Joseph Silva Jr.

11/22/2020

CS 310 Final Project Check-In 4-3

SNHU

**Version Control Summary**

**1. Describe the benefits of version control tools in software development projects.**

Version control tools provide many advantages in the development of software projects. The following is some of the advantages of version control tools: collaboration, organization, and backups. First, version control tools allow the collaboration between team members during the creation of the software development project. Collaboration between team members is always beneficial because it makes it easier for static testing to find any defects early in the project. Team members can use many techniques such as peer review, code inspection, and code walkthrough to find these defects / errors to prevent higher cost testing later on in the project. Second, version control tools keep track of any changes throughout the project. Keeping track of any changes, developers will be informed about these new changes and be able keep the project organized instead of wondering who made recent changes to the code. Lastly, version control tools can be used to make backups of previous work prior to any new modifications. Being able to essentially go back in time, developers have a “safety net” in case any new modification to the code causes more errors. These “safety nets” or saved work of code can prevent wasted time in recreating the prior code and they also can protect the budget from undesired expenses within a project.

**2. Detail how version control helped you manage your source code in developing the calculator app**

Version control tools were great when looking throughout my calculator app. In the beginning of the term (week 2), I had a problem with being able to run the calculator project because of Eclipse configuration problem. However, I was able to use version control tools to see when I added new classes such as Power.Java and Divide.Java to the calculator project. I was also able to keep a track record that I added new classes such as Power and Divide to the Calculator.Java class. The changelog was a great tool to show the future me on when and why I changed or added to the coding for these classes.

**II. Development**

**A. Explain the benefits of pushing local repositories to remote shared repositories specific to expanding the functionality of repositories. Be sure to explain how configuration and cloning can be used to expand the functionality of repositories.**

Pushing local repositories to remote shared repositories is beneficial because it allows the local repositories to be shared and stored within the remote shared repositories. Local repositories can only be reviewed by the user because they are stored in the local file, while the remote shared repositories can be seen and accessed by the entire team on a project. These remote repositories show the team the additional changes made to the code, which can prevent duplication of work for completed features. The remote shared repositories are the most up-to-date work, and it keeps the members of a team in constant communication with each other by being able to access to the different changes made to the code.

Configuration and cloning allow the user to clone the remote shared repositories into their local files, which allows the user to work on the up-to-date code as local repositories instead of making changes to the code stored in the remote shared repositories. Being able to work on the code as local repositories, users can update and/or change the code without affecting any other team member’s work, which is beneficial because it can prevent delays and conflicts to other team members.

**B. Describe your experience with branching and merging in your development process, being sure to discuss conflicts, blaming, stashing, and rebasing.**

During the development process, I gained experience with branching and merging. Branching allows a user to create a copy of code in order to conduct in-progress code creation to parallel the main line of code without hampering progress of other developers. Merging allows a user to merge the completed branching code into the main line of code. During my calculator tutorial, I did not run into any conflicts while committing the multiplication, divide, and power code. When it came to blaming, I was the only one user writing the code for the tutorials in my local folder. However, I still was able to find out when and who certain changes were made when I reviewed the history log. I used the stashing technique during week 3 tutorial module in order to save messages and changes to a file in case a higher priority event occurred. With the stashing technique, I was able to continue the stashed code from where I left off. I was able to use rebasing technique to merge branches with a cleaner history log by merging my divide.java without the multiplication.java and power.java in the same branch.

**C. Specify how you utilized the states of a source file, such as committed, modified, and staged, in relation to the file status lifecycle.**

I was able to utilize the states of a source file in relation to the file status lifecycle by the following states: committed, modified, and staged. I used these states during my work in the calculator folder. I was able to place multiple files in the modified state such as the multiplication.java and calculator.java because they were either being created, or new code was being added to prior created code. After creating the multiplication.java file and modifying other files, these files were shown in the staged state because the code was completed, but they were not committed into the main line. After, I was able to commit all my newly created files and modified files into the main line of the calculator folder code, which was the committed state. Once the code was committed, I was notified about no new changes were updated to the main line because the up-to-date changes were already committed.