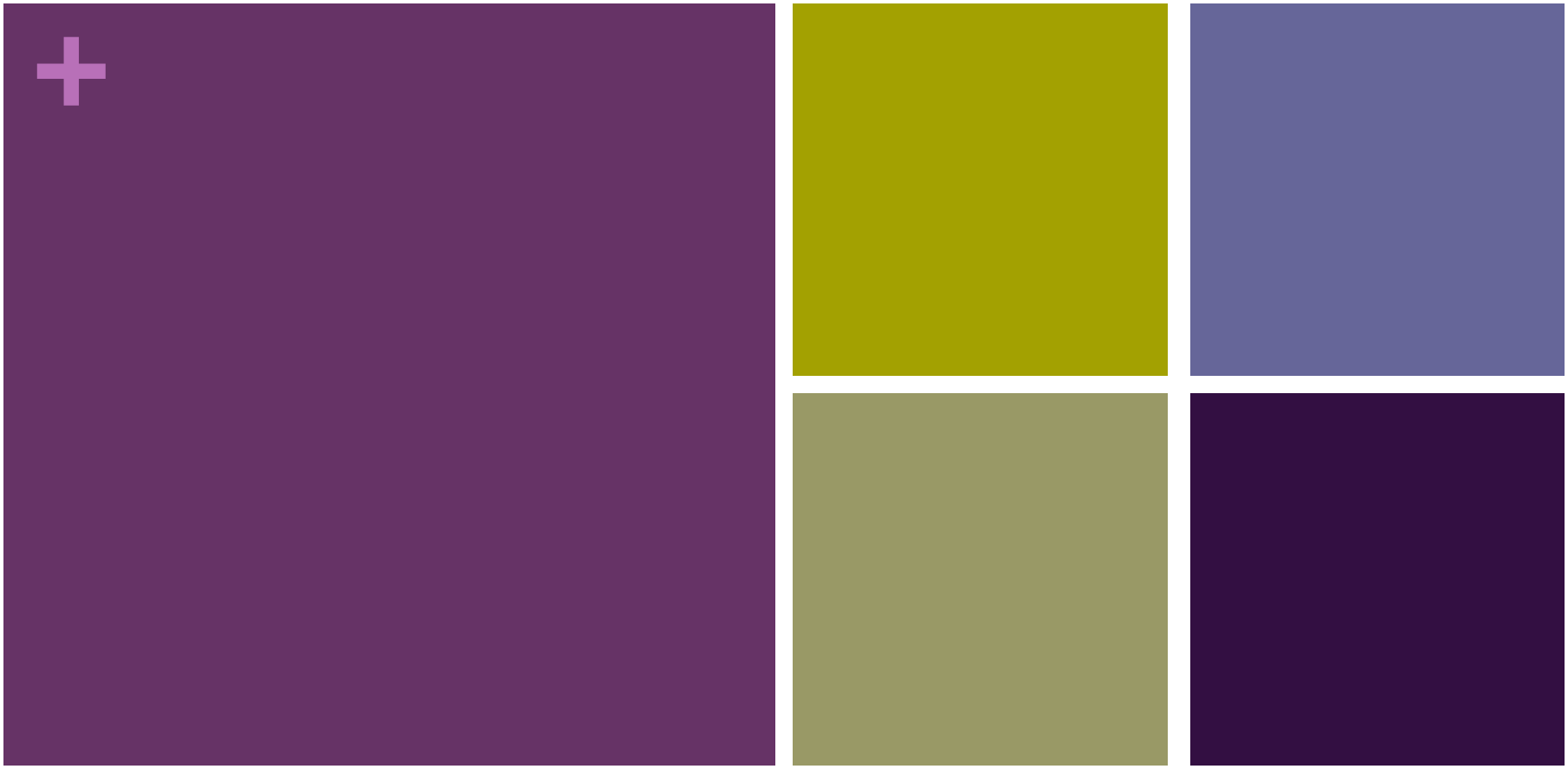


Sistemas Distribuidos
Control de versiones

Rodrigo Santamaría

Control de versiones

Introducción

Sistemas centralizados: CVS/SVN

Sistemas distribuidos: Git

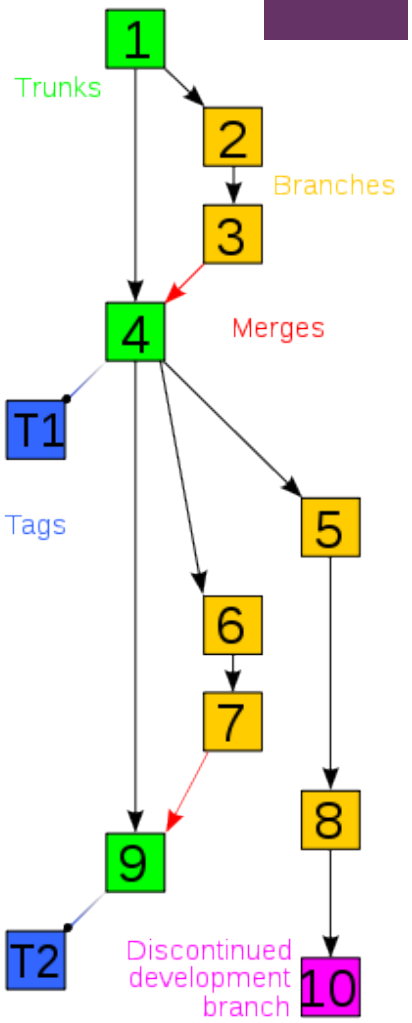
Introducción

- Control de versiones
  - Gestión de los cambios en documentos, programas, páginas web o cualquier otra colección de información
- Sistema de control de versiones (CVS)
  - Aplicaciones para control de versiones
  - Independientes o integradas
  - Locales o distribuidas
  - En equipos de desarrollo de software, un CVS es fundamental
    - Mantener varias versiones a la vez en desarrollo
    - Fusionar las versiones en una única versión principal

Introducción

Conceptos básicos

- El desarrollo de aplicaciones implica:
  - Solución de errores (bugs) → estabilidad
  - Adición de nuevas funcionalidades → última versión
- Tronco (**trunk**)
  - Versión base de un proyecto
  - Dependiendo de la política de actualización, será la más estable, o la última versión, o un punto intermedio
- Rama (**branch**)
  - Versión(es) en la que se están solucionando errores o añadiendo nuevas funcionalidades
  - Cuando la rama es estable (errores solucionados, funcionalidades añadidas) se fusiona (**merge**) con el tronco



Introducción

Otros conceptos

- **Change**: modificación del documento (granularidad variable)
- **Commit**: fusión de los cambios con la copia del repositorio
  - También llamado checkin
  - **Import**: primer commit (todavía no hay copia en el repositorio)
  - **Head**: versión resultante del commit más reciente
- **Checkout**: creación de una copia local (*working copy*) a partir de la copia del repositorio
  - **Export**: checkout que no extrae los metadatos del sistema de versiones
- **Tag**: identificación textual para una determinada copia

Introducción

Estrategias

- **Commit atómico**: la copia de una rama al tronco debe garantizar la estabilidad del sistema si es interrumpida
- **Bloqueo de archivos**: el método más simple para evitar accesos concurrentes
  - Cuando un desarrollador hace checkout de un archivo, nadie puede modificarlo salvo él, hasta que haga un commit
  - Muy seguro, pero poco flexible
- **Fusión de versiones**: permite editar el mismo archivo concurrentemente
  - El primer commit siempre tiene éxito
  - Los siguientes commit tendrán que preservar los cambios de los anteriores y añadir los nuevos

Control de versiones

- Introducción
- Sistemas centralizados: CVS/SVN
- Sistemas distribuidos: Git

Sistemas centralizados

- Se basan en un repositorio central
  - Todos los usuarios extraen del repositorio central (check out)
  - Todos los usuarios mandan sus cambios al repositorio central (commit)
- Se pueden compartir copias locales sin pasar por el repositorio central
  - Se comparten como **patches** (descripción de los cambios de la copia central respecto a la copia local)
    - Por ejemplo, la salida de una orden *diff*

Sistemas centralizados

Ventajas y desventajas

- Consistente
  - La versión principal es siempre la misma
  - Cualquier cambio debe pasar por el repositorio central
    - Los *patches* entre usuarios no tienen repercusión
- Información centralizada
  - No hay réplicas de la versión principal
  - Las operaciones con el repositorio pueden ser lentas y restrictivas
    - Commit, comprobación de la historia de cambios, etc.

Sistemas centralizados

CVS (Current Version System)

- Sistema abierto y gratuito
  - Muy usado en la comunidad de software libre
- Desarrollado en los 80, aunque existían sistemas similares en los 70
- Algunas de sus características han sido muy criticadas
  - No asegura commit atómicos
  - Problemas con nombres de archivo en Unicode o UTF-8
  - Las operaciones de branching son costosas

Sistemas centralizados

SVN (Subversion)

- Desarrollado por Apache en 2000
- Es el sistema centralizado más popular
  - Distribuido bajo una licencia de software libre
- Sucesor (casi totalmente compatible) de CVS
  - Los commit son atómicos
  - Soporte para distintas codificaciones
  - Branching más rápido

Sistemas centralizados

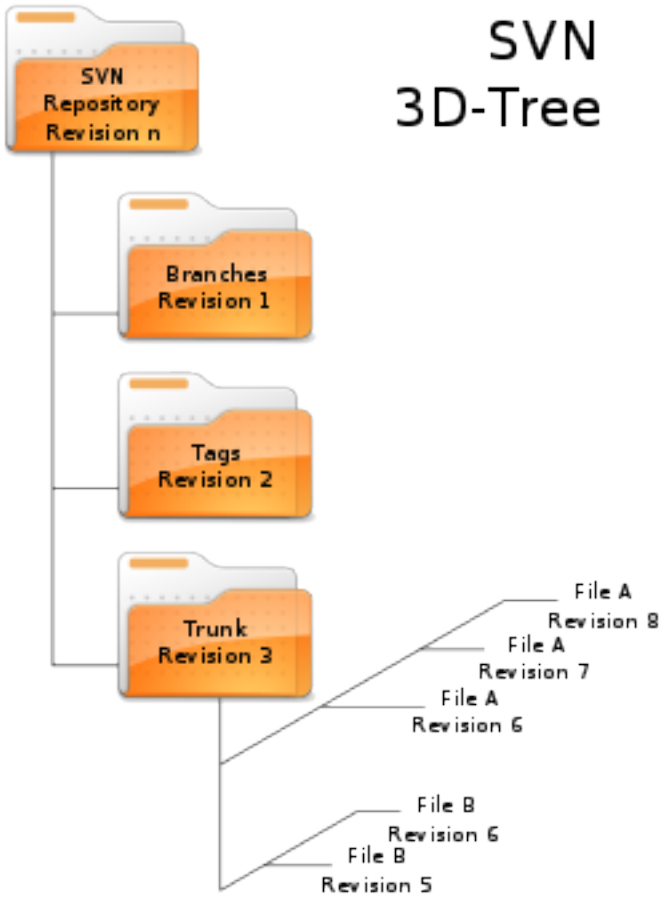
SVN: capas

- SVN cuenta con varias bibliotecas organizadas en capas
  - **Fs**: capa más baja, implementa el almacenamiento de datos
  - **Repos**: añade funcionalidades adicionales al sistema de archivos de Fs.
    - Repos+Fs constituyen la "interfaz del sistema de archivos"
  - **mod_dav_svn**: acceso a WebDAV, una extensión de HTTP para facilitar la colaboración en la edición y administración de documentos almacenados en servidores en la red.
  - **Ra**: acceso al repositorio
    - local (file://path)
    - WebDAV (http://host/path o https://host/path)
    - SVN (svn://host/path/ o svn+ssh://host/path)
  - **Client, Wc**: capa más alta, abstrae el acceso al repositorio y añade tareas de usuario como autenticación o comparación de versiones
    - Wc es la biblioteca utilizada por las implementaciones de SVN

