Learning Journal – Unit 2: Software Requirements and Architecture

This week, I focused on understanding the core concepts of software requirements and architecture. My activities included reading the Learning Guide and assigned readings, completing the discussion post, responding to classmates, submitting the programming assignment, and taking the self-quiz. I also explored the distinctions between functional and non-functional requirements and how to define and test them effectively.

Through the discussion activity, I engaged in a critical reflection on the roles of users and developers during the requirements elicitation phase. I learned that users provide valuable domain expertise but often struggle to articulate their needs in technical terms. Developers, on the other hand, can technically define solutions but may lack an understanding of the business context. This highlighted the need for strong collaboration and clear communication between both parties to ensure well-defined requirements (Kotonya & Sommerville, 1998).

In my programming assignment, I analyzed five non-functional requirements and evaluated whether they were testable or needed refinement. For example, I realized that vague terms like "user-friendly" and "minimal training" should be rephrased into measurable criteria. Usability, for instance, can be tested by defining specific success metrics such as task completion rates or training duration (Pressman & Maxim, 2014). This exercise emphasized the importance of writing precise, verifiable requirements that support effective software validation.

These concepts are directly applicable in real-world software development. Being able to write clear and testable requirements can prevent costly misunderstandings during later development stages. I now understand how to bridge both the technical and human aspects of software design—whether by refining ambiguous requirements or utilizing tools to validate them.

One important thing I'm thinking about is how critical communication and empathy are in the requirements engineering process. It's not only a technical task but also a collaborative effort. Building mutual understanding between users and developers plays a major role in creating successful systems. I'd like to further explore tools and techniques that support this process, such as prototypes, user stories, and interactive design sessions.

Wordcount: 334

References

Kotonya, G., & Sommerville, I. (1998). *Requirements engineering: A good practice guide*. Wiley. https://www.wiley.com/en-us/Requirements+Engineering%3A+A+Good+Practice+Guide-p-9780471974444

Pressman, R. S., & Maxim, B. R. (2014). *Software engineering: A practitioner's approach* (8th ed.). McGraw-Hill Education. https://www.amazon.com/Software-Engineering-Practitioners-Roger-Pressman/dp/0078022126