Sistemas de control de versiones

Sistema de control de versiones. Software que administra el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como código fuente, documentación o ficheros de configuración. [1]

Sistemas de control de versiones

Contenido

Características

Un sistema de control de versiones debe proporcionar:

- Mecanismo de almacenamiento de los elementos que deba gestionar.
- Posibilidad de realizar cambios sobre los elementos almacenados.
- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos.


Sistema de control de versiones

Clasificación

Los sistemas de control de versiones se pueden clasifica en 2 grandes grupos:

Centralizados

En un sistema de control de versiones centralizado todos nuestros fuentes y sus versiones están almacenados en un único directorio (llamado repositorio de fuentes) de un ordenador (un servidor). Todos los desarrolladores que quieran trabajar con esos fuentes, deben pedirle al sistema de control de versiones una copia local para trabajar. En ella realizan todos sus cambios y cuando están listos y funcionando, le dicen al sistema de control de versiones que guarde los fuentes modificados como una nueva versión.Algunos ejemplos son CVS y subversión.

Distribuidos

En un sistema de control de versiones distribuido no hay un repositorio central. Todos los desarrolladores tienen su propia copia del repositorio, con todas las versiones y toda la historia. Por supuesto, según van desarrollando y haciendo cambios, sus fuentes y versiones van siendo distintas unas de otras. Sin embargo, los sistemas de control de versiones distribuidos permiten que en cualquier momento dos desarrolladores cualesquiera puedan "sincronizar" sus repositorios. Si uno de los desarrolladores ha tocado determinados fuentes y el otro no, los modificados se convierten en la versión más moderna.Ejemplos:Git y Mercurial.

Control de versiones

Normalmente consiste en una copia maestra en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local. Esto permite compartir los cambios sobre un mismo conjunto de ficheros. Además, el repositorio guarda registro de los cambios realizados por cada usuario, y permite volver a un estado anterior en caso de necesidad. [2]

Ejemplos

- Guardar distintas copias de los ficheros nombrándolos adecuadamente.
- Hacer scripts para automatizar las copias.
- Usar un software específico para realizar el control de versiones.

Forma habitual de trabajo

- Mantener una copia en local y modificarla.Después actualizarla en el repositorio. Esto nos brinda grandes ventajas,ya que,no necesita acceso continuo al repositorio y asegurarse de que lo actualizado esté bien.

- Con algunos sistemas de control de versiones es posible trabajar directamente contra el repositorio.Esto aunque tiene una ventaja muy grande es que nos facilita la transparerencia de las versiones también

provoca como inconveniente el bloqueo de ficheros.

Funcionamiento

Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados.

Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial. Este proceso se suele conocer como checkout o desproteger. Para modificar la copia local existen dos semánticas básicas:

Exclusivos

Para poder realizar un cambio es necesario marcar en el repositorio el elemento que se desea modificar y el sistema se encargará de impedir que otro usuario pueda modificar dicho elemento.

Colaborativos

En el que cada usuario se descarga la copia, la modifica, y el sistema automáticamente combina las diversas modificaciones. El principal problema es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas. Además, esta semántica no es apropiada para ficheros binarios.

Procedimiento de uso habitual de un sistema de control de versiones

1. Descarga de ficheros inicial (Checkout)
2. Ciclo de trabajo habitual:

- Modificación de los ficheros
- Actualización de ficheros en local (Update)
- Resolución de conflictos (si los hay)
- Actualización de ficheros en repositorio (Commit).

Subversión como evolución de CVS

- Renombar, copiar y mover ficheros y directorios sin perdida del histórico.
- Commits atómicos.
- Implementación de tres tipos de acceso:
- Stand-alone
- Local
- Apache + webDAV
- Mejorado el sistema de permisos.
- Reducción del riesgo de vulnerabilidades por sus distintos RA (repository access).
- Integración con Project Software Manager (Por ejemplo: trac).

