Universidad de las Ciencias Informáticas

Facultad X

Gestión de la Configuración del Software
de Intranet II de la UCI.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor
Leynier Díaz Navarro.
Tutor
Kelvys Gálvez Cabrera.

Ciudad de la Habana
Julio 2007.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Leynier Díaz Navarro	Kelvys Gálvez Cabrera
_____	_____
Firma del Autor	Firma del Tutor

Datos de contacto.

Ing. Kelvys Gálvez Cabrera

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Desde su graduación se vinculó a trabajar en la universidad como profesor, donde ha impartido clases de Programación II y Programación IV. Se desempeña como líder de proyecto y ha participado en varios proyectos de desarrollo de software, fundamentalmente en el desarrollo de aplicaciones Web. Actualmente se encuentra trabajando como profesor de la especialidad.

Considero más valiente al que conquista sus deseos que al que conquista a sus enemigos, ya que la victoria más dura es la victoria sobre uno mismo.

Aristóteles

Agradecimientos.

A mi familia, mi novia, mis amistades y aquellas personas que de una forma u otra han colaborado con la realización de este trabajo, a la Universidad de las Ciencias Informáticas, a la Revolución cubana y a Fidel.

Dedicatoria.

A mi gran abuela Eugenia, a mi padrastro, a mi tía Mayda, a mi novia, y a una persona muy especial para mí, por lo mucho que se lo merece, más que nadie, mi MAMÁ.

Resumen.

Gestión de la Configuración del Software de Intranet II de la UCI es un trabajo que se realiza con el objetivo de proponerle al proyecto un plan para su gestión de configuraciones. Para ello primeramente se exponen y analizan varios conceptos brindados por diferentes autores, luego se describen elementos básicos de importancia para la realización de las tareas de la Gestión de la Configuración del Software (GCS) las cuales son detalladas de forma conceptual también, conformando así el estado del arte de la GCS, el cual facilita al lector de una panorámica sobre el tema. Además en aras de fomentar su desempeño en la Intranet II como rol, se refieren características del sistema sobre el cual se va a trabajar, con el propósito de ubicar al usuario en el ambiente real de trabajo y presentar de forma concreta algunos de los elementos de configuración del software construidos hasta el momento. Luego un estudio de las herramientas propuestas, brindando un análisis detallado de las más usadas en la actualidad. Se incluye además la justificación del estándar utilizado para finalmente presentar el Plan de Gestión de la Configuración, el cual resume los objetivos del gestor en el proyecto.

Palabras clave.

Gestión de la Configuración del Software, Rol, Intranet II, UCI, Elementos de Configuración del Software, Línea base, Control de Versiones, Gestión de Cambios, Subversion, Repositorio, Trac, Estándar, Plan, Política, Estrategia, Calidad.

Tabla de Contenidos.

Introducción.

La información, es la base del conocimiento en el mundo. Cuando se tiene que resolver un determinado problema o tomar una decisión, se emplean diversas fuentes de información. El manejo de esta ha ido ganando en tamaño y dificultad, por ejemplo: hasta hace algún tiempo, controlar la información que fluía en los centros de trabajos para mantener su funcionamiento de una forma coherente, ya fueran compañías, escuelas, hospitales, pequeñas empresas, requería de compromiso, dedicación, consagración, trabajo y de esfuerzo a realizar. Grandes cantidades de papeles archivados, que almacenaban la información necesaria, era la forma de mantener el conocimiento vital del lugar.

Con el desarrollo de las Tecnologías de la Información y la Comunicación (TIC), surgieron las Intranet: red de ordenadores de una red de área local (LAN) privada, empresarial o educativa [1].

Técnicamente las intranets son una solución basada en tecnología Web, accesible desde cualquier ordenador conectado, que permiten publicar contenido sin necesidad de conocimientos informáticos, archivar e indexar cualquier tipo de ficheros, crear perfiles de acceso para los usuarios y para los gestores de esta [2].

Las redes internas corporativas son unas potentes herramientas de comunicación, colaboración e intercambio que permiten la divulgación con efectividad de información en la compañía a los empleados, consiguiendo que estos estén permanentemente informados con las últimas novedades y datos de la organización, en otras palabras, incrementar el conocimiento compartido.

Permite minimizar el tiempo de búsqueda y consulta de la información, el costo de imprimir, almacenar, distribuir y archivar documentos estándares, reciclar documentos como calendarios y listas de precios actualizados centralmente, introducir sencillas y útiles herramientas metodológicas en su gestión empresarial, con un sentido más global en cuanto a su uso: los procesos informatizados se interrelacionan entre sí y los recursos se comparten, perfeccionar la comunicación empresa-clientes y empresa-proveedores, abaratar considerablemente los costos de las comunicaciones [3].

Tienen gran valor como repositorio documental, convirtiéndose en un factor determinante para conseguir el objetivo de oficina sin papeles.

En Cuba son empleadas fundamentalmente en la diversificación de la cultura y el conocimiento. Por tanto las universidades, son unas de las mayores portadoras de la tecnología de redes y de la implementación de Intranets.

La Universidad de las Ciencias Informáticas (UCI), cuenta con su propia Intranet, conformada desde sus inicios, año 2002-2003. En la actualidad, el crecimiento en gran medida de la población universitaria, trae consigo el aumento de usuarios que acceden a esta intranet, que a su vez, hacen que generalmente el servidor se encuentre congestionado, la falta del personal disponible con conocimientos del código de la Intranet, y no existe documentación sobre esta información, y el hecho que este software esté en decadencia natural, debido a la falta de mantenimiento, han creado una ***situación problémica*** en el centro.

En aras de solucionar este problema, la UCI ha creado un grupo de trabajo (Proyecto Intranet) en el que cada integrante tiene un rol bien definido. Háblese de:

- Analista de Sistema.
- Arquitecto.
- Diseñador Principal de Base de Datos.
- Gestor de Configuración.
- Diseñador de Sistema.
- Ingeniero de prueba.
- Revisor Técnico.
- Planificador.
- Líder de Proyecto.
- Programador.
- Factoría.
- Web en general.

El valor de la teoría de los roles de equipo radica en permitir a una persona en particular, o a un equipo, beneficiarse del conocimiento personal y de esta manera adaptarse a las exigencias del entorno.

El Proyecto Intranet tiene la misión fundamental de desarrollar una nueva Intranet que satisfaga las características de la UCI, que posea un nuevo diseño, tanto visual como arquitectónico y que además responda a las normas de una intranet corporativa, orientada a servicios.

Se entiende por *intranet corporativa* a aquella intranet o red privada perteneciente a una empresa o corporación. Además permiten gestionar el Conocimiento de la Empresa y establecer herramientas de Comunicación interna con Empleados referentes a Calidad, Producción, Recursos Humanos, Indicadores de Gestión [4].
Dentro del Proyecto Intranet la labor a desarrollar por el Gestor de Configuración será el tema que guíe este trabajo.

Por lo que se pudiera definir como ***problema científico***: la necesidad de desarrollar una estrategia de Gestión de la Configuración del Software que satisfaga las características de la nueva Intranet de la UCI. Donde nuestro ***objeto de estudio*** sería**:** La Gestión de la Configuración del Software. ***Campo de acción:*** Gestión de Configuración de la Intranet II de la UCI.

Para darle solución a este problema se traza como ***Objetivo de la Investigación:***

- ***Objetivos Generales:***

Definir una estrategia de Gestión de la Configuración para el Proyecto Intranet II.

- ***Objetivos específicos:***

1. Definición de la estrategia para el control de cambios.
2. Definición de la estrategia para el control de las versiones.
3. Definición de los elementos de configuración del software.
4. Selección de las herramientas a utilizar.

Tareas:

- Investigar acerca de las Intranets y la Gestión de la Configuración del Software.
- Analizar y estudiar las características del sistema a implementar, Intranet II.
- Investigar sobre las herramientas posibles a utilizar y valorar sus tendencias.
- Elaborar Plan de Gestión de la Configuración de Intranet.

Luego se puede plantear como **hipótesis** de este trabajo: el desarrollo de una estrategia general para la gestión de configuraciones del proyecto productivo Intranet II, posibilitará la administración de cambios de forma organizada, permitirá mejorar el control sobre las actividades en desarrollo y a su vez una mayor calidad en el producto final.

Estructura del contenido.

Capítulo 1

Este capítulo trata de una forma bien detallada qué es la gestión de configuración, elementos fundamentales para esta, tareas a realizar y algunas herramientas que pueden utilizarse.

Capítulo 2

Este capítulo describe las características del proyecto al cual se le esta gestionando la configuración, Intranet II, con dos objetivos fundamentales: uno, dar idea del software sobre el cual se esta trabajando y dos, mostrar de forma explicita los primeros elementos de configuración del software del proyecto (unidad básica para la gestión de la configuración del software).

Capítulo 3

El capítulo muestra el análisis desarrollado para seleccionar las herramientas a utilizar para el apoyo de la Gestión de Configuración del proyecto Intranet, entre un diverso grupo de sistemas para el control de versiones, con una breve descripción final de la selección.

