Working in the Cloud:

Web-based Version Control System for Task-oriented Group and Individual Projects

Aim to develop this version control system with these features:

1. Fully web-based, without need of installation of any software at client side, and can be accessed at anywhere. It meets the concept of Cloud Computing.

2. Task-oriented development friendly, avoid concurrent and overlap editing, avoid confusing.

# Literature review

This section was about the literatures related to this project – a web-based version control system.

需要修改：1 LR介绍，2 existing部分，3 programming部分，4 脚注部分

## Version Control

Version control, usually be called as revision control or source control, was a method of managing files related to development of projects through their whole life cycle [1]. Lots of popular version control software includes CVS and Subversion run as client-server model, so they support more than one users working especially programming for a same project, because the multi-clients can be connected to a central version control server to be synchronised [2] [3]. The general features provided by version control include storing each commit/version of files or directories, allowing rollback, showing changing history and assist merge/integration [1]. Based on the features, many kinds or parts of project can use version control to be effective managed, such as projects of software development and documentation writing. Each commit/change can be marked with properties of which author did this commit and what time it committed. The commits also comes with a unique version code to identify the times of changes of it. Users can also add some comments to the version when they commit it, so it is very clear to identify what the users changed in there committed version. When a version commit has been identified as worse than an older version, the changes can be easily roll backed to a previous version at any time [1].

Even though the existing version control systems sound wonderful enough, however there still an important point should be noticed, that is the “concurrent access” problem exists in almost all the version control systems [4]. Every project developer can access an entire project, and have privilege in changing any part of the project. It is hard to assign responsibility as tasks to the developers, so the developers may forget where their positions in developing the projects are. When a developer mixed up his/her range of development, and changed some files which someone else is working for, it may leading to serious problem, or waste of time in combining works manually, even though merge policy can do combination of most their works, the result cannot be guaranteed as the most expected one due to the complicacy of different kinds of working [4] [5] [6]. Some version control system, such as Subversion, has designed the “lock” feature to prevent this problem. However, even though a file has been locked by a developer, other developers can also get copy of it from their local update before file locked, or from archive of older versions. It is hard to restrict this problem under the current version control policy [5].

To solve this problem, this MSc project was aimed to develop a version control system which has “task-oriented” feature avoids developers forget their role in development, also prevents the happen of concurrent editing of files. Task-oriented development supports more than one people working with one project at its separated tasks, without interrupt and overlap. Assignment of tasks for the developers can be confirmed by a discussion in a group of each developer’s strong points and weak points, and finish assigning by group leader in the new system.

## Existing Version Control System

As coordination and work management were important features of the system, some design can be learnt from operating mechanism of existing version control system. In the following, the popular software-based client-server version control system CVS [2] and Subversion [3] will be analysed by their important features.

### Unique version number

Every change of directory and file will generate a new version to the directory and file itself. Copying, renaming and deleting will be considered as a change. Each historical version will be kept for in-case use [3].

### Atomic commits

For coordination considering, every submitting of change set will generate a version of change history, even only a part of the set has been changed [3]. As this project was task-based, so it could be different from the SVN versioning mechanism, which is every files will update to a latest same version code after an every submitting command. Task-based feature limited area of change set to tasks, therefore, the versioning mechanism could be designed as every time of file and directory change will increase the version number of file and directory individually, and the task version will be increased when a submit operation has been executed, no matter how many changes of file and directory have been made, even only renamed a file.

### Locking

To avoid unsynchronised editing, lock-work-submit-unlock is a very good feature provided by SVN for conflicts free editing [3]. In this project, the task as a minimum assignable unit, could be locked when a user start doing the task. When a lock has been set to a task, other user except the user who sets the lock cannot request to do the task, even he/she was already been assigned to the task.

The project was designed as task-based, also helps members in a group doing their common project under schedule. In contrast, none of existing version control system provides scheduling features, including the most popular software-based version control system – Subversion. To achieve the schedulable feature, in the system of this MSc project development, the tasks could be set up with relationship with other tasks. In 1910, Henry Gantt first published the concept of predecessor in his famous Gantt chart [7]. It resolved the organisation problem of group collaboration in scheduling very well. This project uses the concept from Gantt chart to scheduling tasks. In the system, a task could have a predecessor/father task, which means a task can be start only when its predecessor task already finished.

