UNIVERSIDAD GALILEO

Técnico en Desarrollo de Software

Profesor: [Erwin Eugenio Gonzalez Gonzalez](https://ges.galileo.edu/dotlrn/community-member?user%5fid=437440103)

Tarea de Puntos Extra #1 Actividad #3

Tipos de Controladores de Versiones

Byron Othoniel Burgos Paniagua

IDE-9713763

FECHA: 8 de febrero DE 2021

¿Qué es un sistema de control de versiones?

Un sistema de control de versiones (o VCS, por sus siglas en inglés), también conocido como sistema de control de revisiones o de fuentes, es una herramienta de software que monitoriza y gestiona cambios en un sistema de archivos. Asimismo, un VCS ofrece herramientas de colaboración para compartir e integrar dichos cambios en otros usuarios del VCS. Al operar al nivel del sistema de archivos, un VCS monitorizará las acciones de adición, eliminación y modificación aplicadas a archivos y directorios.

Clasificación de los SCV

Podemos clasificar los sistemas de control de versiones según la arquitectura para almacenar la información en locales, centralizados o distribuidos.

**Locales**: La información se guarda en un ordenador o repositorio local con lo que no sirve para trabajar en forma colaborativa. Ejemplo: RCS (Revision Control System), esta es una de las herramientas de control de versiones más popular fue un sistema llamado RCS, que todavía podemos encontrar en muchas de las computadoras actuales. Incluso el famoso sistema operativo Mac OS X incluye el comando rcs cuando instalas las herramientas de desarrollo. Esta herramienta funciona guardando conjuntos de parches (es decir, las diferencias entre archivos) en un formato especial en disco, y es capaz de recrear cómo era un archivo en cualquier momento a partir de dichos parches.



**Centralizados o CVCS (Centralized Version Control System):** El siguiente gran problema con el que se encuentran las personas es que necesitan colaborar con desarrolladores en otros sistemas. Los sistemas de Control de Versiones Centralizados (CVCS por sus siglas en inglés) fueron desarrollados para solucionar este problema. Estos sistemas, como CVS, Subversion y Perforce, tienen un único servidor que contiene todos los archivos versionados y varios clientes que descargan los archivos desde ese lugar central. Este ha sido el estándar para el control de versiones por muchos años. La información se guarda en un servidor dentro de un repositorio centralizado. Existe un usuario o usuarios responsables con capacidad de realizar tareas administrativas a cambio de reducir flexibilidad, necesitan la aprobación del responsable para realizar acciones, como crear una rama nueva. Ejemplos: Subversión y CVS.

Esta configuración ofrece muchas ventajas, especialmente frente a VCS locales. Por ejemplo, todas las personas saben hasta cierto punto en qué están trabajando los otros colaboradores del proyecto. Los administradores tienen control detallado sobre qué puede hacer cada usuario, y es mucho más fácil administrar un CVCS que tener que lidiar con bases de datos locales en cada cliente.

Sin embargo, esta configuración también tiene serias desventajas. La más obvia es el punto único de fallo que representa el servidor centralizado. Si ese servidor se cae durante una hora, entonces durante esa hora nadie podrá colaborar o guardar cambios en archivos en los que hayan estado trabajando. Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han realizado copias de seguridad adecuadamente, se perderá toda la información del proyecto, con excepción de las copias instantáneas que las personas tengan en sus máquinas locales. Los VCS locales sufren de este mismo problema: Cuando tienes toda la historia del proyecto en un mismo lugar, te arriesgas a perderlo todo.



**Distribuidos o DVCS (Distributed Version Control System):** Los sistemas de Control de Versiones Distribuidos (DVCS por sus siglas en inglés) ofrecen soluciones para los problemas que han sido mencionados. En un DVCS (como Git, Mercurial, Bazaar o Darcs), los clientes no solo descargan la última copia instantánea de los archivos, sino que se replica completamente el repositorio. De esta manera, si un servidor deja de funcionar y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios disponibles en los clientes puede ser copiado al servidor con el fin de restaurarlo. Cada clon es realmente una copia completa de todos los datos.



## Comparación de software de control de versiones

A continuación, tenemos una descripción general y una comparación de las opciones populares de VCS. Las principales observaciones a partir de estas comparaciones son que las opciones de VCS que usan un modelo cliente-servidor no están admitidas por las soluciones modernas de alojamiento en VCS como Bitbucket. El sector de VCS ha avanzado hacia un modelo distribuido.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

Fuentes

<https://blog.irontec.com/historia-informal-de-los-sistemas-de-control-de-versiones/>

<https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>

<https://www.profesordeinformatica.com/descargas/controldeversiones.pdf>

<https://bitbucket.org/product/es/version-control-software>