Capítulo 4

Este capítulo presenta la propuesta del Plan de Gestión de la Configuración del Software de la Intranet II, el cual muestra los resultados del trabajo del gestor de la configuración del proyecto. Además de una fundamentación del estándar utilizado.

Capítulo 1

Fundamentación Teórica de la Gestión de la Configuración del Software.

La Gestión de la Configuración del Software (GCS) es uno de los procesos clave para toda organización dedicada a la Ingeniería del Software, ya que posibilita una mejor organización del desarrollo y mantenimiento del producto, facilitando el resto de los procesos de producción [5].

Durante el proceso de construcción de un software, los cambios son inevitables. Estos provocan confusión e incertidumbre, sobre todo cuando no se han analizado o pronosticado correctamente. Es importante considerar ciertas modificaciones que pueden ocurrirle al software dentro de todo el proceso de ingeniería. Este capítulo incluye el estado del arte de la Gestión de Configuración del Software, temáticas como: el cambio, línea base, elementos de configuración del software y tareas a realizar por la GCS.

El arte de coordinar el desarrollo del software para minimizar la confusión, se denomina ***gestión de la configuración.*** La gestión es el arte de identificar, organizar y controlar las modificaciones que sufre el software, la meta de la misma es maximizar la productividad minimizando errores [6].

Según Rational, “la GCS describe la estructura del producto e identifica los elementos que lo constituyen y que son tratado como entidades que pueden ser puestas bajo control de versiones en el proceso de GCS. La GC tiene que ver con la definición de la configuración, así como la construcción, el etiquetado y recolección de versiones de los artefactos” [7].

Una de las definiciones más utilizadas es la que brinda la IEEE, donde plantea:

“Gestión de Configuración es la disciplina que abarca todo el ciclo de vida de la producción de software y productos asociados. Específicamente, requiere de la identificación de los componentes a controlar y la estructura del producto, controla todos los cambios sobre los elementos y garantiza mecanismos para auditar todas las acciones” [8].

Ivar Jacobson en una definición muy completa establece:

"Proceso de soporte cuyo propósito es identificar, definir y almacenar en una línea base los elementos de software; controla los cambios, reporta y registra el estado de los elementos y de las solicitudes de cambio; asegura la completitud, consistencia y corrección de los elementos; controla, almacena, maneja y libera los elementos asociados al producto de software" [9].

Se puede decir entonces que la GCS es la coordinación del proceso de desarrollo de un software. Conjunto de actividades que permite identificar, organizar y definir los elementos de configuración en una línea base, controlar los cambios a los que se somete el proceso, en aras de alcanzar una mayor calidad del producto final.

La Gestión de Configuración del Software comienza con el inicio del proyecto y perdura hasta que el software queda fuera de circulación [10].

1.1 "Bases de la Gestión de Configuración del Software"

1.1.1 El "Cambio".

Una de las variables de la Ingeniería del Software durante el desarrollo de un software es "*el cambio*", la cual tiene gran importancia para el desarrollo de un proyecto de software.

La primera Ley de la ingeniería de sistemas establece: "Sin importar en que momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida" [11].

Las causas fundamentales que provocan estos cambios pueden centrarse en:

1. Nuevos negocios o condiciones comerciales que dictan los cambios en los requisitos del producto o en las normas comerciales.
2. Nuevas necesidades del cliente que demandan la modificación de los datos producidos por sistemas de información, funcionalidades entregadas por productos o servicios entregados por un sistema basado en computadora.
3. Reorganización y/o reducción del volumen comercial que provoca cambios en las prioridades del proyecto o en la estructura del equipo de ingeniería del software.
4. Restricciones presupuestarias o de planificaciones que provocan una redefinición del sistema o del producto.

Como el cambio se puede producir en cualquier momento, las actividades de GCS sirven para:

1. Identificar el cambio.
2. Controlar el cambio.
3. Garantizar que el cambio se implementa adecuadamente.

Informar del cambio a todos aquellos que puedan estar interesados.

1.1.2 *Líneas Base.*

Una línea base es un concepto de GCS que ayuda a controlar los cambios sin perjuicio de aquellos que sean necesarios [10]. La IEEE define una línea base como:

Una especificación o producto que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios [12].

Los cambios a los elementos de configuración del software pueden ser realizados de forma rápida e informal. Una vez que este se convierte en línea base, los cambios deberán ser efectuados a través de un proceso formal.

Por ejemplo: los elementos de una especificación de diseño se documentan y se revisan. Se encuentran errores y se corrigen cuando todas las partes de las especificaciones se han revisado corregido y aprobado, la especificación de diseño se convierte en línea base. Solo se pueden realizar cambios futuros en la arquitectura del software (contenidos en la especificación del diseño) tras haber sido evaluados y aprobados. Aunque se puedan definir las líneas base con cualquier nivel de detalle las líneas base más comunes son las que se muestran en la figura 1.0.

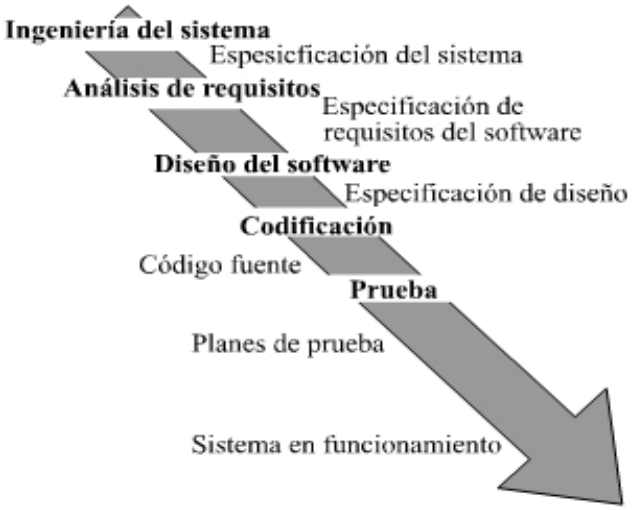

Fig.1.1 Líneas base

Se puede decir que ***línea base*** es la especificación o producto básico, el cual es el resultado de un consenso y que debe ser modificado solo en casos realmente necesarios y a partir de un proceso formal de control de cambio. Un proyecto tiene varias líneas bases.

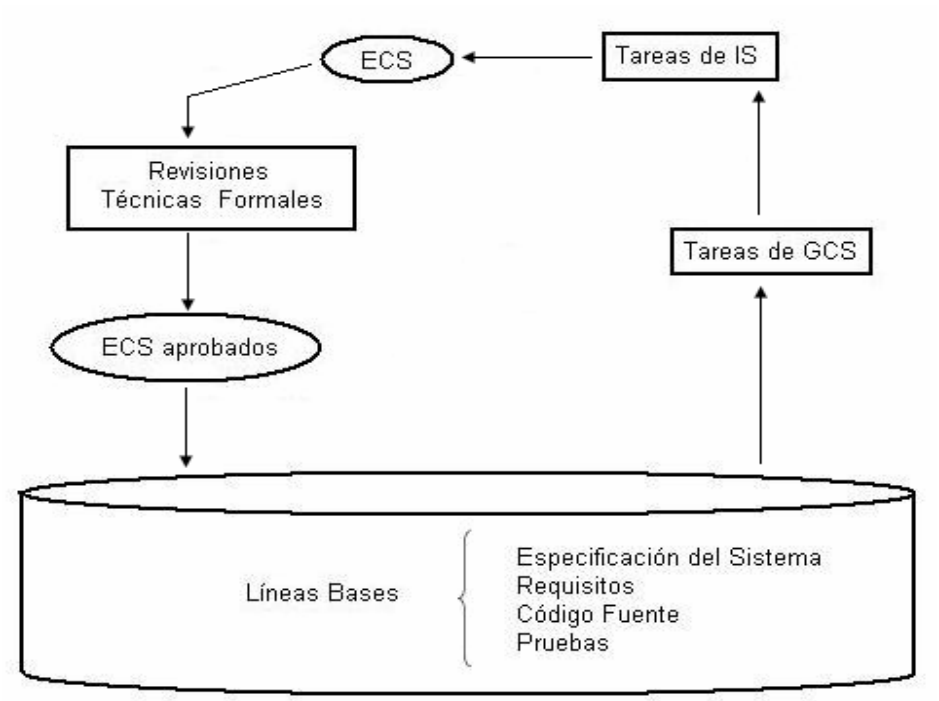

Fig. 1.2 Idea lógica de línea base.

1.1.3 *Elementos de Configuración del Software.*

Como resultado de la aplicación del proceso de desarrollo, se obtienen y emplean a lo largo del ciclo de vida, un conjunto de elementos que pueden ser clasificados en ejecutables, fuentes, ficheros de datos, documentación, ayudas, entre otros, los cuales visto como conjunto son denominados por [13] como configuración del Software. A cada uno de los componentes de la configuración del software se le va a llamar Elemento de Configuración del Software (ECS). Es decir, un ECS es la información creada como parte del proceso de ingeniería. Además es la unidad de trabajo para la GCS [14]. Durante el proceso de desarrollo, el número de los ECS al igual que las relaciones que se establecen entre ellos crece a medida que va transitando el proyecto por el ciclo de vida, provocando confusión entre los involucrados en la realización del proyecto [15].

