Blaine Simcox

CIT 275

Jeff Straw

Source Control

**Mitigating Source Control**

For the entirety of this class, we have been using GitHub.com to mitigate source control tasks. This is important to team-level projects as it keeps an accurate log of when and how an application or file has been changed. When working on projects that consists of millions, if not billions of lines of code, this mitigation is necessary in establishing the when and what changed if your multi-month project suddenly stops working from a new update.

In an office environment this is also useful in keeping track of personal performance of individual employees as it helps employers to see strengths and weaknesses among each employee. This data becomes very useful when assigning work roles in future sprints. You do not want your best Entity Framework coder handling your GUI and your best artists handling the database, this would disastrously inefficient and may result in project failure.

GitHub has proven to be a qualified tool for handling the task of source control. As you make changes to a repository folder, you are required to comment out the details of the changes and even set a quick summary so that all team members can read explicit details on the update. This Check-in/Check-out system leaves a very detailed record of all changes made in a project repository, and if it comes down to it, helps reinforce the blame when needed. GitHub does an amazing amount of the source control documentation for you, autonomously attaching times, dates, and user names that are stored on the site for all team members or employers to view. Every change can be put on separate branches or forks so that they do not interfere with a main project folder until they are tested and cleared for integration. Each team member can even have their own fork to use as their own private cubicle within the repository to propose design ideas and tests. When a fork is tested and the team has cleared the fork for integration, the files can then be easily moved into the main project. Though there is an initially high learning curve due to poor instruction manuals, GitHub is relatively easy to use once set up properly.

Another great use for a source-control manager such as GitHub is stopping bad updates from getting on the project site. In the case of someone other than the repository owner, updates cannot be installed to the repo without final permission of the repo administrator. I assume I speak for the majority of us when I say programmers tend to run off lots of caffeine and very little sleep, and because of this, critical errors can all too easily infect a master project file. I don’t know how many times I have had to resubmit projects due to rushing and zombification from staring at my screen. Because of this human tendency, stop management controls such as the above can be vital to any multi-user project in preventing untested code from entering or replacing vital, tested files and causing a setback.