OOSE Lab Week 8

Contents

[1 Upload Reflective Journal for Part A 1](#_Toc86745343)

Week 8

# Upload Reflective Journal for Part A

Now that you have completed Part A it is a good time to make sure that your reflective journal is up to date and to re-evaluate your approach to the reflective journal.

Are you happy that your reflective journal is successfully capturing your experience of the Project?

Do you feel that adjustments might be beneficial or is your current approach satisfactory?

The exercise is to upload your journal to date for review. It won’t be marked but once uploaded feedback will be provided.

**Week 1 - part 1: Project Initiation, UML, Use Case and Class Modelling**

* Learning Outcomes:

1. Understanding of UML, Use Case and Class Diagrams: We started the project focusing on constructing a use case diagram for our "Newsagent" domain. This involved identifying actors, defining use cases, and establishing relationships. As a team, we used “Lucidchart“ to do it collaboratively.
2. Application of Agile and Scrum: As we began the project, we picked scrum as our development lifecycle framework. Daily stand-ups and weekly sprints were introduced to ensure adequate team collaboration.

* Reflection:

This task improved my understanding of UML and the practical application of use case diagrams. It highlighted the importance of clear communication in system design. Challenges emerged in defining complex relationships, where I needed to research UML and lookup UML’s spec to refresh my memory on how to draw certain relationships.

* Project Goals and Objectives:

Goal: Develop a clear representation of the system's functionalities.

Objective: Create a use case diagram for the "Newsagent" domain.

Additional Learning: Introduced to Agile methodologies and principles, setting the tone for collaborative project management.

* Outcomes:

Successful creation of a use case diagram with actors "Newsagent" and "Customer."

Identified functionalities and their relationships for both actors.

* What I Learned:

1. Gained proficiency in creating a use case diagram using UML standards.
2. Acquired insights into Agile and Scrum principles and their application in project management.

* Tools and Techniques Utilized:

We extensively used LucidChart to create the use case diagram, and Microsoft Teams for communication.

* What Could Have Been Done Differently:

More frequent check-ins on the diagram's progress could have mitigated challenges encountered during the drawing phase.

**Week 1 - Part 2: Detailed Use Case Analysis and Glossary Refinement**

* Learning Outcomes:

1. Detailed Use Case description: After successfully completing the use case diagram, the focus was on describing in detail two specific use cases within our "Newsagent" domain, one per team-member. We selected "Accessing the App" and "Customer Checkout", incorporating alternate and exceptional flows.
2. Glossary: Collaboratively, we worked on creating a project-specific glossary. The aim was to establish a shared vocabulary, ensuring clarity and effective communication among team members.

* Reflection:

This phase deepened my understanding of constructing detailed use cases and emphasized the importance of foreseeing potential deviations for a robust system design. I got a deeper understanding on UML use cases, alternate flows and exceptional flows.

* Project Goals and Objectives:

Goal: Develop a clear representation of the system's functionalities.

Objectives: Describe in detail two use cases and write a project glossary.

* Challenges Encountered:

1. "Accessing the App" Use Case:

Detailing the alternate flow for user login or account creation required anticipating diverse user scenarios.

1. "Customer Checkout" Use Case:

Specifying conditions and actions for failed payment attempts posed a challenge. Defining the exceptional flow involved outlining the logic for redirecting users back to the "Checkout" page and, if unsuccessful after three attempts, blocking the transaction.

What I Learned:

1. The process of detailing use cases brought to light the necessity of considering alternate and exceptional scenarios. This added layer of complexity highlighted the critical role of user experience and error handling in system design.
2. Contributing to glossary emphasized the significance of a shared vocabulary within the team. Clarity in terminology is essential to prevent misunderstandings and facilitate effective communication.

* What Could Have Been Done Differently:

Exploring additional use cases to gain a comprehensive understanding of the system's behaviour.

