Instalación y Uso Básico del Sistema para Control de Versiones Subversion

Tomás Balderas Contreras

Instituto Nacional de Astrofísica, Óptica y Electrónica

Coordinación de Ciencias Computacionales

2011

Contenido

1 Introducción

El presente documento describe los conceptos generales sobre los sistemas para control de versiones y los rasgos específicos del sistema Subversion. También reporta el proceso de instalación y configuración de una infraestructura basada en Subversion que utiliza el modelo cliente-servidor y está a disposición de los estudiantes de los programas de posgrado de la Coordinación de Ciencias Computacionales del INAOE.

1.1 Control de versiones

Los sistemas para control de versiones son programas que permiten centralizar el almacenamiento de los archivos de código fuente de un proyecto de software en un único lugar llamado depósito o repositorio (repository). Los ingenieros de software pueden utilizar el sistema de control de versiones para crear copias locales del código, someter los cambios que realicen al código fuente desde las copias locales hacia el depósito, actualizar las copias locales a partir del depósito para incluir los cambios sometidos por otros desarrolladores, llevar un control de quiénes realizaron qué modificaciones, crear ramificaciones del código fuente correspondientes a nuevas versiones del mismo, e identificar condiciones en las que haya cambios incompatibles de diferentes usuarios sobre una misma sección de código.

Opciones comerciales y propietarias:

- IBM Rational ClearCase
- Microsoft Visual SourceSafe

Opciones de código abierto (open source):

- Concurrent Version System (CVS)
- Subversion (SVN)

Por muchos años el sistema CVS ha sido el estándar de facto en el proceso de desarrollo de proyectos de software de código abierto (open source). Sin embargo, con el paso del tiempo se hicieron notorias algunas desventajas en CVS y surgió la necesidad de contar con un sistema para control de versiones moderno que corrigiera los problemas existentes en CVS y fuera conceptualmente compatible con él.

1.2 Subversion

Subversion es un sistema de control de versiones moderno de código abierto que se considera sucesor de CVS, pues corrige sus limitaciones y defectos [1]. En una década, Subversion se ha convertido en el nuevo estándar para control de versiones para proyectos de software de código abierto. Las herramientas proporcionadas por Subversion se pueden invocar mediante comandos de línea o mediante una serie de interfaces gráficas con el usuario, las cuales invocan a los comandos de Subversion. Algunas de estas interfaces son:

- **TortoiseSVN:** Funciona como un plug-in al explorador de archivos de Windows y permite tener acceso a las funciones de Subversion a través de comandos en un menú contextual [2].
- **Subversive, Subeclipse:** Son dos plug-ins que se instalan sobre el ambiente de desarrollo Eclipse que permiten tener acceso a las funciones de Subversion dentro de Eclipse. Estas herramientas contribuyen a que el desarrollador pueda realizar las operaciones de modelado, codificación, y gestión de las versiones del proyecto sin abandonar Eclipse.

La Figura 1 ilustra los componentes de una infraestructura de control de versiones basada en Subversion. La línea discontinua de la parte superior indica los límites del programa cliente utilizado por los desarrolladores para crear sus copias locales y gestionar sus cambios. La línea discontinua de la parte inferior indica los límites del depósito de archivos almacenado en disco mediante algún sistema de archivos. Existen variantes del programa que accede al depósito y proporciona los servicios solicitados por los clientes.

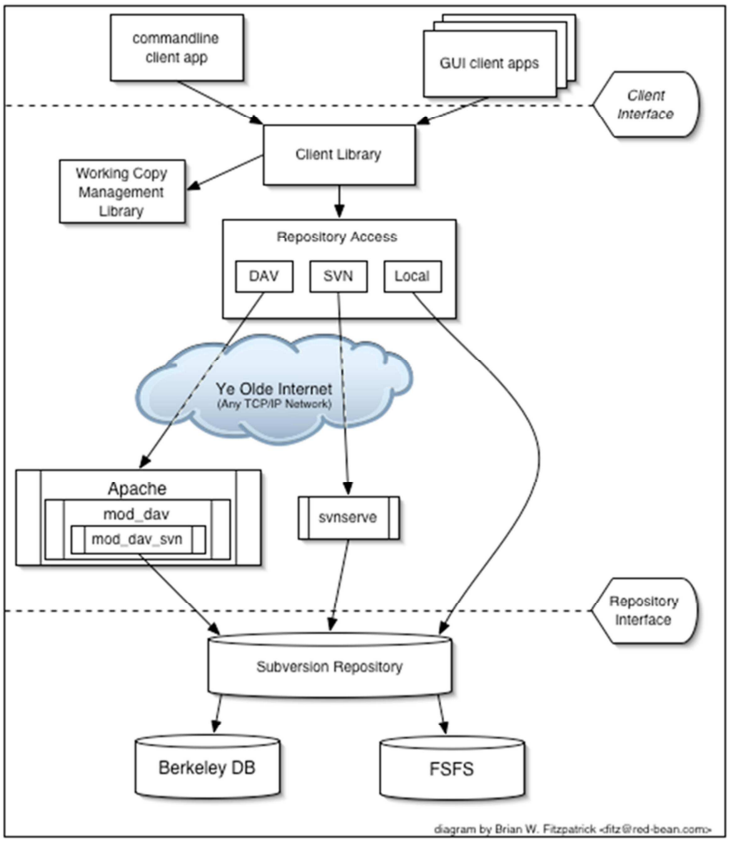


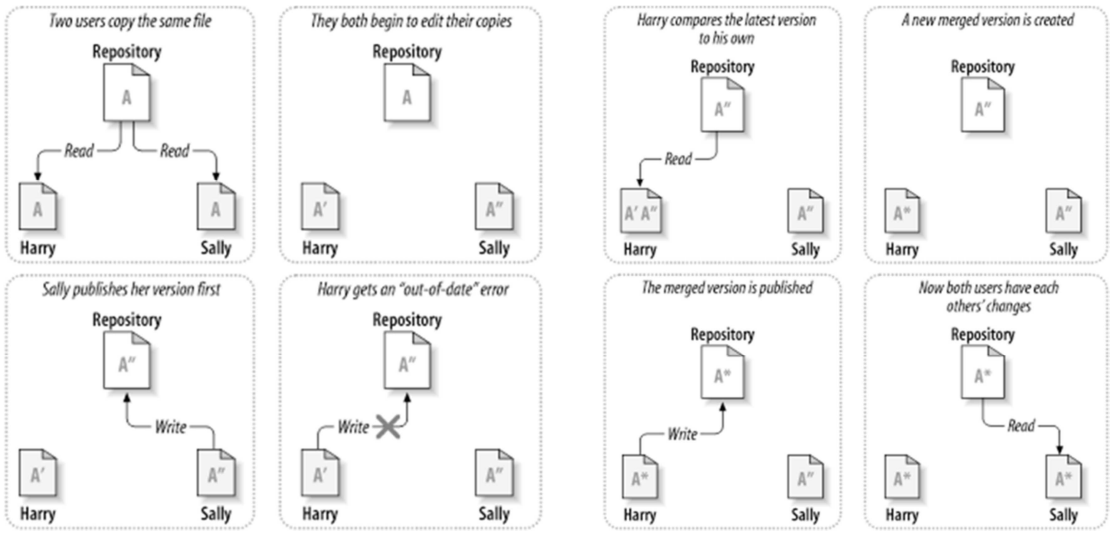
Figura 1. Infraestructura general de Subversion (de [1]).

Entre el cliente y el depósito en la infraestructura mostrada en la Figura 1 se encuentran las diferentes interfaces que permiten establecer la comunicación entre dichos componentes de la infraestructura. Existen tres mecanismos de comunicación entre los clientes utilizados por los

ingenieros de software y el repositorio. Las tres interfaces utilizadas por los clientes para acceder al depósito de archivos son las siguientes:

- **Acceso local:** El depósito de archivos y las copias locales de los proyectos se almacenan en la misma máquina. Se utiliza cuando el ingeniero de software trabaja en un proyecto no publicado y desea llevar un control de los cambios que ha hecho al sistema durante las primeras etapas de desarrollo.
- **Acceso remoto mediante svnserve:** Los ingenieros de software acceden al depósito en un servidor remoto a través de Internet. En este caso, los clientes se comunican con el programa svnserve que se encuentra en ejecución en el servidor remoto mediante un protocolo específico que permite transferir archivos del depósito hacia los clientes y someter cambios desde los clientes al repositorio.
- **Acceso remoto mediante DAV:** Permite acceder de forma remota al depósito a través de una interfaz web. Requiere que el servidor web Apache se encuentre instalado y en ejecución en la máquina que almacena el repositorio.

