Control de Versiones con Subversion

Franco M. Luque

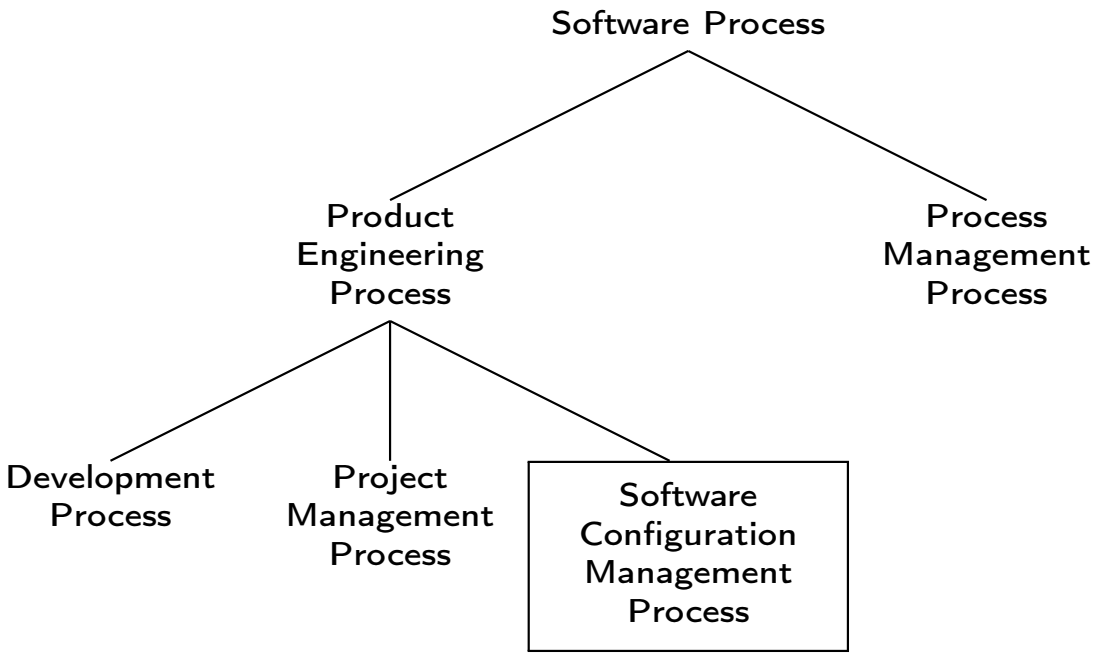
Ingeniería del Software I
Fa.M.A.F., Universidad Nacional de Córdoba

1 de septiembre de 2008

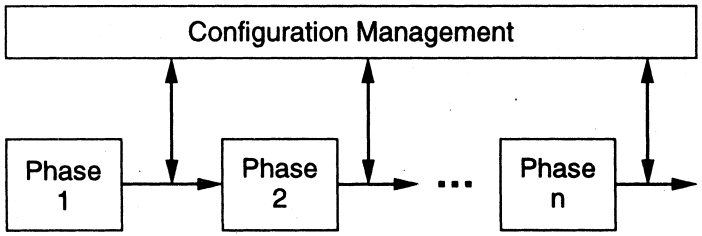
Esquema de la charla

El Proceso de Software

- ¿A dónde estamos parados?



El Proceso de *Software Configuration Management*



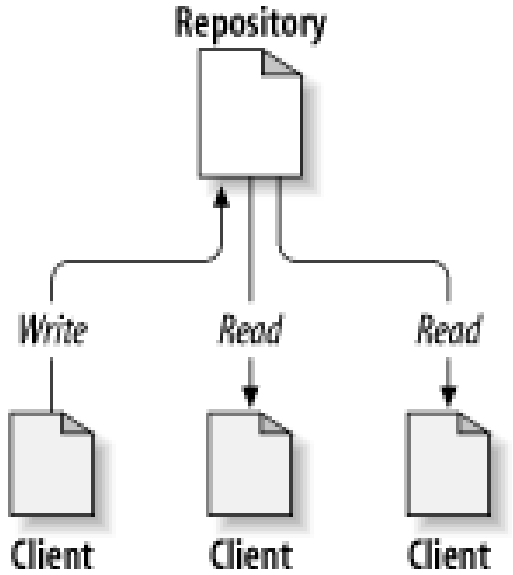
- Administración de los productos de los distintos procesos (código, manuales, documentos, etc.).
- Elementos:
  - Identificación de los ítems del sistema.
  - Control de los cambios de estos ítems.
  - Registro del estado de los ítems y de los pedidos de cambio.
  - Verificación de la correctitud y completitud de los ítems.
- Subprocesos: Control de Versiones, Bug Tracking, Build Management, etc.