## Web and Cloud

Lots of popular version control systems are traditional client-server model based [8], even though some version control systems were built with the distributed approach, like Git[[1]](#footnote-1), however, it is very complicated for small group and individual projects to use, due to it is hard to understand and use for non-experience users [9]. To set up a traditional version control system, it requires sets up three parts which will be configured to work together: centralised server runs server side software, client(s) runs client side software and reliable network connections between server and client(s). Server stores every versions/commits of each files, client(s) stores a copy of latest version of project files and allow user working and changing it for future commit [1] [10] [11]. Network connection is a very important component in version control system, because it connects server and client(s) for both of them can be synchronised to latest status and keep files up-to-date. In normal way of doing a project in a version control system, users as clients usually sign in by client side software to the repository of the project in version control server, download (update) latest version of copy of all files related to the project from server to their own computer, and work for the files. The users may have arrangement in advance to avoid possible overlap working in same files, at least in same class of a programming project or same paragraph of a documentation writing project, because the existing merge algorithm in the most version control systems could not combine overlapped work in different commits without conflicts faultlessly. When a user finished a milestone of working in his/her files, he/she needs to re-login by client software into project repository in the server, commits all the working he/she did to the server. After commit, a new version code will be generated. When other users as clients tried to download/update the project, if newer version of files on the server has been detected after compared to the local version, the files in the client side would be updated to the latest version [1] [12].

The helpful features made version control systems work great for managing many kinds of project. However, it needs to install software at both server side and client side to perform version control actions [1] [12]. For entry-level users, it may be hard for them to install and configure client side software to work with server which providing version control service; For users who often working in different kinds of environment instead of their own computer, such as working in computer lab and on mobile devices like iPad, they may have not privilege to install client side software of version control systems, or the mobile device does not support software for version control.

To allow client side user can working at most environment with version control support, it is important to find out a way deliver the service without need of installing software. By the inspiring of more and more popular cloud computing concepts like Google’s Cloud [13], this project was considered to develop a fully web-based version control system which does not require any installation of client side software. Web-based design can overcomes some drawbacks of software-based design, such as hard to use at anywhere and hard to configure by entry-level user. It can be easily accessed at any computer, even a mobile phone, just need a browser and network access to the version control server. When transfer from traditional version control system to web-based system, user can not only benefit from the “access everywhere” feature, but also gain from an important feature – “easy upgrade at cloud with less disruption”. If the system needs to upgrade to latest release, it does not need to ask user to do upgrade of their client side software like traditional way. To upgrade the web-based system, just need to change the server side software, and all users could start using the new system as usual via their own web browsers [13]. The concept of “network”, “cloud” and “web-based” not indicates the system is only Internet-based. However, it can run at a local area network as a “local cloud”, because the Ethernet supports same technology of Internet’s, such as IP based TCP connection and HTTP protocol, which allows web-based system run at local area network similar to run at Internet [14]. To set up the web-based system in a company-wide network, it needs to allocate a computer as server, configure its Apache, PHP and MySQL running environment, and install web-based version control systems in it. After installation, users can direct access the server’s domain name[[2]](#footnote-2) or IP address, even a private IP address[[3]](#footnote-3), on any browsers by devices connected to the same local area network with the server. Due to the centralised structure and network connection dependency of traditional version control systems, the cost of migration from traditional systems to the new web-based system might be very low – just need do configuration on server side with new web-based system, and tells users to access the new system via their browsers, even mobile browsers, and then continue their working.

## Programming

In order to develop the system to be used at web, there are several programming languages available, such as Java/JSP[[4]](#footnote-4), ASP[[5]](#footnote-5), ASP.net[[6]](#footnote-6), CGI[[7]](#footnote-7) and PHP. In these languages, Java as a popular programming language in object-oriented software development can also provide web service by working with JSP on Servlet[[8]](#footnote-8), even though it provides the most object-oriented structure for programming, however, it is hard to set up server side environment and also hard to programming in the scale of this kind of project; ASP, an Microsoft’s outdated web application uses Visual Basic programming language, which is very popular at the era of Microsoft Windows NT 4.0 and Windows 2000, with many down sides such as lack of library support and low safety design; ASP.net is the latest Microsoft’s web application uses C# as programming language, overcomes many drawbacks existed at ASP, however it still not an open source platform like before; the last one, PHP, is the most shining web programming language with lots of great features, such as fully open source, object-oriented support, abundant built-in library functions and rich high quality open source resources. The running environment of PHP is also easy to be set up. It can run with open source web server software such as Apache[[9]](#footnote-9) and Nginx[[10]](#footnote-10), also Microsoft’s IIS[[11]](#footnote-11) series. So this MSc project has been chosen PHP as the programming language.

## Interaction Design

As a web app with user interface, interaction design is an important part of this project.

