***Team A – Tank Wars***

***Project Management Tools Report***

***September 19, 2014***

***John Aromando***

***Mike Bunger***

***Leah Austin***

***Jason Rivera***

**Overview**

After creating a prioritized list, the team has analyzed three different configuration management tools to see which will be used in the coming weeks. The team has come up with a pros and cons list to see which will be the most beneficial in developing the “Tank Wars” project. The team has also come up with a plan to train a new recruit in case one is added to the team.

**Process Management Tools**

**Assembla**

**Installation**

1. Create an account (free or paid).
2. Follow Assembla prompts.
3. Create a name and url for project (only 1 project for free version).

 **Features**

* Tickets – Create tasks on a prioritized list with team member allocation.
* Several task management and notification systems.
* File storage up to 500mb (for free version).
* Integration with SVN and Git.
* Communication in the form of messages and updates.

**Google Docs**

*\*See below in the Configuration Management Tools section\**

**Blackboard**

**Installation**

1. Must be a student or teacher.
2. No installation.

 **Features**

* Discussion Board – A team forum.
* File Exchange – Share files with team members.
* Group Tasks – Simple task list.
* Integrated with school email.

**Configuration Management Tools**

**Google Docs**

Google Docs is a service provided by Google. Documents can be shared, store, and downloaded amongst many users.

**Installation**

1. All team members must have a Google account.
2. Document owner must invite other team members.

**Features**

* More than one developer can work on a project at the same time.
* One team member owns and manages the documents.
* Text file depository.
* Free spreadsheet tool (ideal for schedules and tasks).
* Integrated with Gmail, Google Drive, and other Google operations.
* Edit or share files with team members.

**Subversion (SVN)**

**Installation**

*\*Eclipse is required in order to use SVN\**

1. Open an internet browser and go to <https://subversion.apache.org/download/?Preferred=http%3A%2F%2Fmirror.reverse.net%2Fpub%2Fapache%2F>
2. If the user is using Windows, download the zip file. All other operating systems recommend downloading the .tar.gz or .tar.bz2 files.
3. Open Eclipse and press “help”> “install new software”.
4. Press the drop down arrow next to “work with” and look for “- -All Available Sites- -” and click on it.
5. Click the drop down arrow next to “Collaboration” and look for items that have “Subversive” in the title and select them.
6. Click “Next” twice, agree to the terms and press “Finish” and wait for it to install.
7. Restart Eclipse.
8. Once Eclipse is back up, go to “Window” > “Open perspective” > “Other”.
9. Click “SVN Repository Exploring”.
10. Click “SVN Kit 1.8.6” and “Native JavaHL 1.8.9” and finish the installation.

**Features**

* Allows members to access different versions of the software.
* Supports “locked files” so that in case multiple users try to access the same file, a warning will appear to prevent ruining each other’s work.
* Displays local messages that give useful information.

**Resources**

* Users can use Eclipse to update project worked on by team.

# Git

# Installation

You can install Git in a number of ways; the two major ones are to install it from source or to install an existing package for your platform. Generally, it is useful to install Git from source, because it will provide its most recent version. Newer versions include very upgraded tools and are especially helpful in regards to the Linux version which has very dated packages.

If you want to install Git on Linux, you will want to go through a binary installer and then follow instructions. Installation on the Mac can be done by a graphical Git installer. It can also be installed through MacPorts and Homebrew, but are not as easily accomplished as through the graphical installer. Installation on Windows is by far the simplest, just downloading the .exe file from the Git page and running it through the command line can accomplish the installation. All versions of Git installation can be done, and are recommended to be, through the different command line like features of each operating system.

Exact detail can be found on the Git webpage at: <http://git-scm.com/book/en/Getting-Started-Installing-Git>

Refer to the Git webpage: <http://git-scm.com/downloads> for more download information.

# Git Features

* You can get a Git project using two main approaches; taking an existing project or directory and importing it into Git or cloning an existing Git repository from another server.
* In this repository, you will want to commit changes to make them save.
* Each file can be in one of two states; trackedoruntracked*.* Tracked files are files that were in the last version or what they call snapshot; they can be unmodified*,*modified*,* or staged*.* Untracked files are everything else.
* When you first clone a repository, all of your files will be tracked and unmodified because nothing has been done to those files.
* To be able to collaborate on a Git project, managing the projects remote repositories is something that must be done. Remote repositories are hosted on the Internet or a network and can include different versions of the project. Several of them can be made, each with the ability to make read only or read and write. Collaborating with others involves pushing and pulling data to and from them when you need to share work. Managing remote repositories includes adding remote repositories, removing no longer valid remotes, managing various remote branches, and defining whether they are tracked.
* Git has the ability to tag specific points in history as being important. This is an important tool in marking release points.
* Branching which means diverging from the main line of development, continuing to do work, but without messing with that main line.
* Git branching is easy, allowing branching to be nearly instantaneous. Switching back and forth between branches are just as fast. Git is set up to encourage working with branching and merging.
* A "Git server", is the preferred method for collaborating with someone. Here an immediate repository is setup and the project members use that to push to or pull from.
* Stashing takes an unfinished state of work (the modified tracked files and staged changes) and saves it on a stack of unfinished changes.
* Git is customizable to the user.
* Git can take on external packages.
* All this and more can be found on the Git Webpage: <http://git-scm.com/doc>

# Git Commands and Terms

# Bare repository - Git repository that has no working directory.

# Commands - <http://git-scm.com/book/commands>

**Training**

When a new team member is added to the group, one member will be assigned to overlook his/her training and monitor their progress.

1. The senior member with explain the goals of the project, how the team stays organized, and review team ethics.
2. The senior member will help install Git and explain how it is being used by the team.
3. The new member will watch the senior member as he/she works on the project.
4. The new member will be assign small and simple task to ensure he/she understand the team’s dynamics.
5. Tasks will gradually increase in size and difficulty as the team’s confidence in the new member increases.

**Plan Moving Forward**

* Learn how to use Git and its features.
* Install all the files to the repository.
* Make sure all members have Git running.