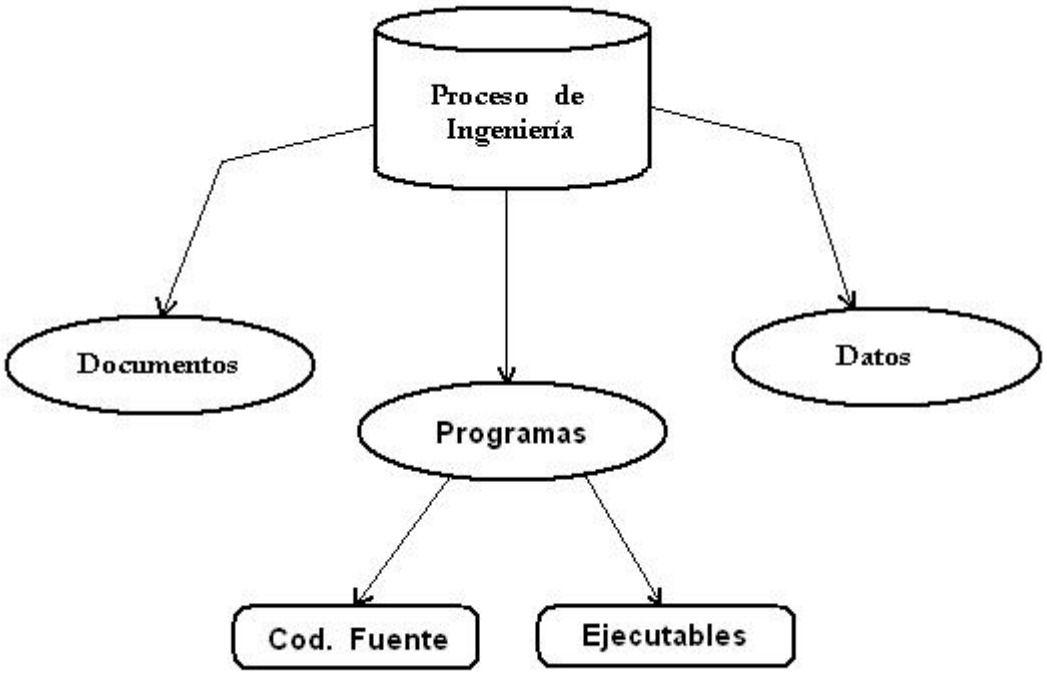

Fig. 1.3 Elementos resultantes del proceso de Ingeniería del Software

Elementos de Configuración del Software:

1. Especificación del sistema.

2. Plan de proyecto del software.
3. Especificación de requisitos del software.
4. Prototipo ejecutable o “en papel”.
5. Manual de usuario preliminar.
6. Especificación de diseños:

    a. Descripción del diseño de datos.

    b. Descripción del diseño arquitectónico.

    c. Descripciones del diseño de los módulos.

    d. Descripciones del diseño de interfaces. Descripciones de los objetos (si se utilizan técnicas de P.O.O).

7. Listados del código fuente.
8. a. Plan y procedimiento de pruebas.

    b. Casos de prueba y resultados registrados.

9. Manuales de operación de y de instalación.
10. Programas ejecutables:

    a. Módulos, código ejecutable.

    b. Módulos enlazados.

11. Descripción de la base de datos:

    a. Esquema y estructura de archivos.

    b. contenido inicial.

12. Manual del usuario final.
13. Documentos de mantenimiento:

a. Informes de problemas del software.

b. Peticiones de mantenimiento.

c. Ordenes de cambios e ingeniería.

14. Estándares y procedimientos de ingeniería del software.

Los ECS se organizan como objetos de configuración que deben ser catalogados por la base de datos del proyecto con un nombre único. Un ECS tiene un nombre y atributos, y está conectado a otros objetos mediante relaciones.

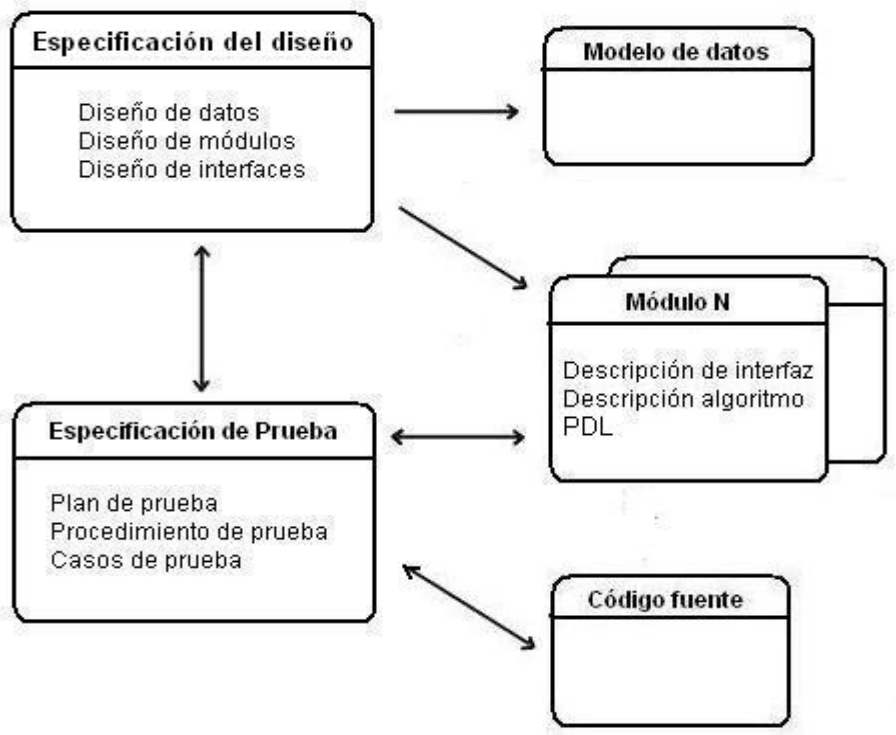

Fig.1.4 Objetos configuración.

La figura 1.4 muestra el modelo de datos de los elementos de la configuración, cada objeto esta relacionado con otro, si se lleva a cabo un cambio sobre un objeto la interrelaciones señalan a que otros objetos afectará.

Los objetos en la GCS se pueden clasificar en:

1. Objetos Básicos.
2. Objetos Compuestos.

Un *objeto básico* es una unidad de información de texto creada durante alguna fase de IS (análisis, diseño, codificación, prueba) [16].

Ejemplo: Parte de un documento (“Unidad de Texto”)

Un *objeto compuesto* es una colección de objetos básicos u objetos compuestos.

Cada objeto tiene un conjunto de características que los identifican como únicos. El nombre del objeto es una cadena de caracteres que identifica al objeto sin ambigüedad. La descripción del objeto es una lista de elementos de datos que identifican:

- El tipo de ECS (documento, programa, datos) que está representado por el objeto.
- Un identificador del proyecto.
- La información de la versión y/o el cambio.

El esquema de identificación de los objetos de software debe tener en cuenta que los objetos evolucionan a lo largo del proceso de ingeniería, por lo que se puede crear un grafo de evolución (figura 1.5)

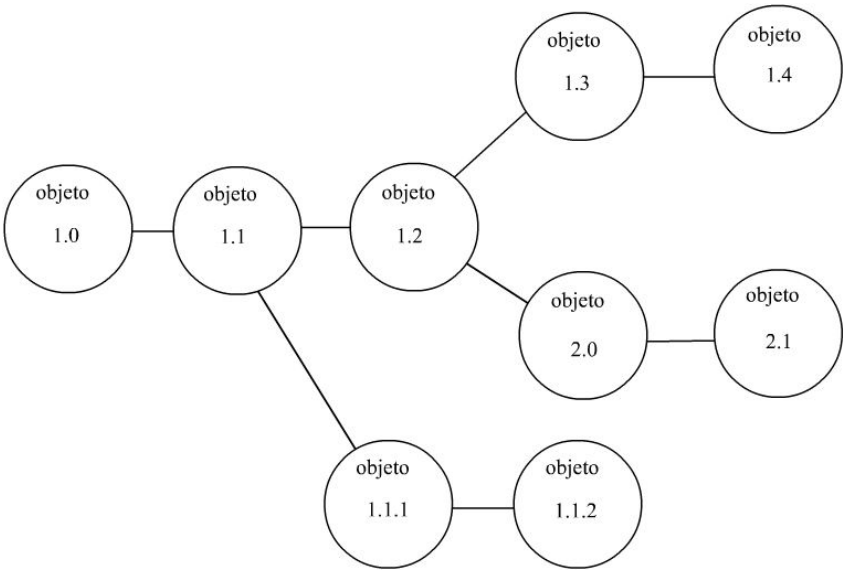

Fig.1.5 Grafo de evolución

En el grafo de evolución se describe la historia del objeto y sus cambios, las grandes modificaciones hacen que un objeto cambie, por lo que cambia el número de versión principal.