Ahora, analicemos la estrategia utilizada por Subversion para crear las copias locales de los clientes y gestionar los cambios realizados por ellos. La Figura 2 muestra a dos programadores (Harry y Sally) que acceden a un mismo archivo almacenado en el depósito del proyecto utilizando la infraestructura Subversion. La Figura 2(a) muestra que inicialmente Sally y Harry acceden al repositorio para crear sus copias locales del archivo A. Después, ambos realizan modificaciones a sus copias del archivo A y generan versiones modificadas de dicho archivo llamadas A' para Harry y A'' para Sally. A continuación, Sally hace públicos sus cambios mediante una operación de modificación del archivo A en el repositorio. En este momento, el archivo A en el repositorio está sincronizado con la versión de Sally (A''), pero no con la versión de Harry (A´). Por lo tanto, cuando Harry quiera hacer públicos sus cambios en el archivo local A' no lo podrá hacer pues el archivo en el repositorio contiene información que no está presente en la copia de Harry.



(a) (b)

Figura 2. Estrategia de gestión de copias locales y modificaciones (de [1]).

Antes de publicar sus cambios, Harry debe actualizar su copia local A' para que incluya los cambios publicados por Sally anteriormente. En la Figura 2(b) se muestra que la actualización se realiza mediante una operación que mezcla las modificaciones en la copia local de Harry con el archivo en el depósito que incluye las modificaciones publicadas anteriormente por Sally. Esta mezcla (merge) genera una versión A* del archivo en la copia local de Harry. En este momento, Harry puede publicar sus cambios e incorporarlos al depósito, lo que hace que el archivo en el depósito se sincronice con la versión A* en la copia local de Harry. Finalmente, Sally puede actualizar su copia local desde el depósito mediante la incorporación de las modificaciones publicadas por Harry anteriormente. Como resultado de esta actualización, las copias locales de cada usuario y el archivo en el repositorio se mantienen sincronizados.

Es posible que los cambios realizados por Harry entren en conflicto con los cambios publicados anteriormente por Sally en el momento de hacer la mezcla en la Figura 2(b). Esto puede ocurrir si tanto Sally como Harry modifican la misma sección del archivo A; por ejemplo, si ambos modifican una función llamada a() en el archivo A de diferente manera. En este caso, las herramientas proporcionadas por Subversion detectarían el conflicto y notificarían al ingeniero de software que intente publicar la última modificación. El conflicto debe ser resuelto por el equipo de desarrollo involucrado después de llegar a un consenso sobre la mejor manera de corregir la situación.

2 Instalación de Subversion

En esta sección se describe el proceso de instalación de los componentes de la infraestructura basada en Subversion, incluyendo el depósito y los clientes. El repositorio y el programa servidor (svnserve) se instalan en un servidor con las siguientes características:

- **Modelo:** Dell PowerEdge 750
- **Procesador:** Intel Pentium 4 2.8 GHz con FSB a 800 MHz con Hyperthreading
- **Disco duro:** SATA de 250 GB
- **Memoria:** 512 MB
- **Nombre del servidor:** cpulabserver.inaoep.mx
- **Dirección MAC:** 00:11:43:cd:60:0a
- **Dirección IP:** 192.100.172.85
- **Sistema operativo:** Linux Ubuntu Server 10.04

Es necesario aclarar que el depósito con todos los proyectos de software, así como el programa servidor svnserve, estarán siempre almacenados en la plataforma GNU/Linux descrita anteriormente. Sin embargo, los programas cliente y las copias locales pueden ser instalados en diferentes plataformas ejecutando una variedad de sistemas operativos, incluyendo Windows, Mac OS X, GNU/Linux, Solaris, etc.

2.1 Recursos

El proyecto que dio origen a Subversion fue iniciado por CollabNet, Inc. Actualmente, Subversion es parte de los proyectos de software mantenidos por la Apache Software Foundation y permanece como un sistema de código abierto. El código fuente de la infraestructura y su documentación están disponibles en el siguiente portal:

http://subversion.apache.org/

En la mayoría de los casos, no se desea compilar el código fuente de Subversion, sino que se necesita descargar una distribución que se pueda instalar en la plataforma deseada y comenzar a trabajar. En este caso, se pueden obtener paquetes pre-compilados con software ejecutable para sistemas operativos como AIX, HP-UX, Windows, Solaris y varias distribuciones de GNU/Linux desde el siguiente vínculo:

http://subversion.apache.org/packages.html

También es posible utilizar el comando apt-get para acceder a la herramienta de empaquetado avanzada (Advanced Packaging Tool - APT) de la distribución de GNU/Linux utilizada para descargar e instalar paquetes de software disponibles. Este medio de instalación se utilizó para instalar Subversion en el servidor que almacena el depósito de proyectos de software. Los paquetes incluyen el programa servidor svnserve y los comandos de línea que el ingeniero de software utiliza desde su terminal para gestionar su copia local y establecer comunicación hacia el repositorio de software.

En plataformas basadas en Windows es posible instalar componentes que permiten al ingeniero de software manipular su copia local y enviar modificaciones al depósito mediante el uso de menús visibles en el explorador de archivos. Uno de esos componentes es TortoiseSVN, cuyo paquete de instalación y documentación se pueden obtener del siguiente portal:

http://tortoisesvn.net/

Antes de instalar TortoiseSVN es necesario determinar si el procesador de la computadora tiene capacidad de cómputo de 64 bits o no. Si se trata de instalar la versión de 64 bits de TortoiseSVN en una computadora con un procesador Intel que sólo tiene capacidad de 32 bits, el programa de instalación abortará indicando un error.

2.2 Instalación

Para contar con una infraestructura de control de versiones basada en Subversion es necesario crear el depósito de los proyectos de software en el servidor cpulabserver.inaoep.mx e invocar al programa svnserve en dicho servidor para que permanezca en ejecución y atienda las solicitudes de los clientes. Este reporte únicamente documenta la instalación de servidores Subversion en sistemas basados en UNIX, específicamente el sistema operativo GNU/Linux, distribución Ubuntu Server.

También, es necesario instalar clientes Subversion en las terminales utilizadas por los ingenieros de software para crear y manipular copias locales. En este reporte se documenta la instalación y configuración de clientes Subversion que se pueden ejecutar en sistemas operativos Windows, UNIX y Mac OS X.

2.2.1 Servidor

Los siguientes pasos para la instalación del servidor de Subversion requieren privilegios de super-usuario en el servidor cpulabserver.inaoep.mx. Estos privilegios pueden adquirirse al ingresar a la cuenta root del sistema o al ingresar a la cuenta de algún usuario en el grupo de administradores del sistema y utilizar el comando sudo. En lo que resta de este documento se ilustra el proceso de instalación obteniendo privilegios de súper-usuario con el comando sudo. El único pre-requisito es que la persona que realice la instalación pertenezca al grupo de administradores del sistema y sea capaz de autenticarse utilizando una contraseña UNIX válida. La instalación y configuración mediante la cuenta root se realiza con los mismos comandos, pero sin la invocación al comando sudo.

