Subversion

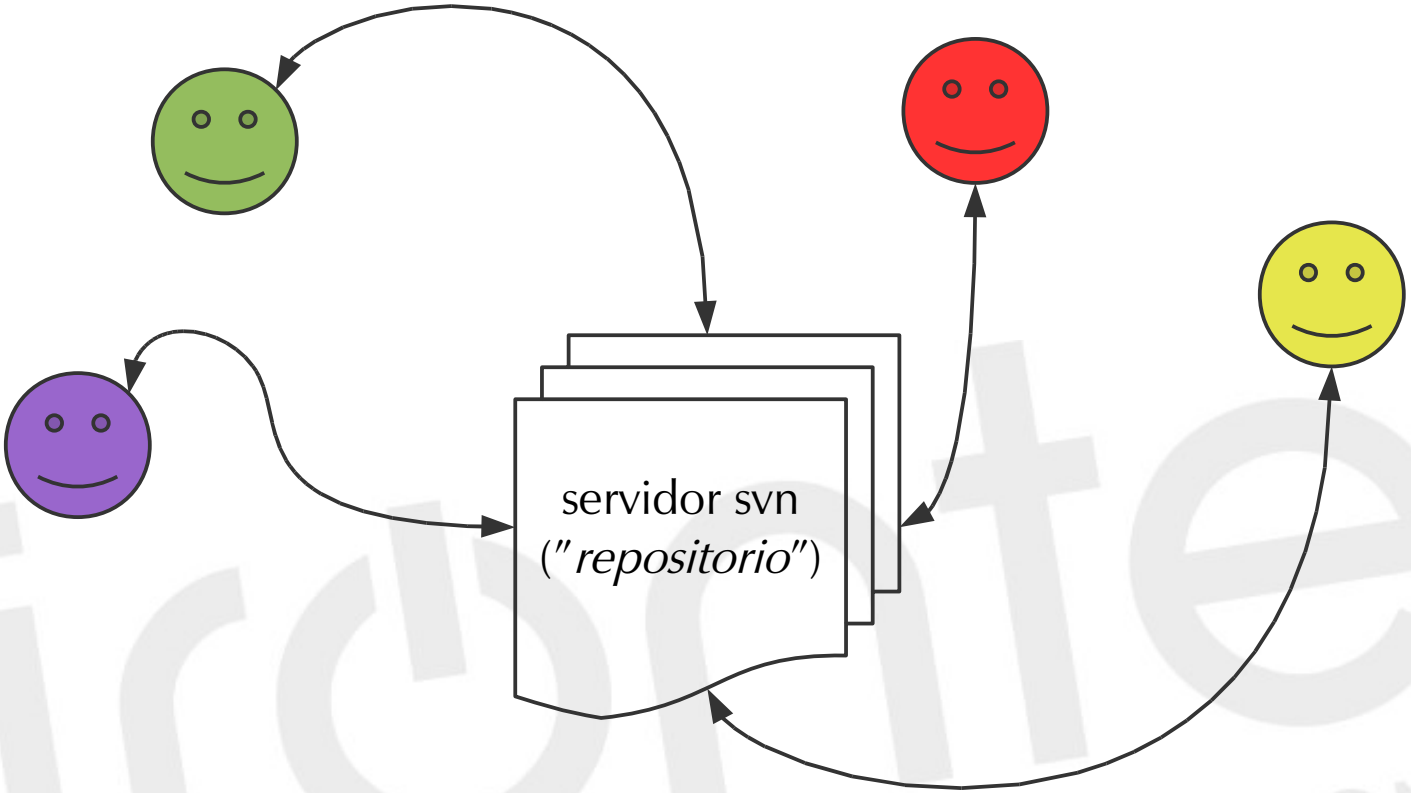
- Introducción a subversion

irontec

Internet *y* Sistemas sobre GNU/Linux

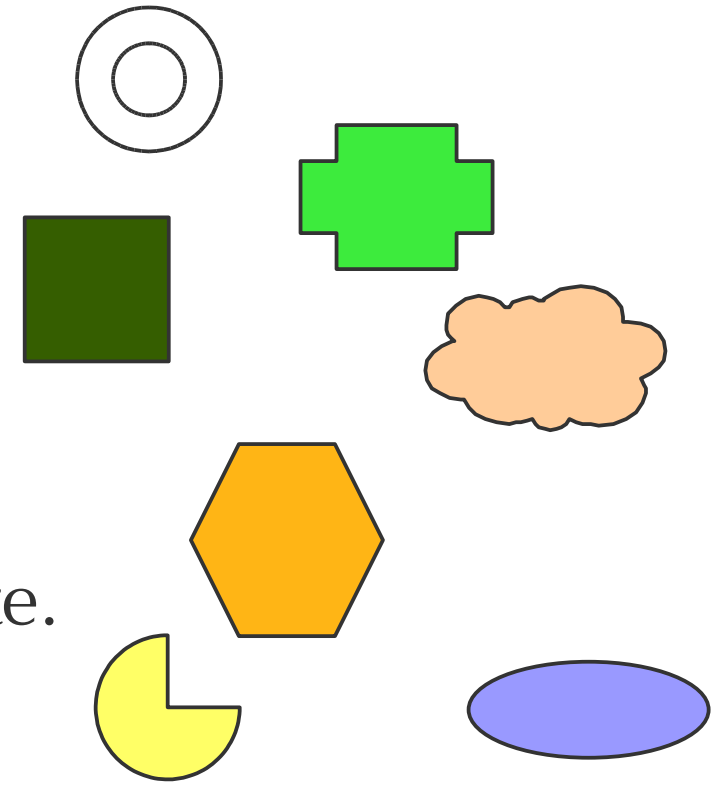
¿Qué es subversion? (aka *svn*)

