



INGENIERÍA DEL SOFTWARE II

Práctica 3

Gestión de la Configuración

Control de versiones con Subversion

Univ. Cantabria – Fac. de Ciencias

Andrea Delgado



Objetivos

- Conocer la herramienta de control de versiones Subversion
 - Repositorio centralizado, usuarios y permisos
 - Clientes Subversion (ejemplos)
 - Subclipse plugin Eclipse
 - Tortoise SVN
- Realizar un caso de estudio simple de control de versiones y probar algunas de las características clave de Subversion



Contenido

- Introducción a Subversion
 - Qué es, para que sirve
 - Arquitectura de la solución
- Conceptos Fundamentales
 - Comandos principales
 - Check-in / check-out
 - Repositorio
 - Organización y dirección de acceso
 - Modelos de versionado
 - El problema de compartir archivos
 - Soluciones
 - Bloquear-Modificar-Desbloquear (lock-modify-unlock)
 - Copiar-Modificar-Combinar (copy-modify-merge)
- Uso de Subversion (cont.)
 - Vista de sincronización
 - Revisiones
 - Marcar versiones (tags)
 - Ramificar y combinar (branching & merging)

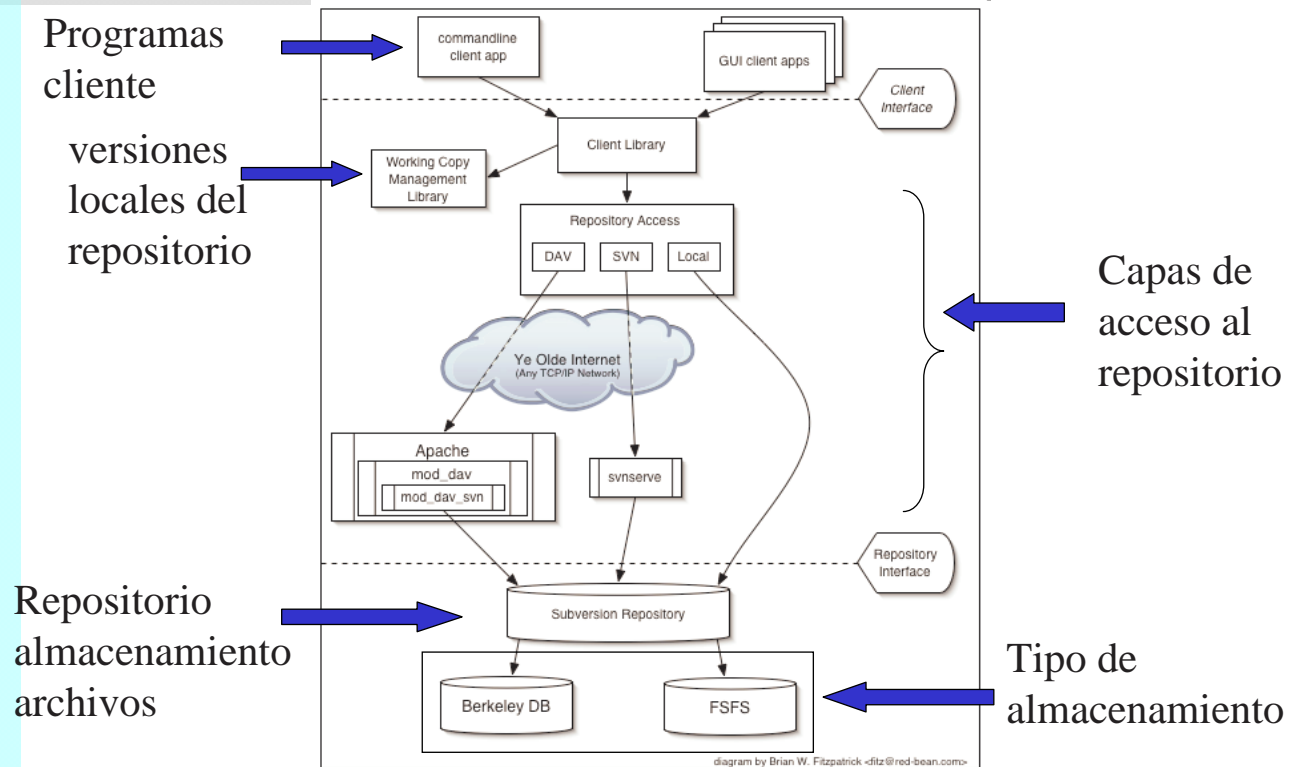


Introducción

- Qué es subversion (SVN)?
 - Sistema de control de versiones libre de código abierto
 - Incluye las principales características del CVS (concurrent version system) mejorando varios aspectos
- Para que sirve ?
 - Gestiona archivos y directorios, y los cambios realizados a éstos en el tiempo.