Se puede resumir diciendo que un ***elemento de configuración del software*** es la información creada como resultado del proceso de Ingeniería del Software, que pueden ser código fuente, casos de prueba, etc. Que se organizan como objetos de configuración, guardados y reconocidos por la base de datos como un elemento único, poseen nombre e id y están relacionados entre si. Constituye la unidad de trabajo para la GCS.

1.2 "Tareas de la Gestión de Configuración del Software."

1.2.1 El Proceso de Gestión de la Configuración del Software.

Varios autores afirman que los procesos asociados a la GCS que aparecen en el estándar de la IEEE son suficientes para conseguir sus objetivos principales, (Gervás-2002, Antonio-2001, IEEE-1987, IEEE-1990) estas actividades son las siguientes:

1. Identificación de la Configuración.
2. Control de Cambios en la Configuración.
3. Generación de Informes de Estado.
4. Auditoria de la Configuración.

En Mconfig.PM [17] se proponen un conjunto de procesos que resulta nuevo en su composición donde los elementos bases en cada una de estas son los Elementos de Configuración de Software (ECS):

1. Identificación de los Elementos de Configuración (**IEC**).
2. Gestión de Cambios en la Configuración (**GCC**).
3. Gestión de Versiones (**GVE**).
4. Planificación de la Gestión de Configuración (**PGC**).
5. Generación de Informes y Comunicación del Estado de la Configuración (**GIE**).

ISO sugiere para esta área los siguientes procesos [18]:

1. Identificación de la Configuración.
2. Control de Cambios de la Configuración.
3. Informe del estado de la configuración.
4. Auditoria de la Configuración.

Se puede concluir diciendo que existe una similitud entre los conceptos de varios autores acerca de los procesos a incluir en la Gestión de Configuración.

Fig. 1.6 Procesos de la GCS.

Identificación de la Configuración.

Para poder llevar a cabo de forma exitosa el proceso de Identificación de la Configuración de Software, es necesario realizar un conjunto de actividades [19].

- Establecer jerarquía preliminar del software.

- Selección de los Elementos de Configuración.
- Definición de las relaciones en la Configuración.
- Definición de un esquema de Identificación.
- Definición y establecimiento de líneas base.
- Definición y establecimiento de bibliotecas de software.

Selección de los Elementos de Configuración.

Consiste en recolectar los ECS (la información creada como parte del proceso de ingeniería).

Definición de las relaciones en la Configuración.

Los elementos de configuración se relacionan entre si. Cuando se efectúa un cambio resulta de gran importancia valorar la repercusión de este, para ello debemos conocer cuantos ECS se ven afectados ya sea de forma directa o indirecta. Por tanto definir la relación de los ECS no es más que plasmar la relación, valga la redundancia, que existe entre todos y cada uno de los ECS.

Definición de un esquema de Identificación.

Describe como serán nombrados, marcados y numerados los Elementos de Configuración del proyecto, estos pueden comprender código fuente, ejecutables, documentación, modelos, hardware.

Definición y establecimiento de líneas base.

La línea base provee un estándar oficial en el que se basan los siguientes trabajos y en el que solo se realizan los cambios autorizados.

Describe en que punto del ciclo de vida son establecidas las líneas bases. Las líneas bases más comunes son las que se generan al final de cada fase. Describe quien autoriza la línea base y que se hace con ella.

Definición y establecimiento de bibliotecas de software.

Describe los tipos de bibliotecas que se usaran para el desarrollo de las aplicaciones, así como los niveles de acceso y la dirección física de las mismas.

Control de Versiones.

Se puede definir una versión como una instancia de un elemento de configuración, en un momento dado del proceso de desarrollo, que es almacenada en un repositorio, y que puede ser recuperada en cualquier momento para su uso o modificación.

Se llama *control de versiones* a la gestión de versiones (revisiones) de todos los elementos de configuración que forman la línea base de un producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado junto a las posibles especializaciones realizadas para algún cliente específico.

El control de versiones combina procedimientos y herramientas para gestionar las versiones de los objetos de configuración creadas durante el proceso de ingeniería del software.

Es posible realizar configuraciones alternativas del sistema de software si se seleccionan bien las versiones adecuadas, esto sucede gracias a los atributos, es decir, a la asociación de atributos a cada versión del software.

Los atributos pueden ser tan sencillos como un número específico de versión asociado a cada objeto o tan complejos como una cadena de variables lógicas que especifiquen tipos de cambios funcionales aplicados al sistema.

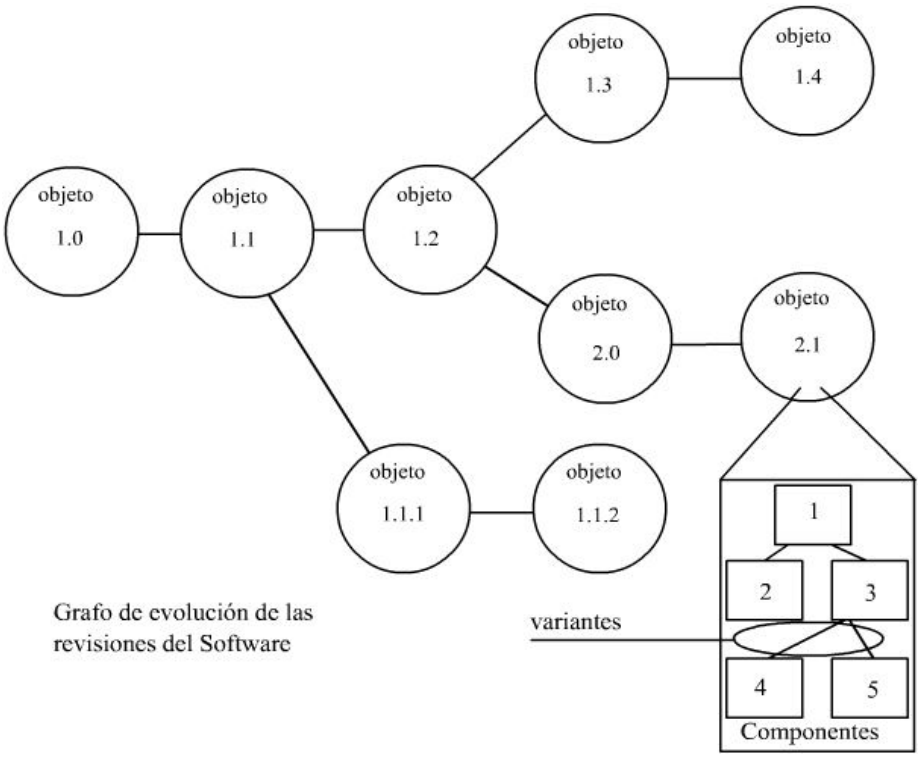

Fig. 1.8 Versiones y variantes.

Para construir la variante adecuada de una determinada versión de un programa, a cada componente se le asigna una tupla de atributos. A cada variante se le asigna uno o más atributos [20].

Otra forma de establecer los conceptos de la relación entre componentes, variantes y versiones es representarlas como un fondo de objetos [21].

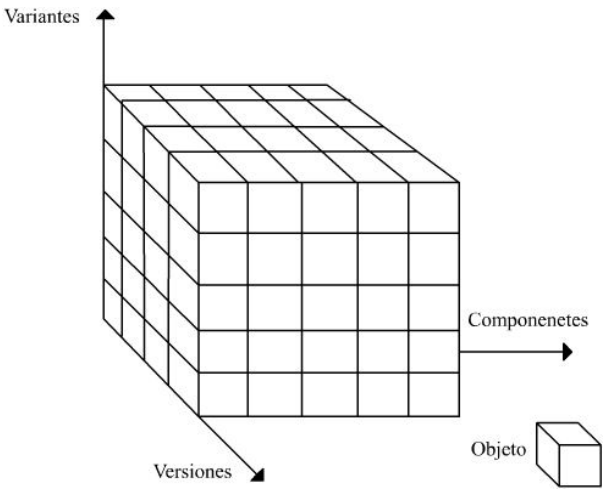

Fig.1.9 Representación de objetos, componentes, variantes y versiones.

Características.

Un sistema de control de versiones debe proporcionar:

- Mecanismo de almacenaje de cada uno de los ítems que deba gestionarse (archivos de texto, imágenes, documentación, etc.).
- Posibilidad de modificar, mover, borrar cada uno de los elementos.
- Histórico de las acciones realizadas con cada elemento pudiendo volver a un estado anterior dentro de ese historial.

Aunque no es estrictamente necesario, suele ser muy útil la generación de informes con los cambios introducidos entre dos versiones, informes de estado, marcado con nombre identificativo de la versión de un conjunto de ficheros, etcétera.

Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados.

Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial. Para modificar la copia local existen dos semánticas básicas:

Exclusivos: para poder realizar un cambio es necesario marcar en el repositorio el elemento que se desea modificar y el sistema se encargará de impedir que otro usuario pueda modificar dicho elemento. Este proceso se suele conocer como check out o desproteger.

Colaborativos: en el que cada usuario se descarga la copia la modifica y el sistema automáticamente mezcla las diversas modificaciones. El principal problema es la aparición de conflictos que deben ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas. Además, esta semántica no es apropiada para ficheros binarios.

Tras realizar la modificación es necesario actualizar el repositorio con los cambios realizados. Habitualmente este proceso se denomina commit, check in o proteger.

Control de Cambios.

En un gran proyecto de desarrollo de software, el cambio incontrolado lleva rápidamente al caos. El control de cambios combina los procedimientos humanos y las herramientas automáticas para proporcionar un mecanismo para el mismo, que no es más que el proceso que incluye proponer un cambio, evaluarlo, aprobarlo o rechazarlo, planificarlo y hacerle un seguimiento.

Antes de que un ECS se convierta en una línea base, sólo es necesario aplicar un control de cambios informal. El que haya desarrollado el ECS en cuestión podrá hacer cualquier cambio justificado por el proyecto y por los requisitos técnicos. Una vez que el objeto ha pasado la revisión técnica formal y ha sido aprobada, se crea la línea base.

Una vez que el ECS se convierte en una línea base, aparece el control de cambios a nivel de proyecto. Para hacer un cambio, el encargado del desarrollo debe recibir la aprobación del gestor del proyecto (si el cambio es local), o de la Autoridad de Control de Cambio (ACC) si el cambio impacta en otros ECS. En algunos casos, se dispensa de generar formalmente las peticiones de cambio, los informes de cambio y las Órdenes de Cambio de Ingeniería (OCI). Sin embargo, hay que hacer una evaluación de cada cambio así como un seguimiento y revisión de los mismos [22].

Cuando se distribuye el producto de software a los clientes, se instituye el control de cambios formal.

La autoridad de control de cambios desempeña un papel activo en el segundo y tercer nivel de control. El papel de la ACC es el de tener una visión general, o sea, evaluar el impacto del cambio fuera del ECS en cuestión. ¿Cómo impactará el cambio en el hardware? ¿Cómo impactará en el rendimiento? ¿Cómo alterará el cambio la percepción del cliente sobre el producto? [23]

Las tareas a realizar durante el Control de Cambios se pueden resumir a:

- Actualizar la copia local.
- Generar un mecanismo para llevar el control de los cambios.
- Protocolos de actuación para personas.
- Herramientas de automatización de cambios.
- Aplicación de políticas de gestión de cambios informales salvo en el proceso de conversión de ECO a Línea Base (crítico).
- Procesos a controlar.

- Baja del objeto en la biblioteca.
- Alta del objeto en la biblioteca.
- Aplicación de la política de versiones sobre los objetos _ Control de versiones.
- Elementos a definir:
    a. Control de acceso al repositorio (seguridad y permisos) _ “Autoridad de Control de Cambios”.
    b. Control de sincronización (concurrencia de acceso al repositorio).
    c. Orden de Cambio (decisión final)

Auditoria de la Configuración.

Para asegurarse que los cambios han sido efectuados de forma correcta pueden realizarse dos procedimientos:

1. Revisiones Técnicas Formales.
2. Auditorias de Configuración del Software.

Las Revisiones Técnicas Formales se centran en la corrección técnica del elemento de configuración que ha sido modificado. Los revisores evalúan el ECS para determinar la consistencia con otros ECS, las omisiones o los posibles efectos secundarios.

Una auditoria de configuración del software complementa la revisión técnica formal al comprobar características que generalmente no tiene en cuenta la revisión. Además es el complemento de las tareas anteriores (identificación, control de cambios y versiones). La auditoria se plantea y responde con las siguientes preguntas:

- ¿Se ha hecho el cambio especificado en la OCI? ¿Se han incorporado modificaciones adicionales?
- ¿Se ha llevado a cabo una revisión técnica formal para evaluar la corrección técnica?
- ¿Se han seguido adecuadamente los estándares de ingeniería de software?
- ¿Se han "recalcado" los cambios en el ECS? ¿Se han especificado la fecha del cambio y el autor? ¿Reflejan los cambios los atributos del objeto de configuración?
- ¿Se han seguido procedimientos del GCS para señalar el cambio, registrarlo y divulgarlo?

- ¿Se han actualizado adecuadamente todos los ECS relacionados?

La auditoria es realizada por el grupo de Calidad.

Informes de Estado.

La generación de informes de estado de la configuración (IEC) consiste en elaborar un informe, que contenga:

1) ¿Qué pasó?
2) ¿Quién lo hizo?
3) ¿Cuándo pasó?
4) ¿Que más se vio afectado?

Se debe generar un IEC:

- Cada vez que se asigna una nueva identificación a un ECS.
- Cada vez que la ACC expide una OCI.
- Cada vez que se lleva a cabo una auditoria de configuración.
- Regularmente, para mantener informados a los desarrolladores de los cambios importantes.

Los IEC se pueden depositar en una base de datos [24].
Generar informes de estado resulta de gran importancia para que el proyecto alcance el éxito, sobre todo si en el desarrollo de este están involucradas muchas personas. Los Informes de Estado facilitan la comunicación entre los desarrolladores del proyecto.

1.3 Conclusiones.

La gestión de la configuración de un software posibilita una mejor organización del desarrollo y mantenimiento del producto, facilitando el resto de los procesos de producción. Permite identificar, definir y almacenar en una línea base los elementos de software; controla los cambios, reporta y registra el estado de los elementos y de las solicitudes de cambio; asegura la completitud, consistencia y corrección de los elementos; controla, almacena, maneja y libera los elementos asociados al producto de software.

Capítulo 2

Características y estructura del sistema, Intranet II.

La Intranet de la Universidad de la Ciencias Informáticas es actualmente el espacio informativo de funcionamiento constante más visitado dentro de las opciones informativas existentes, pues la dinámica propia de este centro en el que las tecnologías de la información y las comunicaciones tienen un peso inigualable, así lo propicia. La aspiración fundamental es convertir a la Intranet en el portal de la Ciudad Digital, un espacio que permita el acceso a los servicios que se ofrecen en la universidad y que por su alcance refleje el acontecer de esta comunidad. De forma general con este capítulo se pretende describir el sistema a implementar, con dos objetivos fundamentales: Primero brindar una noción del software a construir, creando una idea del sistema a gestionar y segundo aprovechar los medios a través de los cuales se describe el sistema (Diagramas, Requisitos, Vistas) para presentar los primeros elementos de configuración del software (ECS) del proyecto Intranet II. Recordar que los ECS constituyen la célula de la GCS. De forma mas especifica para el análisis del contenido, el capítulo estará dividido en tres temáticas fundamentales, Análisis, Diseño, Arquitectura.

2.1 Análisis

El proceso de modelado posibilita obtener una visión de la organización permitiendo definir los procesos y roles relacionado con la obtención de requerimientos y análisis-diseño. Sí se determina que no es necesario un modelo completo del negocio se realizará lo que se conoce como un *modelamiento del dominio*. [25]

2.1.1 Modelo del dominio.

Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. [26]

Debido al bajo nivel de estructuración que presenta el negocio que se está estudiando y que está altamente centrado en las tecnologías informáticas, se propone un modelo del dominio ayudando a los

usuarios, clientes, desarrolladores y demás interesados, a utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema.

El modelo del dominio se describe mediante diagramas UML, específicamente con un diagrama de clases conceptuales significativas en el dominio del problema. Y va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema.

2.1.2 Requisitos del Sistema.

Reúne todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan acerca de lo que debe hacer el sistema. Los requisitos se pueden clasificar en: funcionales y no funcionales.

Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, no alteran la funcionalidad del producto y se mantienen invariables sin importarle con que propiedades o cualidades se relacionen. Son el punto de partida para identificar qué debe hacer el sistema. [27]

A continuación se relacionan los que deben cumplir el portal de la Intranet 2:

R1. Autenticar usuario.

- El sistema debe ser capaz de autenticar los usuarios por el dominio.
- El sistema debe ser capaz de identificar los grupos de usuarios en dependencia al rol que desempeñan.

Esta información requiere:

a-) Nombre de usuario.

b-) Contraseña.

R2. Adicionar noticias.

- El sistema debe ser capaz de adicionar las noticias por categorías, confeccionadas por los periodistas o encontradas en los diferentes medios de comunicación.

Esta información requiere:

a-) Categoría.

b-) Título.

c-) Resumen.

d-) Texto de la noticia.

e-) Fotos.

f-) Fuentes.

g-) Archivos.

h-) Fecha de publicación.
i-) Fecha de retirada.
j-) Autor.

R3. Actualizar noticias.

- El sistema debe ser capaz de actualizar las noticias diariamente en dependencia de las diferentes categorías.

Esta información requiere
a-) Categoría.
b-) Título.
c-) Resumen.
d-) Texto de la noticia.
e-) Fotos.
f-) Fuentes.
g-) Archivos.
h-) Fecha de publicación.
i-) Fecha de retirada.
j-) Autor.

R4. Aprobar noticias.

- El sistema debe ser capaz de decidir si la noticia cumple con los requisitos para ser publicada.