- Un "sistema de control de versiones".
- Almacena ficheros en un servidor central.
- Mantiene el historial (diferencial) completo de los datos: nunca se pierde nada.



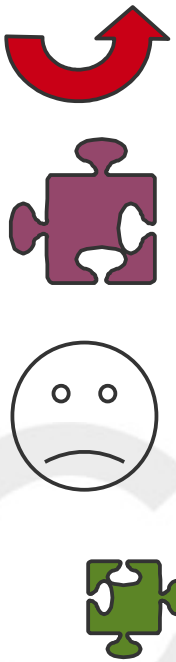
¿Qué es subversion?

- Es genérico:
  - código fuente
  - imagenes
  - /etc/*
  - documentos
  - Lo que sea!
- Es maduro y estable, buen soporte.
- Es multiusuario.



Forma de trabajo tradicional

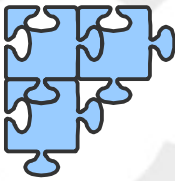
- Modificas tus archivos.
- Decides subir los archivos al servidor común.
- Descubres que otro compañero ha modificado cosas en los mismos archivos.
- Te das cuenta de las horas que supone integrar sus cambios con los tuyos.
- Corres en círculos y te tiras por la ventana.



Forma de trabajo con svn

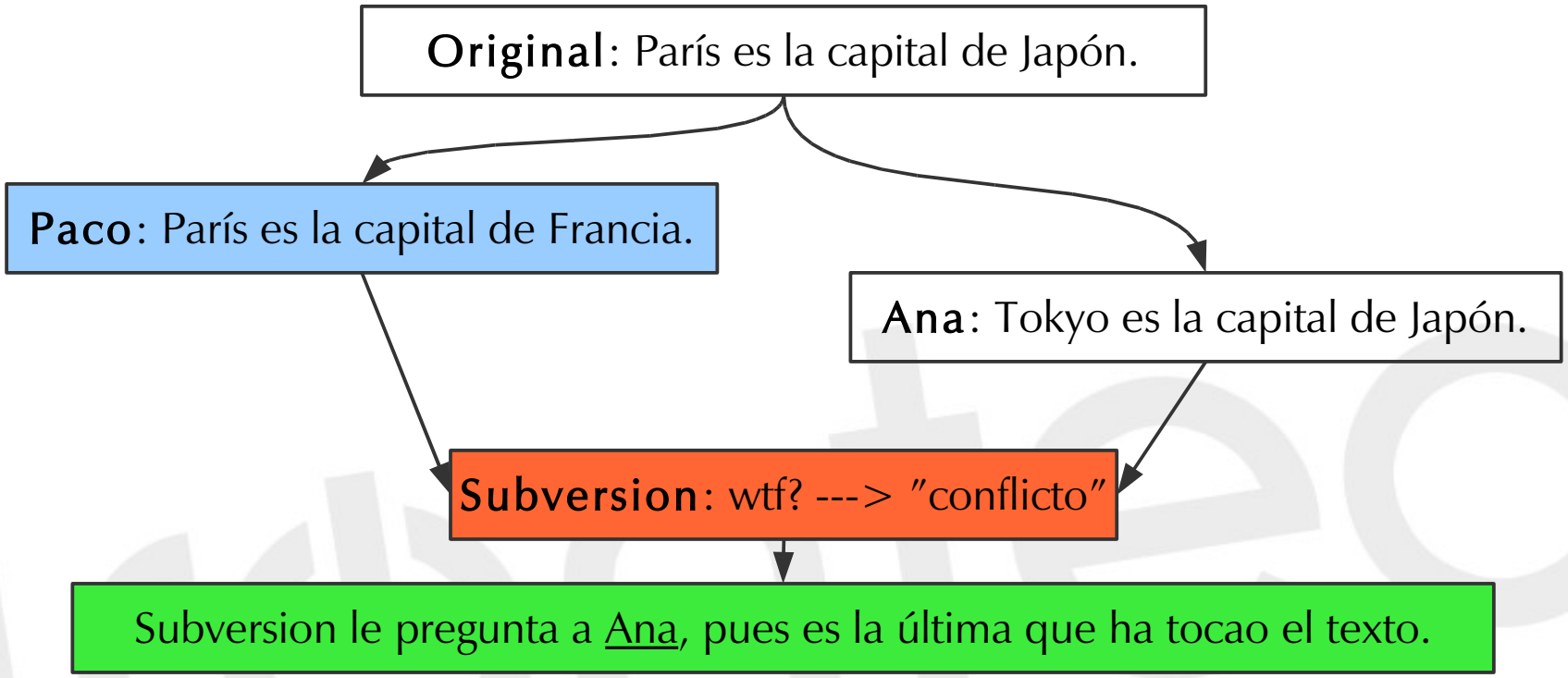
- Modificas tus archivos.
- Decides subir los archivos al repositorio.
- Subversion descubre que otro compañero ha modificado cosas en los mismos archivos.