Introducción (2)



Andrea Delgado - IS2

P3.5



Introducción (3)

- Permite:
 - Múltiples desarrolladores modificando el mismo código
 - Almacenar una única copia maestra del código
 - Automatizar las actualizaciones entre versiones
 - Acceder a cualquier estado previo del código
- Previene:
 - Sobreescribir código
 - Pérdida de trabajo por falta de respaldo
- No sustituye la necesidad de gestionar ni la comunicación entre desarrolladores

Andrea Delgado - IS2

P3.6



Trabajo del laboratorio – inicio

- Ubicarse sentados por grupo para comunicarse mejor
 - Nombrar un responsable de GCS del grupo (2 minutos !!)
 - Abrir el eclipse java
 - Abrir la vista de SVN repository
 - Botón derecho: new repository location
- URL = <http://vm.ciencias.unican.es/svn/reposx/>

USUARIOS:

- grupo 1: user1 al user3
- grupo 2: user4 al user6
- grupo 3: user7 al user9
- grupo 4: user10 al user13
- grupo 5: user14 al user17



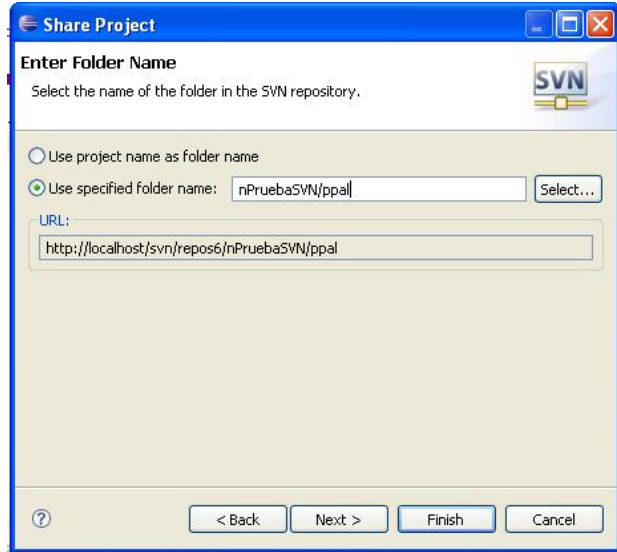
Trabajo del laboratorio – inicio (cont.)

- Solo el Responsable de GCS de cada grupo
 - En el eclipse vista proyectos
 - New java project nombre pruebaSVN
 - New java class HelloWorld default package
 - ```
public class HelloWorld {
 public static void main(String args[]) {
 System.out.println("Hello World!");
 }
}
```
  - Botón derecho sobre el proyecto team share project



## Trabajo del laboratorio – inicio (cont.)

- Solo el Responsable de GCS de cada grupo (cont.)
  - Elegir SVN y existing repository location
  - Use specific folder name poner pruebaSVN/ppal
  - Sobre el proyecto botón derecho commit
- Resto del grupo
  - En la vista repositorio hacer check-out del proyecto



## Comandos SVN – resumen

- Forma general de los comandos SVN:  
\$ svn [opciones] [subcomandos] [objetivo] (en cualquier orden)
- Checkout URL[@REV]... [PATH]
  - Recupera una copia de trabajo
- Commit [PATH...]
  - Confirma en el repositorio los cambios realizados
- Add [PATH...]
  - Agrega un nuevo archivo/directorio en el espacio de trabajo (solo registra que hay un nuevo archivo, para agregarlo al repositorio hay que hacer commit de la copia local)



## Comandos SVN – resumen (2)

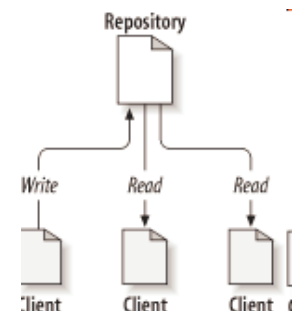
- Update [PATH...]
  - Sincroniza el directorio de trabajo con el repositorio
- Diff OLD-URL[@OLDREV] NEW-URL[@NEWREV]
  - Permite ver diferencias entre dos revisiones o path
- Log URL[@REV] [PATH...]
  - Muestra los mensajes de log de los commit en el repositorio para los archivos/path/versión seleccionados
- Status [PATH...]
  - Muestra información de estado en el directorio de trabajo, con opciones muestra del repositorio también



## Conceptos fundamentales - repositorio - acceso

- Almacenamiento central de datos, tipo sistema de archivos
  - jerarquía de árbol de directorios y archivos
- Múltiples clientes se conectan al repositorio para leer y escribir archivos.
  - Escribir → hacer que la información esté disponible para otros
  - Leer → recibir información que otros han hecho disponible
- El acceso es siempre mediante una URL

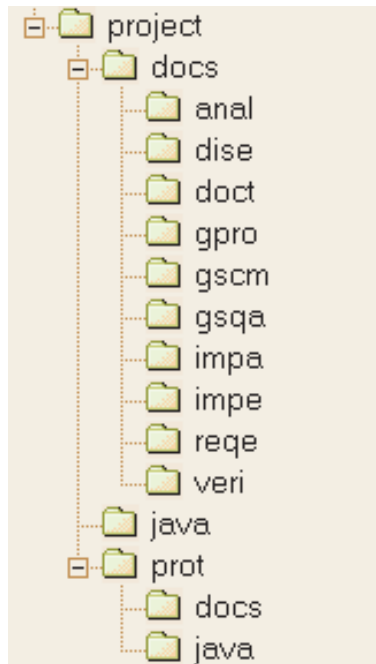
| Schema     | Access Method                                                |
|------------|--------------------------------------------------------------|
| file:///   | direct repository access (on local disk)                     |
| http://    | access via WebDAV protocol to Subversion-aware Apache server |
| https://   | same as http://, but with SSL encryption.                    |
| svn://     | access via custom protocol to an svnserve server             |
| svn+ssh:// | same as svn://, but through an SSH tunnel.                   |





# Conceptos fundamentales – repositorio - organización

## Grupo 1 - 2002



## Grupo 5 - 2002



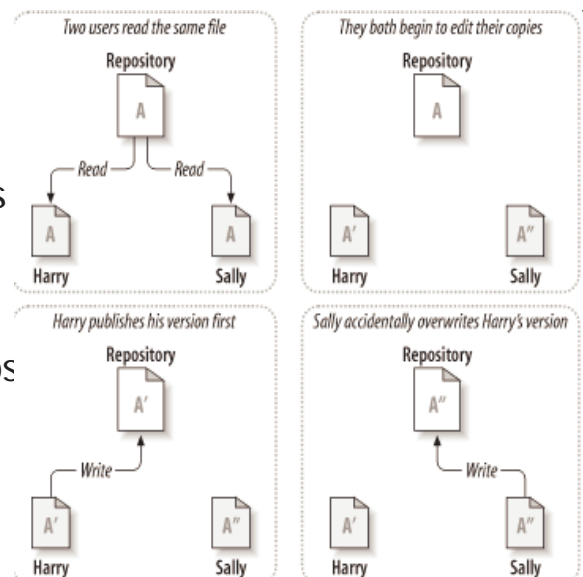
Andrea Delgado - IS2

P3.13



# Modelos versionado

- El problema de compartir archivos
  - Harry y sally hacen check-out del mismo archivo A
  - Cada uno hace sus modificaciones en su copia de trabajo
  - Harry hace commit de sus cambios antes que Sally
  - Cuando Sally hace commit de sus cambios sobrescribe los de Harry



