Joseph Silva Jr.

12/13/2020

CS 310 Final Project

SNHU

**I. Overview/General Remarks:**

**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.

**3. Discuss your experience in using version control for work efficiencies and accuracy when working with peers in the Jukebox application.**

My experience was great when it came to working with peers. I was able to have my work reviewed by my peers and I was also able to see other members’ work. Being able to see other members’ work helped in the creation and review process of my own code. During creation of my code, I was able to compare my work to other members work as a reference. The review process of other members’ work allowed me to see and update my own knowledge on blocks of code being written because each member had different writing styles.

**Describe the benefits and drawbacks you experienced in working with the distributed workflow in the Jukebox app.**

During the Jukebox app, I did not experience any drawbacks when writing code and/ committing to Bitbucket. I did have drawbacks when it came to remembering the steps on how to create branches and uploading the branches, but it was because I have not had many experiences besides this class in working with branches. I had to go back through the tutorials to act as a guide to refresh my memory on each step. The benefits I had with the distributed workflow is the program worked very smooth for me while finishing up my section of code. A benefit was only having to worry about writing my own code and not having to write hundreds of lines of code for the whole team. The distribution of work was very helpful in time management and the prevention of code replication.

**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.

**III. Improvements/Enhancements**

**Provide a specific example of how performing code reviews informed your own development work.**

While conducting reviews, I was able to compare my team members’ work with my own to see if my code was on the right track, but it also showed me if my skill level was matching up with others from the class. While reviewing Shaun Richardson’s code, I was able to see we had similar code to each other, and the code worked. Being able to compare my work with other members, I was able to see my written code could keep up with the “pack” instead of the one holding my team mates back.

**Evaluate the degree to which review feedback informed your development work. Remember to consider not just feedback you implemented, but feedback you elected not to implement into code, and why.**

I implemented all my feedback given to me on BitBucket by my peers. I did not leave out any feedback because the feedback was just telling me about how my work was written well, and it did not need any changes. Since my code did not need any changes, I did not see a point in adding any new, which could have affected my entire code.

**Specify what ways a team could resolve issues brought up in code reviews. Think about scenarios in which team-based development projects encountered issues during the code review.**

A way a team could resolve issues is by giving honest feedback and keeping great communication with each other during the review of the code. Good communication can be used to benefit a team and each of the team’s members. Communication can make sure a team member stays in the parameters in order to protect the entire project’s goal and it can also help a member even if the member’s code failed because another member can explain why the code failed instead of only saying the code is a failure. I believe a team can encounter issued during a code review for many reasons. The issues can range from an individual member cannot put aside their own pride for the betterment of the team or it can be multiple members have different visions of how the code could be written. For these reasons, I believe communication is one of the most important aspects when working together as a team.

**Assess the usefulness of history and newsfeed tools in monitoring development.**

The usefulness of history and newsfeed tools was helpful in monitoring the development of the whole project and my own individual branch. I was able to not only see when and what I changed in my own branch, but I was able to see any changes made to the main project. I believe this is a very useful tool when the main source of code can grow and change constantly throughout the project. The history and newsfeed not only keeps everyone up to date with information, but it also allows the team to be able to figure out why changes were made to the main source file.

**Summarize your due diligence in preparing your code prior to the code review you engaged in with peers. Consider your use of unit tests and approaches to readability as techniques in preparing your code.**

After writing my code, I reviewed my code multiple times for any spelling mistakes. I used programs such as Eclipse to check my lines of code to make sure there was no errors/failures within my added code to the project. I then used Junit Tests to check if the program worked in order to obtain the expected outcome. I conducted all these individual tests before committing my branch to be reviewed by my peers. I wanted to make sure I was not writing half done work in order to show my team I was reliable, and my code should be up to standards.

**IV. Distribution and Deployment and Troubleshooting Tactics and Strategies**

**Assess the most valuable features of the tools used in your development. Provide specific examples of the benefits they provided.**

To me, the most valuable features of the tools to help my development was the history tool. Using history, I was able to keep my added work organized and I was able to keep a record of all my changes. I think the history tool would be helpful for an individual member project if the project is very large and over time early amount of work can be forgotten. I think the history tool also helped out while working on the Jukebox app because it gave the whole team knowledge on when certain sections of code were changed. I believe the history keeps everyone organized and allows up-to-date knowledge to be accessible to the whole team.

**Contrast the workflow strategies in relation to their application in individual and team development projects.**

The workflow strategies for the individual and the team project were different. The one workflow I mainly saw that differed between the two projects is the review. While working on the calculator app, I moved a lot quicker on the review of the code and sometimes I paid for it later because I had bias to my own work. I missed very small errors because I had written the code and I overlooked it on my own personal review. For the team, the review took longer because every detail of the code was written by someone else. I realized my own attention to detail when I was reviewing others work over mine because I did not want to give my team member bad feedback. I also had to look at the code in more detail because I had to see the route my team member was going while writing their code.

**Explain the standard developer version control best practices you considered to be the most effective.**

The most effective version control practice were two practices: Not putting only half-done work and test often. Since I did not put half done work, I did not have to go back and fix multiple errors, which would have wasted a lot of time during my assignment of the project. Since I tested my work often, I did not have to worry about having to go back and fix any problems later on. I think these were very helpful with keeping me on a set schedule to get assignments in on time.

**Illustrate how version control tools provided a benefit in identifying or resolving an issue in developing the Calculator app.**

Version control was very helpful when it came to the Calculator app because it is very difficult to make sure I was staying on track with the tutorials because results did not always match with the tutorial. I also had an issue with the Calculator app being uploaded as a Maven file and it was hard to stay on pace with the tutorials because I had a weight holding my work down. However, I was able to use tools such as the history tool to make sure I could always find my way back to my previous work when I got lost and made a mistake in the project.

**Illustrate how version control tools provided a benefit in identifying or resolving an issue in developing the Jukebox Playlist app.**

Version Control helped identify any issues during the Jukebox Playlist app by allowing me to see up-to-date knowledge on any changes made throughout the project. I was also able to go back to any previous files if I made a mistake in different files of the code because the changes were saved on the history tool. While working on the team project, I was able to identify any issues immediately especially when it came to code that I wrote in the student list file. I had an issue with creating my own personal playlist and it was immediately discovered that I had a spelling error.

**Provide a specific example of a version control technique used in both the Calculator and Jukebox Playlist apps that you envision using in your software development moving forward. Explain why.**

A version control technique I used in both Calculator and Jukebox was the centralized version. I was able to have my own files of code stored in my local repository. I felt comfortable knowing I could work on the project at my own pace and I did not have to rush any of my work because I was afraid the code would change drastically when I got back to it. I was able to test my work often and I did not put half done work because I was not worried about rushing to write my code and commit it.