Esta información requiere:
a-) Categoría.
b-) Autor o fuente.
c-) Tema.
d-) Palabras en el título.
e-) Fecha de publicación.
f-) Fecha de caducidad.
g-) Palabras claves.

R6. Adicionar Avisos.

- El sistema debe ser capaz de adicionar avisos creados por los responsables de aéreas que pueden ser de interés para ciertos usuarios o para todos en general.
- El sistema debe ser capaz de identificar los avisos según sean ser docentes, eventuales o de otras categorías.
- Pueden estar enmarcados de importancia o de prioridad.

  Esta información requiere:

  a-) Autor.
  b-) Nivel de prioridad.
  c-) Categorías.
  d-) Fechas de publicación.
  e-) Categoría.
  f-) Título.
  g-) Texto.
  h-) Fecha de retirada.
  i-) Solicitud de área.

R7. Actualizar avisos.

- El sistema debe ser capaz de actualizar los avisos según vayan caducando en la fecha de vigencia.
- Debe ser capaz de identificar los avisos por categorías y nivel de prioridad.

  Esta información requiere:

  a-) Contenido nuevo.
  b-) Categoría.
  c-) Nivel de prioridad.
  f-) Título.
  g-) Texto.
  h-) Fecha de publicación.
  i-) Fecha de retirada.
  j-) Autor.
  k-) Solicitud de área.

R8. Aprobar avisos.

- El sistema debe ser capaz de aprobar los avisos que serán publicados a partir de que cumplan con los requisitos para la publicación.

Esta información requiere:

a-) Autor.

b-) Título.

c-) Nivel de prioridad.

d-) Categorías.

e-) Fechas de publicación.

f-) Fecha de retirada.

g-) Solicitud del área.

R10. Adicionar efemérides.

- El sistema debe ser capaz de adicionar efemérides de acuerdo al día en curso.

Esta información requiere:

a-) Texto.

b-) Fecha.

c-) Título.

e-) Aprobación.

f-) Fecha de publicación.

g-) Archivos.

R11. Actualizar efemérides.

- El sistema debe ser capaz de actualizar diariamente las efemérides en correspondencia a la fecha.

Esta información requiere:

a-) Texto.

b-) Fecha.

c-) Título.

e-) Aprobación.

f-) Fecha de publicación.

g-) Archivos.

R12. Aprobar efemérides.

- El sistema debe ser capaz de aprobar las efemérides que serán publicadas diariamente.

Esta información requiere:

a-) Texto.
b-) Fecha.
c-) Título.
e-) Aprobación.
f-) Fecha de publicación.
g-) Archivos.

R14. Enviar solicitud de publicación SW.

- El sistema debe ser capaz enviar las solicitudes confeccionadas por los diferentes responsables de áreas.

Esta información requiere:

a-) Descripción del sitio.
b-) Nombre del solicitante.
c-) Clasificación del sitio.

R15. Aprobar solicitud de publicación SW.

- El sistema debe ser capaz de aprobar los sitios web que se publicarán.

Esta información requiere:
a-) Descripción del sitio.
b-) Clasificación.

R16. Publicar Sitios Web.

- El sistema debe ser capaz de publicar los sitios web de las diferentes aéreas de la universidad de acuerdo con la aprobación del consejo editor.

Esta información requiere:

a-) Descripción del sitio.

b-) Clasificación.

R17. Enviar solicitud de publicación de páginas Informativas.

- El sistema debe ser capaz de recibir las solicitudes enviadas por los diferentes responsables de aéreas para publicar las páginas informativas, estas recogen la información de determinado usuario, área o servicio perteneciente a la universidad.

Esta información requiere:

a-) Título.

b-) Fuente (Área de la universidad).

c-) Resumen del contenido de la página (Texto).

R18. Aprobar solicitud de publicación de páginas Informativas.

- El sistema debe ser capaz de enviar la aprobación de la solicitud de publicación de páginas Informativas.

Esta información requiere:

a-) Título.

b-) Fuente (Área de la universidad).

c-) Resumen del contenido de la página (Texto).

d-) Respuesta.

R19. Publicar Páginas informativas.

- El sistema debe ser capaz de publicar las páginas informativas que han sido aprobadas.

Esta información requiere:

a-) Título.

b-) Fuente (Área de la universidad).

c-) Página a publicar.

R20. Adicionar usuario.

- El sistema debe ser capaz de adicionar usuario en dependencia del rol a desempeñar.

Esta información requiere:

a-) Nombre del usuario.

b-) Fecha de creación.

c-) Área a la pertenece.

d-) Rol.

R21. Modificar usuario.

- El sistema debe ser capaz de modificar los usuarios, en dependencia del rol que desempeñe en el momento.

Esta información requiere:

a-) Nombre del usuario.

b-) Fecha de creación.

c-) Área a la pertenece.

d-) Rol.

R22.Eliminar usuario

- El sistema debe ser capaz de eliminar usuario y rol correspondiente.

Esta información requiere:

a-) Nombre del usuario.

b-) Fecha de creación.

c-) Área a la pertenece.

d-) Rol.

R23. Buscar información del contenido del portal.

- El sistema debe ser capaz de realizar una búsqueda y recopilación de cualquier información perteneciente a la universidad existente en el portal.

Esta información requiere:

a-) Información que se va a buscar.

b-) Palabras claves.

c-) Autor.

R24. Mostrar estadísticas del portal.

- El sistema debe ser capaz de mostrar las estadísticas del portal, usuario que han visitados, sitios más visitados, información más buscada y servicios más utilizado.

Esta información requiere:

a-) Usuarios.
b-) Sitios visitados.
c-) Servicios.

Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características que hacen al producto atractivo, usable, rápido o confiable, son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales. [28]

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. [29]

Entre las diferentes categorías de los requerimientos no funcionales se encuentran:

1. Usabilidad.
   - El portal podrá ser accedido por todos los usuarios de la UCI.
   - Debe tener un diseño sencillo.
   - La navegación debe ser fácil para el usuario.
   - No se debe recargar de contenido que resulte molesto.
2. Portabilidad.

El sistema será multiplataforma.

3. Seguridad.
   - El sistema se regirá por el reglamento disciplinario de navegación de la universidad, así como la seguridad que ofrece PLONE.
   - Debe estar protegida con la seguridad de Linux. Los usuarios que accedan a la aplicación solo tienen acceso según el rol que cumplan dentro de esta.
4. Implementación.

Utilizar el servidor Apache para remplazar el Zope Web Server. Para manejar las BD utilizar PostgreSQL en reemplazo de ZopeDB. Usar CMS Plone y el lenguaje que trae incorporado Python para el desarrollo de la aplicación.

5. Apariencia.

La apariencia del sistema debe ser lo más accesible y usable posible, cumplir las normas de la UCI, en dependencia del rol que cumpla el usuario.

6. Software.

Cliente:

- Cualquier sistema operativo con interfaz gráfica y red.
- Navegador Web.

Servidor:

- Linux cualquier distribución (Debian, Ubuntu, KDE, etc.).
- Servidor de BD PostgreSQL en reemplazo de ZopeDB.
- Apache para remplazar el Zope Web Server.

7. Hardware.

Para el desarrollo:

- Pentium IV o superior.
- 256 de memoria RAM o superior.
- Disco duro con capacidad de 80 GB o superior.

Para la explotación:

Clientes:

- Pentium IV o superior.
- 240 de RAM o superior.

8. Legales.

El sistema contara con normas de de estilos y pautas de diseño para cada uno de los servicios de la intranet aprobadas por el consejo de dirección de la Universidad.

9. Confiabilidad.

El sistema solo será visitado por los usuarios que cuenten con acceso al dominio UCI.

10. Interfaz.

- La interfaz del sistema será muy legible.
- También será fácil de usar.

- Debe regirse por el manual de estilo de la Universidad.

11. Ayuda y documentación en línea.

El sistema cuenta con ayuda para navegar en el portal, una guía de estilos y pautas de diseño.

2.1.3 Casos de uso.

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema. Entre ellos se encuentran entre las actividades a automatizar. [30]

Definición de los actores.

Actores del Sistema: Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

Nombre del Actor	Descripción
Productor	Usuario que realiza la acción de crear los contenidos que se publicarán en el portal.
Editor	Usuario encargado de los arreglos, edición y de cumplir con las normas y estilos establecidos por la UCI, para la publicación de contenidos.
Revisor	Consejo editorial, compuesto por directivos y periodistas, que se encargan de decidir que contenido será publicado en el portal y de que cumpla con las normas de estilos establecidas por la UCI.
Publicador	Usuario con permisos de administración, que publica el contenido en el portal.

Tabla 2.1 Actores del Sistema.

Listado de casos de uso del sistema.

Casos de Uso del Sistema (CU): son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para los actores. Un CU especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia. Cada forma en que los actores usan el sistema se representa con un caso de uso.