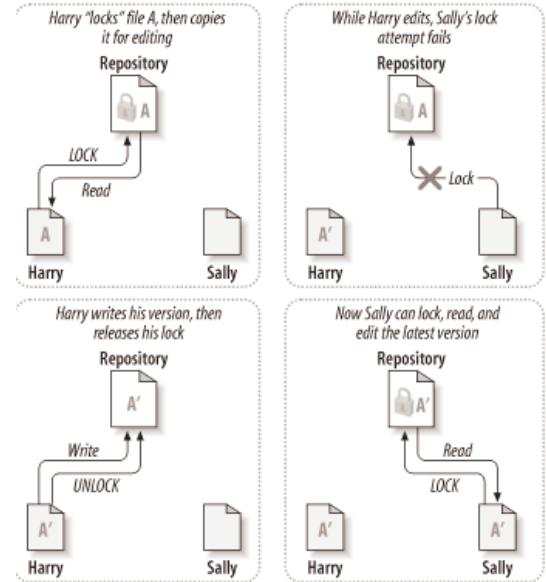
Andrea Delgado - IS2

P3.14



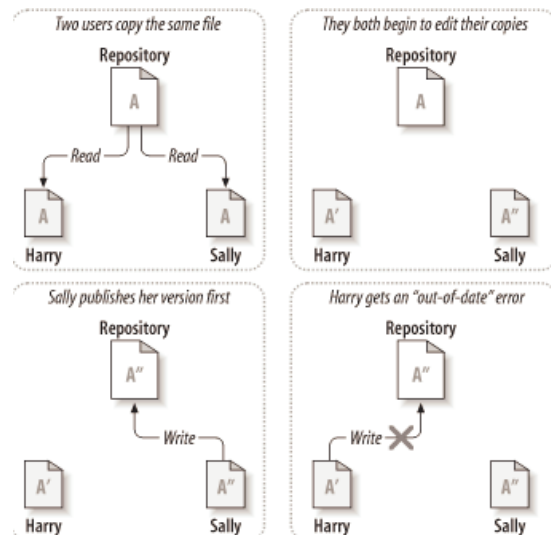
## Modelos versionado (2)

- Solución Bloquear-Modificar-Desbloquear
  - Harry bloquea el archivo y lo copia para trabajar con él
  - Sally quiere bloquear el archivo pero como ya está bloqueado por Harry falla
  - Harry hace commit de sus cambios y desbloquea el archivo
  - Cuando Sally intenta bloquear nuevamente lo logra sin problemas



## Modelos versionado (3)

- Solución Copiar-Modificar-Combinar
  - Harry y Sally copian el mismo archivo para trabajar en él
  - Sally hace commit de sus cambios
  - Cuando Harry hace commit de sus cambios obtiene un error de out-of-date

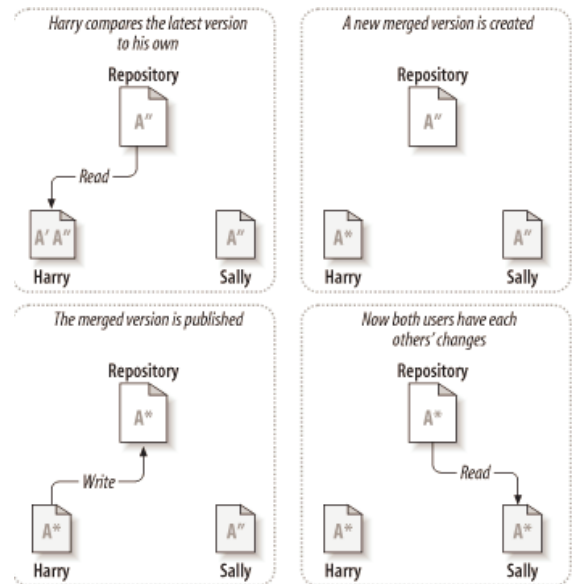




## Modelos versionado (4)

- Solución Copiar-Modificar-Combinar (cont.)

- Harry compara la última versión del repositorio con la suya
- Combina ambas versiones y se obtiene una nueva
- Hace commit en el repositorio de la versión combinada que incluye todos los cambios



## Trabajo del laboratorio – probemos !!

- Cada integrante hace cambios en la clase HelloWorld
  - Dos personas cambian la línea que hace el print de hello world agregando que diga algo más
  - El resto de los integrantes agregan una nueva línea a continuación de la que hay
  - Grabar y parados en la clase botón derecho commit

Qué ha sucedido ??

