Module 8: Portfolio Project

Lessons Learned Reflection

Nolan Byrnes

CSC505 – Principles of Software Development

Colorado State University – Global Campus

Professor Steven Evans

September 9th, 2022

Lessons Learned Reflection

In the course Principals of Software Development (CSC-505) we covered the human aspects of being a software engineer, modelling software solutions, and what is considered to be high-quality software. This paper highlights the lessons that I learned from this course, and how it has impacted my life.

Bridge Global claims that “Empathetically understanding the need of the project helps you build a product that precisely solves a problem” (Bridge Global, 2019, para. 9). I was working on a solution where we had to grab a list of email addresses that my bot would need to look for in its email inbox. Initially, to meet this requirement, I had our stakeholders place this list of email addresses in an Excel spreadsheet. After this feature was implemented, I thought about the end user’s perspective and realized how much of a task it would be for them to manage this email list through Excel. By taking the time to think this through their perspective, I was able to come up with a better solution, where my bot would automatically gather the list of required emails by grabbing all the email addresses through a SharePoint Security group, which is easier for the end users to manage.

Having an eagerness to learn can help a developer have a broader range of tools that they can use to achieve the end goal for a given project, and open opportunities for what they can accomplish. I have had stakeholders come up with automation ideas that involve Atlassian’s Confluence, and since I was not as familiar with it, I have claimed that their ideas were not feasible. Remembering to have an eagerness to learn, I started to learn more about Confluence, which has since opened opportunities for automations to develop.

In this course, I learned how important modeling your software solution prior to development is. Pressman explains that a use case "tells a stylized story about how an end user interacts with the system under a specific set of circumstances" (Pressman, 2019, pg. 114). In Module 5, I created a Use Case diagram to facilitate understanding of how end users would interact with the Pothole Tracking and Repair System (PHTRS). By taking the time to create the Use Case diagram prior to development, I was able to model the behavior of the PHTRS. By referring to my use case diagram during development, I was able to develop a solution that met all the requirements through the end user’s perspective.

In Module 6, I made a UML diagram of the same PHTRS system to plan the architectural design, and by doing so, I realized how much of an improvement that I made to the final solution. The resulting solution was more organized, easier to follow, and if I was approached with more requirements to be implemented, it would be easier to do so based on the Model-View-Controller (MVC) architectural approach that I took.

Prior to this course, when I would implement a software solution, I would jump right into the code without creating any diagrams. When I would do so, I would come up with a solution that would work, but the result would end up being sloppy, and would take more time as I would have to do a lot more refactoring of code as I figure out the solution directly in my code editor. If I had taken the time to think about how the software would behave, and the architectural design of how the solution would be, I would have a cleaner solution that would be easier to follow.

Modelling the solution before development not only helps the developer create more efficient solutions, but the resulting diagrams that are created also serve as a communication tool to stakeholders and other developers who may be joining the project in the future. To stakeholders, a use case diagram helps to show that the developers understand the use cases that need to be implemented and show a logical progression of the actions that the software will perform. Haan claims that “if someone joins a project it is far easier to understand the high-level model of the software application compared to trying to understand the behavior of the application by reading source code” (Haan, 2009, para. 11). A picture is worth a thousand words, and by having diagrams, it helps to communicate to others about how the software will behave, and in a way provides a map of the software being created.

High-quality software is useful and provides value to both the user of the software, as well as to the ones who created it. For a software solution to be useful, it needs meet all the requirements of the stakeholders in a reliable way. A high-quality software solution provides value to the ones who developed it by requiring less effort for maintenance and the reduced amount of bugs that would have to be resolved, allowing them to spend their time developing new solutions.

**Conclusion**

The lessons that I have learned throughout this course will help me to become a more successful developer. Remembering to have empathy helps to connect with stakeholders and understand the problems that they are trying to solve through their perspective.

Continuing to have an eagerness to learn helps to keep up with the rapid pace of technology and opens opportunities for more problems that I can resolve with software. Remembering the importance of modelling software solutions will help me communicate with stakeholders, ensuring that the software solutions aligns with their needs, and helps me come up with a clean and efficient software solution that is easier for myself and other developers to maintain. Overall, the lessons that I have learned in this course not only has helped me become a more successful developer, but by using the things that I have learned through the course, I am more capable of developing high-quality software.

**REFERENCES**

Bridge Global. (2019, August 8). 3 brilliant reasons why software developers should have empathy? Medium. <https://bridge-tweed.medium.com/3-brilliant-reasons-why-software-developers-should-have-empathy-ec50ddb86f7e>

Haan, J. D. (2009, November 11). 15 reasons why you should start using model driven development. The Enterprise Architect. <https://www.theenterprisearchitect.eu/blog/2009/11/25/15-reasons-why-you-should-start-using-model-driven-development/>

Pressman, R. S., & Maxim, B. R. (2019). Software engineering: A practitioner's approach. McGraw-Hill Higher Education.