Id. Caso de Uso	Nombre	Actor
CU-1	Autenticar usuario	Usuario
CU-2	Adicionar Contenido	Productor.
CU-3	Actualizar Contenido	Editor
CU-4	Aprobar Contenido	Revisor
CU-5	Enviar solicitud de publicación	Productor
CU-6	Aprobar solicitud de publicación	Revisor
CU-7	Publicar SW	Publicador.
CU-8	Publicar Página Informativa	Publicador.
CU-9	Gestionar usuario	Publicador.
CU-10	Buscar información del contenido del Portal	Usuario.
CU-11	Mostrar estadísticas del portal	Publicador.

Tabla 2.2

Diagrama de Casos de Uso del Sistema.

En el diagrama de casos de uso del sistema se representan las relaciones (inclusión, extensión y generalización/especialización) entre los casos de uso, los actores (generalización/especialización), así como los casos de uso asociados con la seguridad. Estos se pueden estructurar en paquetes y poner el diagrama con las dependencias entre ellos.

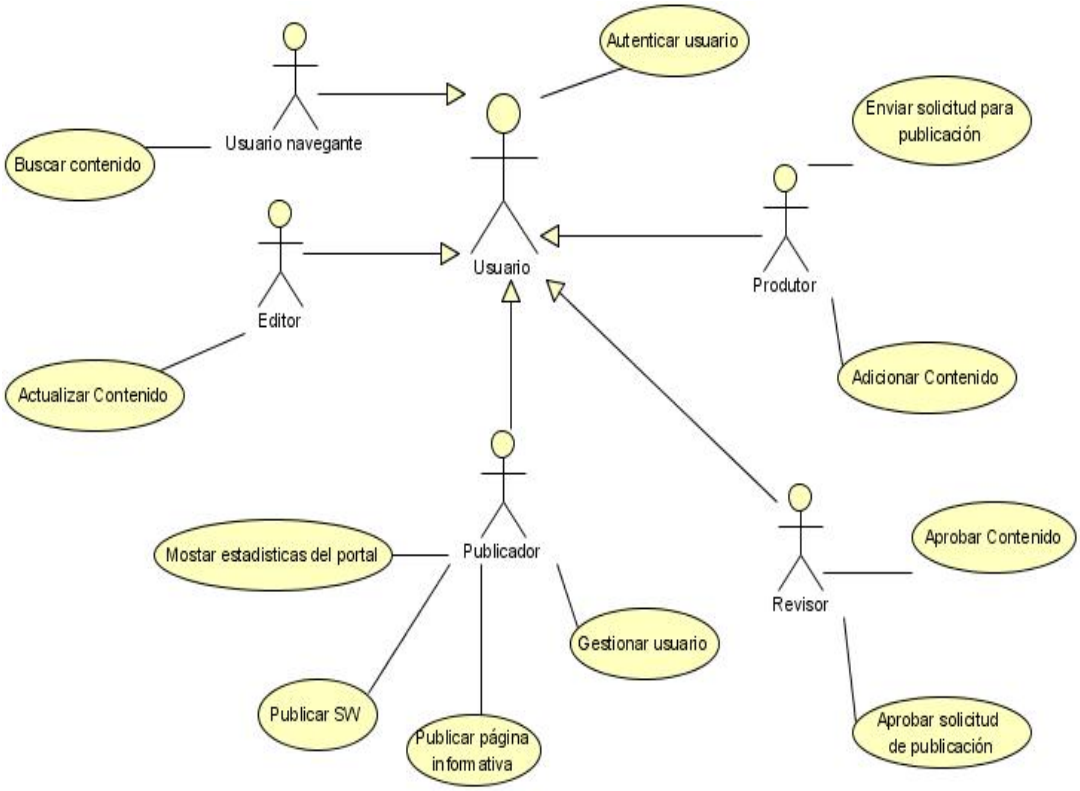

Fig. 2.1 Diagrama de CUS

2.2 Diseño.

El proceso de Diseño es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del Sistema a construir. El objetivo principal es traducir los requisitos a una especificación que describe cómo implementar el sistema.

2.2.1 Diseño del Sistema.

El diseño es donde se fomenta la calidad del software, es la única manera de materializar los requerimientos del cliente.

Diagrama Clases del Diseño.

- Clases Web.
- Colaboración.

Diagrama de Clases Web (un ejemplo).

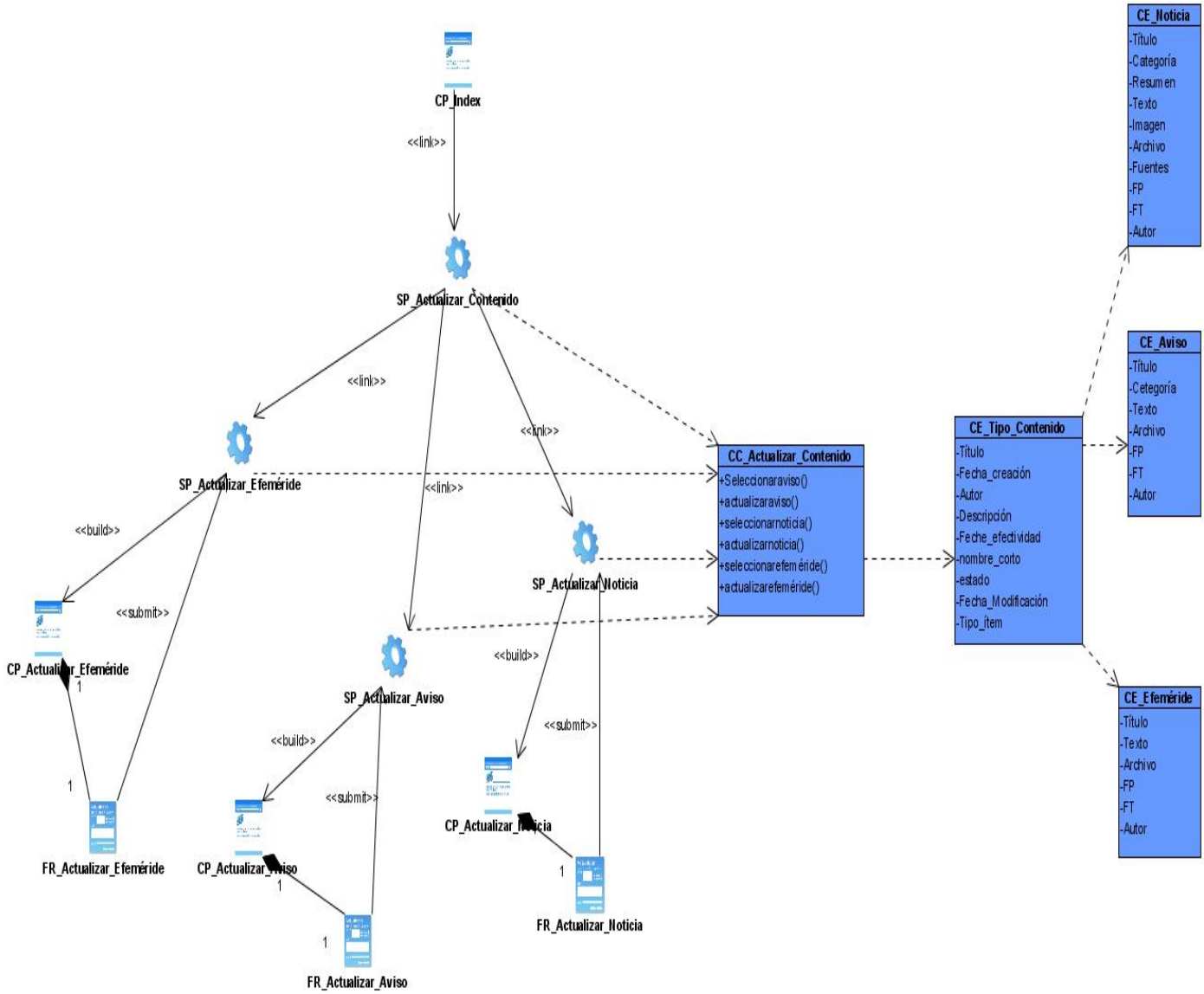

Fig. 2.2 Diagrama de clase del diseño para el caso de uso Actualizar Contenido.

Diagramas de Colaboración del Diseño (un ejemplo).

