

College of Arts and Science

School of Science and Technology

**SOFT20111: Software Engineering**

# Portfolio 5 (P5)

By

N0485381: Darren Langley

Merging: Theory and Practice

**Lab Tutor:** Dr Richard Hibberd

Contents

[Portfolio 5 (P5) 1](#_Toc417497025)

[5.1 – Version Control Background (Theory) 2](#_Toc417497026)

[What is version control? 2](#_Toc417497027)

[Why is merging necessary? 3](#_Toc417497028)

[5.2 – Mercurial SCM Demonstration (Practice) 4](#_Toc417497029)

[References 6](#_Toc417497030)

[Bibliography 7](#_Toc417497031)

## 5.1 – Version Control Background (Theory)

### What is version control?

A version or revision control system or software is a container (repository) of files with monitored access. Often those particular files tend to be source code for computer programs to aid software development projects (Yeates, 2005).

Here are some examples of version control software, which explains what each one provides:

#### Mercurial

(Rawson, 2011) explains that Mercurial was originally made to compete with Git for Linux kernel development and also explains that Mercurial is easier to learn than Git. Noticing that Mercurial is essentially developed in Python, with certain aspects of C, Python developers have considered a switch which will allow non-core developers/students to have easier access to understand the basics on how a particular version control software works/functions.

#### Git

Git was originally developed for Linux in order to make this particular version control software faster to deal with large projects efficiently (Future Publishing Ltd., 2009).

Git uses a stash list feature where if developers are currently working on large projects, they can stash that particular bit of implementation to one side and can potentially come back to it later when ready to commit to the repository without worrying about interruption. It is more complex than a simple branching approach, however the stashing approach is more powerful and will eventually prevent confusion on what stash to work on once familiarised (McBride, 2008).

#### Apache Subversion (SVN)

(Rawson, 2011) explains that SVN was created essentially as an alternative to CVS, ideally for fixing bugs in the CVS system making it faster.

It transmits less information through the network and supports more operations for offline mode. SVN achieves the efficiency over CVS by full back up of all project files. SVN is not frequently used unlike CVS and Git, where there are particular features of the version control system that are still not implemented as expecting. However the code developed is expandable, allowing room for future development to be made, potentially based on those areas still not implemented. (PushOk Software Ltd., 2005).

#### Visual SourceSafe (VSS)

VSS is more of a hierarchy based version control software that allow developers to keep track of any differences/updates regarding projects, files and the history of committed projects. VSS keeps all checked in versions onto the repository, where developers can only see the latest version of software development. This makes file management easier to maintain, especially when it comes to tight scheduled projects. Where previous versions are needed, a “Show History” feature is also incorporated to access previous projects/ working code. (Gao, 2008)

#### Team Foundation Server (TFS)

Built within Visual Studio, TFS builds into Team Explorer as an interface for team projects. With each team projects folder usually contains a Source Control folder for viewing and managing source code, which opens Source Control Explorer. This is where many of the source control functions are performed, such as branching and merging and checking in and out of the repository. TFS provides a workspace feature, where this will map projects contained on the server to a local area on a drive and is essentially a core feature to have, as developers cannot check in/out files to the repository otherwise. When checking out a project, this can be done through Solution Explorer or Source Control Explorer. (Minisi, 2011)

### Why is merging necessary?

Merging is used to manage and maintain software development when 2 or more people are working on the same project, potentially the same coding file as a form of pair programming. There are situations when developers want to experiment or commit changes on the repository that could break a working program by updating/modifying code.

