Unidad didáctica 4. Herramientas para el control y documentación de software

IES CONSUELO ARANDA

- En esta unidad vamos a manejar herramientas de control de versiones, herramientas para documentar los programas y herramientas de refactorización.

  → Vital importancia cuando trabajamos en equipos de software.

- Este tipo de herramientas garantiza que los proyectos estén bien documentados, informando de lo que hace cada una de las clases y métodos que lo forman, y bien refactorizados.

  → Se favorece la lectura y el mantenimiento.

IES CONSUELO ARANDA

CONTROL DE VERSIONES

CONTROL DE VERSIONES

- Se puede definir como la capacidad de recordar todos los cambios que se realizan, tanto en la estructura de directorios como en el contenido de los archivos.

  → Útil cuando de desea recuperar un documento, archivo, proyecto.

  → Útil cuando se comparten archivos o proyectos.

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Algunas funcionalidades:
  - Comparar cambios en los diferentes archivos a lo largo del tiempo.
  - Reducir problemas de coordinación entre diferentes programadores.
  - Posibilidad de acceder a versiones anteriores de un documento o código fuente.
  - Ver qué programador ha realizado un determinado cambio de código.
  - Acceso al historial de cambios de todos los archivos, durante la realización del proyecto.
  - Volver un archivo a un estado anterior.

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Funcionalidades específicas para proyecto informático:
  - Control histórico detallado de cada archivo.
  - Control de usuarios con permisos para acceder a los archivos.
  - Creación de ramas de un mismo proyecto.
  - Fusionar dos versiones de un mismo archivo.

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Componentes y terminología de un sistema de control de versiones:
  - Repositorio
  - Revisión o versión
  - Etiquetar o rotular (tag)
  - Tronco (trunk)
  - Rama o ramificar (branch)
  - Desplegar (checkout)
  - Confirmar (commit o check-in)
  - Exportación (export)
  - Importación (import)
  - Actualizar (update)
  - Fusion (merge)
  - Conflicto
  - Resolver conflicto

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Componentes y terminología de un sistema de control de versiones:
  - Repositorio: lugar donde se almacenan todos los datos y los cambios. Puede ser un sistema de archivos en un disco duro, un banco de datos, un servidor, etc.
  - Revisión o versión: versión concreta de los datos almacenados. Algunos sistemas identifican las revisiones con un número contador. Otros utilizan código de detección de modificaciones.
  - Etiquetar o rotular (tag): cuando se crea una versión concreta en un momento determinado del desarrollo, se le pone una etiqueta, de forma que se pueda localizar y recuperar en cualquier momento. Las etiquetas permiten identificar de forma fácil revisiones importantes en el proyecto (por ejemplo, versión publicada).

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Componentes y terminología de un sistema de control de versiones:
  - Tronco (trunk): tronco o línea principal del desarrollo de un proyecto.
  - Rama o ramificar (branch): son copias de archivos, carpetas o proyectos. Cuando se crea una rama se crea una bifurcación del proyectos y se crean dos lineas de desarrollo. Motivos de ramas son nuevas funcionalidades o la corrección de errores.
  - Desplegar (checkout): crear una copia del trabajo del proyecto, o de archivos y carpetas del repositorio en el equipo local. Por defecto se obtiene la última versión pero también se puede indicar una versión completa. (se vincula la carpeta de trabajo local con un repositorio).

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Componentes y terminología de un sistema de control de versiones:
  - Confirmar (commit o check-in): se realiza cuando se confirman datos en local para integrarlos en el repositorio.
  - Exportación (export): similar a checkout pero sin vincular la copia en el repositorio. Es una copia limpia sin metadatos de control de versiones.
  - Importación (import): subida de archivos y carpetas del equipo local al repositorio.

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Componentes y terminología de un sistema de control de versiones:
  - Actualizar (update): cuando se desea integrar los cambios realizados en el repositorio en la copia local. Los cambios pueden ser realizados por cualquier persona del equipo de trabajo.
  - Fusion (merge): unir los diferentes cambios en una única revisión. (se usa cuando hay varias lineas separadas en ramas).
  - Conflicto: cuando dos usuarios crean una copia local y uno de ellos no actualizan.
  - Resolver conflicto: actualización del usuario para atender un conflicto entre diferentes cambios de un mismo documento.

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Componentes y terminología de un sistema de control de versiones:
  - Para trabajar en proyectos utilizando un sistemas de control de versiones, lo primero que hay que hacer es crearse una copia en local de la información del repositorio (checkout) → así copia vinculada.
  - Modificaremos y subiremos modificaciones con commit.
  - Si la copia ya está vinculada al repositorio, antes de modificar y realizar cambios → update (para asegurarnos que los cambios se realizan sobre la última versión).

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- **Subversion**.

  → Herramienta multiplataforma de código abierto para el control de versiones.

  → Usa una BBDD central, el repositorio, que contiene los archivos cuyas versiones y respectivas historias se controlan.

  → El repositorio actúa como un servidor de ficheros, con la capacidad de recordar todos los cambios que se hacen, tanto en sus directorios como en sus ficheros.

CONTROL DE VERSIONES

- **Git**.

