Portfolio Project: Lessons Learned Reflection

Alexander Ricciardi

Colorado State University Global

CSC505: Principles of Programming

Professor: Dr. Joseph Issa

January 11, 2025

Portfolio Project: Lessons Learned Reflection

Through my Bachelor of Science in Computer Science, obtained last August 2025, I became familiar with Software Engineering (SE). The CSC505: Principles of Software Development course, for me, was not only an excellent refresher but also an opportunity to learn more about SE. The course allowed me to dig deeper into the software architecture and project management aspects of SE. Although I joined the course with a technical foundation, the course curriculum allowed me to explore further Computer Science topics and dig deeper into the complexity of software development. This reflection describes the lessons that I have learned, which range from algorithmic refinement to process modeling, and how I applied these concepts to a real-world project, such as my personal project Omega.py. Furthermore, I will discuss how learning further about software development methodologies has changed my personal approach to work, specifically in managing my goal of perfectionism through iterative development.

Significant Modules

In Module 1, I learned that software development needs to be a managed process that follows software engineering practices rather than a process that just implements a sequence of coding tasks. The module reinforced my perspective on what software is, as a product of high quality rather than a simple set of executable code. I applied this concept in my Module 1 Critical Thinking Assignment (CTA), where I implemented a Python script to read and analyze a CSV file. I learned through these exercises that software robustness is heavily dependent on how gracefully an application handles external data inputs, like the quality of a CSV file data parsing process. This lesson on data integrity is now a cornerstone for my work with AI agents, where data sanitation can make a substantial difference between an AI agent generating a useful response and an AI agent generating a hallucinated response.

In Modules 2 and 3, I had to evaluate the traditional software development Waterfall Model against the Agile and Scrum processes. Module 3 CTA asked to identify the limitations of the Waterfall model, specifically its rigidity, and to compare it to modern and adaptive design models. The assignment led me to develop my own Ricciardi Adaptive Iteration Model. The model is an incremental development, implementation, and release approach to software development based on the Agile Scrum methodology, which is based on iterative and incremental deliveries (Schwaber & Sutherland, 2020. It focuses on releasing software incrementally, rather than delivering a finished product in one deployment. The process of designing this model taught me that modern software is never truly done; it is an ever-changing and ever-improving product that is delivered in releases. This helps me understand better the importance of the Software Development Life Cycle (SDLC). This aligns with the course textbook, Software engineering: A practitioner's approach (Pressman & Maxim, 2020), describing why incremental/agile models are better suited for projects with evolving requirements due to stakeholders evolving throughout the lifecycle of the project.

Module 6 introduced me to the concept of Stepwise Refinement (or top-down design), which is a strategy used to break down complex problems into manageable parts or modules. I applied this concept directly to the module CTA, where I built a Root Solver to solve the roots of a transcendental equation using methods like the Newton-Raphson method. The assignment was challenging; however, by using the Stepwise Refinement strategy, I completed the assignment successfully. I had to break down the problem from a high-level goal, Find Root, into mid-level logic, Calculate Derivative, Check Tolerance, and low-level implementation, code. This exercise reinforced my understanding of what a robust software architecture requires, and it allowed me to acquire skills that can be applied to complex real-world projects or problems.

Real-World Application and Modules

I used the lessons learned from this course, and I applied them to my personal project, Omega.py, an AI Agent and RAG (Retrieval-Augmented Generation) system utilizing a Neo4j knowledge graph. I am using my Ricciardi Adaptive Iteration Model for developing Omega.py, as integrating the Neo4j graph database and RAG with novel system AI has significant technical uncertainty, and using the Waterfall approach would have likely introduced issues during the planning phase of the project. So, I utilize my model's Incremental Release strategy, focusing on an incremental software release that started with a simple RAG connection to the database release. I am now focusing on an AI agent connecting to the RAG system. Furthermore, the Stepwise Refinement logic practiced in the Root Solver assignment is allowing me to debug the agent's outputs.

Additionally, the lessons in Module 4, the Human Aspects of Software Engineering, and Module 5, Understanding Requirements and Requirements Modeling, had the most impact on my personal life. As someone who has described himself as having OCD tendencies regarding work processes and perfectionism, I often struggle with the desire to over-engineer solutions, adding unnecessary requirements, and prioritizing perfection and ignoring the human aspect of a project. This is often counterproductive, consuming time unnecessarily and creating more human stress and friction.

Furthermore, my own Ricciardi Adaptive Iteration model permitted me to be imperfect. I learned that shipping a Bug Fix Release or a Missing Feature Release is not a sign of failure; it is a planned part of the development lifecycle. This has lowered my anxiety associated with complex projects. I no longer feel the need to solve every problem and implement all features on

the first attempt. I can now accept an incremental approach to developing a project or solving problems, rather than trying to produce at once a very complex and flawless project or solution.

Moreover, Module 7, Component-Level and User Experience Design, and Module 8, Software Quality Concepts, Assurance, Security Engineering, and Testing. I learned to integrate in my projects Component-Level Design, which helps to reduce the gap between high-level architecture/perspective and actual coding. The Software Quality Assurance lesson taught me that security and testing are not afterthoughts but integral parts of the engineering lifecycle. Now, I apply these concepts when developing my own software projects.

Conclusion

CSC505 has helped me turn what I learned in Computer Science into real-life problem solutions and software engineering projects. Additionally, the course gave me further understanding of when the use of the rigidity of the Waterfall approach is appropriate and allowed me to create my own more flexible Ricciardi Adaptive Iteration Model, which is well-suited for my real-life project. This allowed me to acquire the tools to tackle more complex issues and to project effectively. Additionally, the course has transformed my approach to my current Omega.py project from a personal hobby project into a professional engineering portfolio project, and it has also provided me with tools to manage the pressures created by my own often misplaced perfectionism and OCD tendencies. Finally, I leave this course equipped with a better understanding of the Software Development Life Cycle and ready to approach future software projects not just as a programmer but as a Software Engineer.

References

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill. ISBN: 9781260423297

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The definitive guide to Scrum: The rules of the game* [PDF]. Scrum Guides. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

Souppaya, M., Scarfone, K., & Dodson, D. (2022). *Secure software development framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities* [PDF]. National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf