**Module 8 Portfolio Project**

**Lessons Learned Reflection**

Joshua Humphreys

Colorado State University Global

Course Code:CSC505

Dr. Luis Gonzalez

January 28, 2023

**Module 8 Portfolio Project**

In the following pages I will discuss my lessons learned from this course. My key takeaways are related to recursion and procedural abstraction. I will discuss the ways each are applicable to software engineering and problem solving in general. I will relate these concepts to my primary lessons learned from this course. Finally, I will discuss how these concepts will change the way I approach problem solving in the future.

Throughout this course we’ve discussed the idea of recursion in a variety of ways. In the context of software development, recursion is at the core of Agile methods. That is, there are a variety of ways to apply Agile principles. However, they all look a little like developing an understanding of the problem, doing some work, revisiting our understanding of the problem, and repeat.

The concept of procedural abstraction goes hand-in-hand with recursion. Procedural abstraction is the process of breaking a problem into component parts, then breaking component parts into their subcomponent parts and so on. That is, there is recursion built into abstraction in that each subcomponent becomes the component of a new layer that is broken down into further subcomponents.

Both ideas apply to both the project management side and the software development side of software engineering. The software project itself can be abstracted and solved recursively. In other words, a problem can be broken into sub problems that are broken into sub problems, etc. Then, the sub-problems can be solved recursively. Software components can be broken into smaller and smaller design parts and then built recursively. Additionally, the diagrams involved in communicating the design of software follow this same pattern. That is, the diagrams begin at a very high-level view that captures the “essence” of what the software system is supposed to do. From there, the diagrams simply provide more and more details related to the behavior and structure of the system. Eventually, these refinements lead to a level of detail that can be translated to a set of instructions for a computer.

While these ideas are simple, for me, this was a profound revelation. I started this course with some background in using a couple of different computer languages. Mostly, my experience was with the application of a couple languages in data analysis and mathematics. What I didn’t understand was how to connect those concepts to the creation of an application for example. What I understand now is that, for starters, I was working in reverse. That is, I was trying to think about starting by writing simple scripts and then adding complexity. I also can now picture, at least in concept, how I would go about designing a software system. In module 3 we initiated the design process for a shopping cart app. When I started that project, I couldn’t imagine how I would go about building that application. However, at this point, I feel confident that I at least have a road map to follow.

In conclusion, what I learned in this course ultimately comes down to abstraction and recursion. These concepts are applicable not only to software engineering but to problem solving in general. As I approach large problem sets in the future, I will consider how to break the problem into smaller and smaller parts until the parts are manageable. I will then solve each problem while reflecting on how the process of solving each small problem develops my understanding of the big problem.

**References**

Brush, K. & Silverthoren, V.(2022, November). *Agile software development.* TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development

Fakhroutdinov, K. (2024). *The Unified Modeling Language.* uml-diagrams.org https://www.uml-diagrams.org/

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

Sharma, R.K.(2018, July 18). *Procedural abstraction.* Medium. <https://medium.com/@rshkntshrm/power-of-procedural-abstraction-249fd7529473>