- Subversion integra tus cambios con los de tu compañero y los sube al servidor.
- Te vas a tomar unas birras con los amigos.



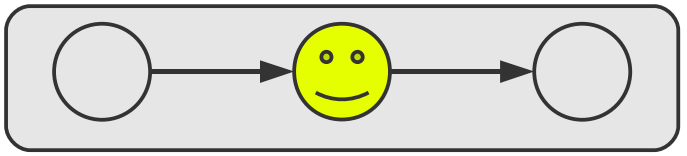
Sí, claro...

- En efecto, svn no siempre sabe como integrar los cambios.
  - Ej: dos usuarios modifican la misma línea de texto.

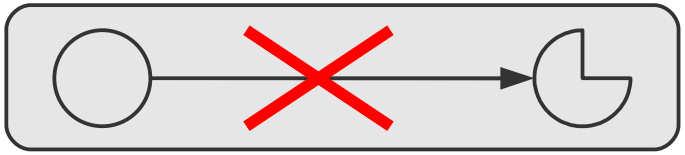


No me fío de svn y sus movidas...

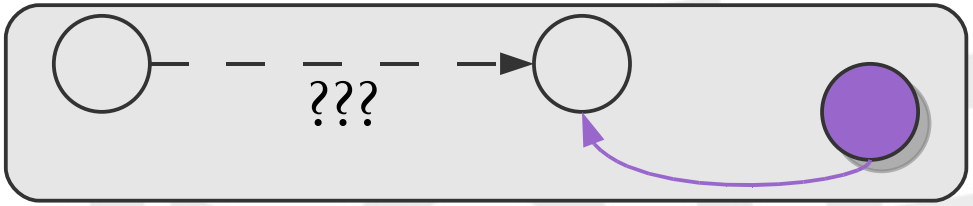
- Siempre puedes revisar los datos antes de que se suban al repositorio.



- Tu set de modificaciones nunca quedará a medio subir: atomicidad (todo o nada).

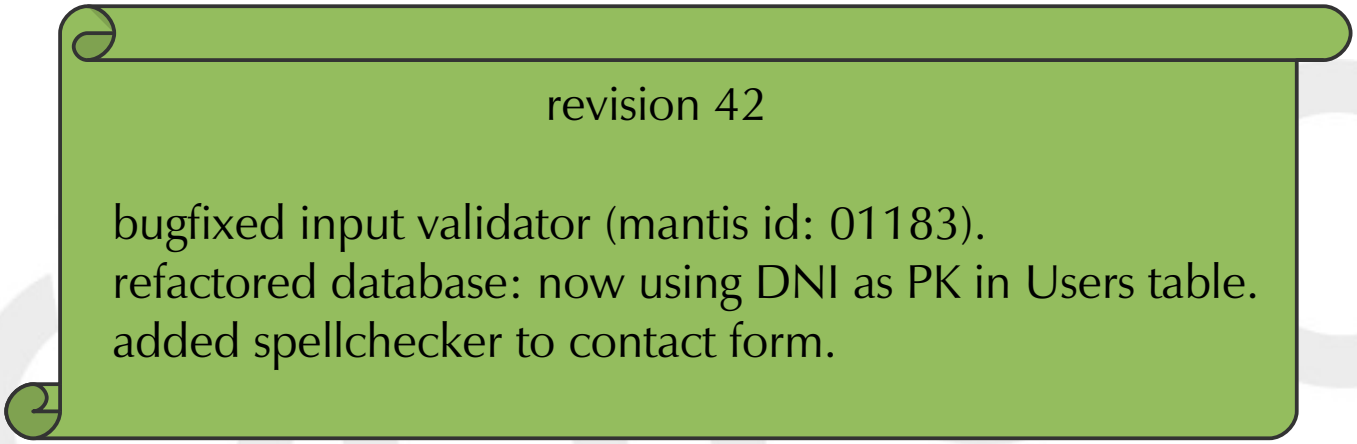


- Antes de subir nada, svn siempre comprueba si otra persona ha tocado el repositorio.