Como se mencionó anteriormente, para instalar el paquete Subversion en Ubuntu Server en el servidor cpulabserver.inaoep.mx se utilizará el comando apt-get con atributos de súper-usuario adquiridos mediante sudo:

```
balderas@cpulabserver:~$ sudo apt-get install subversion
[sudo] password for balderas:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

La instantánea anterior ilustra el uso de la opción install del comando apt-get para descargar e instalar el paquete subversion. Ya que el comando debe ser ejecutado con permisos de súper-usuario, se utiliza el comando sudo para adquirir dichos privilegios. Antes de invocar la ejecución de apt-get, sudo realiza la autenticación del usuario solicitando que proporcione su contraseña del sistema. Si sudo autentica al usuario correctamente, entonces apt-get procede a descargar los paquetes solicitados para instalar el software correspondiente. Cuando apt-get intenta instalar un paquete, es posible que determine que algún software requerido no está instalado aún; en éste caso el usuario debe instalar dicho software antes de continuar con la instalación inicial.

2.2.2 Cliente

El paso de instalación anterior empleando sudo y apt-get se puede utilizar también para instalar los comandos utilizados por el sistema cliente para crear y gestionar las copias locales de un proyecto. Por lo tanto, para instalar el cliente de la infraestructura Subversion en una plataforma con GNU/Linux basta con repetir el proceso anterior.

La distribución de Subversion para Mac OS X también incluye todas las herramientas para configurar un servidor de Subversion y para trabajar en las terminales cliente. Antes de instalar, es necesario elegir el paquete adecuado de acuerdo al procesador del sistema con el que se cuenta (Intel x86 o PowerPC) y la versión del sistema operativo Mac OS X instalada (10.5, 10.6 o 10.7). El

proceso de instalación es simple y sólo requiere aceptar los términos de licencia y la autenticación del administrador para permitir los cambios al sistema. La Figura 3 ilustra el proceso de instalación de la versión 1.6.17-1 de la distribución de Subversion liberada por CollabNet, Inc en una computadora con Mac OS X 10.7 (Lion).

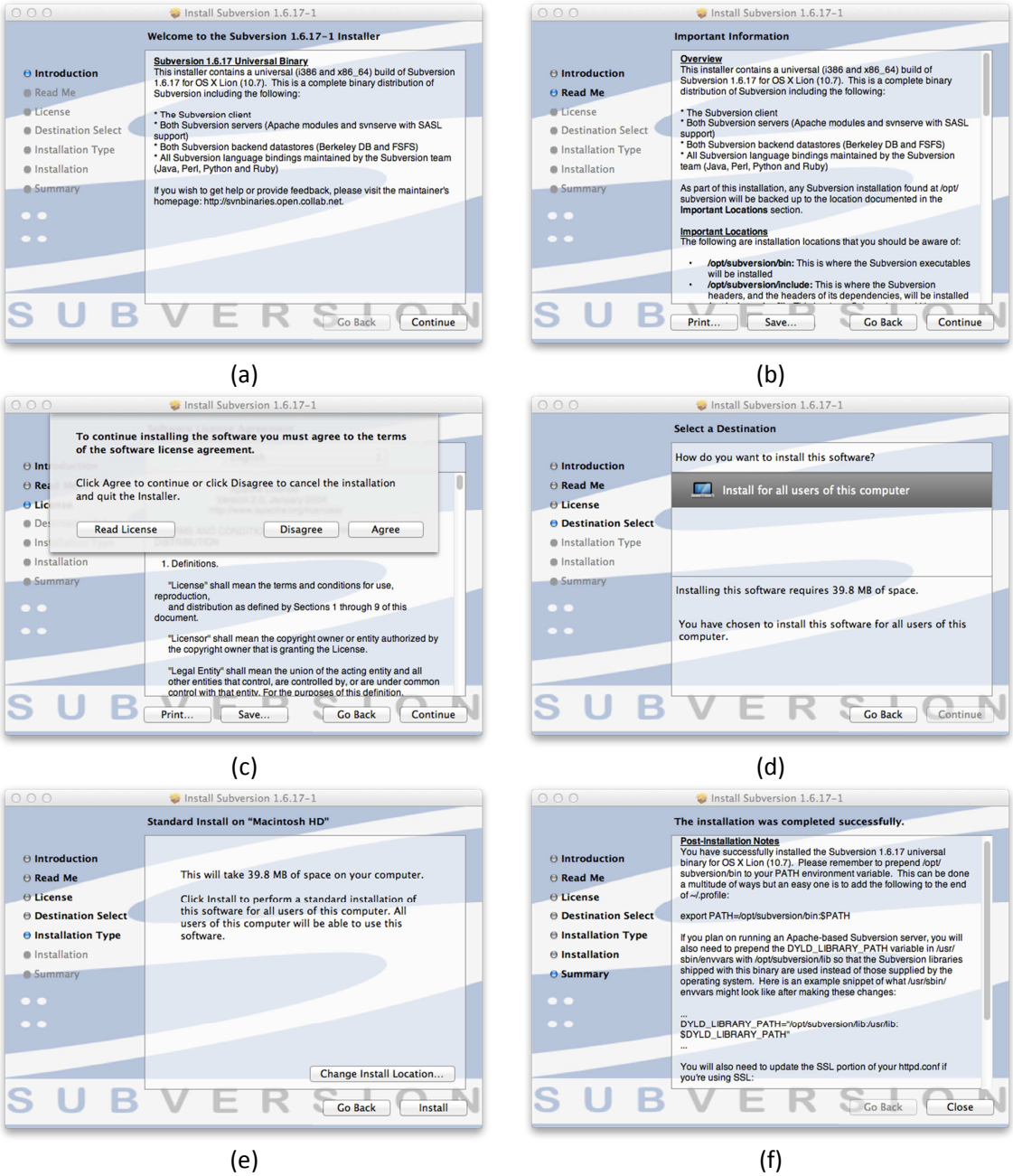


Figura 3. Instalación de la distribución de Subversion para Mac OS X.

La Figura 4 ilustra el proceso de instalación del cliente TortoiseSVN versión 1.7.1. El procedimiento es bastante convencional y similar a la instalación de cualquier otro software en Windows. Es necesario simplemente aceptar los acuerdos de licitación. En la Figura 4(c) se indica que se

instalen las herramientas en línea de comandos, que por omisión no se instalan. Después de concluida la instalación es posible utilizar las herramientas proporcionadas por TortoiseSVN para gestionar las copias locales mediante el explorador de archivos en Windows.

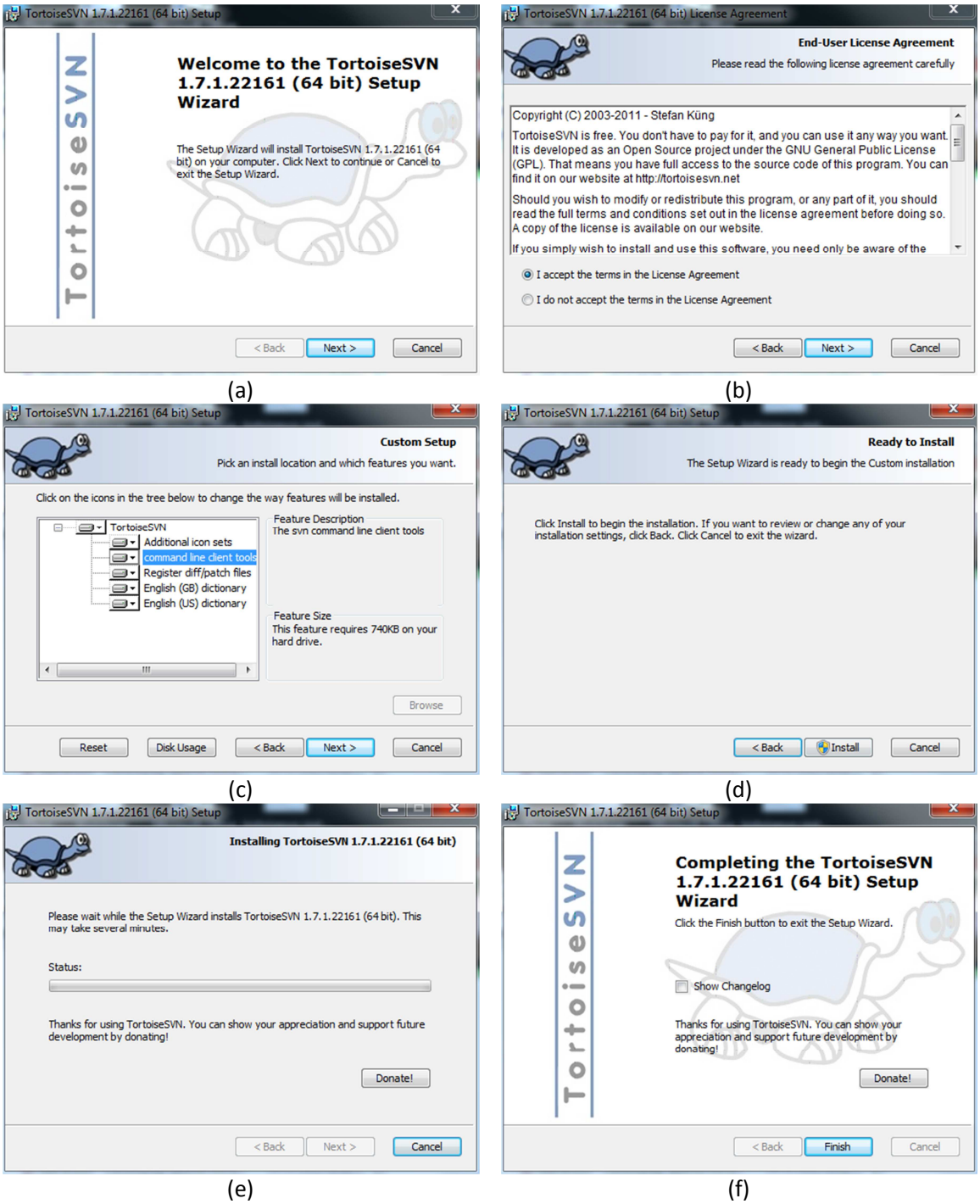


Figura 4. Proceso de instalación de TortoiseSVN.

3 Configuración de Subversion

En esta sección se describe el proceso para colocar y configurar el directorio en el servidor cpulabserver.inaoep.mx que almacena los repositorios para los proyectos. Esta descripción incluye los archivos de configuración que se deben modificar para otorgar permisos de lectura y escritura a los archivos en cada repositorio, así como sugerencias sobre la organización de la jerarquía de directorios para los repositorios. Finalmente, se describe cómo iniciar el proceso svnserve que atiende las solicitudes de los clientes y accede a los archivos en los repositorios.

3.1 Configuración del repositorio

El directorio que contiene los repositorios está localizado en el sistema de archivos del servidor cpulabserver.inaoep.mx en la ruta /usr/local/svn/repositories/, para crear este directorio se utiliza el comando mkdir con privilegios de súper-usuario adquiridos mediante el comando sudo:

```
balderas@cpulabserver:~$ sudo mkdir /usr/local/svn/repositories
[sudo] password for balderas:
```

Después de autenticar al administrador, sudo invoca a mkdir para crear el directorio que contendrá los repositorios para los proyectos de software. En algunas instalaciones los repositorios se colocan en el directorio /home/svn/ y se crea una cuenta especial (svn) para realizar la gestión de dicho directorio. En nuestro caso, los repositorios se colocan en /usr/local/ para no colocarlos en el mismo directorio que contiene la información de los usuarios (/home/).

A continuación, se restringe el acceso a los repositorios a los usuarios que forman parte del grupo svn para evitar modificaciones no controladas en dichos archivos. Esto se realiza creando el grupo svn, estableciendo que los archivos en los repositorios pertenecen al grupo svn y anexando a los usuarios adecuados a dicho grupo. Los comandos necesarios para realizar lo anterior son los siguientes:

```
balderas@cpulabserver:~$ sudo groupadd svn
balderas@cpulabserver:~$ sudo chgrp svn /usr/local/svn/repositories
balderas@cpulabserver:~$ sudo chmod g+w /usr/local/svn/repositories
balderas@cpulabserver:~$ sudo chmod g+s /usr/local/svn/repositories
balderas@cpulabserver:~$ sudo usermod –a –G svn balderas
```

El comando groupadd crea el grupo svn y el comando chgrp establece que los integrantes del grupo svn son propietarios del directorio /usr/local/svn/repositories/. La primera invocación al comando chmod establece permisos de escritura para los miembros del grupo svn. La segunda invocación a chmod establece que el grupo svn va a ser propietario de todo archivo y directorio creado dentro del directorio /usr/local/svn/repositories/. Finalmente, la invocación usermod anexa la cuenta de un usuario existente (en este caso balderas) al grupo svn, es necesario invocarlo tantas veces como usuarios sea necesario añadir a dicho grupo.

La anexión de una cuenta de usuario a un grupo se hace efectiva hasta la siguiente sesión del usuario. Un usuario debe cerrar todas sus sesiones abiertas e ingresar al sistema con una sesión distinta para que su pertenencia al grupo svn se lleve a cabo. Para cerciorarse de que ha sido anexado al grupo svn el usuario puede invocar al comando groups de la siguiente manera:

```
balderas@cpulabserver:~$ groups
balderas adm dialout cdrom plugdev lpadmin sambashare admin svn
```

En el ejemplo anterior la cuenta de usuario balderas es parte de los grupos svn y admin. La pertenencia al grupo admin le permite al usuario realizar actividades con privilegios de súper-usuario mediante el comando sudo.

Hasta ahora se ha configurado el directorio raíz que contendrá los repositorios. Dentro de dicho directorio se crea un sub-directorio para cada repositorio que va a ser controlado por Subversion. Cuando se crean dichos sub-directorios, mediante el programa svnadmin, se establece una estructura de archivos y directorios especial. Los repositorios son manipulados exclusivamente por las herramientas de Subversion cuando se anexan nuevos archivos al proyecto, cuando se eliminan archivos o cuando se incorporan cambios a dichos archivos. Ni el administrador ni algún otro usuario deben alterar manualmente la estructura de un repositorio.

Antes de crear el sub-directorio para un nuevo repositorio, es necesario establecer que los miembros del grupo svn tienen permiso de modificar los archivos que conforman dicho sub-directorio mediante el comando umask. El sub-directorio para el nuevo repositorio se crea mediante el comando svnadmin, que es una de las herramientas proporcionadas por Subversion. Finalmente, es necesario restablecer los permisos por omisión volviendo a invocar al comando umask. El siguiente ejemplo ilustra los pasos anteriores para crear un nuevo repositorio llamado test en el directorio /usr/local/repositories/:

```
balderas@cpulabserver:~$ umask 002
balderas@cpulabserver:~$ svnadmin create /usr/local/svn/repositories/test
balderas@cpulabserver:~$ umask 022
balderas@cpulabserver:~$ ls -l /usr/local/svn/repositories/
total 12
drwxrwsr-x 6 balderas svn 4096 2011-11-22 06:26 hellow
drwxrwsr-x 6 balderas svn 4096 2011-11-23 05:07 rna
drwxrwsr-x 6 balderas svn 4096 2011-12-08 03:41 test
```

La cuenta de usuario que invoca a svnadmin puede crear un repositorio en el directorio /usr/local/svn/repositories/ porque pertenece al grupo svn. En este momento se cuenta con un repositorio vacío que no contiene ningún archivo del proyecto de software pero que contiene otros archivos necesarios para la gestión de dicho repositorio, como se muestra a continuación:

```
balderas@cpulabserver:~$ ls -l /usr/local/svn/repositories/test/
total 24
drwxrwsr-x 2 balderas svn 4096 2011-12-08 03:41 conf
drwxrwsr-x 6 balderas svn 4096 2011-12-08 03:41 db
-r--r--r-- 1 balderas svn    2 2011-12-08 03:41 format
drwxrwsr-x 2 balderas svn 4096 2011-12-08 03:41 hooks
drwxrwsr-x 2 balderas svn 4096 2011-12-08 03:41 locks
-rw-rw-r-- 1 balderas svn  229 2011-12-08 03:41 README.txt
balderas@cpulabserver:~$
```

Antes de mostrar cómo añadir archivos de código fuente al repositorio es necesario ilustrar cómo permitir a los usuarios realizar modificaciones al repositorio mediante las herramientas cliente en

Subversion. Vamos a crear un archivo de contraseñas global que especifica qué usuarios tienen acceso a los repositorios colocados en el directorio /usr/local/svn/repositories/, dicho archivo se coloca en el directorio /usr/local/svn/. Para crear el archivo de contraseñas para los miembros del equipo se utiliza cualquier editor de texto con privilegios de súper-usuario, en este caso emacs:

```
balderas@cpulabserver:~$ sudo emacs /usr/local/svn/passwd-team
[sudo] password for balderas:
```

El archivo passwd-team incluye una sección llamada users que contiene un conjunto de entradas compuestas por un identificador para cada usuario y su contraseña. El siguiente es un ejemplo del contenido de un archivo passwd-team:

```
[users]
balderas = svnpassword
harry = harrysvnpassword
sally = sallysvnpassword
```

Debido a que las contraseñas se almacenan sin ningún tipo de cifrado es necesario proteger el archivo passwd-team para evitar que pueda ser leído por cualquier persona. El siguiente comando establece que el archivo passwd-team sólo puede ser leído y modificado por el super-usuario del sistema (root):

```
balderas@cpulabserver:~$ sudo chmod 600 /usr/local/svn/passwd-team
[sudo] password for balderas:
```

Ahora es necesario modificar el archivo de configuración svnserve.conf en el repositorio test recién creado. Dicho archivo está ubicado en el directorio /usr/local/svn/repositories/test/conf/. Utilizando un editor de texto abrir el archivo svnserve.conf para modificar su configuración:

```
balderas@cpulabserver:~$ sudo emacs
/usr/local/svn/repositories/test/conf/svnserve.conf
[sudo] password for balderas:
```

El archivo svnserve.conf tiene diferentes secciones y varios comentarios que documentan las opciones contenidas en cada sección. Las modificaciones necesarias se realizan en las opciones de la sección general del archivo svnserve-conf y están indicadas por la siguiente instantánea:

```
[general]
anon-access = none
password-db = /usr/local/svn/passwd-team
realm = Team
```

Las opciones anteriores definen qué usuarios de Subversion tienen acceso al repositorio test y cómo autenticarlos. También establecen que el acceso de usuarios anónimos al repositorio no está permitido. La línea que define password-db indica que los usuarios a los que se concede acceso al repositorio test son aquellos listados, junto con sus contraseñas, en el archivo passwd-team definido anteriormente. El valor asignado a anon-access prohíbe el acceso al repositorio test de usuarios no autenticados. La última opción establece el dominio de autenticación.

Es conveniente mencionar que en cada repositorio existe otro archivo que contiene una base de datos de identificadores y usuarios. En nuestro caso, este archivo se llama passwd y está localizado en el directorio /usr/local/svn/repositories/test/conf/. La siguiente instantánea muestra el contenido por omisión del archivo passwd:

```
### This file is an example password file for svnserve.
### Its format is similar to that of svnserve.conf. As shown in the
### example below it contains one section labelled [users].
### The name and password for each user follow, one account per line.

[users]
# harry = harryssecret
# sally = sallyssecret
```

Para usar este archivo simplemente es necesario reemplazar la ruta y el nombre del archivo passwd-team en la línea que establece el valor de la opción password-db en svnserve.conf por la siguiente línea:

```
password-db = passwd
```

En este caso, la literatura no indica que sea necesario establecer el dominio de autenticación, por lo que la línea que asigna un valor a la propiedad realm en svnserve.conf podría estar comentada.

3.2 Ejecución de svnserve

Ahora que los directorios de los repositorios para cada proyecto están configurados y que los permisos de acceso para los usuarios de los repositorios están definidos, es posible ejecutar el programa que atiende las solicitudes de los clientes para manipular los repositorios. Este programa se llama svnserve y debe permanecer siempre en ejecución en el servidor cpulabserver.inaoep.mx esperando por las conexiones remotas de los clientes.

Se tienen que tomar en cuenta las siguientes consideraciones antes de invocar a svnserve y que entre en ejecución:

1. svnserve debe ser ejecutado por un proceso al fondo (daemon) que no interfiere con las acciones de los usuarios pero que está al pendiente de las solicitudes de los clientes que llegan a través del puerto 3690.
2. svnserve debe tener en cuenta que el directorio donde se encuentran los repositorios es /usr/local/svn/repositories/. Los clientes que se comunican con svnserve no deben saber la ubicación exacta de los repositorios en el servidor. Los clientes indican a svnserve únicamente el nombre del repositorio que desean manipular y este nombre se mapea a una ruta relativa a la ruta donde se encuentran los repositorios.
3. El proceso que ejecuta svnserve debe ser invocado cada vez que se reinicia el sistema operativo del servidor cpulabserver.inaoep.mx. De esta forma se garantiza que la infraestructura de Subversion estará siempre disponible.

La forma de invocar a svnserve para tomar en cuenta los requerimientos 1 y 2 es la siguiente:

```
balderas@cpulabserver:~$ svnserve -d -r /usr/local/svn/repositories/
```

Con la opción –d se indica que svnserver se debe ejecutar como un proceso al fondo, mientras que con la opción –r se establece la ruta en donde están colocados los repositorios y que se utilizará para determinar la ruta efectiva de cada uno.

Para indicar que svnserve se ejecute siempre al término del proceso de inicio del sistema operativo es necesario programar el despachador cron para que inicie un proceso para svnserve. cron permite a los usuarios programar la ejecución de diversas tareas periódicamente en ciertos instantes de tiempo; en este caso, cron debe invocar a svnserve siempre que el sistema operativo se reinicie (reboot). Para realizar dicha programación es necesario añadir una entrada a la tabla de cron (crontable) indicando qué tarea se debe ejecutar (svnserve) y en qué momento se desea que se ejecute (reboot). Para modificar la tabla de cron utilizando el editor de preferencia con privilegios de súper usuario es necesario invocar el siguiente comando:

```
balderas@cpulabserver:~$ sudo crontab –e
[sudo] password for balderas:
```

En la tabla de cron hay que añadir la siguiente línea y salvar el archivo:

```
@reboot svnserve -d -r /usr/local/svn/repositories
```

Esta última modificación permitirá que svnserve sea invocado siempre que se reinicie el sistema operativo y que el servidor de Subversion atienda las solicitudes de los clientes.

Las últimas consideraciones respecto al repositorio tienen que ver con la estructura del directorio que contiene los repositorios. Hasta el momento se ha considerado que dicho directorio contiene uno o más repositorios para diferentes proyectos, como se muestra a continuación:

```
balderas@cpulabserver:~$ ls -l /usr/local/svn/repositories/
total 12
drwxrwsr-x 6 balderas svn 4096 2011-11-22 06:26 hellow
drwxrwsr-x 6 balderas svn 4096 2011-11-23 05:07 rna
drwxrwsr-x 6 balderas svn 4096 2011-12-08 03:41 test
```

¿Cómo podría organizarse un directorio que contenga repositorios pertenecientes a dos organizaciones distintas? Una opción es colocar todos los repositorios en el directorio base /usr/local/svn/repositories/ e identificar la organización a la que pertenecen con una clave en el nombre del repositorio. Otra solución es crear dos o más subdirectorios dentro del directorio base, uno para cada organización de la institución, y colocar en cada sub-directorio todos los repositorios asociados a los proyectos de la organización correspondiente. En ambos casos el programa svnserve debe invocarse sin modificaciones en la ruta base de los repositorios; especificar la ruta de un subdirectorio podría traer como consecuencia descartar otros repositorios en otro directorio. Organizar los repositorios en sub-directorios de la ruta base puede ser una buena solución, pero a costa de escribir el complemento de la ruta base necesaria para llegar al repositorio deseado en alguna operación de modificación o lectura.

Como ejemplo suponga que el directorio /usr/local/svn/repositories/ contiene dos sub-directorios llamados orga y orgb, y que cada uno de estos sub-directorios contiene varios repositorios. Tendríamos una organización de la siguiente manera:

```
/usr/local/svn/repositories/
                            orga/
                                  project1/
                                  project2/
                            orgb/
                                  project1/
                                  project2/
```

Si svnserve se invoca especificando la ruta base de los repositorios como /usr/local/svn/repositories/ entonces un desarrollador que desee acceder al repositorio project1 de la organización A deberá indicar a su cliente que debe acceder al repositorio colocado en /orga/project1/, que es una ruta relativa a la ruta especificada durante la invocación de svnserve.

4 Uso básico de Subversion

Esta sección ilustra cómo manipular el contenido de un repositorio a través de las herramientas cliente de Subversion, utilizadas por los ingenieros de software durante el proceso de desarrollo de un proyecto de software. Se muestra cómo anexar archivos a un repositorio vacío, crear copias locales de los archivos en un repositorio, enviar cambios realizados en una copia local al repositorio, actualizar una copia local desde el repositorio y cómo eliminar archivos en el repositorio. Se ilustra cómo realizar las operaciones anteriores mediante comandos en línea en un sistema basado en UNIX puesto que es la forma más común de utilizar Subversion en proyectos de fuente abierta.

4.1 Adición de archivos en un repositorio vacío

En la sección anterior se creó el sub-directorio para el repositorio del proyecto test, el cual no contiene archivos de código fuente en este momento. El procedimiento para anexar archivos de código fuene en el repositorio es el siguiente:

1. Crear una copia local del repositorio test en una terminal remota mediante un cliente de Subversion.
2. Crear los archivos de código fuente en la copia local.
3. Enviar los archivos de código fuente creados anteriormente al repositorio.

El paso 1 crea una copia local en la terminal del ingeniero de software que va a trabajar en el proyecto test. Dicha copia local consite en un directorio llamado test/ que inicialmente no contiene archivos de código fuente sino archivos utilizados por las herramientas de Subversion para gestionar la copia local y la sincronización con el servidor. La creación de la copia local se realiza con la opción checkout del comando svn como se muestra a continuación:

```
$ svn checkout svn://cpulabserver.inaoep.mx/test --username harry
Authentication realm: <svn://cpulabserver.inaoep.mx:3690> Team
```

```
Password for 'harry':
Checked out revision 0.
$ ls –l
total 0
drwxr-xr-x  3 harry staff  102 Dec 12 09:41 test
$ ls -la ./test/
total 0
drwxr-xr-x  3 harry  staff  102 Dec 12 09:41 .
drwxr-xr-x  4 harry  staff  136 Dec 12 09:41 ..
drwxr-xr-x  7 harry  staff  238 Dec 12 09:41 .svn
```

La invocación de la operación checkout para crear la copia local requiere indicar la URL del repositorio test, la cual está formada por el nombre y dominio del servidor de Subversion (cpulabserver.inaoep.mx) y por el nombre del repositorio a extraer (test). También es necesario indicar en nombre de qué usuario se realiza dicha operación mediante la directiva --username seguida del identificador de usuario. Los identificadores de usuario que se pueden proporcionar al comando svn son aquellos que están listados en el archivo /usr/local/svn/passwd-team descrito en la sección anterior. El comando svn trata de autenticar al usuario harry solicitando su contraseña y verificando si ésta coincide con la almacenada en el archivo /usr/local/svn/passwd-team. Si la autenticación es exitosa entonces svn crea el directorio test/ en el sistema utilizado por harry, el cual contiene inicialmente un sub-directorio oculto llamado .svn de uso interno de svn.

Ahora anexemos archivos de código fuente a la copia local en el directorio test/. Anexemos dos archivos llamados test.c y Makefile; el primer archivo contiene código en lenguaje C y el segundo dirige la compilación del archivo test.c. El contenido de test.c es el siguiente:

```
#include <stdio.h>

main () {
  printf("Hello World!\n");

  return 0;
}
```

El contenido de Makefile es el siguiente:

```
test:  test.c
       gcc -o test test.c
```

Después de anexar los archivos, el contenido de la copia local queda como sigue:

```
$ ls -la ./test/
total 16
drwxr-xr-x  5 harry  staff  170 Dec 12 10:11 .
drwxr-xr-x  4 harry  staff  136 Dec 12 09:41 ..
drwxr-xr-x  7 harry  staff  238 Dec 12 09:41 .svn
-rwxr-xr-x  1 harry  staff   33 Dec 12 10:11 Makefile
-rwxr-xr-x  1 harry  staff  295 Dec 12 10:11 test.c
```

En este momento se puede compilar el programa en C en la copia local y generar un archivo ejecutable como se muestra a continuación:

```
$ make
gcc -o test test.c
$ ls –la
total 40
drwxr-xr-x  6 harry  staff   204 Dec 12 10:33 .
drwxr-xr-x  4 harry  staff   136 Dec 12 09:41 ..
drwxr-xr-x  7 harry  staff   238 Dec 12 09:41 .svn
-rwxr-xr-x  1 harry  staff    33 Dec 12 10:11 Makefile
-rwxr-xr-x  1 harry  staff  8688 Dec 12 10:33 test
-rwxr-xr-x  1 harry  staff    73 Dec 12 10:29 test.c
$ ./test
Hello World!
```

Ahora que se ha comprobado que el programa en test.c es correcto y que el archivo Makefile construye el programa ejecutable de manera correcta, se deben enviar los dos archivos al repositorio test en el servidor. Para añadir los archivos al repositorio vacío se emplea la opción add del comando svn para cada archivo y enseguida la opción commit, como se muestra a continuación:

```
$ svn add test.c
A         test.c
$ svn add Makefile
A         Makefile
$ svn commit -m "test.c and Makefile have been added to the repository"
Adding         Makefile
Adding         test.c
Transmitting file data ..
Committed revision 1.
```

Hay que notar que la comunicación entre el cliente de Subversion y el servidor para llevar a cabo la transferencia de los archivos se establece hasta el momento en que se invoca la operación commit. Como el repositorio test estaba inicialmente vacío, la transferencia de los archivos test.c y Makefile conforman la primera revisión del proyecto en el repositorio. Finalmente, siempre que se realice alguna modificación en el repositorio mediante la operación commit es necesario describir de forma breve y sucinta en qué consiste dicho cambio. El comando svn permite especificar una descripción de los cambios realizados mediante la directiva –m; la descripción se incorpora al registro de cambios en el repositorio gestionado por Subversion.

4.2 Envío de cambios en la copia local al repositorio

En la cláusula anterior, el usuario henry añadió los archivos test.c y Makefile al repositorio y los hizo públicos a otros usuarios, quienes al crear sus copias locales descargarán las primeras revisiones de los dos archivos. En este apartado se ilustra cómo el usuario sally crea su propia copia local de los archivos, realiza y publica sus modificaciones al archivo test.c, y cómo el usuario harry incorpora los cambios de sally a su copia local.

Al igual que el usuario harry, el usuario sally utiliza la opción checkout del comando svn en su terminal remota para obtener los archivos contenidos en el repositorio. La siguiente instantánea ilustra el uso de svn por sally y el contenido de la copia local generada:

```
$ svn checkout svn://cpulabserver.inaoep.mx/test --username Sally
Authentication realm: <svn://cpulabserver.inaoep.mx:3690> Team
Password for 'sally':
A    test/test.c
A    test/Makefile
Checked out revision 1.
$ ls -la ./test/
total 16
drwxr-xr-x  5 sally  staff  170 Dec 12 15:26 .
drwxr-xr-x  3 sally  staff  102 Dec 12 15:25 ..
drwxr-xr-x  7 sally  staff  238 Dec 12 15:26 .svn
-rwxr-xr-x  1 sally  staff   33 Dec 12 15:26 Makefile
-rwxr-xr-x  1 sally  staff   73 Dec 12 15:26 test.c
```

Después de ejecutar svn con la opción checkout y autenticarse, el usuario sally ha creado su copia local y ha descargado la primera revisión de los archivos test.c y Makefile. En este momento tanto las copias locales de los usuarios harry y sally como el repositorio se encuentran sincronizados. Ahora, supongamos que el usuario sally introduce una nueva línea de código y modifica el archivo test.c de la siguiente manera:

```
#include <stdio.h>

main () {
  printf("Hello World!\n");
  printf("Hello World! ... again\n");

  return 0;
}
```

Después de compilar el nuevo código y verificar que hace lo que espera, el usuario sally decide enviar el archivo modificado test.c al repositorio. La siguiente instantánea ilustra el uso de la opción commit del comando svn para enviar los cambios en el archivo al repositorio.

```
$ svn commit -m "A new printf sentence was added to test.c"
Sending        test.c
Transmitting file data .
Committed revision 2.
```

El repositorio se actualiza con los cambios sometidos por el usuario sally y Subversion crea la segunda revisión, como se indica al final de la transmisión del archivo modificado.

En este momento, el repositorio y la copia local del usuario sally se encuentran sincronizados, pero la copia local del usuario harry se ha vuelto obsoleta pues aún contiene la primera revisión del proyecto. Para actualizar su copia local a la revisión más reciente, el usuario harry debe invocar al comando svn con la opción update como se muestra a continuación:

```
$ svn update
U    test.c
Updated to revision 2.
$ cat test.c
#include <stdio.h>
```

```
main () {
  printf("Hello World!\n");
  printf("Hello World! ... again\n");

  return 0;
}
```

Es así como las dos copias locales y el repositorio se sincronizan a la misma revisión. Cada miembro del equipo de desarrollo puede invocar a la opción update del comando svn cada día antes de comenzar su trabajo para incorporar en su copia local las modificaciones sometidas por sus compañeros el día anterior.

4.3 Conflictos en las modificaciones de los usuarios

¿Qué ocurre cuando dos usuarios realizan cambios en el mismo lugar de un mismo archivo y usan Subversion para enviar dichas modificaciones al repositorio? En este caso puede ocurrir un conflicto. Aunque Subversion es capaz de detectar este tipo de situaciones conflictivas, deja a los usuarios decidir cuál es la mejor manera de arreglarlas.

Supongamos que el usuario harry añade una sentencia al archivo test.c en su copia local de la siguiente forma:

```
#include <stdio.h>

main () {
  printf("Hello World!\n");
  printf("Hello World! ... again\n");
  printf("Hello World! ... yet again!\n");

  return 0;
}
```

En otro momento, el usuario sally modifica el archivo test.c de su copia local de la siguiente manera:

```
#include <stdio.h>

main () {
  printf("Hello World!\n");
  printf("Hello World! ... again\n");
  printf("Good bye World!");

  return 0;
}
```

Notar que ambas modificaciones ocurren en el mismo lugar del archivo test.c y son contradictorias. Si ambos usuarios desean publicar sus modificaciones, Subversion eventualmente detectará una situación de conflicto entre las modificaciones realizadas por los usuarios.

Supongamos que el usuario sally es quien envía sus modificaciones al repositorio primero, por lo que invoca a la opción commit del comando svn como se muestra a continuación:

```
$ svn commit -m "A new printf statement has been added to test.c"
Sending        test.c
Transmitting file data .
Committed revision 3.
```

Subversion incorpora las modificaciones del usuario sally al archivo test.c en el repositorio y genera la tercera revisión del proyecto con dichos cambios. Ahora, el usuario harry intenta someter sus cambios al repositorio sin saber que anteriormente sally sometió sus modificaciones:

```
$ svn commit -m "A new printf statement has been added to test.c"
Sending        test.c
Transmitting file data .svn: Commit failed (details follow):
svn: File '/test.c' is out of date
```

Subversión determina que la copia local del usuario harry (revisión 2) no está sincronizada con el contenido del repositorio (revisión 3), por lo que le envía un mensaje al usuario indicando dicha situación. En este momento, el usuario harry invoca a la operación update del comando svn para actualizar el archivo test.c en su copia local para que incorpore los cambios existentes en el repositorio. La operación de actualización intentaría mezclar el contenido del archivo test.c en la copia local con el contenido del mismo archivo en el repositorio, pero detectaría el conflicto.