  → Herramienta de gestión de versiones desarrollada para programadores del núcleo Linux.

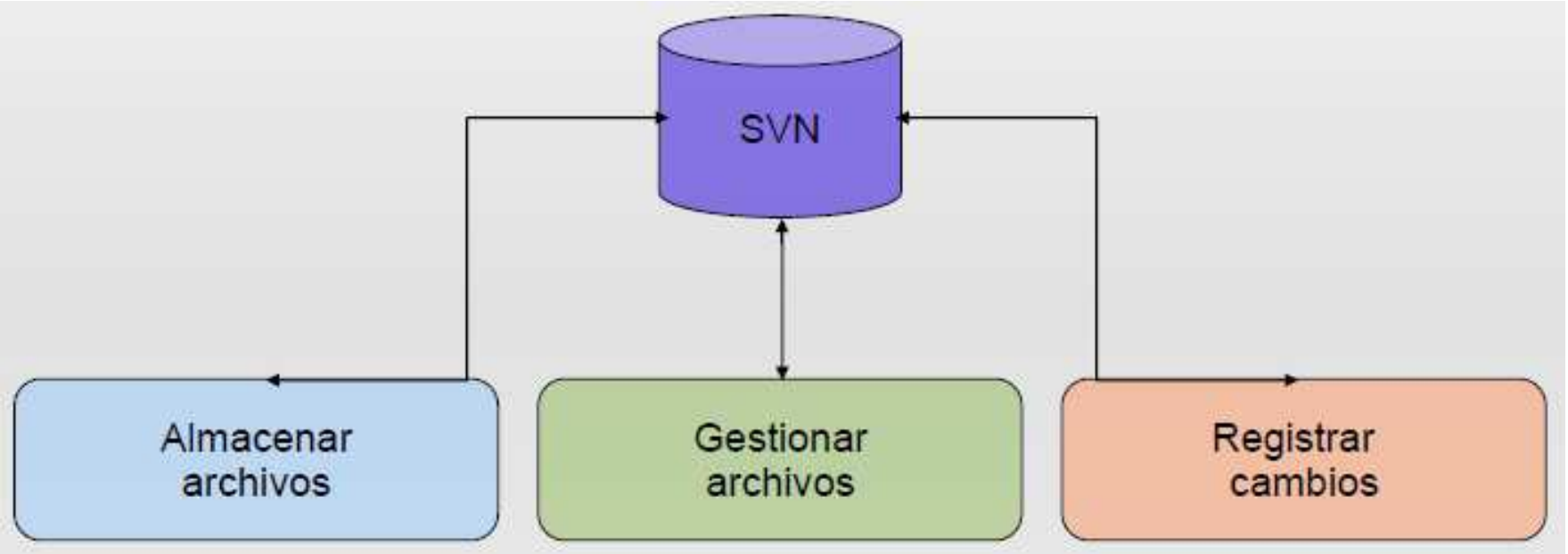
  → Subversion no cubría las necesidades de los desarrolladores de proyectos Linux.

CONTROL DE VERSIONES

- Otras herramientas para el control de versiones
  - Team Fundation Server → propietaria (Microsoft).
  - Clear Case
  - Darcs
  - CVS – Concurrent Versions System
  - MyLyn
  - GNU Arch
  - RedMine
  - Mercurial (ALSA, MoinMoin, Mutt, Xen)
  - PHP Collab

CONTROL DE VERSIONES

- Esquema conceptual de un repositorio



IES CONSUELO ARANDA

CONTROL DE VERSIONES

- Como funciona *subversion*.
  - El proyecto debe verse como un arbol que tiene su tronco (trunk) → línea principal de su desarrollo.
  - Tiene sus ramas (branches) → se añadiran nuevas funciones o se corregirán errores.
  - Además tiene sus etiquetas (tags) → para marcar situaciones importantes o versiones finalizadas.

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- **Client TortoiseSVN** (*Windows*).

  - Es un cliente de código abierto para el sistema de control de versiones subversion.

    → Habrá que buscar el equivalente en ubuntu o utilizar la máquina virtual...

IES CONSUELO ARANDA

CONTROL DE VERSIONES

- ¿Qué necesitamos para poder utilizar Subversion?
  - Un cliente, como puede ser TortoiseSVN o el propio que incorpora subversion.
  - Un repositorio que podemos crear en nuestro servidor SVN, si disponemos de uno, o bien en local con el propio TortoiseSVN.

CONTROL DE VERSIONES

- ¿Necesito un servidor SVN?
  - Solo en el caso de que me interese mantener una gestión de usuarios con sus controles de accesos, y permitir el acceso mediante HTTP (webdav).
  - Si se trata de un repositorio privado, puedo crearlo en mi propio PC, sin necesidad de un servidor SVN. En este caso, puedo hacerlo directamente con TortoiseSVN, incluso en una unidad de red.

CONTROL DE VERSIONES

- # ¿Necesito un servidor SVN?

  - Si finalmente opto por la opción del servidor SVN, en la máquina destinada a tal fin, tendré que instalar un servidor web Apache (en el caso de que quiera habilitar el acceso mediante HTTP).