**Week 2 - part 1: UML, Class Diagram, Communication Diagram, and System Operation Contracts**

* Learning Outcomes:

1. Communication Diagrams: Gain proficiency in creating communication diagrams to illustrate the interactions between system elements.
2. System Operation Contracts: Develop detailed contracts for system operations.

* Reflection:

This week's tasks deepened my understanding of communication diagrams and system operation contracts. Creating a communication diagram improved my ability to visualise the flow of interactions. Developing contracts required a meticulous approach to detail, considering various preconditions and postconditions.

* Project Goals and Objectives:

Goal: Develop a detailed conceptual class diagram and system operation contracts.

Objectives:

1. Construct Conceptual Class Diagram: Create a visual representation of the system's architecture using UML, emphasizing attributes, relationships, and other key elements.
2. System Operation Contracts: Develop contracts for two essential system operations: "addToShoppingCart" and "processPayment."

* Outcomes:

1. Successful creation of a detailed class diagram illustrating relationships between key classes in the "Newsagent" system.
2. Development of system operation contracts providing clear specifications for "addToShoppingCart" and "processPayment."

* What I Learned:

1. Communication Diagrams: Gained proficiency in creating communication diagrams to visualize system interactions.
2. System Operation Contracts: Acquired skills in developing detailed contracts, considering

preconditions and postconditions.

* What Could Have Been Done Differently:

A more thorough exploration of advanced UML elements could have been beneficial for a more comprehensive class diagram.

**Week 3 - risk, quality and communication, software development methodology.**

• Learning Outcomes:

1. Risk, Quality and Communication management: Gain an understanding on how to manage risk, quality, and communication, and understand how it can differently impact a project.
2. Software development methodologies: Gain proficiency on different software development lifecycle methodologies, and learn how to differentiate and select the most appropriate one.

• Reflection:

This phase gave me a better understanding of risk, quality and communication management.

How techniques like effective communication, a deep and thoughtful knowledge of the project in hands are mandatory to fully understand what the outcome will be and how mismanagement of those pillars may affect the project.

Not only that, but these phase also contributed to my understanding of software development lifecycle methodologies, how many distinct options exist, and how they suit.

• Project Goals and Objectives:

Goal: Discuss and justify risk, quality and communication management, and the software development software lifecycle methodology used in the project.

Objectives:

1. Risk, Quality and Communication: Discuss and justify how risk, quality and

communication are managed.

1. Software Development lifecycle methodologies: Describe and justify the methodology used.

• Challenges Encountered:

Identifying specific risks early in the project was challenging, demanding a flexible risk management strategy. Balancing the growth of diagrams and sustaining clear communication in a small team required continuous reassessment, leading to adaptable quality and communication management approaches. Regular reflections and a commitment to adaptability are crucial for navigating the dynamic nature of the project effectively.

• What I Learned:

1. Risk, Quality and Communication
   1. Adaptive Risk Management: Recognized the importance of an adaptable risk management approach, understanding that specific risks may not be immediately evident in the early stages of a project.
   2. Dynamic Quality Management: Learned to balance the growth and complexity of project diagrams through a dynamic quality management approach, emphasizing regular reassessment and adjustments.
   3. Sustaining Communication in Small Teams: Acknowledged the challenge of sustaining clear communication in small teams and discovered the value of regular meetings and diverse communication tools for ongoing collaboration.
2. Software Development lifecycle methodologies
   1. Continuous Adaptability: Embraced the need for continuous reflection and adaptability, understanding that the iterative and evolving nature of projects requires a commitment to refining approaches for effective project management.

• What Could Have Been Done Differently:

1. Early Risk Identification:

A more proactive approach to early risk identification should be implemented in future projects.

1. Communication Strategies:

Communication strategies could be improved. Establish clearer communication protocols and set expectations for ongoing interaction to address the challenges of sustained communication in a small team.

1. Explicit Roles in Agile/Scrum Framework:

Even in a small team, we should define explicit roles if using Agile/Scrum-inspired methodologies, as was the case for this project. This could involve rotating responsibilities for key roles like Scrum Master or Product Owner to ensure a more structured approach to project management.