Author: Clinton Lohr

Date: 03/10/2023

Course: CS 362 – Software Engineering II

Continuous Integration Workflow Report

Previous Team Projects

As of now, my experience working as part of a programming team has been solely through computer science courses taken at Oregon State University. Though these experiences are limited, I have gathered an immense understanding of not only tools, such as Git, but also development practices such as continuous integration and Agile.

These teams have ranged from working with one other developer and up to five developers all using the same codebase to work on a project. In the course Software Engineering I, I worked as part of a five-person team in which we were building separate web applications that incorporated software built by other team members.

To integrate others software into our own, we had to play the role of both client and developer. We would discuss with the team member developing the software, what our requirements were and how we envisioned the final product working with our own software. On the other end, we played the role of developer in which we took the requirements given to us by team members and designed software that met their needs and could be successfully integrated into their own projects.

This experience was nothing but positive, as my team was motivated and eager to help each individual member succeed. The greatest lesson that I took away from this project was the importance of communication. Not only is it important to be responsive to team members, but to work with them to find solutions. Having a team that is willing to go above and beyond motivated me to meet their work ethic and commitment.

Working with Continuous Integration

I have worked with continuous integration before, so I had an idea of the workflow that is required for this process. I have found that CI is a powerful tool for maintaining standards in a codebase while understanding that there is a learning curve for using this process. Knowing that our team would be working from the same codebase, I expected that at some point we would encounter merge conflicts.

With all of us being new to CI, I was unsure how we would handle a situation like this when it arose. I think the best approach for instances like these is communication with one another, which is ultimately what we did to resolve these issues. Using Discord, we let one another know when we made a pull request and tried to review these requests in first in first out order. This approach proved successful, and we were able to avoid any unnecessary headaches integrating our code.

All group work leading up to this project never required code reviews before submitting into the codebase, so this process was new to me. I found the code reviews very valuable as my team helped me point out areas where my code could be tweaked to be more descriptive and efficient. Once instance

was in the test.py file of ours. My tests were all using the 'assertTrue' function to verify input. My team member helped me realize that it would be more straightforward if I used a 'assertEqual' function instead, so that I could directly compare the output to the expected output. This helped drive home the importance of getting a second or third pair of eyes to view the software and helped maintain a standard for our code. When writing a code review, I tried my best to provide meaningful feedback and alternate ways of going about something without coming off as abrasive.

I did experience one of the disadvantages of code reviews which is the downtime you have waiting for your code to be reviewed. Luckily, I had other parts of the code that I could be working on during that waiting period, but I can see how code reviews could holdup a developer's workflow.

One part of the continuous integration process is daily commits. I would say that for this project, daily commits were something that we could have worked more on. The way went about this project was to assign one of the three functions to each of us to work on. With such a small piece of code, it was easy for us to complete the whole function and send a pull request for the completed part. We did however tweak out code based off code reviews and tried our best to submit the updated version daily. I was however able to grasp the importance of daily commits to avoid merge conflicts and assure that everyone is on the same page as far as structure of our codebase.

As far as testing our software, our team took a white box testing approach rather than test driven development. Because the requirements for input and output were clearly defined, we found it easier to build our code to meet these requirements and then develop tests to confirm that our code was functioning as expected.

Though I've had experience as a team member working on a group programming project, I feel like I take away a new valuable lesson each time I get to collaborate with others on an assignment. Working solo, I find that I get very set in my ways and tend to stick with what I know. Working as a team and receiving code reviews, I was able to open my mind to other design approaches and see different ways of thinking about a problem. One point made in the Code Review lecture for Software Engineering II was to put yourself in the shoes of the reviewer. For this project, I really focused on writing my code so that when it is finally reviewed, even a reviewer that has no knowledge of the project could look at my code, and based on docstrings, descriptive variable names, and well-placed comments, and determine the purpose of my function.

Having worked in a team setting on programming projects, I've come to understand some of the more important qualities and behaviors that make for successful collaboration. In my eyes, the most important thing is communication. There is nothing worse than being stalled from continuing your work due to unresponsive team members. Knowing my own frustrations working in a group provides me with an example of what not to do when working with others. I tried my best to be responsive when my team members had questions and provide them with meaningful input when doing code reviews. This project helped me put into practice all the qualities that I expect from others when working towards a common goal.

As I mentioned before, this project is my first introduction to code reviews. I think the combination of taking what I already knew about giving respectful and constructive criticism, along with the teachings in the Code Review lecture, helped me form feedback that was well thought-out and considerate. I've always had a hard time critiquing others and would tend to lean towards avoiding any conflict. I think

that by practicing code reviews, I was forced to overcome that obstacle and assume the role of mentor in areas that I felt I could be of contribution. There's also something enjoyable about providing your own insight and seeing others understand and place value on your input. That's not something that I had experienced much of throughout my time at Oregon State.

Lessons for the Future

Having had the chance to work with continuous integration throughout this project, I feel that I have a better sense of what to expect in terms of workflow in a professional setting. One thing I have been yearning for throughout my time at Oregon State, is the opportunity to see and experience what a day as a professional software developer is like. I think that with this project I was able to get a taste of what this might be like.

I think using CI helped me to think about my code from a different perspective. Prior to this project, I was very focused on writing code that met my own expectations. Through working with CI, I noticed that I was much more considerate of how my code would be interpreted. This includes readability, designing more efficient approaches, writing code that is modular with use of helper functions. Also, seeing my teammates integrate their code helped me get a feel for their approach and I could tailor my code to fit their expectations.

The code reviews that make up a part of CI had a bigger impact than I had anticipated. Having received valuable input as the reviewee really helped shine a light on the importance of giving valuable feedback as a reviewer. I put more effort into understanding the code my teammates developed and tried to really see the reasoning behind their approach. Analyzing the code not only helped me understand alternative approaches but gave me the opportunity to impart some knowledge on my ideas for an approach.

Though we took a white box testing approach, test driven development would have been a valuable tool to use for this project. Speaking for myself, I had a clear idea of the requirements for this assignment, so I developed my function to meet those requirements and then developed tests to confirm that the function was working as intended. I had gone back a few times to combine conditionals and, in some cases, remove altogether. I think had I used test driven development, I could have built more concise code as I went along instead of modifying my code after completion. Looking back, TDD is the route I would have taken which probably would have saved me time and effort in the end.

Because we each worked on our own function and designed our tests individually, we never ran into issues with conflicting code. With that said, if we each did a part of each of the three functions, it would have been even more important for us to have a well-developed test suite. There could have been instances where one of us changed the code without the other team member knowing. Without a solid test suite, changes made to the code may have altered the design enough that what had worked before may no longer provide valid output. Having the test suite established beforehand, any changes to the code that altered output would have been caught immediately and saved us a lot of panic and headaches.