not only streamline project workflows but also create a more resilient and responsive team environment.

Peer Collaboration Insights:

I learned how theory applies to real-life projects, emphasizing that careful consideration of various factors is crucial, as poor judgments can lead to costly modifications later. Each project is unique, requiring thorough analysis of its complexity before making any decision. It involves many rounds of meetings and consensus and thorough analysis before finalising any decision. Discussing how project management is applied in real time projects through past software project managers in class helped me understand the nuances and high expertise required to become a successful software project manager. These are few of the insights are gained throughout the course. Complexity and quality of a software product determines the effort required. Past similar projects are a very good baseline for future projects. An efficient and secure configuration and version control system is important to ensure concurrent work without mishaps can take place.

Personal Growth:

I gained a clearer understanding of the nuances of project management processes, recognizing that it is not straightforward and that project managers play a critical role in a project's success. They work on top of software developers and ensure they can deliver the software project to the customer within budget, quality and schedule constraints.