**Student Name:** Andres Ardila – 40274138

**Journal URL:** https://github.com/AndresArdila544/SOEN-6841-LJ

**Dates Range of activities:** 26 October -9 November 2024

**Date of the journal: 9 November 2024**

**Key Concepts Learned**

**Project** closure marks the formal completion of a project, signifying the transition from execution to operational use or archiving. This phase involves several critical activities, including **finalizing deliverables** to ensure they meet project goals and quality standards**, archiving project** data such as source code and documentation for future reference, and **evaluating performance** metrics like timelines, budgets, and resource utilization.An essential part of closure is **documenting lessons learned insights** into what worked well, challenges encountered, and areas for improvement. These lessons serve as valuable resources for enhancing processes, reducing risks, and improving the success rates of future projects.

**Software engineering** is a systematic discipline focused on applying engineering principles to the development, maintenance, and operation of software. It involves a structured **Software Development Life Cycle**, which includes phases like *requirement analysis, design, implementation, testing, deployment,* and *maintenance*. Each phase contributes specific **work products**, such as *requirements documents, design blueprints, source code,* and *user manuals*, which collectively shape the software product.

**Metrics** such as *productivity*, *defect density*, and *adherence to timelines* help measure development efficiency and product quality. **Quality assurance** is embedded in every phase through practices like *quality gates*, *testing*, and *reviews* to ensure that the software meets its specifications and standards. Various models, such as the traditional **Waterfall Model** and more flexible iterative models like **SCRUM** and **Extreme Programming**, provide frameworks for managing the development process.

**Customer requirements** are the foundation of software projects, defining **functional requirements**, what the software must do; and **non-functional requirements**, how it should perform. Gathering these requirements involves *interacting with stakeholders* through interviews, workshops, surveys, and observations, and *refining them* through techniques like prototyping. Effective management of requirements is essential for project success. This includes *documenting them systematically, ensuring traceability across the project lifecycle,* and *handling changes efficiently with minimal disruption*.

**Configuration management systems** play a vital role in tracking requirement changes, maintaining consistency, and ensuring that all team members work with up-to-date information. **Quality assurance** during this phase involves *validating requirements* for clarity, completeness, and feasibility, often through iterative review cycles and robust change control processes.

**Application in Real Projects:**

These concepts can significantly enhance the efficiency and quality of real-world projects when applied effectively. Project closure ensures that the team reflects on successes and shortcomings, offering insights to refine future processes. Archiving source code and performance metrics allows organizations to maintain a repository of reusable assets and benchmark data. However, implementing closure activities in a fast-paced environment may be challenging due to the pressure to move to new projects quickly, potentially leading to skipped steps or incomplete documentation.

Software engineering ensures that projects are built systematically, reducing the likelihood of errors and improving maintainability. By adhering to the software development life cycle phases and utilizing models like SCRUM, teams can adapt their processes based on project needs, whether they require rigid structures or iterative flexibility. Metrics enable organizations to monitor progress and quality, allowing for timely interventions. However, a key challenge in real-world projects is balancing the rigor of quality assurance with tight deadlines. Testing and quality gates, although essential, may require additional time and resources that could strain project budgets or schedules.

Customer requirements management is essential in ensuring software aligns with user needs, but its implementation poses challenges. Effective requirement gathering mitigate the risk of misaligned expectations and costly rework. Configuration management systems ensure that changes are tracked and integrated seamlessly, providing stability in dynamic environments. These practices ensure that projects deliver high-value outcomes, customer satisfaction, and enhance the adaptability of the development process. However, managing frequent changes in requirements can be resource-intensive and may lead to scope creep if not controlled effectively.

**Peer Interactions**

This past 2 weeks we had the midterm and a lecture. During the lecture I discussed with my group about the midterm and our expectations. We also discussed the next delivery for the project and started working on it, applying the concepts learned throughout the past lectures.

**Challenges Faced:**

The midterm was challenging. MCQ were tricky and I hope I do well. Studying took me longer than expected rereading all the chapters.

**Personal development activities:**

I re read all the chapters preparing for the midterm. This helped me develop different study methodologies while contrasting the book with the slides and different sources and extracting key notes for future studying.

**Goals for the Next Week:**

Next week I’ll start my preparation for the midterm. We also have a quiz and I hope I’ll do well on it.