Sistema de control de versiones

Luis Fernando Vásquez Fabián

Universidad de la Sierra Sur

9 de mayo de 2022

1. Introducción

La aplicación de métodos sistemáticos para implementar ciclos de vida para desarrollo de software seguro, abarca los procesos de creación de software de calidad, dando al cliente la fiabilidad de contar con un Sistema de información seguro, que desde el primer instante del planteamiento ha contado con estrategias, procesos y prácticas que garantizaran el producto. El desarrollo de un repositorio versionador que cuente con cada paso, o actividad que puede ser implementado en cada una de las fases del proyecto de desarrollo, facilita a los programadores herramientas que permitan minimizar las amenazas que se presenten Este ensayo se enfoca desde el punto de vista de la seguridad en procesos de desarrollo de software, una guía que permita fortalecer y adoptar una orientación crítica respecto al mejoramiento de la calidad y la eliminación de vulnerabilidades frente a ataques que puedan poner en riesgo la integridad del Software desarrollado, mediante la construcción de un software versionador

2. Sistema de control de versiones

Un Sistema Control de Versiones puede definirse como un Software encargado de llevar el control de las diferentes versiones o modificaciones realizadas en el desarrollo de software, en este, es posible administrar y llevar el registro sistemático de los avances que se puedan tener en el mismo. Generalmente este tipo de software es utilizado en el proceso de gestión de código dado al trabajo de varios desarrolladores en un mismo producto, conservando una copia de cada uno de los cambios realizados y garantizando de esta manera el trabajo integrado del equipo del proyecto. Dentro de las ventajas encontradas en el uso de estos sistemas tenemos:

- Reconstrucción de archivos o versiones anteriores cuando se requieran.

- Bitácora de modificaciones y cambios.

- Registro de todo tipo de gestión en el código fuente.

- Recuperación de versiones antiguas desarmando de alguna manera los cambios realizados en la última entrega o modificación.(Borrell,)

3. Características que debe tener un buen control de versiones

Antes de mostrar algunas de las características más importantes que debe tener un buen control de versiones, es necesario pensar en algunas cuestiones que ayuden a comprender el tema y la importancia del control de versiones. Cuando se habla de control de versiones, se piensa en el código fuente de las aplicaciones, pero es necesario comprender que el control de versiones no solamente debe abarcar este aspecto sino que debe cubrir otros como los requisitos, los diagramas, la documentación.

Establecer cuándo se llega a una versión es algo importante y debe obedecer a políticas institucionales de desarrollo de versiones. No se deben establecer versiones definitivas sin antes haber pasado dichos productos por otro tipo de versiones. En este sentido, un buen mecanismo para establecer versiones es definir tipos de versiones, por ejemplo: alfa, beta y finales.

Veamos entonces algunas de las características que debe tener un buen control de versiones:

• Un sistema de control de versiones debe, primero que todo, darle al desarrollador la posibilidad de establecer el momento en que desea indicarle al sistema que tiene una versión. Obviamente este momento tiene que estar acompañado de un almacenamiento confiable de toda la información del proyecto en el momento de tomar la decisión de establecer la fijación de una versión.
• De otra característica surge otra bien importante: la posibilidad que debe tener el desarrollador de sistemas de conocer cuáles son las diferencias entre lo que se tiene actualmente y la última versión, con el fin de saber si es el momento preciso para una nueva versión, o si todavía los esfuerzos hechos no conlleva el lanzamiento de una nueva versión
.(Trujillo Silva y cols.,)

4. Tipos de sistemas de control de versiones

4.1. 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 desarrolladores 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.
Posibles desventajas:

• Conexión continua a una red para tener acceso al servidor central.

• Posibles retrasos en los tiempos en caso de presentarse caída de red o del servidor.

Dentro de este tipo de Sistemas Control de Versiones se encuentran:

• CVS (Concurrent Version Systems): Favorece y facilita el trabajo colaborativo entre diferentes desarrolladores mediante la arquitectura Cliente-Servidor. Son usados principalmente en el control de cambios al código fuente de un proyecto, sin embargo, su uso incluye al versionador de documentos de toda índole que puedan tener diferentes modificaciones. En cuanto a su funcionamiento, en caso de concurrir dos desarrolladores en el trabajo de un mismo código, el CVS permite el proceso de actualización generando una nueva versión del código fuente, esto, en caso de no presentar conflicto al intentar tocar una misma línea por ambos actores, si este es el caso, se indicará al usuario y no será generada

la nueva versión. Adicional, el sistema ofrece un proceso de log de proceso en el cual se registrarán los datos básicos de la modificación realizada como fechas y usuarios usados.

• Subvesion: Sistema Control de Versiones de código abierto cuya utilización se extiende desde documentos hasta estructura y diferentes versiones de directorios, permitiendo su repositorio. Este sistema facilita el proceso de trabajo en paralelo de diferentes desarrolladores, cuya actividad final será la integración conformando una última versión usando el proceso "Copiar – Modificar – Unificar", de esta manera, evita el bloqueo de usuarios al intentar ejecutar cambios en la misma línea de código. Cada integrante deberá crear una copia y de esta manera, trabajará en un documento personalizado que por medio de "Subversión" será compactada en la versión final.(Trujillo Silva y cols.,)

4.2. 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 generación 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.

• No es necesario trabajar por red, ya que cada uno podrá trabajar en una copia local.

• Permiten el envío de versiones de un desarrollador a otro sin necesidad de pasar por el servidor.

• Uno de los desarrolladores será el encargado de recopilar la información y enviarla al repositorio remoto encontrado en el servidor.(Wanumen Silva,)

5. Entornos gráficos para el sistema de control de versiones

5.1. Sistema Control de Versiones GIT

Desarrollado por Linux, funcionalmente, el desarrollador deberá descargar una versión local del proyecto, en la cual realizará las modificaciones que hubiera lugar, posteriormente Git generará un listado 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 proyecto, esto sin afectar el resto de proyecto.
Está compuesta por 3 unidades:

• Directorio de Git: Repositorio de objetos en los cuales se ha desarrollado modificación, contiene así mismo el historial de cambios.

• Directorio de trabajo: Almacena los archivos iniciales sobre los cuales se realizarán los cambios.

• Área de preparación: Contiene la información de los archivos modificados que serán enviados al ser confirmados por el desarrollador.(Ruiz-Bertol y Zarazaga-Soria,)

5.2. Mercurial

Con funcionamiento similar al 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(Ruiz-Bertol y Zarazaga-Soria,)

5.3. Conclusiones

Las características de desarrollo en parale-lo, como el branching, permiten que los equi-pos separen el proceso de desarrollo en pro-yectos y archivos paralelos, creando copias idénticas que heredan toda la documenta-ción de versiones, pero a las que se les pue-de dar seguimiento como proyectos nuevos e individuales . Esto obviamente debe estar soportado por una alta tecnología que permita reconciliar diferencias entre diferen-tes versiones del mismo archivo y ver cómo se comportaría una determinada versión al ser incluida dentro del proyecto, pero esto bajo la premisa de evitar al máximo la pérdida potencial de cambios valiosos.Al igual 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 políticas de seguri-dad y niveles de permisos.

Referencias

Borrell, G. (2006). Control de versiones. *Revista guillen Borrell nogueras*.

Ruiz-Bertol, F. J., Zarazaga-Soria, F. J. (2007). El control de versiones en el aprendizaje de la ingeniería informática: Un enfoque práctico. *Actas de las XIII Jornadas de Enseñanza Universitaria de Informática*.

Trujillo Silva, D. M., y cols. (2016). Sistema de control de versiones para el desarrollo de software seguro.

Wanumen Silva, L. F. (s.f.). Los sistemas de control de versiones.