Clientes e integración con IDEs

1.Para Concurrent Version Control (CVS):

- Eclipse (integrado)
- Kdevelop (integrado)
- TortoriseCVS. [3]

2.WinCVS[4]

- Cervisia[5]

Para Subversión:

- TortoiseSVN [6]
- Plugin para Eclipse (subclipse)

Sistemas de Control de Versiones Libres

CVS

CVS[7] ha estado durante mucho tiempo, y muchos desarrolladores están ya familiarizados con él. En su día fue revolucionario: fue el primer sistema de control de versiones de código abierto con acceso a redes de área amplia para desarrolladores (que yo sepa), y el primero que ofreció ""checkouts"" anónimos de sólo lectura, los que dieron a los desarrolladores una manera fácil de implicarse en los proyectos. CVS sólo versiona ficheros, no directorios; ofrece ramificaciones, etiquetado, y un buen rendimiento en la parte del cliente, pero no maneja muy bien ficheros grandes ni ficheros binarios. Tampoco soporta cambios atómicos.

Subversion

Subversion[8] fue escrito ante todo para reemplazar a CVS—es decir, para acceder al control de versiones aproximadamente de la misma manera que CVS lo hace, pero sin los problemas o falta de utilidades que más frecuentemente molestan a los usuarios de CVS. Uno de los objetivos de Subversion es encontrar la transición a Subversion relativamente suave para la gente que ya está acostumbrada a CVS. Aquí no hay sitio para entrar en detalles sobre las características de Subversion; acceda a su sitio web para más información. [Descargo: Estoy implicado en el desarrollo de Subversion, y es el único de estos sistemas que uso habitualmente.]

SVK

Aunque se ha construido sobre Subversion, probablemente SVK[9] se parece más a algunos de los anteriores sistemas descentralizados que a Subversión. SVK soporta desarrollo distribuido, cambios locales, mezcla sofisticada de cambios, y la habilidad de ""reflejar/clonar"" árboles desde sistemas de control de versiones que no son SVK. Vea su sitio web para más detalles.

Mercurial

Mercurial[10] es un sistemas de control de versiones distribuido que ofrece, entre otras cosas, "una completa ""indexación cruzada"" de ficheros y conjutos de cambios; unosprocotolos de sincronización SSH y HTTP eficientes respecto al uso de CPU y ancho de banda; una fusión arbitraria entre ramas de desarrolladores; una interfaz web autónoma integrada; [portabilidad a] UNIX, MacOS X, y Windows" y más (la anterior lista de

características ha sido parafraseada del sitio web de Mercurial).

Mercurial

GIT

GIT[11] es un proyecto empezado por Linus Torvalds para manejar el arbol fuente del ""kernel"" de Linux. Al principio GIT se enfocó bastante en las necesidades del desarrollo del ""kernel"", pero se ha expandido más allá que eso y ahora es usado por otros proyectos aparte del ""kernel"" de Linux. Su página web dice que está "... diseñado para manejar proyectos muy grandes eficaz y velozmente; se usa sobre todo en varios proyectos de código abierto, entre los cuales el más notable es el ""kernel"" de Linux. GIT cae en la categoría de herramientas de administración de código abierto distribuído, similar al, por ejemplo, GNU Arch o Monotone (o bitKeeper en el mundo comercial). Cada directorio de trabajo de GIT es un repositorio completo con plenas capacidades de gestión de revisiones, sin depender del acceso a la red o de un servidor central."

Bazaar

Bazaar[12] está todavía en desarrollo. Será una implementación del protocolo GNU Arch, mantendrá compatibilidad con el procotolo GNU Arch a medida que evolucione, y trabajará con el proceso de la comunidad GNU Arch para cualquier cambio de protocolo que fuera requerido a favor del agrado del usuario.

Referencias

1. ↑ http://polaris.dit.upm.es/~rubentb/docs/subversion/TutorialSubversion/index.html
2. ↑ http://polaris.dit.upm.es/~rubentb/docs/subversion/TutorialSubversion/index.html
3. ↑ http://www.tortoisecvs.org
4. ↑ http://www.wincvs.org
5. ↑ http://www.kde.org/apps/cervisia
6. ↑ http://tortoisesvn.tigris.org
7. ↑ http://www.nongnu.org/cvs
8. ↑ http://subversion.tigris.org/
9. ↑ http://svk.elixus.org/
10. ↑ http://www.selenic.com/mercurial/
11. ↑ http://git.or.cz/
12. ↑ http://bazaar.canonical.com/

Fuentes

- Jourmoly (http://www.jourmoly.com.ar/debes-usar-un-sistema-de-control-de-versiones/)
- Sistema de control de versiones libres (http://producingoss.com/es/vc-systems.html)