# SISTEMAS DE CONTROL DE VERSIONES

Introducción

Los sistemas de control de versiones permiten gestionar los distintos cambios que se producen sobre un producto o una configuración del mismo durante el proceso de desarrollo del mismo.

En el ámbito informático estos sistemas permiten a los programadores que pertenecen a un proyecto centralizar y coordinar sus trabajos.

Los gestores de versiones guardan todos los ficheros del proyecto en un repositorio,

durante el desarrollo, de forma que en cualquier momento podemos deshacer los cambios hechos, volviendo a una versión anterior. Para ello las diferentes versiones se numeran (también llamadas revisiones) para identificarlas.

Un SVC permite:

* Crear copias de seguridad y restaurarlas.
* Sincronizar (mantener al día) a los desarrolladores respecto a la última versión de desarrollo.
* Deshacer cambios.
* Gestionar la autoría del código e histórico.

Funcionamiento básico

Los sistemas de control de versiones se basan en el modelo cliente-servidor donde existe un repositorio principal en el servidor que contiene toda la información gestionada.

Cada uno de los usuarios puede crearse una copia local en el cliente obteniendo información del repositorio pudiendo obtener la última versión o cualquiera de las almacenadas en el historial.

A continuación, el usuario puede modificar la copia. Existen dos técnicas para gestionar los cambios:

* **Exclusivos**: bloquear el elemento que va a ser modificado, previa notificación del mismo, para impedir que otro usuario pueda hacerlo a la vez. De esta forma se impide que aparezcan conflictos. Algunos sistemas funcionan con este sistema (SourceSafe) y otros permiten implementarlo (Subversión).
* **Colaborativos**: en el que cada usuario se descarga la copia, la modifica, y el sistema automáticamente combina las diversas modificaciones. El inconveniente de esta técnica es que pueden aparecer conflictos si las mismas líneas de código se modifican simultáneamente por varias personas. El sistema detecta los conflictos que deben ser solucionados manualmente. Subversion o Git permiten implementar este modo de funcionamiento.

Tras realizar la modificación es necesario actualizar el repositorio principal con los cambios realizados.

Clasificación

En función del almacenamiento del código los sistemas de control de versiones pueden ser:

* **Centralizados**: existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). Se simplifican las tareas administrativas, pero las decisiones importantes (como crear una nueva rama) necesitan el permiso del responsable. Ejemplos: CVS y Subversion.
* **Distribuidos**: Cada usuario tiene su propio repositorio. No es necesario tomar decisiones centralizadamente. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Ejemplos: Git y Mercurial.

Elementos básicos

*Repositorio* -> Almacén que guarda toda la información del proyecto y que habitualmente tiene estructura de árbol.

*Servidor* -> Máquina donde está alojado el repositorio.

*Working Copy/Working Set* -> Copia local donde el desarrollador trabaja.

*Trunk/Main/master* -> La rama principal de desarrollo.

Operaciones básicas

**Fetch/Pull**: Cuando hacemos pull de un proyecto, nos traemos todos los cambios existentes en el repositorio central a nuestro repositorio local.

**Commit**: Con esta operación llevamos los cambios de nuestro workspace al repositorio GIT local.

**Push**: Con esta operación llevamos los cambios de nuestro repositorio GIT local al repositorio GIT central.

**Revert/Rollback**: Deshacer un cambio y volver a la versión anterior.