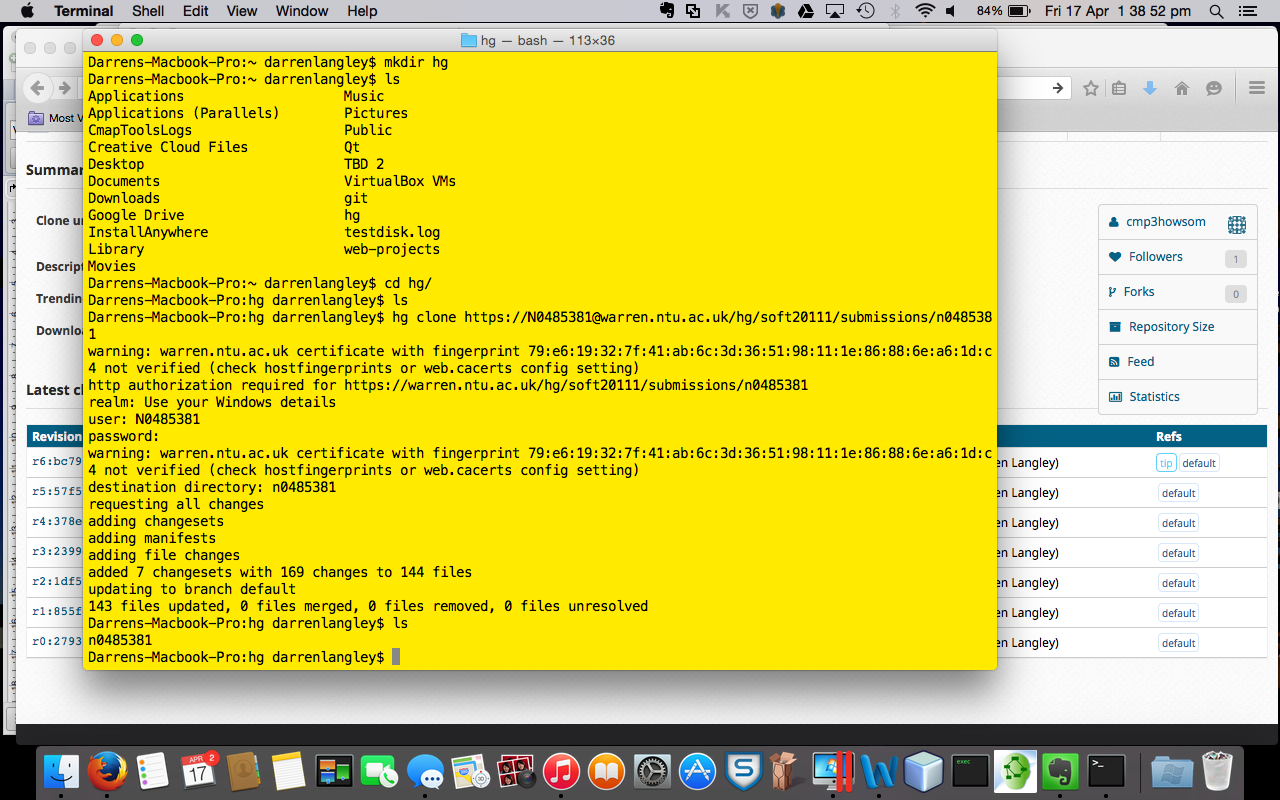
A branch is created which takes a working copy of the repository without editing the main code/file source (usually called the master or trunk), whereas developers may commit to the main set of code without affecting the branch version. As branching and committing are in a continuous process, there is a point where merging comes into place which creates a repository of the working program. (Wildbit, 2015).

Version control software knows how each file has been altered. A changeset represents a graphical representation of branching and committing changes which forms a unique revision number containing an incrementing number or unique hashing/identifier depending on the system. This will eventually form a final representation (master) based on those branches and commits. This is another form of making backups. Sometimes the version control system may not know which change to apply to between two revisions or different branches when merging, causing a conflict. When a conflict occurs, manual interception is required to decide on which revision contains sufficient file contents/code when working on particular projects whilst branching. (Wildbit, 2015).

Once changes have been committed onto a repository, also known as a centralised version control, developers will then need to push those changes onto the main repository using the push command. The main repository represents the new master/updated version of the working program, also known as a distributed version control. It will essentially get the current master/version of file contents by pulling them into the central repository. This will combine the contents of what has been gathered from the main repository with the current contents that has been committed onto the central repository and together, they form the new main repository to be pushed back. This is essentially a merging process, subtracting the deletions made in the current repository (Ernst, 2012).