Control de Versiones

- Elemento clave del *Software Configuration Management*.
- Administración de las distintas versiones del software y los cambios que se le realizan.
- Características:
  - Guarda todas las versiones y la historia de cambios realizados (es una máquina del tiempo).
  - Administra *releases* (entregas) y *branches* (ramificaciones) del desarrollo.
  - Administra el trabajo en equipo proveyendo "locks" y/o integrando los cambios (merging).
- Herramientas:
  - CVS, Subversion, otros.
  - Plugins para IDEs como Eclipse y Netbeans.
  - GUIs (Cervisia) e interfaces Web (Trac).

Subversion: Introducción

- Historia:
  - Creado por CollabNet Inc. en el 2000.
  - Creado en base a la experiencia con CVS. Es su sucesor.
  - Hoy parte de Tigris.org (Open Source).
- Características principales:
  - Cliente-servidor: Un repositorio de donde se bajan copias locales de trabajo.
  - Multiusuario: Integra automáticamente cambios simultáneos.

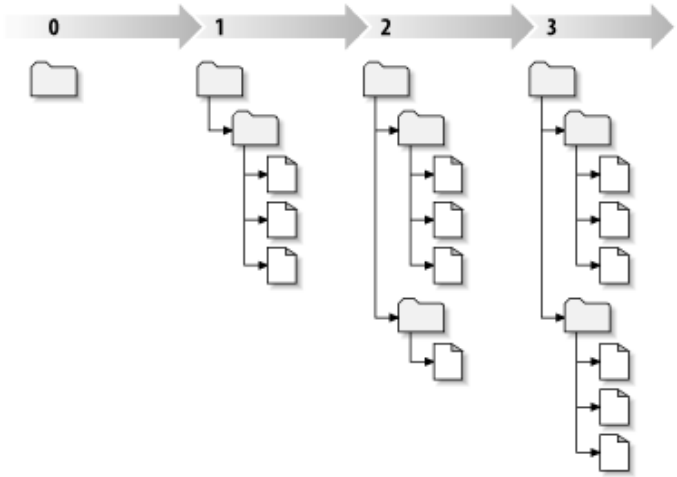


Subversion: Características

- Los usuarios trabajan sobre copias locales.
- Cuando realizan los cambios deseados, suben al repositorio la nueva versión (*commit*).
- Permite trabajo simultáneo de varios usuarios sobre los mismos archivos:
  - Antes del *commit*, se integran los cambios de los otros usuarios en la copia local (*merge*).
  - Si hay conflictos, se informa al usuario para que los resuelva.
- Cada *commit* es atómico: o se sube todo o no se sube nada.

Subversion: Características

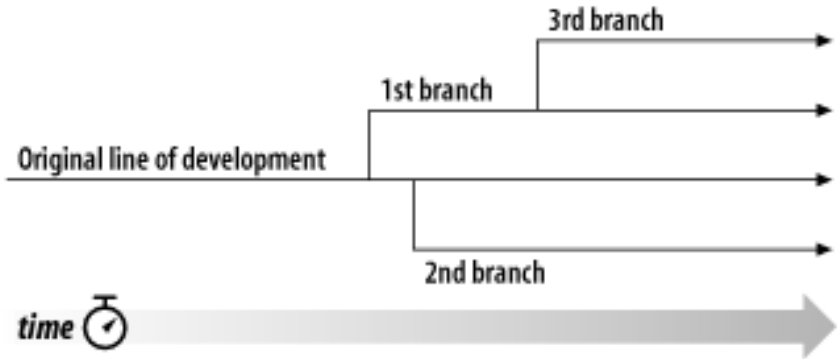
- Cada versión (= revisión) se identifica con un número.
- Empieza de 0 y se incrementa una vez por cada *commit*.
- *Global revision numbers*: Un mismo número para todos los archivos del repositorio.