Sistemas centralizados

SVN: sistema de archivos

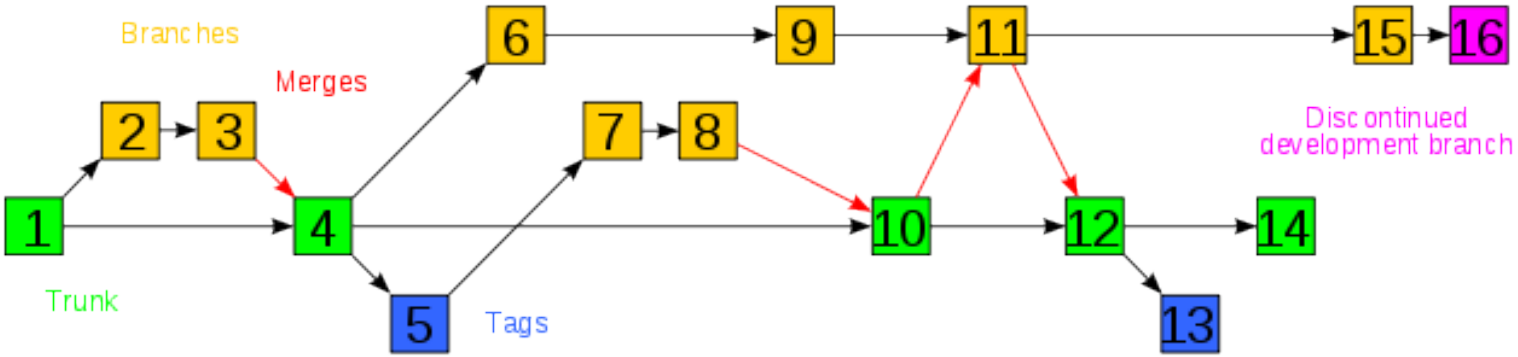
- Sistema de archivos ”tridimensional”
  - Dimensiones de la ruta (ancho y alto)
    - Contiene la estructura de archivos (tronco, ramas y tags)
  - Dimensión de la revisión (profundo)
    - Cambios en la estructura en la revisión
- Estructura muy compacta
  - Cada revisión sólo almacena cambios



Sistemas centralizados

SVN: branching y tagging

- Branching
  - Cuando se crea una rama no se crea una nueva estructura de archivos, si no que internamente se enlaza al tronco y se van anotando los cambios
  - Rápido y compacto
  - Se pueden hacer fusiones con el tronco o entre ramas
- Tagging
  - Etiquetas en el tronco para marcar hitos (p. ej. versiones estables)



Sistemas centralizados

SVN: limitaciones y problemas

- Problemas con cambios de nombre en archivos y directorios
  - Confusión con versiones anteriores
    - Especialmente si se renombran y mueven en el mismo commit
- Problemas con el borrado de archivos o directorios
  - Pueden hacerse persistentes y no borrarse adecuadamente
- Almacenamiento de copias locales
  - Los metadatos del control de versiones (.svn) pueden corromperse fácilmente por errores del usuario
- Pequeños problemas de codificación de acentos
  - Acento en el tronco (Linux) + checkout en MacOs

Sistemas centralizados

SVN: implementaciones

- TortoiseSVN: incrusta SVN en el sistema de archivos del SO
- Versiones integradas para IDEs
  - P. ej. Subclipse para Eclipse
- Versiones integradas en hosting de software de código abierto
  - Google Code, SourceForge, etc.
- Una gran variedad de versiones, integradas o independientes
  - http://en.wikipedia.org/wiki/Comparison_of_Subversion_clients