Reverting is a tool where if developers are currently processing changes to a project or coding files, then they have the option of discarding that change without anything happening to the master. This is in the form of *“version control software, revert”* where version control software is the tool that developers use to branch, commit and merge files together to form the currently updated master version. If a particular file needs to change, this would be in the form of *“version control software, revert, filename”* where the filename must provide its extension. If developers commit changes by accident, this would be in the form of “*version control software, un-commit”* (Canonical Ltd., 2011)*.*

## 5.2 – Mercurial SCM Demonstration (Practice)



**Figure 1: Cloning / checking out from the mercurial repository**

Figure 1 shows that the current state of repository contents are downloaded to the hg folder, which is the current directory being used. This shows how many changes, changesets (also known as a set of changes) which have been set as new revision masters of the repository and the total number of files contained on/obtained from the repository.

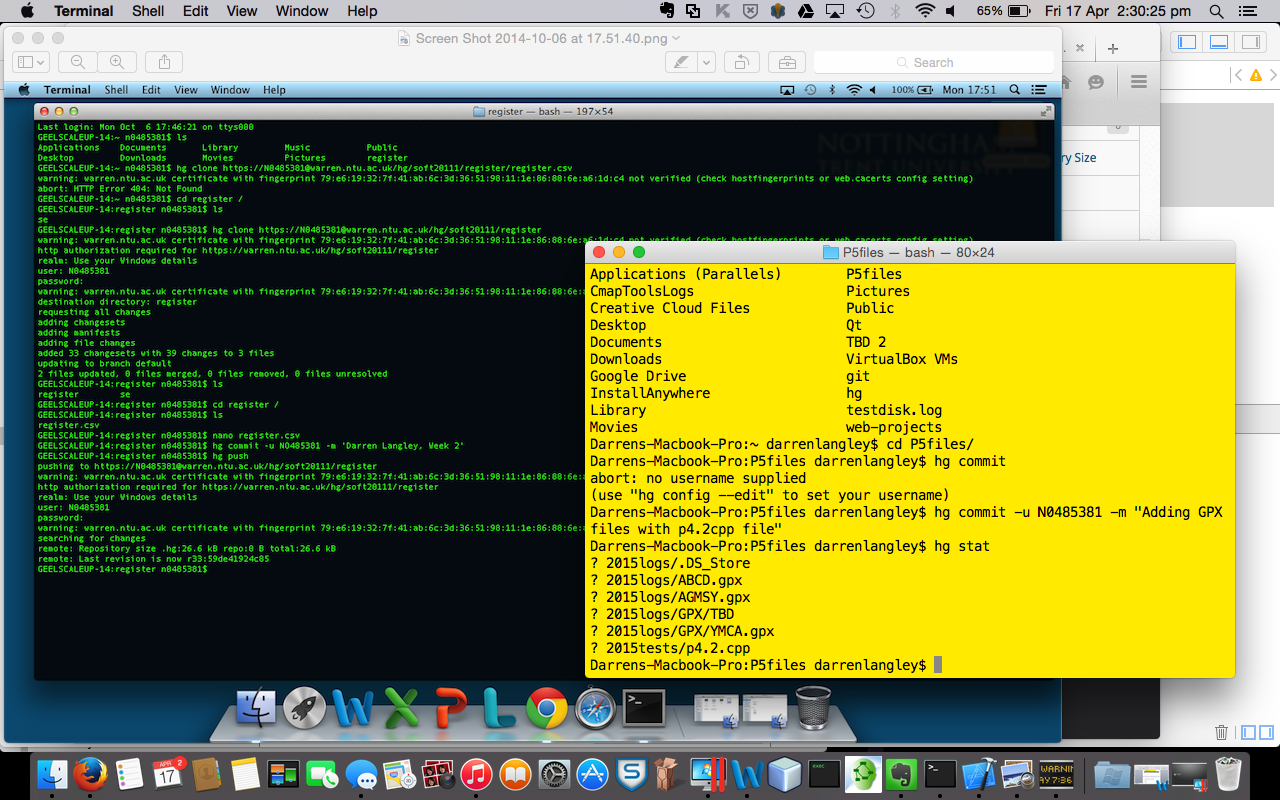
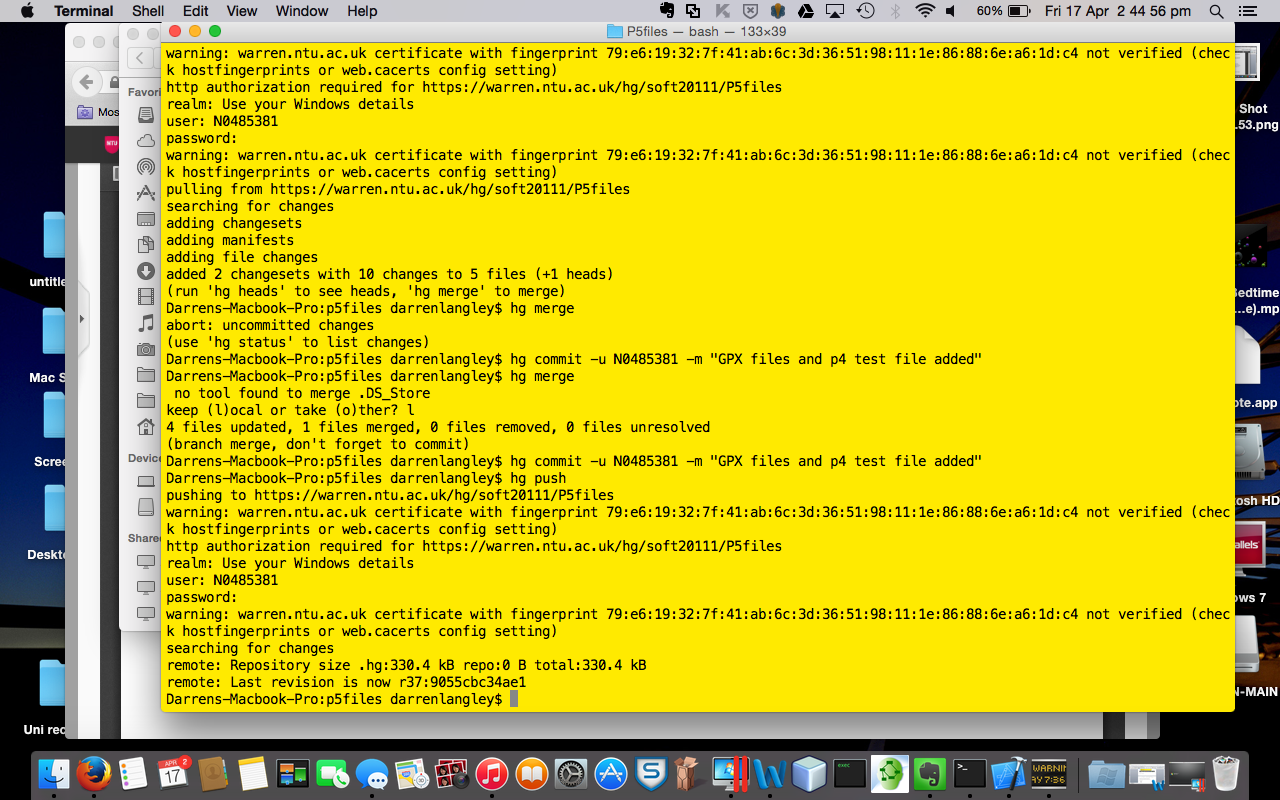
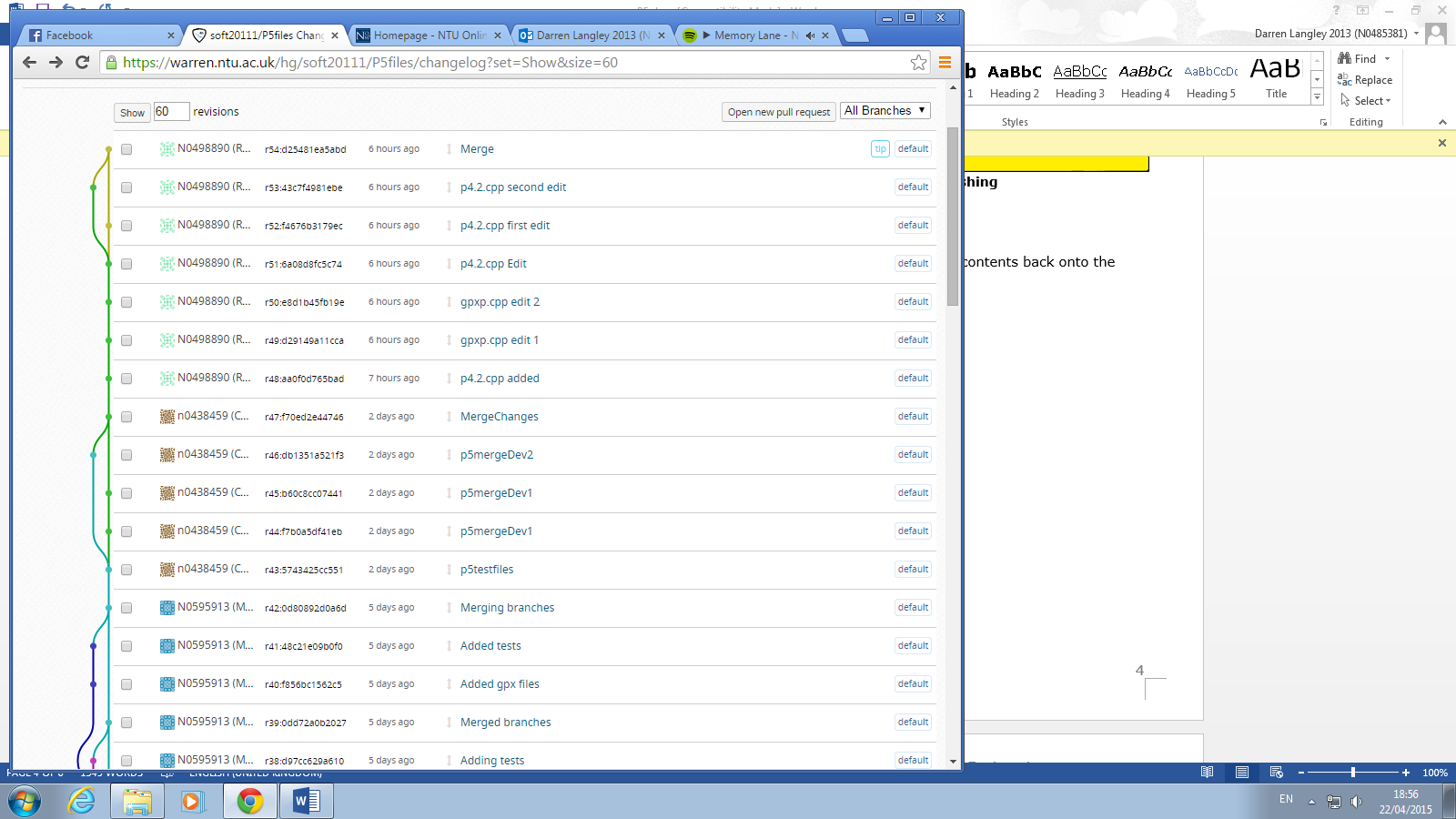