Vamos a lo práctico

- Subir mis cambios al repositorio:
  - svn diff //muestra todas las lineas cambiadas
  - echo "svn diff $* |colordiff |less -R" >> svndiff.sh
  - svn commit//saca tu $EDITOR pa que escribas un
    //resumen de tus cambios, y de seguido los sube
    // si nadie ha cambiado los mismos ficheros
  - Ej. de log:



...y si han tocado los ficheros?

yo: quiero subir mis cambios
`svn commit`

svn: quieto parao! hay modificaciones en el repo

yo: ah. cuáles son?
`svn update`

svn: pues.. te las integro en tu copia local

yo: a ver cómo me has dejao el tinglao...
`svn diff`

opcional: oh, hay conflictos! los resuelvo.

```
vi file.txt
svn resolved file.txt
```

yo: todo ok. quiero subir mis cambios once again
`svn commit`

svn: ahora sí que los subo :-D he creado una nueva *"revision"*

¿Qué es eso de las revisiones?

- Numero entero incremental asignado a cada modificacion (commit).
- Repository-wide: un commit incrementa la revision de todos los ficheros y directorios del repo.
- "head" es un alias para la ultima revision.
- Muchos comandos svn aceptan elegir la revision. Ej: -r 120 -r 120:146 -r 120:head

Comandos basicos

- Crear una copia local del repositorio (al ppio):
  svn checkout <repositorio> <directorio local>
- Eliminar una copia local (al final):
  rm -rf <directorio local>

- Hacer cambios locales:

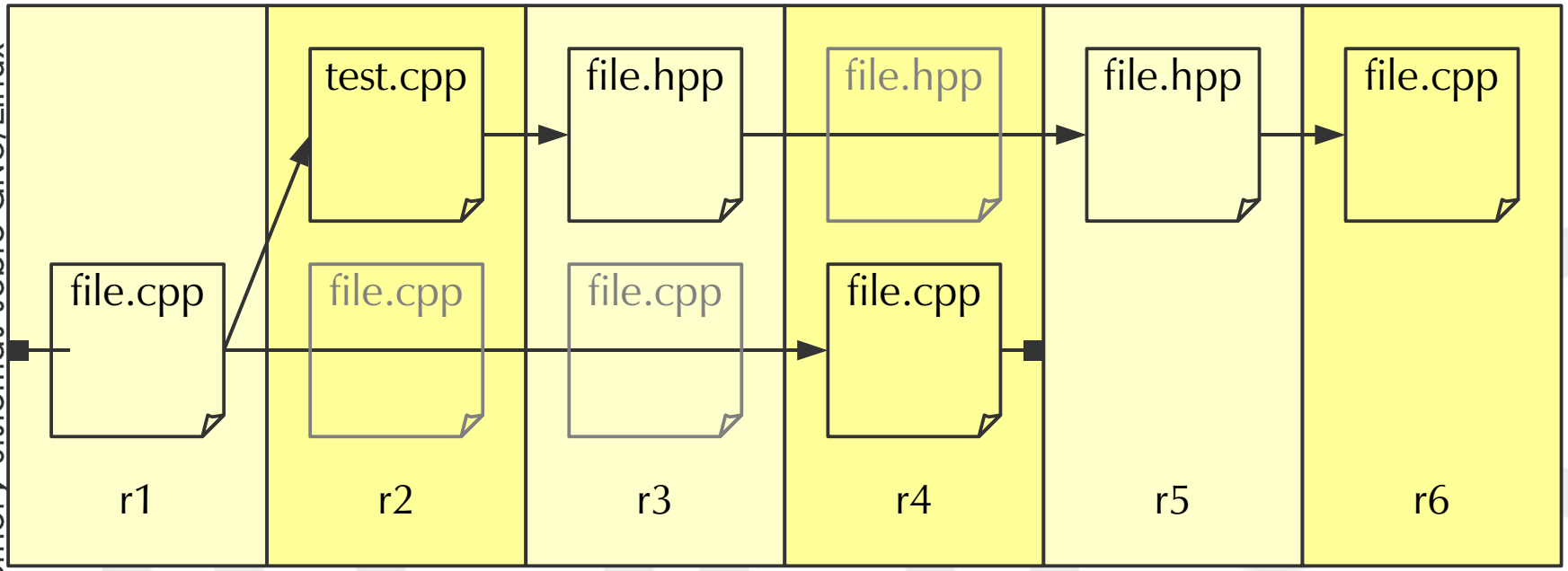
  vi file1 file2 :wqa

- Eliminar cambios locales:
  svn revert file{1,2} //elegantemente
  rm file{1,2}; svn update file{1,2} //warramente

Comandos basicos II

- Añadir, eliminar, copiar, mover, renombrar:
  svn add/rm/cp/mv/mkdir ...
- Se mantiene el historial!

add cp mv (editar) rm mv



Más cosillas...

- Ver número de revision de tu copia local:

  svn info

- Ver los mensajitos de log...

  svn log //todos los logs desde el primero
  svn log -r 150 //log de la revision 150
  svn log -r head //log de la ultima rev.
  svn log -r 150:head //logs de la r150..head

- Volver a la revision 150 en tu copia local:

  svn update -r 150 //tb se puede con fechas

Más cosillas II

- Ver estado de todos los ficheros/dirs:

  svn status

  - ? .configuration.ini.swp //no versionado
  - M Doxyfile //modificado
  - M configuration.ini //modificado
  - A nuevoFichero.txt //añadido

- Ignorar ficheros/dirs (*.bak *.swp binarios ...):

  svn propedit svn:ignore .

  ```
  *.bak
  .*.swp
  log???.txt
  Debug
  ```

  :wq

Uhh... "propedit"?

- Sirve para almacenar metadatos:
  - Propiedades usadas por subversion:
    svn:ignore, svn:externals, svn:eol-style,svn:author, svn:executable...
  - Propiedades usadas por otros:
    svnmerge-integrated...
  - Datos usados por nosotros, por ej: iron:client, iron:trac-url, iron:public-url...
- Subcomandos de svn:
  proplist, propget, propset, propedit, propdel

Ese gran odioso invento: ramas

- Ramas paralelas de desarrollo. Ejs:
  - Una web en castellano vs inglés
  - Un programa estable vs bleeding-edge
  - Un programa completo vs demo
  - Un apache2.conf para windows vs linux
- Cómo se crean?
  - svn cp directorioOriginal ramaNueva
- También mantienen el historial.

Ese gran odioso invento: ramas

- Para hacer:

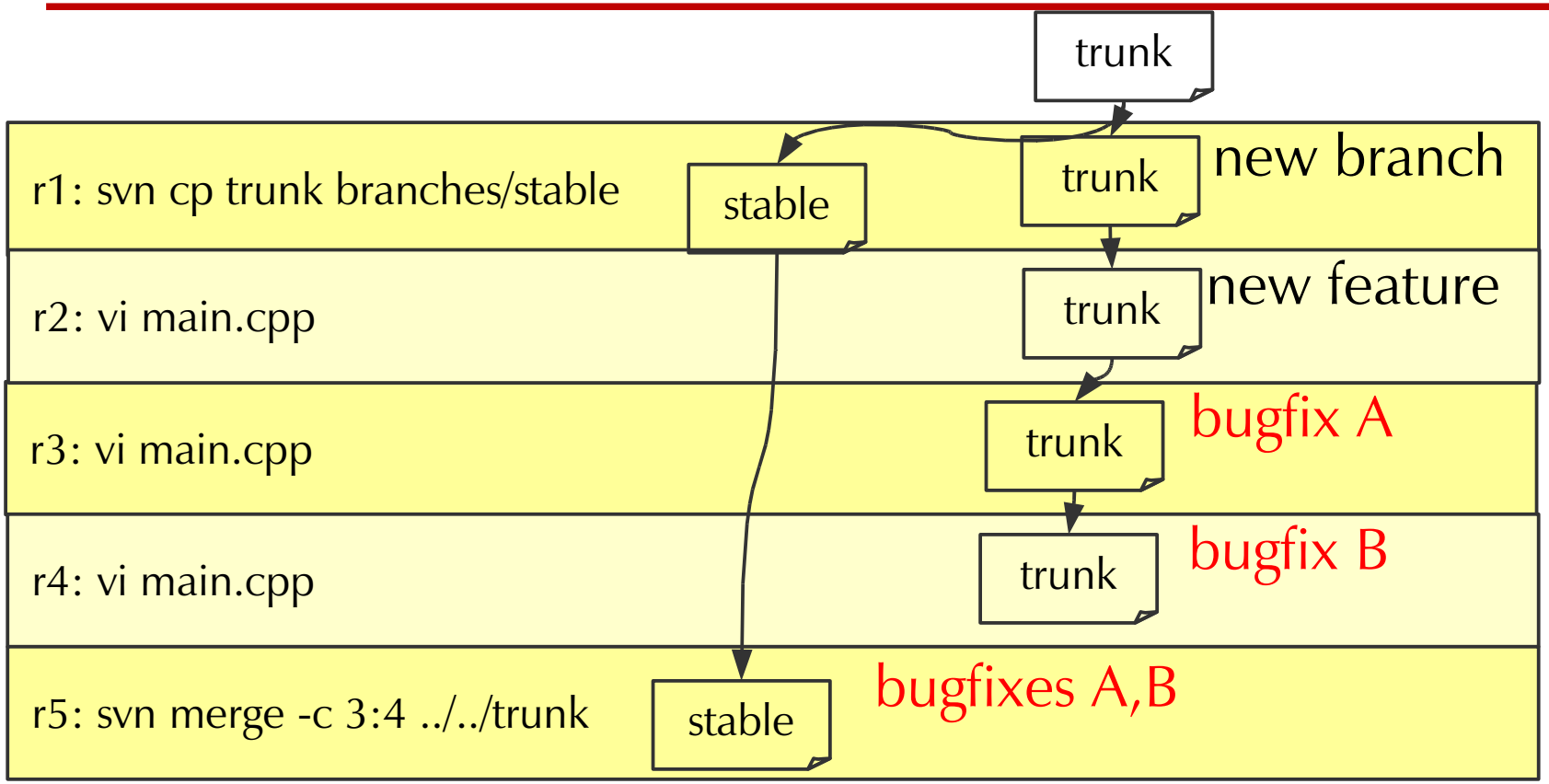
```
svn cp original rama
```

habrá que tener un directorio "original", claro.

- Estándar de-facto:

```
/trunk                  //el original
/branches/estable       //una rama
/branches/demo          //otra rama
/branches/*             //más ramas
```

Sincronizar cambios concretos



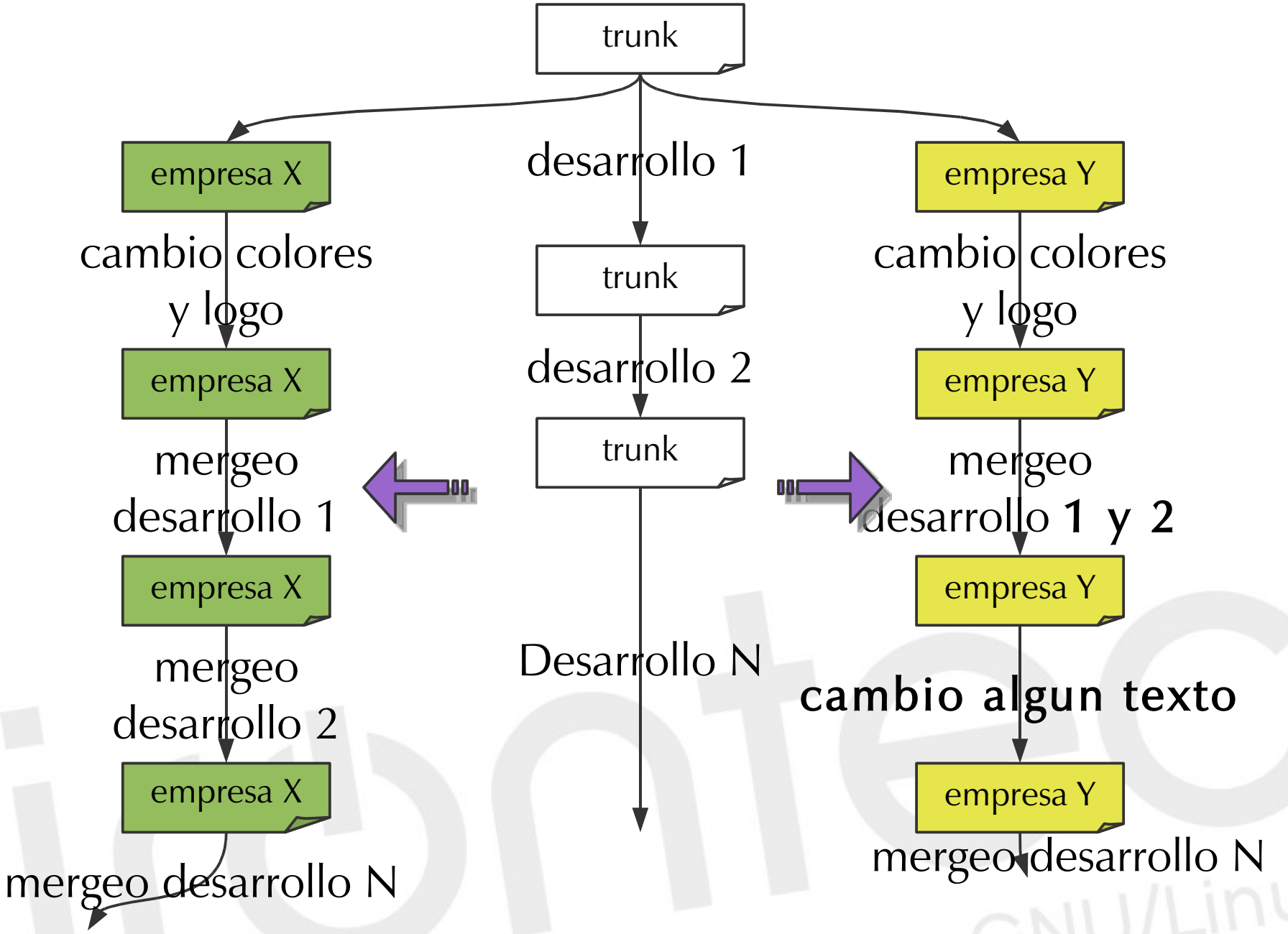
Es como hacer un *update* de los cambios... contra otro directorio.

Hay que mantener manualmente una lista de las revisiones mergeadas :-(

Sincronizar casi todos los cambios

- Caso ejemplo: tenemos una software web para la empresa X y otro para la empresa Y.
  - La empresa X quiere colores verdes, la Y amarillos.
  - La empresa X quiere su logo.gif, la empresa Y quiere tb su propio logo, claro.
  - El resto del software web será siempre idéntico.
  - No vale poner if(empresaX) {...} elif (empresaY) {...} porque se les vende el código fuente también.

Sincronizar casi todos los cambios



Sincronizar casi todos los cambios

- También hay que mantener una lista de las revisiones mergeadas.
  - ¿Por qué? El algoritmo de merging es un poco kk.
- Como es un coñazo, dejamos que un programa lleve la lista por nosotros:
  - svnmerge.py
    - En teoría www.orcaware.com/svn/wiki/Svnmerge.py...
    - ...pero está caída asi que usad su google-cache (solo parece tirar en modo "text only").

Tags (etiquetas)

- Ejemplos: "alpha 0.7", "rc2", "v2.5"...
- Cómo se puede hacer?:
  - Llevar manualmente una lista de esos tags<-->revs
  - svn cp -r 176 repo/trunk tags/beta2
    - Típica estructura svn: trunk, branches/*, tags/*
- Subversion solo almacena las modificaciones, asi que un tag ocupa virtualmente nada.

"Montar" repositorios externos

- Incluir un repositorio de terceras partes:
  - A mano (local, mantenimiento manual):

```
$ cd ~/miRepo
$ svn co      svn://foo.com/libbar/tags/v1.7 libbar
$ svn co -r943 svn://svn.irontec.com/gesti/trunk libgesti
```

  - Con externals (repository-wide, mantenimiento automatico):

```
$ cd ~/miRepo
$ svn propedit svn:externals .
    libbar       svn://foo.com/libbar/tags/v1.7
    libgesti -r943 svn://svn.irontec.com/gesti/trunk
    :wq
$ svn commit -m"Added 3rd party libs and gesti lib"
$ svn update
```

    (recomendable usar siempre -rN: versiones consistentes!)
  - Ignorar externals: svn <subcomando> --ignore-externals

"Montar" repositorios externos

- Los externals son usables... y mejorables:
  - Solo se puede referenciar directorios, no ficheros.
  - Referencias a directorios absolutos (no vale ../algo).
  - La propiedad svn:externals no se actualiza con svn mv/etc.
  - Commits manuales:
    - arreglamos un bug:
      vi libgesti/gesti.inc
    - committamos el apaño (mal!):
      svn commit -m"Fixed login (mantis id:001919)."
    - committamos el apaño (ahora sí):
      cd libgesti
      svn commit -m"Fixed login (mantis id:001919)."

Administración

- Crear un repo:
  - svnadmin create /var/svn/miRepositorio
- Servir un repo:
  - Svnserve: svn://
  - Apache: http:// https://
  - SSH: svn+ssh://
- Backup:
  - svnadmin hotcopy miRepo /backups/miRepo

Recomendaciones mías

- "Save early, save often". Haz commits muy a menudo.
- Cambios independientes, commits independientes (backports de fixes, etc...). Puedes tener varios checkouts.
- Todos los commits deben ser usables/compilables. Puedes usar ramas para versionar tus cambios grandes.
- Revisa el status y el diff antes de cada commit (cambios temporales, archivos no añadidos, etc...).
- Siempre escribe un log. Básate en el diff.
- Referencia número de issue en el log de svn y el número de revisión al cerrar el issue.

Recomendaciones mías

- Los comentarios sobre código van en el código, no en el log.
- Yo suelo usar palabras clave, pueden ayudar a crear un "changelog" publico (a mano o incluso scriptado). Por ejs:
  -

```
fixed login failures.
modified logo image: it had pixelation issues (issue #9).
replaced Exit with Logout class.
added new Pedido class (issue #225).
removed deprecated headers from /common/foo.
bufixed invisible checkbox labels (issue #98).
renamed classes to CamelCase.
refactored footer DIVs (patch by Joe <joe.a@gmail.com>).
etc...
```

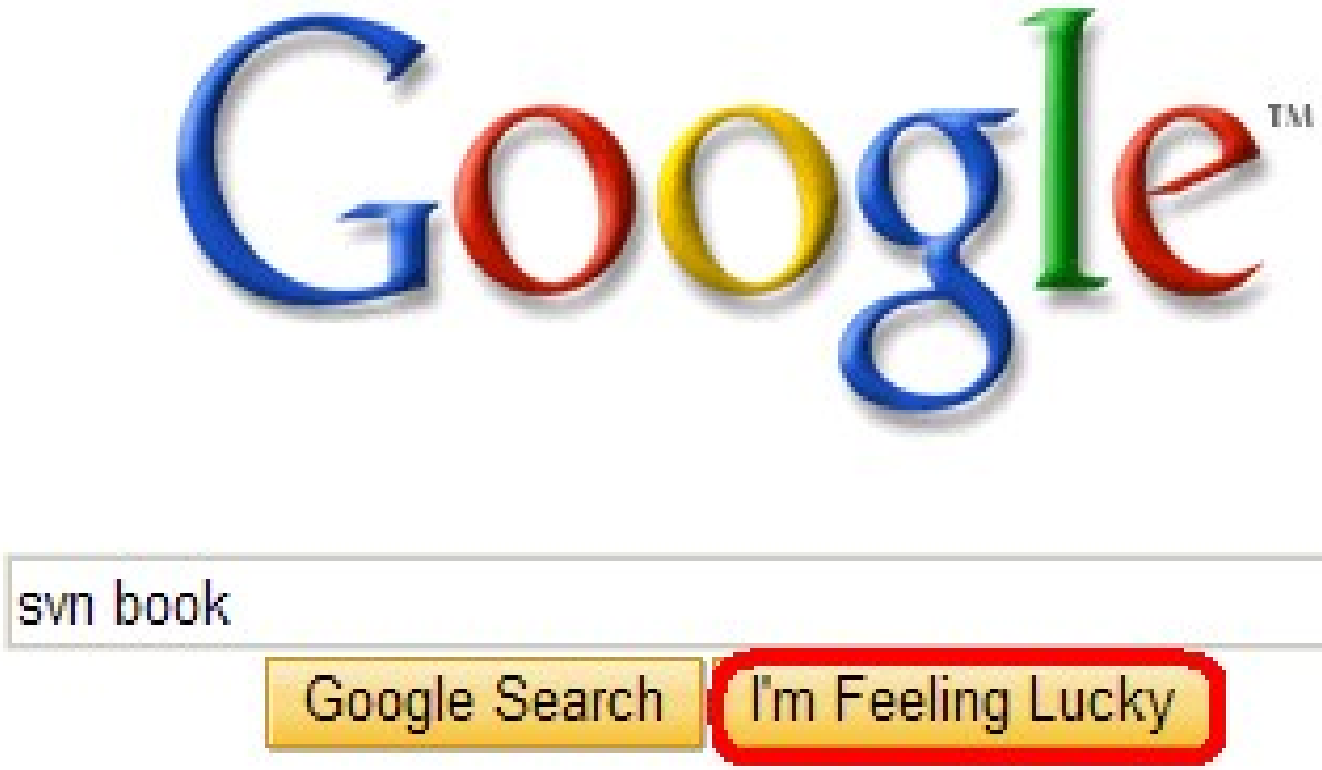
Administración

- FAQ: "¡Auxilio, socorro! Se me ha colado mi número de VISA en un commit!!!!11!1one!! qué hago???"
  - Primer intento (mal):
    - svn rm visa.txt
    - svn commit -m"Removed my visa number"
  - Segundo intento (ahora sí):
    - Se hace un svnadmin dump del repositorio.
    - Se le pasa por svndumpfilter
    - Se recarga el dump al repositorio con svnadmin load.
- Sip, es una warrada... así que revisad los commits!

Más información

- CLIs: apt-get install subversion && ls /usr/bin/svn*
- GUIs (http://subversion.tigris.org/links.html):
  - Standalone: tortoisesvn, rapidsvn, kdesvn, ...
  - Plugin: subclipse, vcscommand.vim, psvn.el, ksvn, ...
- APIs:
  - SvnCPP, PySvn, SvnKit (java), SubversionSharp (C#), PECL SVN (PHP)
- Ayuda:
  - <comando> help [subcomando]
  - Para todo lo demás... el libraco (sig. pag.):

Más información



Aftermatch

- ¿Dudas, sugerencias, insultos, etc?

The End.

- Gracias por asistir! :-D

Licencia Copyleft

- Este documento está protegido bajo la licencia Reconocimiento-SinObraDerivada 2.1 España de Creative Common (http://creativecommons.org/licenses/by-nd/2.1/es/)