Foundations of Quality Software - Week 1 Report

CSC 640 – Software Quality

Nadine Karabaranga

Dr. Samuel Cho

1. Introduction

The first week of the course introduced the foundation of **quality software engineering**.
Software quality was defined not only by functionality but by **maintainability, reliability, and teamwork**.
The lectures emphasized how disciplined processes and communication are essential for producing professional, long-lasting software.

2. Understanding Quality Software

Quality software must be:

- **Reliable** – performs as expected under all conditions.
- **Maintainable** – simple to update and extend.
- **Efficient** – optimizes time and resources.
- **Readable** – understandable for both current and future developers.

> Building for maintainability is building for the future.

3. Managing Complexity

Professional engineers handle complexity through:

- **Modular architecture** and clean code separation.
- **Iterative testing** and verification at each stage.
- Adherence to the **KISS** and **DRY** principles.
- **Clear documentation** to prevent confusion.

Effective complexity management improves performance and reduces error propagation.

4. The “No Surprises” Team Principle

A significant lesson from Week 1 was understanding that **software quality is a team effort**.
The *No Surprises* principle means:

- Communicating openly about progress, challenges, and risks.
- Avoiding unexpected code changes or hidden issues.
- Ensuring transparency through peer reviews and shared updates.

This practice builds **trust**, **predictability**, and **collaboration**, which are the foundation of successful software projects.

5. The ASE Framework

The ASE architecture defines the **4-D Cycle** — *Define*, *Design*, *Develop*, *Deploy* — providing a structured roadmap:

1. **Define** – clarify objectives and success criteria.
2. **Design** – plan components and data flow.
3. **Develop** – implement, test, and refine.
4. **Deploy** – release and evaluate the solution.

This process reinforces quality through consistent feedback and documentation.

6. Tools and Processes

Category	Examples	Purpose
Version Control	GitHub	Track revisions and support team transparency
IDE	VS Code	Provide uniform development environment
Testing	Postman, PHPUnit	Verify correctness and functionality
Documentation	Marp, Markdown	Ensure professional communication

Together, these tools strengthen traceability, accountability, and reproducibility.

7. Reflection

Week 1 established that building **quality software** relies equally on **technical skills** and **team communication**.
Effective collaboration under the *No Surprises* principle minimizes friction and enhances code integrity.
Going forward, I plan to apply these values to my REST API project to create software that is transparent, efficient, and maintainable.

8. Conclusion

Software quality begins with **clear structure**, **disciplined communication**, and **collaborative teamwork**.
The *No Surprises* culture promotes predictability and trust — two elements that transform individual effort into sustainable team success.
These foundations set the tone for continuous improvement in all future course projects.