Principles of Software: Lessons Learned

Michael Stewart

Colorado State University Global

CSC505-1: Principles of Software Development

Professor Jonathan Vannover

Nov 3, 2023

Principles of Software: Lessons Learned

This software class has been a good breath of fresh air in my educational career about learning software development. Most classes in my career have been helpful, but overtechnical and did not spend time on the general mindset behind software. Even though having engineers that are technically adept is important in order to have smart people creating solutions, it's very important to set these smart people up for success. Supporting the team is essential to an engineering operation, and managing it is very hard without the lessons I have learned from this class. I’ve learned great ways to set up engineers in the design phase, implementation phase, and testing phase of the project. These lessons include different methodologies for approaching projects, creating good attainable requirements for software, and creating good tests for software.

Engineering Methodology

One of the most useful things I learned in this course is the different engineering methodologies. This includes things like Waterfall, Agile, Scrum, and other ways that engineers use organized structure to succeed on their projects. Being able to understand the Waterfall methodology has been especially helpful. Since the Waterfall method is very similar to the way I would intuitively plan a project, giving it a name and learning the pros and cons have been helpful for my decision-making ability when it comes to how to perform a project. Knowing the structure of the Waterfall model can be very rigid without much flexibility helps me to know when to choose more flexible models like agile (TutorialsPoint, 2022). This will help me in the real engineering world because it will help the project be more efficient. Many situations occur at my job where the structure of how we are doing the project can impede our progress and result in more costs for the client. This is bad because the client is less likely to work with us in the future and be satisfied with the project. However, if we are able to make more informed decisions, then we can increase the amount of good decisions in our firm.

Requirements Engineering

Another important thing I have learned from this course is the importance of requirements engineering. Being able to identify, elicit, analyze, specify, validate, and manage the expectations of stakeholders within the project is an essential aspect that is handled by good requirements engineering (GeeksForGeeks, n.d.). Things like use cases, user stories, and observations can help lead the engineering team to develop attainable requirements and give the client good estimates of what we are capable of. This skill will be helpful to me in my future careers because it will lead to more clarity between our engineers' expectations and the clients' expectations for the software. This will also help lead our development team by giving them clear objectives to complete.

Software Testing

Finally, software testing and creating good tests is a great skill I have learned from this class. One of the papers I read in this class was about the gamification of software testing (Fraser, 2017). Encouraging engineers to test software by creating fun games associated with it can be very beneficial, but this mainly taught me that no matter the costs, software testing is direly needed in order to deliver a good product. Some engineers will slack on testing because they think they've tested it enough, but being thorough in software testing can literally mean life and death for users in industries like manufacturing, aerospace, etc. Now with this lesson, I am able to put more importance on testing and defend my time spent on it in my engineering job.

Conclusion

In conclusion, this class has been very helpful for my engineering skills and my mindset when developing software. I will use these skills a lot in my career and am very thankful to have had this class. Being able to understand the engineering process better from the design, implementation, and testing phases will help me produce better projects more efficiently. Hopefully that will also come with more generous compensation as well.

References

Fraser, G. (2017). Gamification of software testing. In Proceedings of the 12th International Workshop on Automation of Software Testing (AST '17). IEEE Press, 2–7.

GeeksForGeeks. (n.d.). Requirements engineering process.https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/

Tutorialspoint. (2022). Software engineering overview.

Tutorialspoint. (2022). Waterfall model. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm