Sistema de control de versiones

Leonel Alexis García Juárez
alexrez2000@gmail.com
Universidad de la Sierra Sur

2022/05/09

1. Introducción

El control de versiones hoy en día es uno de los más importantes sistemas que se debe ser capaz de manejar, puesto que hoy en día para lograr un desarrollo de software seguro seguro y de calidad necesitamos pasar por diferentes etapas de nuestro proyecto en donde cada paso que implementemos en nuestro proyecto podría ser de gran cambio para el usuario o cliente final.
El control de versiones hoy en día se a convertido en un sistema usado por muchas empresas ya que existen ocasiones en las cuales se necesita regresar el software a una versión anterior o se necesita conocer los distintos cambios que se han realizado.A continuación se mostraran algunos datos importantes sobre el sistema de control de versiones, tomando en cuenta lo datos proporcionados por distintos autores.

2. Sistema de control de versiones

El sistema de control de versiones se puede definir como el software responsable de mantener el control Para diversas versiones o cambios realizados durante el desarrollo del software, por lo que es posible gestionar y llevar un registro periódico de los avances que se pueden realizar en el mismo. Generalmente, este tipo de software se utiliza en el proceso de gestión de código confiando el trabajo de múltiples desarrolladores en un mismo producto, guardando una copia de cada cambio realizado y asegurando así la integridad del equipo del proyecto.

3. Características

Cuando se trata de control de versiones, piensa en el código El origen de la aplicación, pero debe entenderse que el control de versiones debe cubrir no solo este aspecto, sino también otros aspectos como los requisitos, los esquemas y la documentación. Decidir cuándo se alcanzará una versión es importante y debe seguir las políticas de desarrollo de versiones de la empresa. No se creará una versión final sin mover primero estos productos a otros tipos de versión. En este sentido, un buen mecanismo para establecer versiones es definir tipos de versiones, como alfa, beta y final.
Características imprescindibles de un buen control de versiones: información del proyecto cuando se decida configurar el pasador de liberación.

• Otra característica muy importante que surge de otra: las capacidades que

• El sistema de control de versiones primero debe habilitar al desarrollador Establece durante cuánto tiempo desea que el sistema sepa que tiene una versión. obviamente este punto debe ir acompañado de un almacenamiento fiable de toda la debe tener un desarrollador.Un sistema para saber la diferencia entre la versión actual y la última versión,Para ver si es hora de un nuevo lanzamiento o nuevos esfuerzos no tiene nada que ver con el lanzamiento de una nueva versión.

4. Tipos de sistemas

Sistema Control de Versiones Centralizados

Como su nombre lo indica, este tipo de Sistema de control almacena la información en un servidor centralizado donde se encuentra el proyecto y cada una de sus modificaciones, los diferentes desarro- lladores deberán entrar al servidor, descargar una copia en su equipo, realizar las actualizaciones que hubiera lugar y, subir esta nueva versión para llevar el control de los procesos

El control de versiones centralizado (SCVC) utiliza un servidor central para almacenar todos los archivos y permitir la colaboración en equipo. Se ejecuta en un único repositorio al que pueden acceder los usuarios desde un servidor central.

El repositorio apunta a un servidor central que está directamente conectado a cada estación de trabajo del desarrollador. Todos los programadores pueden actualizar (refrescar) sus estaciones de trabajo con los datos del repositorio o realizar cambios en ellos (commit). Cada operación se realiza directamente en el repositorio.

Sistema Control de Versiones Distribuidos

Permite que cada integrante pueda realizar el proceso de manera local e independiente para ello es necesario clonar el repositorio del proyecto en el equipo local, de esta manera, se permite la gene- ración de versiones independientes en cada desarrollador y que, cuando sea definido podrá sincronizar su información en el servidor. Funcionalmente permite la fragmentación en ramas de las diferentes actividades y que, pueden facilitar el avance de diferentes avances del proyecto.

Los sistemas de control de versiones distribuidas (DCCS) no dependen necesariamente de un servidor central para almacenar versiones de archivos de proyecto.

En SCVD, cada desarrollador tiene una copia local o una copia del repositorio principal. Esto significa que cada desarrollador mantiene su propio repositorio local que contiene todos los archivos y metadatos contenidos en el repositorio principal. Cualquiera puede trabajar con su repositorio local sin interferencias.

5. Entornos Gráficos

Mercurial

Posee un funcionamiento al igual que Git, este Sistema Control de Versiones, guarda una copia de los archivos e historial del proyecto de manera local, pero, no se conecta directamente con el Repositorio Origen,

este proceso debe ser llevado a cabo por el desarrollador, sin embargo, cuenta con un aplicativo WEB que facilita las siguientes funciones:

• Navegación de la composición estructural del proyecto

• Visualización del antecedentes y cambios

• Expansión de archivos y directorios(

Características

Distribuido. Su control de versiones no requiere un servidor central a diferencia de otros sistemas como Subversion.

Rápido. Su implementación y estructura de datos pretenden agilizar el funcionamiento del sistema. Esto le permite marcar la diferencia entre vistas o retroceder en el tiempo en segundos. Es ideal para grandes proyectos como OpenJKD o NetBeans.

Diversidad. Está escrito en Python y C, y está disponible para la mayoría de los sistemas. Se puede ampliar. La funcionalidad del programa se puede ampliar con extensiones, tanto oficiales como escritas por terceros, todas escritas en Python.

Fácil. Mercurial está diseñado para ser fácil de usar y requiere muy poca investigación. Por ejemplo, si usamos usuarios de Subversion (svn), encontraremos muchas similitudes en su uso. Siempre podemos consultar su guía rápida (en inglés)

textbfCódigo fuente abierto. El producto sigue la filosofía de desarrollo de código abierto y es software libre porque utiliza la licencia GNU GPLv2.

GIT

Git generará un lis- tado de cada uno de los archivos modificados, los cuales eran seleccionados para crear la nueva versión; Después de ser confirmados los cambios que se han almacenados en el “área de preparación” y en el Directorio Git, generando de esta manera una modificación en su repositorio local. Al ser trabajado paralelamente por varios desarrolladores, Git realiza la función Merge para compactar las diferentes versiones creadas por cada uno, esto teniendo en cuenta que al usar este sistema es creada una rama maestra de la cual se desprenderán muchas otras ramas que tendrán incluidas todas las historias del proyecto. Una rama es una abstracción que permite trabajar de forma paralela sobre un mismo pro- yecto, esto sin afectar el resto de proyecto.

Git fue diseñado con la más alta prioridad en mantener la integridad del código fuente administrado. El contenido de los archivos y las relaciones reales entre archivos, directorios, versiones, etiquetas, confirmaciones y todos los objetos en un repositorio de Git están protegidos por un algoritmo hash seguro cifrado llamado ”SHA1”. Esto protege su código y el historial de cambios de cambios accidentales y maliciosos, y garantiza que se pueda rastrear todo el historial.

6. Conclusión

Conocer las distintas manerae en que operan los sistemas de control de versiones así como sus características que los integran, permiten a los desarrolladores entender de manera mas clara y precisa el software que se debe de usar de acuerdo a las especificaciones del proyecto que se desea hacer.Es por eso que con los usuarios de cualquier otro sistema, los sistemas de control de ver-siones deben permitir manejar niveles de usuarios; aunque globalmente todos sean desarrolladores de software y de sistemas, es bueno que el sistema de control de versiones permita la gestión de usuarios en el equipo de desarrollo basado en polıticas de seguri-dad y niveles de permisos.

7. Bibliografía

Referencias

[1] . Borrell, G. (2006). Control de versiones. Revista guillen Borrell nogueras.

[2] . Atlassian. (z.d.). Qué es Git: conviértete en todo un experto en Git con esta guía. Geraadpleegd op 2 mei 2022, van https://www.atlassian.com/es/git/tutorials/what-is-git

[3] (2020, 5 februari). ¿Por qué deberías usar un sistema de control de versiones si eres desarrollador? Opentix: Partner Partner Platinum Sage. Geraadpleegd op 2 mei 2022, van https://www.opentix.es/sistema-de-control-de-versiones/