- Pasos para solucionarlo: cambiar a la vista Team sincronización y ver en que estado están los archivos



## Significado de los íconos en la vista sincronización

|  |                                                                                                                                                                                                                                                                                                                                  |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Adición entrante significa que un recurso fue agregado al repositorio. Un Update transferirá el recurso al espacio de trabajo local.                                                                                                                                                                                             |
|  | Cambio entrante significa que el archivo ha cambiado en el repositorio. Un Update transferirá la nueva revisión del archivo al espacio de trabajo local.                                                                                                                                                                         |
|  | Borrado entrante significa que un recurso fue borrado del servidor. Un Update borrará el recurso en el espacio de trabajo local.                                                                                                                                                                                                 |
|  | Adición saliente significa que el archivo fue agregado en el espacio de trabajo local pero aún no está en el repositorio. Para transferirlo se deberá hacer primero Add y luego Commit.                                                                                                                                          |
|  | Cambio saliente significa que el archivo fue cambiado localmente. Haciendo Commit se transferirán los cambios al repositorio y se creará una nueva revisión del archivo.                                                                                                                                                         |
|  | Borrado saliente significa que un archivo ha sido borrado localmente. Haciendo Commit causará que los recursos remotos sean borrados del repositorio. Nota: en CVS los directorios no son realmente borrados del repositorio. En su lugar, los archivos son borrados y los directorios vacíos son podados del espacio de trabajo |
|  | Adición conflictiva significa que el recurso ha sido agregado local y remotamente.                                                                                                                                                                                                                                               |
|  | Un cambio conflictivo significa que un archivo ha sido cambiado local y remotamente. Una combinación manual o automática es requerida. Además, cualquier entrada en la vista que contiene hijos que están en conflicto será también marcada con el ícono de conflicto. Esto es para facilitar la búsqueda de conflictos.         |
|  | Borrado conflictivo significa que el recurso fue borrado local y remotamente.                                                                                                                                                                                                                                                    |



## Trabajo del laboratorio – probemos !!

- Comparar la copia de trabajo con el repositorio

```
Java Source Compare
Workspace file: HelloWorld.java
Repository file: HelloWorld.java

public class HelloWorld {
 public static void main(String args[])
 { System.out.println("Hello World!"); }
 { System.out.println("Hello World! 3"); }
}

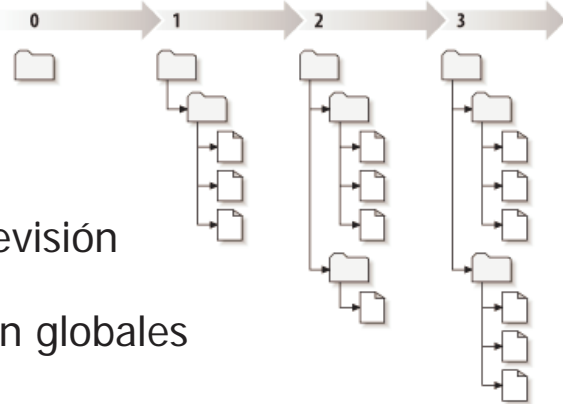
public class HelloWorld {
 public static void main(String args[])
 { System.out.println("Hello World!"); }
 { System.out.println("Hello World! 2"); }
}

SVN
Adding workspace/pruebaSVN/.project
Transmitting file data ...
Committed revision 3.
commit -m "cambio2 andreade" E:/eclipse-java-europa/workspace/pruebaSVN/src/HelloWorld.java
Sending workspace/pruebaSVN/src/HelloWorld.java
Merge conflict during commit
svn: Commit failed (details follow):
svn: File or directory 'HelloWorld.java' is out of date; try updating
svn: resource out of date; try updating
```



## Revisiones en Subversion

- Cada vez que se hace un commit se publican los cambios realizados, como operación atómica
  - Esto es, todos los cambios o ninguno
- Cambio de CVS a SVN en la forma de numerar las versiones