Figure 2 shows what files have been changed or added from the current repository contents, using both the hg commit and hg stat command.

This is submitted through the local repository before the pushing and merging processes take place.

**Figure 2: Adding / Editing code**

**Figure 3: Merging, committing and pushing**



**Figure 4: Result of changelogs**

Figure 3 explains that when committing and pushing puts the contents back onto the repository, a branch is created detailing the edits made. When committing, pushing and merging the contents back onto the repository, it combines the master version downloaded, along with the changes made to that master. As a result, a new master or final version has been produced of a particular development code, as shown in Figure 4.

In conclusion, while merging has not taken place and if committing and pushing onto the repository has been made, this will continue to make branches until a merge has been made. When a merge has been made, this will cut down a branch. As more merges are made, more branches are broken down until a final version has been made. It is worth considering that as huge projects are developed, more branches may be necessary as backups but at the same time, it is also worth considering when to stop making branches.

# References

Canonical Ltd., 2011. *Undoing mistakes.* [Online]   
Available at: http://doc.bazaar.canonical.com/latest/en/user-guide/undoing\_mistakes.html  
[Accessed 29 March 2015].

Ernst, M., 2012. *Version control concepts and best practices.* [Online]   
Available at: https://homes.cs.washington.edu/~mernst/advice/version-control.html  
[Accessed 29 March 2015].