Control de versiones

Introducción

Sistemas centralizados: CVS/SVN

Sistemas distribuidos: Git

Sistemas distribuidos

- En un sistema distribuido de control de versiones (DVCS)
  - Cada usuario tiene una copia completa del proyecto
  - Todas las copias tienen el mismo nivel, no hay repositorio central
  - La divergencia entre versiones (fork) se ve como algo positivo
  - Los repositorios son mucho más pequeños

- La primera generación de DVCS surge en 2000, con Arch y Monotone, muy cercana al versionado de Linux
  - Continúan con BitKeeper y otros
  - En 2007, aparece Git

Sistemas distribuidos

Git

- Desarrollado inicialmente por Linus Torvalds para el desarrollo del kernel de Linux

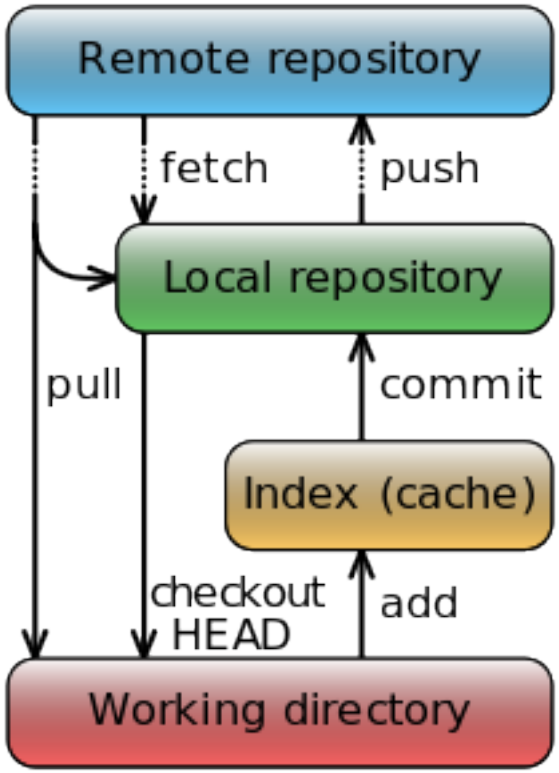
  git (*plural* gits)
  1. (*UK*, *slang*, *pejorative*) A contemptible person.
  2. (*UK*, *slang*, *pejorative*) A silly, incompetent, stupid, annoying or childish person.

- Ideas fundamentales de diseño
  - Tomar CVS como ejemplo de qué *no* hacer
  - Flujo de trabajo distribuido con repositorio *local*
  - Protección contra corrupción de versiones
  - Alto rendimiento

Sistemas distribuidos

Git: flujo de trabajo

- **Directorio de trabajo** es la copia local en la que desarrollamos código
  - `add` para indicar qué archivos han cambiado
  - `commit` para enviarlos al repositorio local
- **Repositorio local** actúa como "borrador" donde resolver posibles conflictos
  - `fetch` para comparar con repositorio remoto
  - `pull` para actualizar desde repositorio remoto
  - `push` para modificar repositorio remoto
- **Repositorio remoto** que puede ser modificado por distintos usuarios



Sistemas distribuidos

Git: estructura de datos

- **Árbol de trabajo**: los archivos y directorios bajo versionado
  - `.gitignore`: archivo de configuración con los patrones de nombres que Git debe considerar invisibles en cuanto a control de versiones
    - p. ej. *.o, *.class, bin/
- **Metadatos**
  - `.git`: directorio en el que se almacenan los datos de control
    - Objetos commit, tags, etc.
- **Configuración global**
  - `.gitconfig`: archivo con preferencias de usuario (nombre, e-mail…)
- A nivel de usuario, el entorno de trabajo parece un directorio normal