  - ¿Puedo montar un servidor SVN sin Apache? Sí, con el servidor nativo “svnserve”, sin acceso HTTP.

IES CONSUELO ARANDA

DOCUMENTACIÓN

IES CONSUELO ARANDA

DOCUMENTACIÓN

- Cualquier texto escrito que acompaña a nuestros proyectos, en nuestro caso, de software.

- Tipos de documentación:
  - Documentación de las especificaciones.
  - Documentación del diseño.
  - Documentación del código fuente.
  - Documentación de usuario final.

IES CONSUELO ARANDA

DOCUMENTACIÓN

- **Documentación de las especificaciones.**

  → Tiene como objeto asegurar que tanto el desarrollador como el cliente tienen la misma idea sobre las funcionalidades del sistema.

  → La norma IEEE 830 contiene las especificaciones de requisitos de Software.

IES CONSUELO ARANDA

DOCUMENTACIÓN

- **Documentación de las especificaciones.**
  - → La norma IEEE 830 recoge:
    - ✓ Introducción: se define fines y objetivos del software.
    - ✓ Descripción de la información: se realiza una descripción detallada del problema, incluyendo hardware y software.
    - ✓ Descripción funcional: descripción de cada función requerida en el sistema, incluyendo diagramas.
    - ✓ Descripción del comportamiento: comportamiento del software ante sucesos, eventos y controles internos.
    - ✓ Criterios de validación: documentación sobre límites de rendimiento, clases de pruebas, respuesta esperada del software y consideraciones especiales.

DOCUMENTACIÓN

- ## Documentación del diseño

  → En la fase de diseño se decide la estructura de datos a utilizar, la forma en que se van a implementar las distintas estructuras, contenido de las clases, sus métodos y sus atributos, los objetos a utilizar.

- ## Documentación del código fuente

  → Cuando se está programando, es necesario comentar convenientemente cada una de las partes que tiene el programa. Estos comentarios se incluyen en el código fuente.

- ## Documentación de usuario final

  → Documentación que se entrega al usuario. Se describe cómo utilizar las aplicaciones del proyecto. La puede redactar cualquier persona.

DOCUMENTACIÓN

- **Documentación del código fuente.**

  Es necesario para explicar claramente lo que hace el programa, de esta manera todo el equipo de desarrollo sabrá lo que se está haciendo y por qué.

  Un programa bien documentado es mucho más fácil reparar errores y añadirle nuevas funcionalidades para adaptarlo a nuevos escenarios que si carece de documentación.

IES CONSUELO ARANDA

DOCUMENTACIÓN

- Documentación del código fuente.

  Hay dos reglas que no se deben olvidar:

  → Todos los programas tienen errores y descubrilos solo es cuestión de tiempo y de que el programa tenga éxito y se utilice frecuentemente.

  → Todos los programas sufren modificaciones a lo largo de su vida, al menos todos aquellos que tienen éxito.

  - Si el programa tiene éxito, normalmente, será modificado en el futuro.

DOCUMENTACIÓN

- **Documentación del código fuente.**

  A la hora de documentar interesa que se explique lo que hace una clase o un método y por qué y para qué se hace.

  Para documentar proyectos existen muchas herramientas. Normalmente cada lenguaje dispone de su propia herramienta (PHPDoc, phpDocumentor, Javadoc, etc.).

IES CONSUELO ARANDA

REFACTORIZACIÓN

IES CONSUELO ARANDA

REFACTORIZACIÓN

- La refactorización es una técnica de la ingeniería del software que permite la optimización de un código previamente escrito.
- ¿Cuándo refactorizar?
  - Código duplicado
  - Métodos muy largos
  - Clases muy grandes
  - Lista de parámetros extensa
  - Cambio divergente
  - Cirugía a tiro pistola
  - Envidia de funcionalidad
  - Clase de solo datos
  - Legado rechazado

IES CONSUELO ARANDA

REFACTORIZACIÓN

- ¿Cuándo refactorizar?
  - Código duplicado: mismo código en más de un lugar → extraer y unificar
  - Métodos muy largos: cuanto más largo, más difícil de entender.
  - Clases muy grandes: clase que resuelve muchos problemas → clase con muchos métodos, atributos o incluso instancias.
  - Lista de parámetros extensa: en POO no se suelen pasar muchos parámetros a los métodos.
  - Cambio divergente: clase modificada frecuentemente por diversos motivos → posiblemente conviene eliminar la clase.
  - Cirugía a tiro pistola: se modifica una clase → implica hacer modificaciones en diversos lugares.

REFACTORIZACIÓN

- # ¿Cuándo refactorizar? II
  - ✓ Envidia de funcionalidad: se observa este síntoma cuando tenemos un método que utiliza más cantidad de elementos de otra clase que de la propia suya → se soluciona pasando el método a la clase cuyos elementos más se utilizan.
  - ✓ Clase de solo datos: clases que solo tienen atributos y métodos de acceso a ellos → deberán cuestionarse, dado que no suelen tener comportamiento alguno.
  - ✓ Legado rechazado: clases que utilizan unas pocas características de superclases → si las subclases no utilizan lo que sus superclases les proveen por herencia.