```
$ svn update
Conflict discovered in 'test.c'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```

Si seleccionamos la opción df para visualizar la ubicación del conflicto y las líneas de código en conflicto, Subversion nos proporciona la siguiente información:

```
        (s) show all options: df
--- .svn/text-base/test.c.svn-base     Mon Dec 12 16:16:33 2011
+++ .svn/tmp/test.c.tmp   Mon Dec 12 17:12:13 2011
@@ -3,6 +3,11 @@
 main () {
   printf("Hello World!\n");
   printf("Hello World! ... again\n");
+<<<<<<< .mine
+  printf("Hello World! ... yet again!\n");
+=======
+  printf("Good bye World!");
+>>>>>>> .r3

   return 0;
 }
Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```

Subversion lista el código en test.c en la copia local, resalta las líneas de código que entran en conflicto con las de dicho archivo en el repositorio y solicita una acción. Si seleccionamos posponer

entonces lo que Subversion producirá en la copia local es un archivo test.c que resalta el conflicto, en espera de ser arreglado mediante consenso entre los desarrolladores:

```
Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: p
C    test.c
Updated to revision 3.
Summary of conflicts:
  Text conflicts: 1
$ cat test.c
#include <stdio.h>

main () {
  printf("Hello World!\n");
  printf("Hello World! ... again\n");
<<<<<<< .mine
  printf("Hello World! ... yet again!\n");
=======
  printf("Good bye World!");
>>>>>>> .r3

  return 0;
}
```

Si la acción correctiva indica que los cambios sometidos por el usuario sally que conforman la revisión 3 del proyecto son los más adecuados entonces el usuario harry deberá modificar el archivo test.c en su copia local para que coincida con el archivo test.c en la revisión 3 en el repositorio.

4.4 Gestión de copias locales usando TortoiseSVN

Las operaciones descritas en la cláusula anterior requieren proporcionar comandos de línea en una terminal de un sistema basado en UNIX. En ambientes Windows existe la posibilidad de realizar dichas operaciones mediante el cliente TortoiseSVN, que es una interfaz visual que permite gestionar copias locales utilizando el explorador de archivos de Windows. Este apartado describe cómo manipular una copia local utilizando TortoiseSVN. El único pre-requisito para esta sección es haber instalado TortoiseSVN como se describe en la sección 2.

4.5 Creación de una copia local a partir del repositorio

La Figura 5 ilustra las operaciones proporcionadas por TortoiseSVN disponibles en el menú contextual del explorador de archivos que se despliega al hacer click con el botón derecho del ratón sobre el explorador de archivos. Para crear una copia local y descargar los archivos de código del proyecto test se deben realizar los siguientes pasos:

1. Elegir el comando **SVN Checkout…** del menú contextual en el explorador de archivos en la ruta donde se desea colocar la copia local, ver la Figura 5.
2. En el cuadro de diálogo para la operación checkout desplegado a continuación, indicar la URL del repositorio del proyecto que se desea acceder y la ruta donde se colocará la copia

local. Para crear una copia local a partir del repositorio del proyecto test se debe especificar la URL svn://cpulabserver.inaoep.mx/test. Ver Figura 6.

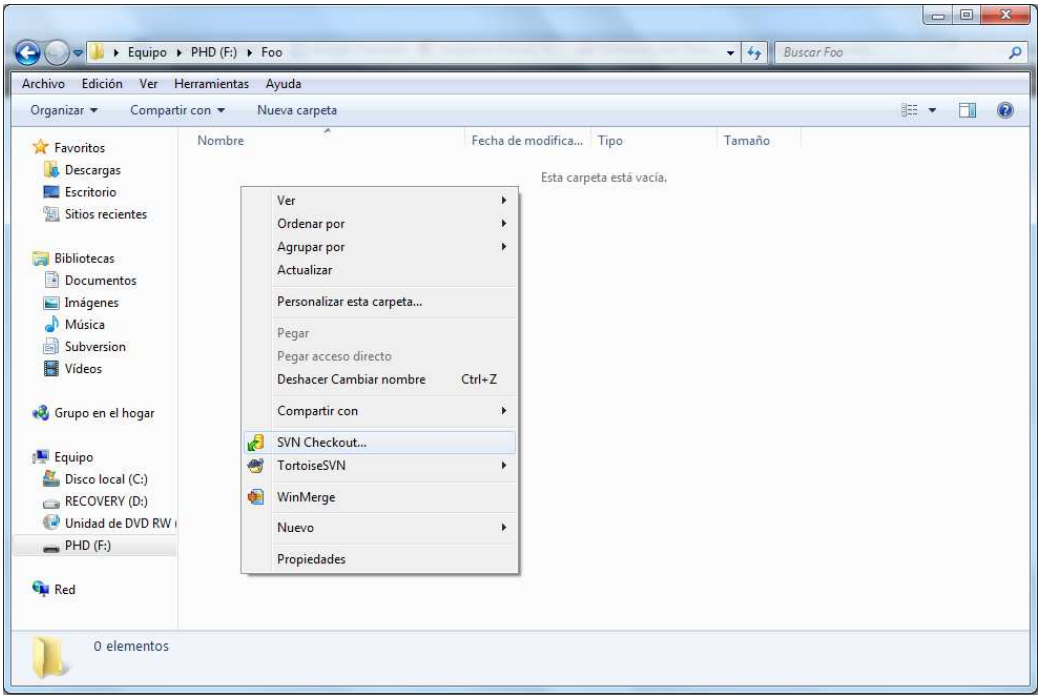


Figura 5. Los comandos de TortoiseSVN están disponibles a través de un menú contextual.

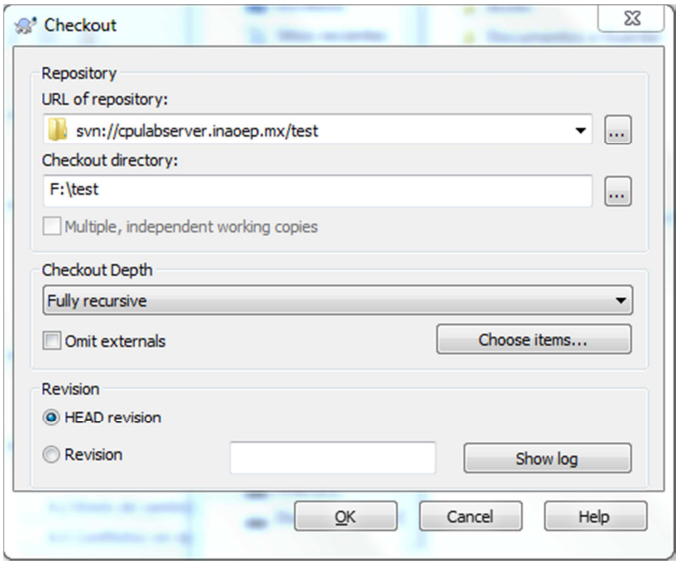


Figura 6. Cuadro de diálogo para la operación checkout.

Después de unos instantes aparece el cuadro de diálogo ilustrado en la Figura 7 que indica el estado de la operación. En este caso se muestra que la operación checkout descargó correctamente los archivos de código correspondientes a la revisión 3 del proyecto. La Figura 8 ilustra el contenido de la copia local generada por TortoiseSVN; notar que están presentes el

archivo de código test.c, el archivo Makefile y el directorio oculto .svn, tal como ocurre con la copia local generada en un sistema UNIX.

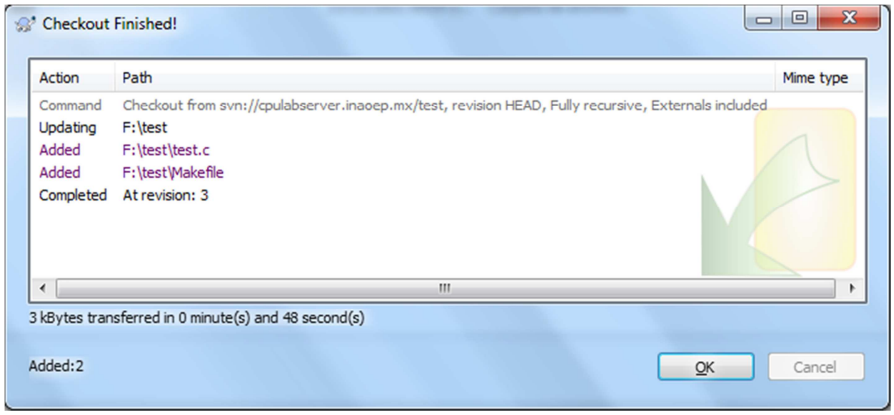


Figura 7. Cuadro de diálogo que muestra el estado de la operación checkout.

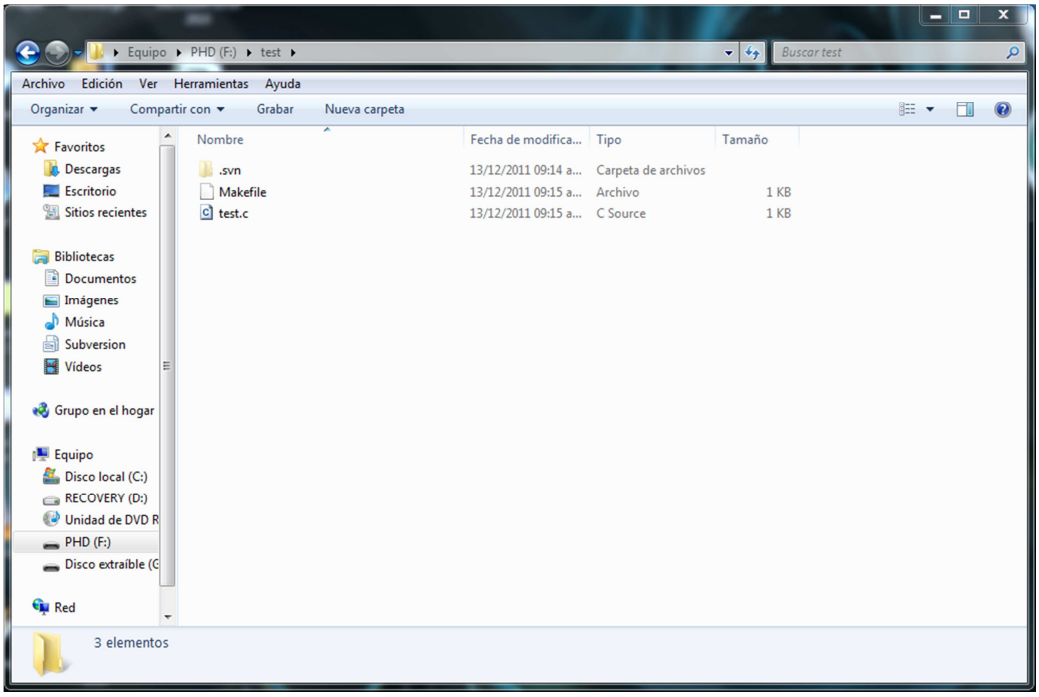


Figura 8. Los archivos de código fuente del proyecto test.

4.6 Modificación a la copia local y envío de cambios al repositorio

Ahora se ilustra cómo someter cambios hechos en una copia local en Windows al repositorio mediante TortoiseSVN. Primero, supongamos que el usuario sally realiza la siguiente modificación al archivo test.c en su copia local:

```
#include <stdio.h>

main () {
  printf("Hello World!\n");
```

```
  printf("Hello World! ... again\n");
  printf("Good bye World!");
  printf("I'm back!");

  return 0;
}
```

Para publicar la modificación anterior, se deben llevar a cabo los siguientes pasos utilizando la infraestructura TortoiseSVN:

1. Seleccionar el archivo test.c en la carpeta que contiene la copia local, hacer click con el botón derecho del ratón y seleccionar el comando **SVN Commit…** del menú contextual que aparece. Ver Figura 9.
2. En el cuadro de diálogo que aparece a continuación se puede ingresar el mensaje que describe la modificación y que se incorporará al registro de modificaciones generados por Subversion. Ver Figura 10.

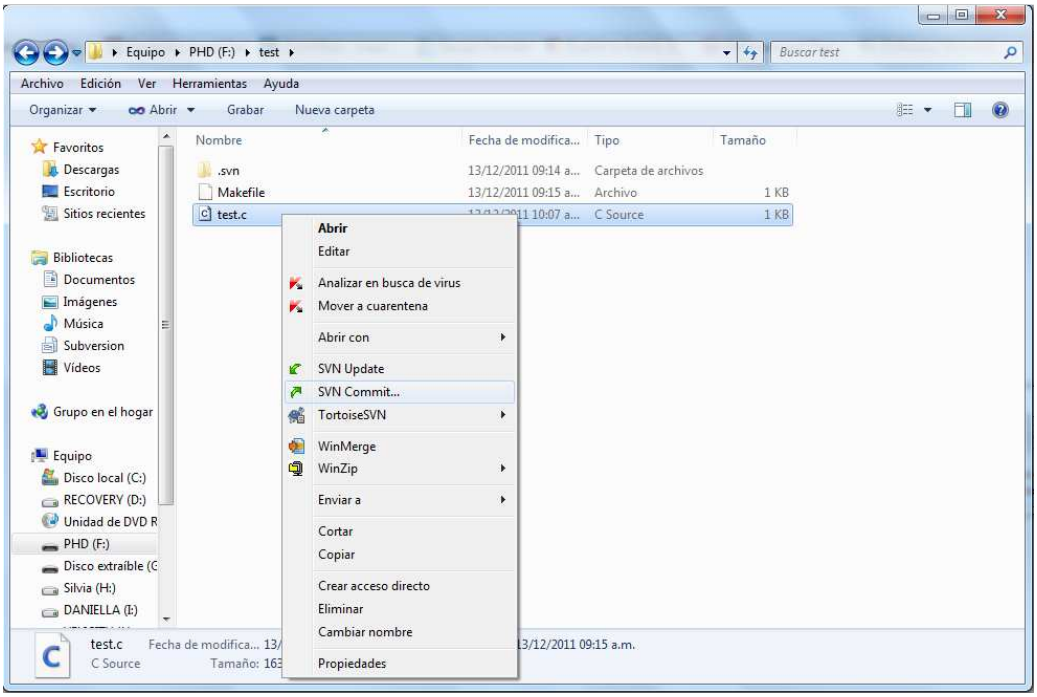


Figura 9. El comando SVN Commit en el menú contextual.

Después de proporcionar el mensaje solicitado, TortoiseSVN se comunica con el servidor cpulabserver.inaoep.mx para actualizar el archivo. El estado de la operación de envío de cambios al repositorio se muestra en el cuadro de diálogo en la Figura 11. Notar que la modificación realizada genera la cuarta revisión del proyecto en el repositorio.

La operación de actualización de las copias locales de otros usuarios se realiza en forma similar a la operación de modificación del repositorio. En este caso, en vez de invocar al comando **SVN**

Commit… en el menú contextual se debe invocar al comando **SVN Update**, que se puede observar en el menú contextual en la Figura 9.

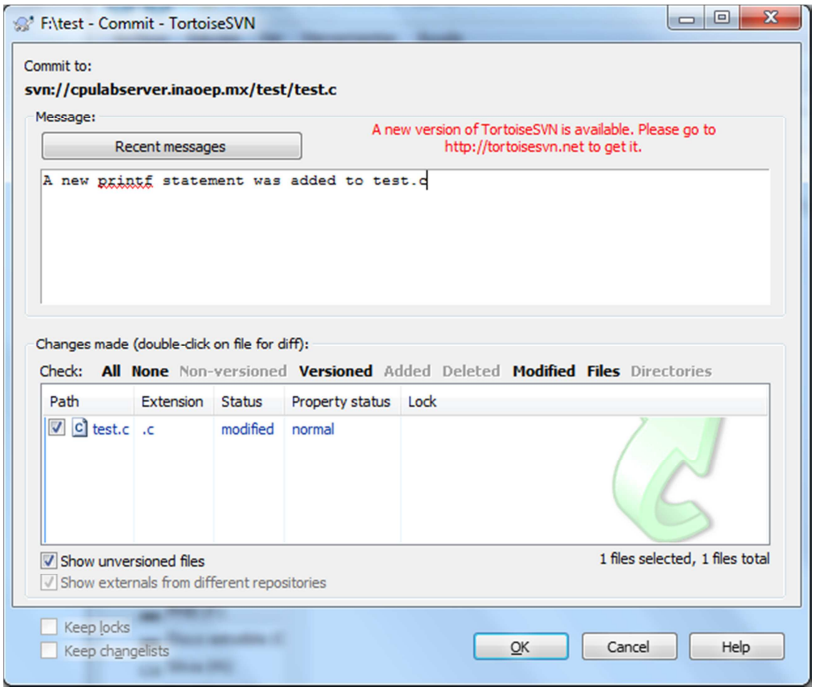


Figura 10. Cuadro de diálogo para la operación commit.

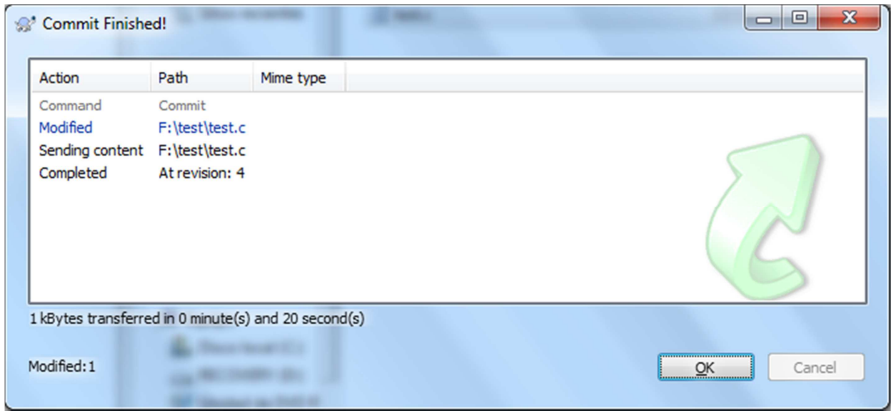


Figura 11. Cuadro de diálogo que muestra el estado de la operación commit.

5 Referencias

[1] Collins-Sussman, B., Fitzpatrick, B. W. y Pilato C. M.; Version Control With Subversion For Subversion 1.6; O'Reilly & Associates, Inc.; CA, USA; 2011; Disponible en http://svnbook.red-bean.com/

[2] Küng, S., Onken, L. y Large, S.; TortoiseSVN A Subversion client for Windows Version 1.7; 2011; Disponible en http://tortoisesvn.net/support.html