Future Publishing Ltd., 2009. *Which version control system is best for you?.* [Online]   
Available at: http://www.tuxradar.com/content/which-version-control-system-best-you  
[Accessed 31 March 2015].

Gao, K., 2008. *Using Visual SourceSafe – What can VSS do?.* [Online]   
Available at: http://www.codepool.biz/version-control/sourcesafe/what-can-sourcesafe-do.html   
[Accessed 31 March 2015].

McBride, M., 2008. *3 Reasons to Switch to Git from Subversion.* [Online]   
Available at: http://markmcb.com/2008/10/18/3-reasons-to-switch-to-git-from-subversion/  
[Accessed 31 March 2015].

Minisi, B., 2011. *Using TFS 2010's Source Control and Version Control Features.* [Online]   
Available at: http://devproconnections.com/visual-studio/using-tfs-2010s-source-control-and-version-control-features  
[Accessed 31 March 2015].

PushOk Software Ltd., 2005. *SVN vs CVS.* [Online]   
Available at: http://www.pushok.com/software/svn-vscvs.html  
[Accessed 31 March 2015].

Rawson, R., 2011. *SVN, Git, Mercurial - Comparison of Version Control Software.* [Online]   
Available at: http://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/  
[Accessed 31 March 2015].

Rawson, R., 2011. *SVN, Git, Mercurial – Comparison of Version Control Software.* [Online]   
Available at: http://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/  
[Accessed 27 March 2015].

Wildbit, 2015. *An introduction to version control.* [Online]   
Available at: http://guides.beanstalkapp.com/version-control/intro-to-version-control.html  
[Accessed 29 March 2015].

Yeates, S., 2005. *What is version control? Why is it important for due diligence?.* [Online]   
Available at: http://oss-watch.ac.uk/resources/versioncontrol  
[Accessed 31 March 2015].

# Bibliography

Canonical Ltd., 2011. *Undoing mistakes.* [Online]   
Available at: http://doc.bazaar.canonical.com/latest/en/user-guide/undoing\_mistakes.html  
[Accessed 29 March 2015].

Ernst, M., 2012. *Version control concepts and best practices.* [Online]   
Available at: https://homes.cs.washington.edu/~mernst/advice/version-control.html  
[Accessed 29 March 2015].

Future Publishing Ltd., 2009. *Which version control system is best for you?.* [Online]   
Available at: http://www.tuxradar.com/content/which-version-control-system-best-you  
[Accessed 31 March 2015].

Gao, K., 2008. *Using Visual SourceSafe – What can VSS do?.* [Online]   
Available at: http://www.codepool.biz/version-control/sourcesafe/what-can-sourcesafe-do.html   
[Accessed 31 March 2015].

McBride, M., 2008. *3 Reasons to Switch to Git from Subversion.* [Online]   
Available at: http://markmcb.com/2008/10/18/3-reasons-to-switch-to-git-from-subversion/  
[Accessed 31 March 2015].

Minisi, B., 2011. *Using TFS 2010's Source Control and Version Control Features.* [Online]   
Available at: http://devproconnections.com/visual-studio/using-tfs-2010s-source-control-and-version-control-features  
[Accessed 31 March 2015].

PushOk Software Ltd., 2005. *SVN vs CVS.* [Online]   
Available at: http://www.pushok.com/software/svn-vscvs.html  
[Accessed 31 March 2015].

Rawson, R., 2011. *SVN, Git, Mercurial - Comparison of Version Control Software.* [Online]   
Available at: http://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/  
[Accessed 31 March 2015].

Rawson, R., 2011. *SVN, Git, Mercurial – Comparison of Version Control Software.* [Online]   
Available at: http://biz30.timedoctor.com/git-mecurial-and-cvs-comparison-of-svn-software/  
[Accessed 27 March 2015].

Wildbit, 2015. *An introduction to version control.* [Online]   
Available at: http://guides.beanstalkapp.com/version-control/intro-to-version-control.html  
[Accessed 29 March 2015].

Yeates, S., 2005. *What is version control? Why is it important for due diligence?.* [Online]   
Available at: http://oss-watch.ac.uk/resources/versioncontrol  
[Accessed 31 March 2015].