# Benefits and Risks of Source Control

The benefits of source control is that you can have a copy of your project up on the web that way if something happened to your computer you only lost what you haven’t committed recently. It also makes it so you can work with a team of people. This team can work on separate pieces of the code and push it up to the source control site in pieces and when everything is complete the team can hook it all up together and there is the completed project ready to be shipped, if that is the goal.

Another benefit of source control is that if you pushed something to the site and realized that what you did made it so the project doesn’t build anymore you can click a button to revert your commit and it will act as if the change never happened. If you wanted to add a new feature and didn’t want it to interfere with the rest of the project you can branch off from what you were working off of and start to code from there. That way if what you wanted to try didn’t work it never would interfere with what you were originally trying to do. Just delete branch and it would have never happened.

A benefit I have heard about especially within github was an open repo that was making a massive program for a nonprofit organization that helps out in disasters and it would put issues up and anybody could go to it assign themselves an issue and start working on it. A good way to get things done with minimal effort. I can’t remember all the details about this project except that I heard it in a podcast.

A risk to source control is if you’re working on an open repo, like what github does, somebody else could come in and wreck your stuff if you didn’t take any type of precautions to prevent it, or have to remember to revert the changes that were made by the interloper. Another risk is something we have been running into within the class is the lack of knowledge of how to use the source control so it has been a massive tangled mess that the person that really understands the source control has to go in and try to fix it all or find a way to figure it all out.

# Check-in & Check-out concepts

To check-out a repository to add features or whatever it is that you were asked to do you can create a branch of from it that way if something happens where what you were doing didn’t work or turned out to be not what was needed you can just delete the branch and it would never have affected the master copy of the code. Name the branch something that makes sense like sort feature or something like that.

Checking-in the code should only be done once it has been tested thoroughly by some type of unit testing preferably some type of automated testing. If the code doesn’t pass the test then it needs to be reworked and reworked until it passes the tests. If it doesn’t pass the test it could make the portions or the entire program not function properly. If the code has passed the test then push the code up and attempt to merge it into the main project leaving it for the leader to double check the code before it gets merged into the master code.

# Constraints when using source control on team development projects

Constraints are rules that used to help organize the program the team is using. Some constraints could be, only code that has been tested can be submitted for merging, or naming conventions in commit titles to help see what you changed and an actual description so there can be a reason why it was added. These constraints seem simple until they are actually put into use. It would be an individual’s responsibility to follow these constraints as they are described.

Without the constraints there could be mass chaos in the organization world. If something did go wrong and you wanted to revert before a certain feature was added you wouldn’t know where to revert from easily. Another possibility is there could possibly be more broken programs during production because somebody didn’t test there code before putting it in, or there simply was no code to test their code against.