Gestión de Configuración con CVS, WinCVS y plug-in para Eclipse

Proyecto de Ingeniería de Software
Curso 2005

Introducción

- SCM permite que el desarrollo se realice en forma ordenada y controlada.
- Definir procedimientos, responsabilidades, tiempos, líneas base, gestión de cambios.
- Las herramientas ayudan a realizar la gestión de configuración pero no la suplantan.
- En Proyecto de Ingeniería de Software:
  - Plan de SCM para el proyecto
  - definición del ambiente controlado y procedimientos
  - control de versiones y cambios, auditorías linea base
  - herramientas CVS, WinCVS y plug-in para Eclipse.
- El Responsable de SCM debe ser la referencia para los integrantes del grupo en lo concerniente a la gestión de configuración del proyecto

CVS – Concurrent Versions System

- Sistema de control de versiones para registrar la historia de los productos generados durante un proyecto de software.
- Los archivos fuentes se almacenan en un único archivo registrando las diferencias entre las distintas versiones.
- Los demás archivos deben ser almacenados como binarios para evitar que se corrompan.
- Organización:
  - Repositorio centralizado en facultad:
    lulu.fing.edu.uy/ens/cvs/ingswXX
  - Un usuario unix por grupo: ingswXX
  - usuarios CVS para cada integrante del grupo que permiten identificar quien realizo acciones sobre el repositorio

CVS – Concurrent Versions System

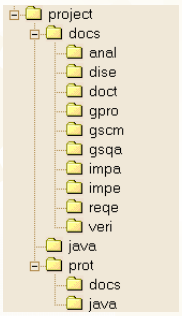
- Permite:
  - Múltiples desarrolladores modificando el mismo código
  - Almacenar una única copia maestra del código
  - Automatizar las actualizaciones entre versiones
  - Acceder a cualquier estado previo del código
- Previene:
  - Sobreescribir código
  - Pérdida de trabajo por falta de respaldo
- No sustituye la necesidad de gestionar ni la comunicación entre desarrolladores
- Es responsabilidad de cada integrante del grupo seguir los procedimientos para la gestión de configuración definidos por el SCM, y del SCM controlar que se cumplan.

Repositorio

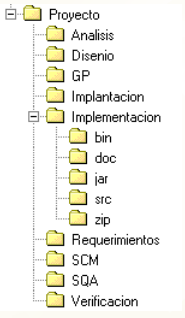
- Almacena los archivos y directorios bajo gestión de configuración.
- El acceso se realiza únicamente mediante comandos CVS.
- La estructura general del repositorio es un árbol de directorios y debe ser definida por el Responsable de SCM.
- El directorio CVSROOT contiene archivos administrativos de CVS y solo debe ser usado por el Responsable de SCM.
- Es responsabilidad de cada integrante del grupo mantener el repositorio y del SCM asegurar la integridad del mismo.

Repositorio - ejemplos

Grupo 1 - 2002

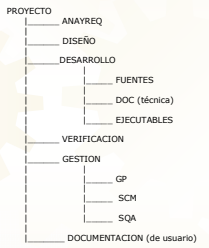


Grupo 5 - 2002

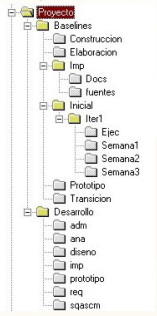


Repositorio - ejemplos

Grupo 7 - 2002



Grupo 3 - 2002



Uso de CVS por los grupos y áreas

- Todos los grupos harán el versionado de documentos utilizando el repositorio en facultad
- Los grupos JAVA harán también el versionado de código utilizando el repositorio en facultad
- Los grupos GENEXUS y .NET almacenarán en el repositorio al final de la iteración, el código obtenido en esa iteración, es decir la versión de la iteración.
- Todos los grupos y áreas pueden trabajar con CVS desde la línea de comandos o WinCVS
- Los implementadores de los grupos JAVA usarán el plug-in CVS de Eclipse

Repositorio – creación de la estructura

- Definir el árbol de directorios en que se organizará el repositorio para el proyecto
- Crear la estructura localmente (vacía), posicionarse en el directorio padre y hacer:
  - Línea de comandos:
    cvs import prueba com fin
    (com y fin son dos tags requeridos por el comando import que pueden tener cualquier nombre)
  - WinCVS:
    Remote→Import Module
  - Recomendación: crear un árbol de directorios de prueba para hacer pruebas antes de crear el definitivo.

Repositorio – creación de la estructura

- Con el plug-in de Eclipse:
  - En la vista CVS Repositories → Navigator:

- Crear la estructura localmente (vacía) incluyendo los folders correspondientes, luego desde el directorio padre elegir Team → Share Project

Comandos – resumen (I)

- Forma general de los comandos CVS:
  cvs [global ops] comando [comand ops] comand args]
- Checkout [options] module
  - Recupera una copia de trabajo
- Commit [options] [files]
  - Confirma en el repositorio los cambios realizados
- Add [options] [files]
  - 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 en la que se está trabajando)
- Update [options] [files]
  - Sincroniza el directorio de trabajo con el repositorio

Comandos – resumen (II)

- Diff [options] [files]
  - Compara versiones del archivo especificado, la acción por defecto es comparar el archivo en el directorio de trabajo con la copia en el repositorio.
- Tag [options] nombre [files]
  - Agrega una marca simbólica a las revisiones de los archivos recuperadas, marca una versión.
- Log [options] [files]
  - Imprime información histórica para los archivos
- Status [options] [files]
  - Muestra información de estado en el directorio de trabajo
- Release [options] directorio
  - Libera el directorio de trabajo en uso

Módulos - definición

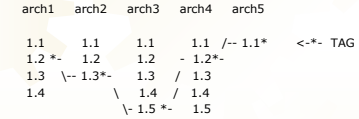
- Trabajar con módulos puede ser conveniente para agrupar archivos y directorios relacionados.
- Permite recuperar el módulo sin sus directorios padres, es útil para trabajar en el mismo directorio sobre distintos subdirectorios.
- Procedimiento: definir los módulos en el archivo administrativo modules de CVS:
  - cvs checkout CVSROOT/modules
  - Editar el archivo modules y definir los módulos:
    Ej. analisis prueba/analisis
  - Hacer commit del cambio y release de la copia de trabajo recuperada.
  - Permite hacer checkout solo de los archivos en el módulo analisis

Números de revisiones (versiones)

- CVS asigna en forma automática los números de revisión (versión) a los archivos.
- En el tronco principal de desarrollo se numera:
  1.1 – 1.2 – 1.3 – 1.4 .....
- El tronco principal puede abrirse en ramificaciones (branches) donde cada ramificación es una línea de desarrollo propia que luego puede incorporarse al tronco principal.
- Las ramificaciones se numeran comenzando con la numeración de la revisión en el tronco principal que se ramificó, agregando el primer número par no utilizado, comenzando con el 2:
  1.2.2 → 1.2.2.1 – 1.2.2.2 ....
  1.2.4 → 1.2.4.1 – 1.2.4.2 – 1.2.4.3 ...

Marcado de versiones (tags)

- Para identificar una versión se realiza una marca (tag) de los archivos que la componen.
- Los nombres de las marcas deben comenzar con una letra y pueden tener letras, dígitos, '_' y '-'.



- Las revisiones de los archivos que tienen * son las que componen la versión marcada.
- En PIS se debe marcar una versión por iteración.

Estados de los archivos (I)

- según las operaciones realizadas localmente sobre un archivo recuperado y las realizadas por otros sobre ese archivo en el repositorio:
  - Up-to-date: es igual al vigente (última versión) en el repositorio
  - Locally modified: modificado localmente pero no se ha hecho commit de los cambios
  - Locally added: agregado localmente pero no se ha hecho commit de los cambios
  - Locally removed: eliminado localmente pero no se ha hecho commit de los cambios
  - Needs checkout: otra persona ha realizado commit de una nueva versión en el repositorio, se debe actualizar el archivo local con el comando update.

Estados de los archivos (II)

- Needs merge:otra persona ha realizado commit de una nueva versión en el repositorio y quien recibe el mensaje también ha realizado cambios en el archivo
- File had conflicts on merge: ha sido modificado localmente y un comando update realizado previamente está dando conflictos.
- Unknown: CVS no tiene información sobre ese archivo (se creó en la copia de trabajo pero no se hizo add)

Todos los desarrolladores deben conocer estos estados para poder resolver conflictos y realizar las acciones correspondientes sobre los archivos.

CVS – uso en PIS05 – línea de comandos

- Usuarios CVS para cada integrante del grupo
- Un directorio de trabajo por integrante del grupo
- Formas de conexión:
  En facultad:
  - Conexión a CVS desde el shell con los comandos:
    - cvsamb <usuariocvs>
    - cvs login
      :pserver:<usuariocvs>@lulu.fing.edu.uy:/ens/cvs/ingswXX
  - Para recuperar una copia de trabajo:
    - Desde el shell haciendo checkout en el directorio de trabajo propio en el home del usuario
    - cvs logout (al finalizar la sesión de trabajo con cvs)

CVS – uso en PIS05 - línea de comandos

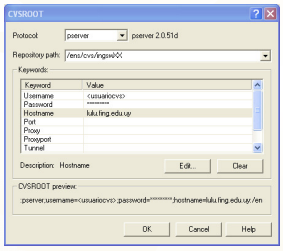
Desde fuera de facultad:

- Conexión a lulu.fing.edu.uy con el usuario unix mediante el shell ssh
- Conexión a CVS desde el shell ssh con los comandos:
  - cvsamb <usuariocvs>
  - cvs login :pserver:<usuariocvs>@lulu.fing.edu.uy:/ens/cvs/ingswXX
- Para recuperar una copia de trabajo:
  - Desde el shell del ssh haciendo checkout en el directorio de trabajo en el home del usuario
  - Copiar los archivos recuperados a la máquina local mediante el cliente ftp del ssh
  - cvs logout (al finalizar la sesión de trabajo con CVS)
- Para devolver una copia de trabajo:
  - Copiar los archivos recuperados al directorio de trabajo propio en el home mediante el cliente ftp del ssh
  - Con los comandos cvs realizar las acciones necesarias para reflejar los cambios en el repositorio

CVS – uso en PIS05 - WinCVS

- En la nueva versión de WinCVS se eliminó el seteo de CVSROOT como variable global, se deben ingresar los valores correspondientes al hacer login y dura la sesión

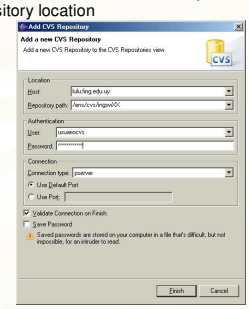
  Admin -> Login y en Login Settings -> CVSROOT



- Cada integrante del grupo se conecta con su usuario <usuariocvs> con la opción pserver

CVS – uso en PIS05 – plug-in de Eclipse

- Setear ubicación, tipo de autenticación y usuario del repositorio CVS en la vista CVS Repositories, New →Repository location



- Cada integrante del grupo se conecta con su usuario <usuariocvs> con la opción pserver

CVS – uso en PIS05

- Resumen conexión al repositorio:
  - En cualquiera de las 3 formas de trabajo (linea de comandos, WinCVS, plug-in de Eclipse) se elige la opción pserver con:
    - host: lulu.fing.edu.uy
    - repositorio del grupo: /ens/cvs/ingswXX
    - usuario: <usuariocvs>
- Ventaja de la nueva forma de conexión:

  se registra automaticamente quién realizó acciones sobre los archivos ingresados, lo que facilita la tarea del SCM y de los integrantes del grupo frente a problemas
- Con la opción "history" se pueden ver las distintas versiones, fechas y usuarios que trabajaron sobre los archivos ingresados

CVS – ejemplo con línea de comandos

- La estructura del repositorio es la siguiente:

```
PRUEBA
   |_____ ANALISIS
   |_____ DISEÑO
```

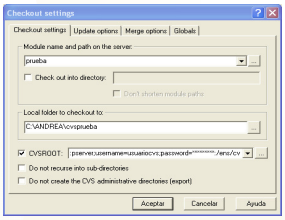
- Para empezar a trabajar con CVS se debe tener un directorio de trabajo por integrante en el home del usuario
- Para hacer un checkout de todo el proyecto en el directorio correspondiente al integrante:
  - cvs checkout prueba
- Se creará la estructura del repositorio incluyendo los archivos en cada subdirectorio, y un subdirectorio CVS de archivos administrativos
- Si se tiene definido el módulo analisis se puede hacer checkout solo de ese módulo de la misma forma:
  - cvs chekout analisis

CVS – ejemplo con línea de comandos

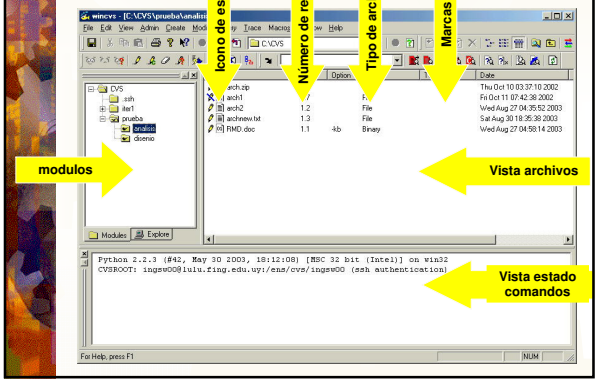
- Para agregar archivos, copiarlos al directorio correspondiente en la copia de trabajo y agregarlos para que CVS registre su existencia:
  - cvs add archnew
- Para registrar los cambios realizados en la copia de trabajo:
  - cvs commit analisis
- Si hay conflictos en algún archivo CVS lo indicará para que sean resueltos, y no permite realizar el commit hasta que hayan sido eliminados los mismos.
- Para ver conflictos en archivos de texto (código) se utiliza:
  - cvs diff arch1.ext (txt, java, etc)
- Muestra las diferencias que deben ser resueltas antes de hacer commit de los cambios.

CVS – ejemplo con WinCVS

- Estructura del repositorio igual ejemplo anterior
- Para empezar a trabajar con WinCVS se debe crear un directorio de trabajo en la máquina local y posicionarse en él:View → browse location → change
- Para hacer un checkout de todo el proyecto (desde prueba): Remote → checkout module

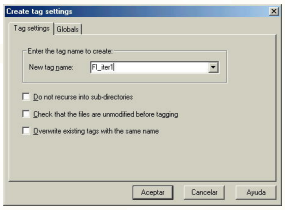


CVS – ejemplo con WinCVS



CVS – ejemplo con WinCVS

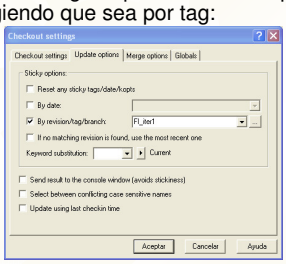
- Para marcar un versión, se seleccionan los archivos que la componen en cada subdirectorio y se van marcando:Modify → create a tag



- Posteriormente se podrá recuperar la versión utilizando el nombre con que se marcó.

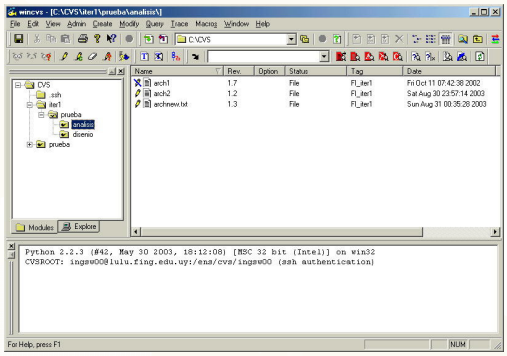
CVS – ejemplo con WinCVS

- Para recuperar una versión por nombre (tag), se hace checkout igual que antes desde prueba pero eligiendo que sea por tag:



- Recupera la versión FI_iter1 en el directorio C:\CVS\iter1

CVS – ejemplo con WinCVS



CVS – plug-in de Eclipse

- En la vista CVS Repositories → CVS Repositories se muestra el arbol existente en el repositorio
  - para hacer check out del proyecto/módulos/archivos elegidos posicionarse sobre el proyecto/módulo/archivo elegido y elegir check out
  - En la vista CVS Repositories → Navigator aparecerán el proyecto/módulo/archivos correspondientes
  - Trabajar normalmente y luego para hacer el check in correspondiente posicionarse en el proyecto/módulo/archivo y elegir Team → Synchronize with Repository
  - Aparece la vista de sincronización donde se mostrarán los conflictos si hubiera, los cuales deberán resolverse para poder hacer el commit de los cambios realizados.
- Para el trabajo con el repositorio de los implementadores de los grupos JAVA, estudiar en el help de Eclipse:
  - Workbench User Guide → Getting started → Team CVS tutorial

CVS – resumen forma de trabajo

- Hacer checkout de los archivos con los que se va a trabajar:
  - con línea de comandos al directorio del integrante en el home del usuario unix
  - con WinCVS y plug-in de Eclipse al directorio de trabajo en la máquina local
- Trabajar con los archivos y con archivos nuevos normalmente (utilizando las herramientas elegidas: editores de texto, planillas, IDE, etc)
- Copiar los archivos que se van a incluir en el repositorio (y las nuevas versiones) al directorio de trabajo de forma que CVS registre los cambios
- Agregar los nuevos con el comando add, hacer commit de cada archivo (directorio/módulo) resolviendo los conflictos que pudieran ocurrir.

Recomendaciones de uso

- Definir una estructura para el repositorio que sea simple y entendible por todo el grupo, y que permita el gestionado de versiones sin demasiado esfuerzo.
- Definir módulos para que las distintas áreas de trabajo puedan hacer checkout solo de los archivos con los que trabajan.
- Para todos los archivos que no sean fuentes almacenarlos como binarios (con la opción –kb en línea de comandos y con la opción correspondiente en WinCVS).
- Marcar versiones por iteración incluyendo todos los archivos que definen la versión en forma lógica (incluyendo los de verificación que se puedan desfasar unos días al comienzo de la iteración siguiente)

Herramientas a instalar

- Uso mediante línea de comandos:
  - Bajar el ssh de:
    http://www.ssh.com
- Uso mediante WinCVS:
  - Bajar el WinCVS, Csdiff, Python de:
    http://www.wincvs.org/download.html versión 2.0.2.3
    http://www.python.org/ Python 2.4.1
    http://www.componentsoftware.com/products/CSDiff/download.htm CSDiff v4.0
- Para WinCVS y plug-in de Eclipse:
  http://www.openssh.com

Material del curso y sitios de interés

- Para el manejo de CVS se debe estudiar la guía del curso que está para bajar en:
  - http://www.fing.edu.uy/inco/cursos/ingsoft/pis/index.htm
    material → Gestión de Configuración en Proyecto de IS
- Links de interés sobre CVS, WinCVS y plug-in Eclipse:
  - http://www.wincvs.org/
  - http://www.cvshome.org/
  - http://www.eclipse.org/
- Para la realización del Plan de SCM utilizar:
  - IEEE Standard for Software Configuration Management Plans Std 828
  - Guía para adaptación del estándar al curso: GuíaSCMP bajar en el mismo link que la guíaCVS