- Versiona creación, movimiento y copia de carpetas y archivos.
- Versiona tanto archivos de texto (código fuente, etc.) como binarios (imágenes, etc.).

Subversion: Características

- *Tags*: Se pueden crear aliases (*tags*) para las revisiones.
- *Branches*:
  - Se pueden crear nuevas ramas de desarrollo.
  - Dos patrones de uso:
    - *Release branches*: una rama por *release*, mantenidas paralelamente.
    - *Feature branches*: una rama principal estable y otras con *features* inestables.
  - Se pueden integrar cambios entre ramas (*merging*).



Creando un Repositorio

1. Elegir ubicación y nombre.

   Ejemplo: en el home con nombre `svn` (repositorio personal).

2. Crear el repositorio.

   ```
   # cd; svnadmin create svn
   ```

3. Importar los archivos que quiera empezar a versionar.

   ```
   # svn import proyecto file:///home/francolq/svn/proyecto
                   -m"Primer import proyecto"
   ```

4. Borrar los archivos importados.

   ```
   # rm -rf proyecto
   ```

5. Bajar los archivos del repositorio.

   ```
   # svn checkout file:///home/francolq/svn/proyecto
   ```

Ciclo de Trabajo

1. Actualizar copia local.

   ```
   # svn update
   ```

2. Trabajar sobre la copia local.
   - Modificar archivos con los programas usuales.
   - Agregar, eliminar, mover y copiar archivos y directorios:

     ```
     # svn [add|remove|mv|cp]  PATH...
     ```

3. Examinar los cambios realizados.

   ```
   # svn status
   ```

   ```
   # svn diff PATH...
   ```

4. Hacer *merge* con los cambios de otros (y resolver conflictos).

   ```
   # svn update
   ```

   ```
   # svn resolve
   ```

5. Subir cambios al repositorio (*commit*).

   ```
   # svn commit -m"Cambie esto y aquello y aquello."
   ```

Examinar los cambios realizados

- No requiere conexión con el repositorio.
- Vista general:

```
# svn status
?      util.pyc  # el archivo no esta versionado
M      main.py   # el archivo ha sido modificado
A      util.py   # el archivo se agendo para ser agregado
```

- Detalles de cambios en un archivo:

```
# svn diff main.py
Index: main.py
===================================================================
--- main.py     (revision 1)
+++ main.py     (working copy)
@@ -1 +1,4 @@
+import util
+
+util.init()
 print "Hola mundo!"
```

Comandos Útiles

- El más importante de todos:

```
# svn --help
# svn <subcommand> --help
```

- Arrepentirse de cambios locales (volver a última versión):

```
# svn revert PATH...
```

- Declarar resuelto un conflicto:

```
# svn resolved PATH
```

- Hacer *tag* de una versión (por ejemplo para marcar una *release*):

```
# svn copy file:///home/francolq/svn/proyecto
           file:///home/francolq/svn/proyecto/tags/release-1.0
```

Subversion en el Proyecto

- El repositorio ya viene creado y vacío.
- La URL del repositorio es de la forma

  `https://kali.famaf.unc.edu.ar/2008/is/grupoXX/svn`

  reemplazando `XX` por el número de grupo.
- Estructura recomendada:

```
proyecto/trunk      # tronco principal de desarrollo
proyecto/branches   # probablemente no lo usemos
proyecto/tags       # para las releases
```

- Se debe hacer un *checkout*, crear la estructura del proyecto, luego eliminar la copia local y bajarse sólo el "tronco":

  `svn checkout --username USUARIO URL proyecto`

  (el resto queda como ejercicio)

Subversion en el Proyecto

- Ignorar los `*.pyc`:
  1. Abrir `~/.subversion/config`.
  2. Buscar `global-ignores`.
  3. Agregar el patrón `*.pyc`:

     ```
     global-ignores = *.o ....*.rej *.rej .*~ *~ .## .DS_Store *.pyc
     ```
- Agregar muchos archivos a la vez:
  1. Poner los archivos en la copia local.
  2. `svn add --force .`
     (no "`svn add *`" porque entran los `pyc`)
  3. `svn commit -m"Agrego archivos generados por django."`
- Usar Trac para navegar el repositorio y ver la historia de revisiones.

Fin
¡Gracias!