Sistemas distribuidos

Git: implementaciones

- Versiones integradas en el sistema de archivos del SO
  - TortoiseGit
- Versiones integradas para IDEs
  - P. ej. EGit para Eclipse
- Versiones integradas en software hosting
  - GoogleCode
  - SourceForge
  - GitHub
  - GitEnterprise
  - Bitbucket
  - …

Sistemas distribuidos

Git vs Mercurial

- Mercurial es otro CVS de filosofía similar a Git
  - Se empezaron a desarrollar a la vez, como respuesta a la retirada de la versión abierta de BitKeeper
- BitKeeper era el CVS utilizado por el kernel de Linux
  - Ahora su CVS es Git
  - Mercurial es el CVS de Mozilla, Facebook o el W3C
- Realmente Git y Mercurial son muy parecidos, aunque su comparación ha dado lugar a "batallas" similares a las de Vim vs Emacs
  - Mercurial: más simple y sencillo de aprender
  - Git: más flexible y potente

Sistemas distribuidos

Distribuido	Centralizado
Aproximación horizontal	Aproximación vertical
No hay referencia canónica	Referencia: repositorio central
Operaciones comunes rápidas - commit, ver historia, reversión de cambios	Operaciones más lentas - Acuerdo con repositorio central
Fusión basada en una red de confianza - Mérito o calidad	Fusión basada en una política rígida - Autorización

Sistemas distribuidos

Linus Torvalds sobre Distribuido vs Centralizado

- Charla de Linus Torvalds sobre Git en 2007 (25 primeros minutos)
  - http://www.youtube.com/watch?v=4XpnKHJAok8
  - Transcripción: http://web.archive.org/web/20110725034129/https://git.wiki.kernel.org/index.php/LinusTalk200705Transcript
- 3:00 → crítica a CVS y SVN, BitKeeper era el más útil para él
- 8:00-9:00 → "CVS users, you shouldn't be here, but in a mental institution [laughs]"
- 11:00 → "if you're not distributed, you're not worth using"
  - "if you don't get me what I put into, you're not worth using" (!)
- 13:20 → "no single place is more important than any other single place"
- 15:00 → importancia de las ramas, en CVS/SVN casi nadie las usa
- 19:00 → importancia de las versiones, cualquiera puede hacer commit sin afectar al resto
- 22:00 → la política suele ser no dejar hacer commit a no ser que la versión sea perfecta, y aún así con mucho cuidado y mediante un proceso tedioso

Sistemas distribuidos

Linus Torvals sobre CVS, SVN y BitKeeper

For the first 10 years of kernel maintenance, we literally used tarballs and patches, which is a much superior source control management system than CVS is, but I did end up using CVS for 7 years at a commercial company and I hate it with a passion. When I say I hate CVS with a passion, I have to also say that if there are any SVN (Subversion) users in the audience, you might want to leave. Because my hatred of CVS has meant that I see Subversion as being the most pointless project ever started. The slogan of Subversion for a while was "CVS done right", or something like that, and if you start with that kind of slogan, there's nowhere you can go. There is no way to do CVS right.

BitKeeper was not only the first source control system that I ever felt was worth using at all, it was also the source control system that taught me why there's a point to them, and how you actually can do things. So Git in many ways, even though from a technical angle it is very very different from BitKeeper (which was another design goal, because I wanted to make it clear that it wasn't a BitKeeper clone), a lot of the flows we use with Git come directly from the flows we learned from BitKeeper.

Referencias

- Control de versiones:
  - http://en.wikipedia.org/wiki/Version_control
- Subversion:
  - http://en.wikipedia.org/wiki/Apache_Subversion
- Git:
  - http://en.wikipedia.org/wiki/Git_(software)
  - http://www.youtube.com/watch?v=4XpnKHJAok8
  - http://wiki.eclipse.org/EGit/Git_For_Eclipse_Users

In case of fire

1. git commit

2. git push

3. leave building