Final Learning Journal and Final Reflections

Student Name: Himangshu Shekhar Baruah, 40229774
Course: Software Project Management
Journal URL: https://github.com/raemonx/SOEN6841/tree/main

Weeks: 10 March – 4 April (Chapter 9, 10 , 11, 12 ,13, 14)
Date: March 14 , March 21 , March 28

Key Concepts Learned/Analysed:

- **Software lifecycle -** series of processes which culminates to build software products.
- **Examples** - waterfall model, SCRUM, eXtreme Programming etc.
- **Waterfall Model** - Waterfall model is the classic model where each development process follows the previous process and is going back is not allowed.
- **Iterative models** - model where any part of the software can be rebuilt using rework even going backwards.
- **Requirement gathering -** process of interacting with end users and noting down their requirements.
- **Types –** Functional and Non-Functional Requirements
- **Software Design Process**: It involves creating and refining the software architecture based on user requirements.
- **Design Approaches**:
  - **Top-Down Approach**: The overall product design is created first, followed by the design of individual parts.
  - **Bottom-Up Approach**: Individual parts are designed first, and then combined to form the complete product design.
- **Version Management**: It is crucial to maintain different versions of software design to handle change requests effectively, ensuring the correct design version is used for construction.
- **Robustness and Flexibility**: Software design should be robust, accommodating change requests without issues. Incremental builds should ensure compatibility with the original design.
- **Design Techniques**: Various techniques can be utilized, such as prototyping, structural models, object-oriented design, and entity-relationship models, to create suitable designs.
- **Refactoring**: This is a technique used in iterative development models to revise the design, incorporating new features without compromising structure or functionality. Refactoring is particularly useful when the design becomes too complex after many iterations.
- Software design - top to bottom method, bottom up method

- **Coding Standards**: Defining coding conventions to prevent defects and ensure consistent quality.
- **Programming Language Selection**: Choosing a language that best meets the project's technical needs.
- **Code Reviews**: Regularly reviewing code to maintain quality and catch defects early.
- **Coding Methods**: Utilizing evolved coding practices like structured and object-oriented programming for better code quality.
- **Configuration Management**: Implementing thoughtful branching strategies for efficient version control.
- **Testing**: Conducting exhaustive unit and integration testing for robust and reliable code.
- **Test Automation**: Actively automating repetitive test case executions to minimize testing efforts in iterative software projects.
- **Test Case Writing**: Automating unit test case writing, while manually crafting test cases for more complex testing types.
- **Manual Testing**: Conducting the first run of test cases manually to set a reference for future test automation.
- **Defect Tracking**: Continuously monitoring and tracking software defects to measure testing effectiveness and unresolved issues.
- **Software Maintenance**: Maintaining software in production through periodic updates and patches, ensuring continuous operation and support.
- **Maintenance Types**: Implementing preventive, corrective, and adaptive maintenance to address different needs and improve software longevity.
- **Maintenance Techniques**: Utilizing re-engineering, reverse engineering, and forward engineering to maintain and enhance software systems.

New Terms - Software lifecycle, Requirement gathering, Software requirements validation cycle, Version Management, Design Techniques, Refactoring, Code Reviews, Configuration Management, Test Automation, Test Case Writing, Manual Testing, Defect Tracking

Methodologies - Waterfall Model, Agile Model, SCRUM, eXtreme Programming, Iterative models, Top-Down Approach, Bottom-Up Approach, Prototyping, Structural models

Application in Real Projects:

Let us consider a real-life scenario of a tech startup aiming to launch a mobile application that helps users manage their personal finances. We will perform the following in this scenario.

Budget365 App Development Project

Background:
The aim is to make financial management accessible and straightforward for the average user.

The SDLC Phases for Budget365 App are:

- **Requirement Phase:**
    1. Conduct market research to identify key features that users want in a personal finance app.
    2. Organize focus groups to gather feedback on proposed functionalities.
    3. Finalize a list of requirements based on user feedback and competitive analysis.
- **Design Phase:**
    1. Create wireframes and prototypes of the Budget365 app for various platforms.
    2. User experience (UX) designers work closely with financial experts to ensure the interface is intuitive.
- **Construction Phase:**
    1. Software engineers begin coding based on finalized design documents.
    2. Backend developers set up servers and databases for secure financial data storage.
- **Testing Phase:**
    1. Conduct a series of unit, integration, and system tests to ensure the app's reliability.
    2. Perform user acceptance testing (UAT) with beta testers to collect user feedback.
- **Release Phase:**
    1. Prepare the app store submissions with marketing materials.
    2. Launch the app on multiple platforms, including iOS and Android.
- **Maintenance Phase:**
    1. Monitor user feedback for any issues and rapidly deploy updates.
    2. Regularly update the app with new features based on user demand and financial trends.

Challenges:

- Keeping the app secure to protect sensitive financial data.
- Integrating with various financial institutions for real-time data syncing.

- Ensuring compliance with financial regulations and data protection laws.

Benefits:

- Offers a comprehensive tool for users to manage their finances efficiently.
- Gains a competitive edge by using AI to provide personalized financial advice.
- Builds a loyal user base by ensuring user data privacy and app reliability.

Peer Interactions:

- We have analyzed Agile and Waterfall methodologies, noting Agile's flexibility for dynamic projects and Waterfall's suitability for fixed requirements.
- We also discussed how bottom-up design is beneficial for prototyping and innovative solutions.
- During weekly interactions, we focused on how to enhance code quality and standards.
- We discussed strategies on test automation to boost efficiency.

Challenges Faced:

- Deciding which software development model best fits project requirements is complex, especially when considering hybrid approaches.
- Ensuring requirements are fully understood and validated between the development team and stakeholders poses communication challenges.
- Selecting the right design technique and approach is critical, and making the wrong choice can impact the project's success.
- Maintaining consistent coding standards across diverse teams is challenging and crucial for preventing defects.
- Balancing between manual testing and automation is tricky, particularly in deciding which tests to automate and maintaining those tests.
- Applying effective maintenance strategies that keep the software functional and relevant over time can involve considerable effort and resources.

Personal development activities:

- I had enrolled in coursera courses to learn more and stay updated with industry trends.
- I had also participated in industry conferences and seminars.
- **I** regularly looked for documents which explained professional literature to expand knowledge.
- I also learnt effective project management techniques through workshops and YouTube videos.

Final Reflections

Overall Course Impact

The course had a significant impact on my understanding of software project management by exploring various management theories and practices in a structured manner. It covered essential concepts like project initiation, planning, execution, monitoring, and control in depth. I also studied methodologies such as iterative development, agile and waterfall models, and the need to tailor project management strategies to specific project requirements. Finally, the examination of risk management and configuration management helped me shed light on the complexities of software projects and the importance of effectively managing changes and risks.

Application in Professional Life:

The knowledge gained from this course will help me a lot in my professional life, which will help me in managing software projects. For example:

Iterative Development: Implementing this approach will help continuous integration and feedback loops that improve product quality and customer satisfaction.
Risk Management: Deploying comprehensive risk assessment and mitigation strategies which can help prevent project delays or budget overruns.
Configuration Management: By following the best practices in version control and change management, it will ensure the stability and reliability of software products.

These skills learned in this course will help me in managing real world projects like developing new software applications where effectively managing scope, time, cost, and quality are essential for achieving success.

Insights from Working with Peers

Interacting with my classmates has greatly enhanced my learning, offering various viewpoints on project management challenges and solutions. Conversations and discussions about estimating techniques, managing risks, and implementing scheduling methods have been really valuable. Collaborative activities, such as creating risk matrices and analyzing project baselines, have helped me in understanding and improved my teamwork skills.

Personal Development

During the course, I have gained a solid grasp of project management concepts. Also applying proactive approach in seeking out extra resources, actively participating in discussions and putting theoretical knowledge into practice, helped me in enhancing my analytical and problem-solving skills. Finally, my ability to reflect on the learning journals helped me increase aptitude for self-directed learning and continuing progress.