- Cada vez que se hace un commit, se crea un nuevo estado del árbol, llamado revisión
- Los números de revisión son globales



## marcas (Tags)

- Es una instantánea (snapshot) de un proyecto en determinado momento
  - para subversion es lo mismo que una revisión.
- Entonces para qué sirve ?
  - Para nombrar amigablemente determinadas revisiones o subdirectorios más pequeños de un gran proyecto, facilitando su recuperación.
  - Ej. release-1.0 es un subdirectorio particular de la revisión 4822, al marcarlo se puede recuperar solo con la marca release-1.0



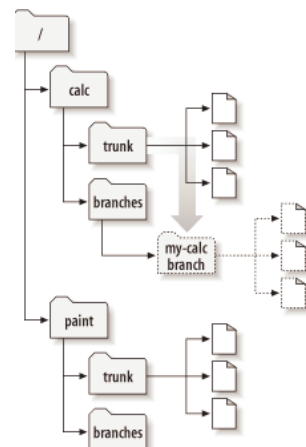
## Trabajo del laboratorio – probemos !!

- El responsable de GCS marca la revisión actual del repositorio como release-1.0
  - Parado en el repositorio en pruebaSVN botón derecho branch/tag
    - Elegir el repositorio y agregar a la URL  
.../pruebaSVN/tags/release-1.0
  - Resto del equipo ir al repositorio y hacer refresh
    - Como se ve la revisión marcada como release-1.0?



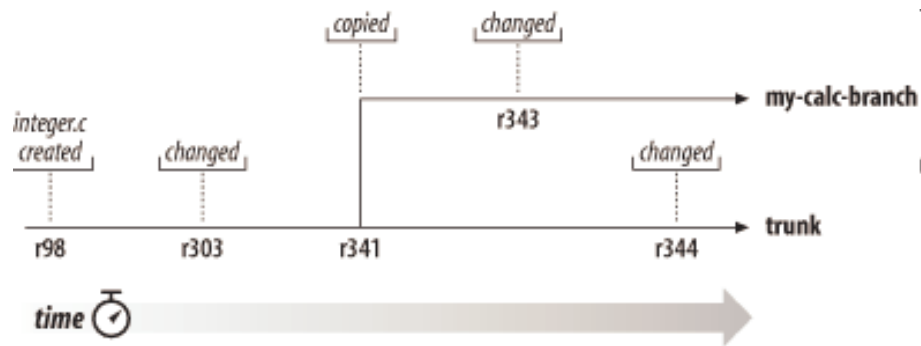
## Ramificar y Combinar (branch & merge)

- Una rama es un desarrollo en paralelo con el tronco principal del desarrollo o con otra rama.
- Se obtiene haciendo una copia de la revisión desde la cual queremos partir y luego cada desarrollo sigue por si mismo





## Ramificar y Combinar (2) (branch & merge)



- Qué se obtiene al ver la historia del archivo `integer.c`?
  - `$ pwd /home/user/my-calc-branch ... $ svn log -v integer.c`
  - `$ pwd /home/sally/calc ..... $ svn log -v integer.c`



## Trabajo del laboratorio – probemos !!

- El responsable de GCS abre una nueva rama desde la revisión actual del repositorio
  - Parado en el repositorio en pruebaSVN botón derecho branch/tag
    - Elegir el repositorio y agregar a la URL  
.../pruebaSVN/branches/branch1
  - Resto del equipo ir al repositorio y hacer refresh
    - Como se ve la nueva rama abierta ?
  - Uno de los integrantes cambia su copia de trabajo a la nueva rama abierta, otro integrante crea una nueva rama branch2



## Trabajo del laboratorio – probemos !!

---

- Para cambiar a la rama deseada
  - Parado en la copia local de trabajo, botón derecho elegir Team switch to another branch/tag/revision
  - elegir del repositorio la rama a la que queremos cambiar
  - Observa en la copia local como indica sobre que rama se está trabajando
- Para combinar ramas entre ellas y con el tronco principal nuevamente
  - Utilizando el comando para merge visto anteriormente, sigue de la misma forma



## Referencias

---

- Libro on-line de Subversion
  - <http://www.open.collab.net/community/subversion/svnbook/>