References

|  |  |
| --- | --- |
| [1] | B. Collins-Sussman, F. W. Brian and C. M. Pilato, Version Control with Subversion, O'Reilly, 2004. |
| [2] | D. Price, “CVS v1.11.23 Manual,” Ximbiot LLC, 8 May 2008. [Online]. Available: http://ximbiot.com/cvs/manual/cvs-1.11.23/cvs.html. [Accessed 1 July 2011]. |
| [3] | The Apache Software Foundation, “Apache Subversion Features,” [Online]. Available: http://subversion.apache.org/features.html. [Accessed 1 July 2011]. |
| [4] | T. Oakden, “None Concurrent Access in Version Control,” 12 Oct 2009. [Online]. Available: http://forum.unity3d.com/threads/36536-None-concurrent-access-in-version-control. [Accessed 22 July 2011]. |
| [5] | P. Roy, “Understanding Subversion's Problems,” 9 Mar 2011. [Online]. Available: http://ventspace.wordpress.com/2011/03/09/understanding-subversions-problems/. [Accessed 22 July 2011]. |
| [6] | Jbcrouigneau, “Task Oriented Development and Validation Space,” 24 Apr 2009. [Online]. Available: http://www.svnforum.org/threads/36840-Task-oriented-development-and-Validation-space?s=cf3b028492de3003320a35e609f4777b. [Accessed 22 July 2011]. |
| [7] | H. Gantt, Work, Wages and Profit, New York: The Engineering Magazine, 1910. |
| [8] | Shawn, “Which is More Popular (Currently, by Recent Install Base) SVN or CVS?,” 23 Apr 2009. [Online]. Available: http://stackoverflow.com/questions/782375/which-is-more-popular-currently-by-recent-install-base-svn-or-cvs. [Accessed 23 July 2011]. |
| [9] | C. Duan, "Understanding Git Conceptually," 17 Apr 2010. [Online]. Available: http://www.eecs.harvard.edu/~cduan/technical/git/. [Accessed 23 July 2011]. |
| [10] | T. Spencer, “Setup a Subversion Server in 4 Minutes,” 2 Mar 2007. [Online]. Available: http://www.tonyspencer.com/2007/03/02/setup-a-subversion-server-in-4-minutes/. [Accessed 23 July 2011]. |
| [11] | “Setting Up Subversion,” July 2006. [Online]. Available: http://systhread.net/texts/200607subver.php. [Accessed 23 July 2011]. |
| [12] | D. Thomas and A. Hunt, Pragmatic Version Control Using CVS, Pragmatic Bookshelf, 2003. |
| [13] | Google, “Top ten advantages of Google's cloud,” 2011. [Online]. Available: http://www.google.com/apps/intl/en/business/cloud.html. [Accessed 24 July 2011]. |
| [14] | L. L. Peterson and S. B. Davie, Computer Networks : A Systems Approach, Amsterdam; London: Morgan Kaufmann, 2007. |
| [15] | Oracle, "JavaServer Pages Technology," [Online]. Available: http://www.oracle.com/technetwork/java/javaee/jsp/index.html. [Accessed 2 July 2011]. |
| [16] | S. Chacon, "Git - The Fast Version Control System," 2011. [Online]. Available: http://git-scm.com/. [Accessed 23 July 2011]. |
| [17] | J. Loeliger, Version Control with Git, O'Reilly, 2009. |
| [18] | R. M. Hinden and B. Haberman, “RFC 4193 : Unique Local IPv6 Unicast Addresses,” Feb 2005. [Online]. Available: http://tools.ietf.org/html/rfc4193. [Accessed 24 July 2011]. |
| [19] | Y. Rekhter, R. G. Moskowitz, D. Karrenberg, G. J. d. Groot and E. Lear, “RFC 1918 : Address Allocation for Private Internets,” Feb 1996. [Online]. Available: http://tools.ietf.org/html/rfc1918. [Accessed 24 July 2011]. |
| [20] | P. Mockapetris, “The Domain Name System,” in *Proceedings of the IFIP 6.5 Working Conference on Computer Message Services*, Nottingham, 1984. |
| [21] | P. Mockapetris, J. Postel and P. Kirton, “Name Server Design for Distributed Systems,” in *Proceedings of the Seventh International Conference on Computer Communication*, Sidney, 1984. |
| [22] | P. Mockapetris, “RFC 882 : Domain Names - Concepts and Facilities,” Nov 1983. [Online]. Available: http://tools.ietf.org/html/rfc882. [Accessed 24 July 2011]. |
| [23] | P. Mockapetris, “RFC 883 : Domain Names - Implementation and Specification,” Nov 1983. [Online]. Available: http://tools.ietf.org/html/rfc883. [Accessed 24 July 2011]. |

1. Git is a distributed version control system developed by father of Linux – Linus Torvalds, used for manages development of Linux kernel originally, now also managing some large scale projects [16] [17]. [↑](#footnote-ref-1)
2. Domain name, an easy to remember name identifies a computer, which have mapping relation (records) to IP addresses, allows user to access various service on a server by only use its domain name, such as http and ftp (A record), mail service (mx record), etc. [22] [23]. [↑](#footnote-ref-2)
3. Private IP address, is IP address in range of pre-reserved network address space. It usually use in local area networks, which can only be accessed by computer in the same network [18] [19]. [↑](#footnote-ref-3)
4. JSP, JavaServer Pages, a technology uses Java language creates dynamic web content [15]. [↑](#footnote-ref-4)
5. ASP [↑](#footnote-ref-5)
6. ASP.net [↑](#footnote-ref-6)
7. CGI [↑](#footnote-ref-7)
8. Servlet [↑](#footnote-ref-8)
9. Apache [↑](#footnote-ref-9)
10. Nginx [↑](#footnote-ref-10)
11. IIS [↑](#footnote-ref-11)