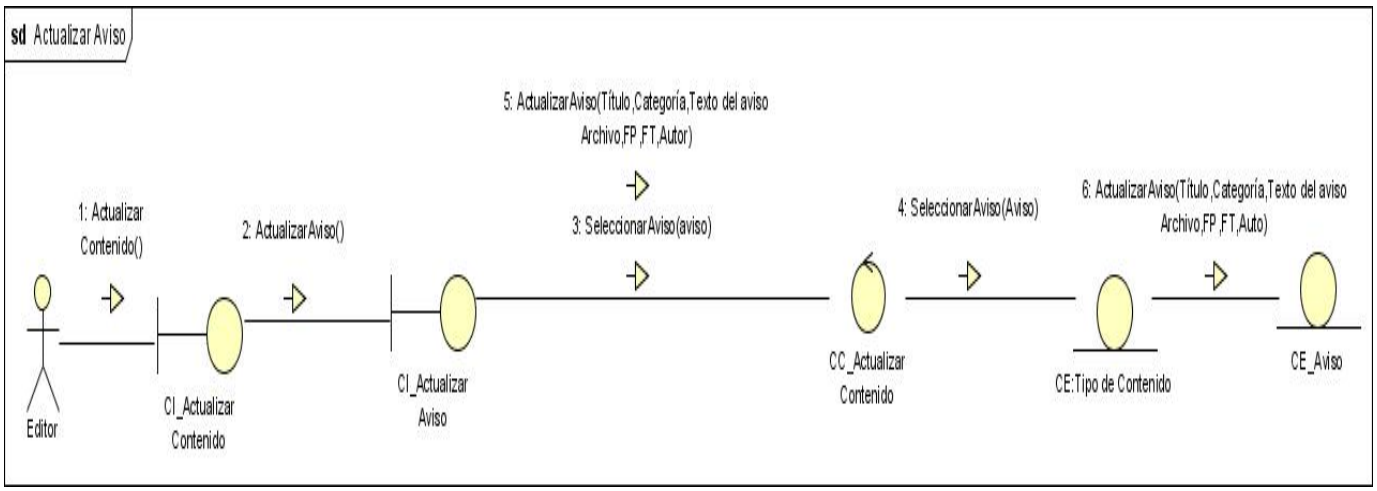

Fig 2.3 Diagrama de colaboración para el caso de uso Actualizar Aviso.

2.2.2 Diseño de las Bases de datos.

Para entender mejor el diagrama de entidad relación o modelo de datos se describen primeramente las clases que persisten [31] que son las tablas de la base de datos. Después de haber obtenido el diagrama de clases persistentes se hace mucho más fácil obtener el diagrama de la Base de datos como se encuentra a continuación.

Diagramas de clases persistentes.

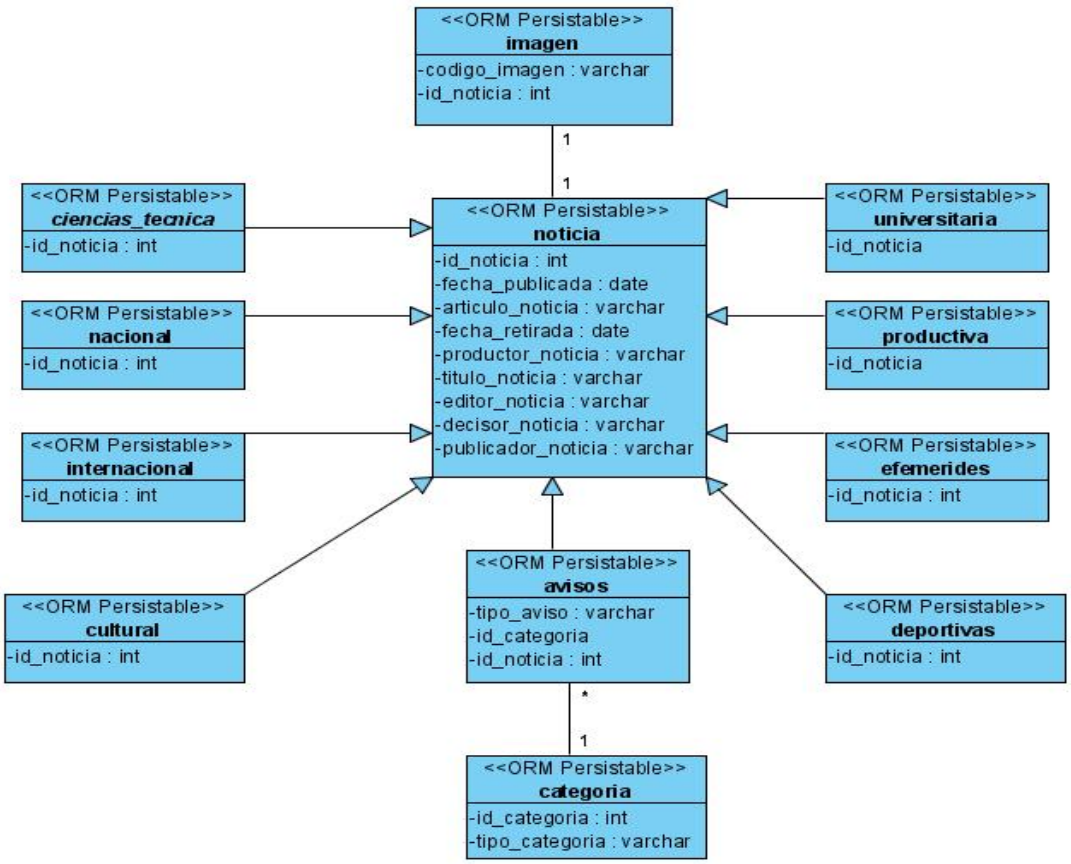

Fig. 2.4 Diagrama de Clases Persistentes, Noticias.

Diagramas Entidad Relación.

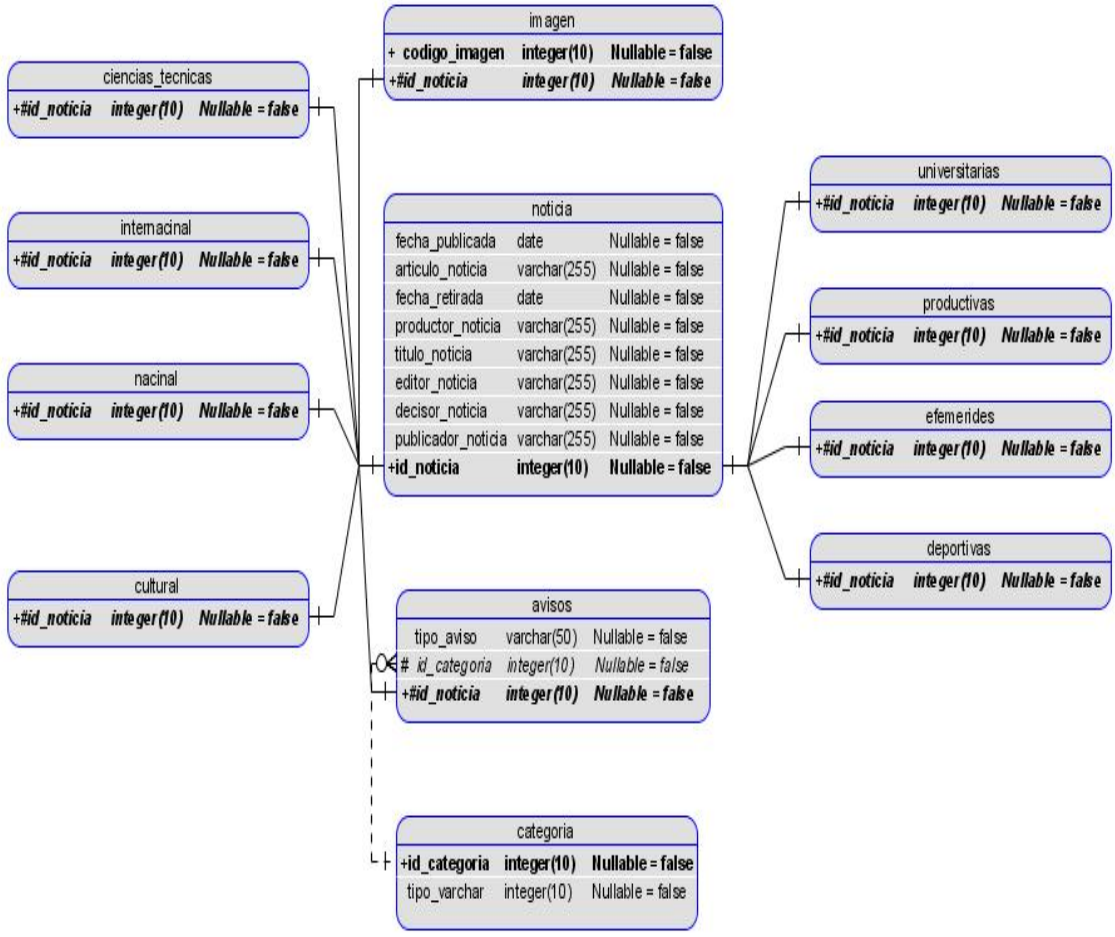

Fig. 2.5 Diagramas Entidad Relación, Noticias.

2.3 Arquitectura de software.

El propósito de desarrollar la arquitectura de software es, primeramente crear la estructura o el esqueleto del sistema que cohesiones las funcionalidades más criticas y relevantes, y que sirva de soporte a al resto de las funcionalidades finales. Luego realizar el ensamblado de las distintas partes. Todo esto hace del diseño de un software particular una tarea generalmente única.

La arquitectura de software constituye un documento guía que se define a partir de un conjunto de requisitos críticos funcionales, de rendimiento o de calidad, donde se especifica detalladamente los principales componentes que integran el sistema y como se relacionan entre si, facilitando el entendimiento de cada uno de los miembros del equipo de desarrollo.

2.3.1 Vista de Caso Uso.

En esta vista se muestra la percepción que tiene el usuario de las funcionalidades del sistema mediante la representación de los actores y casos de usos más importante.

Desde el punto de vista arquitectónico:

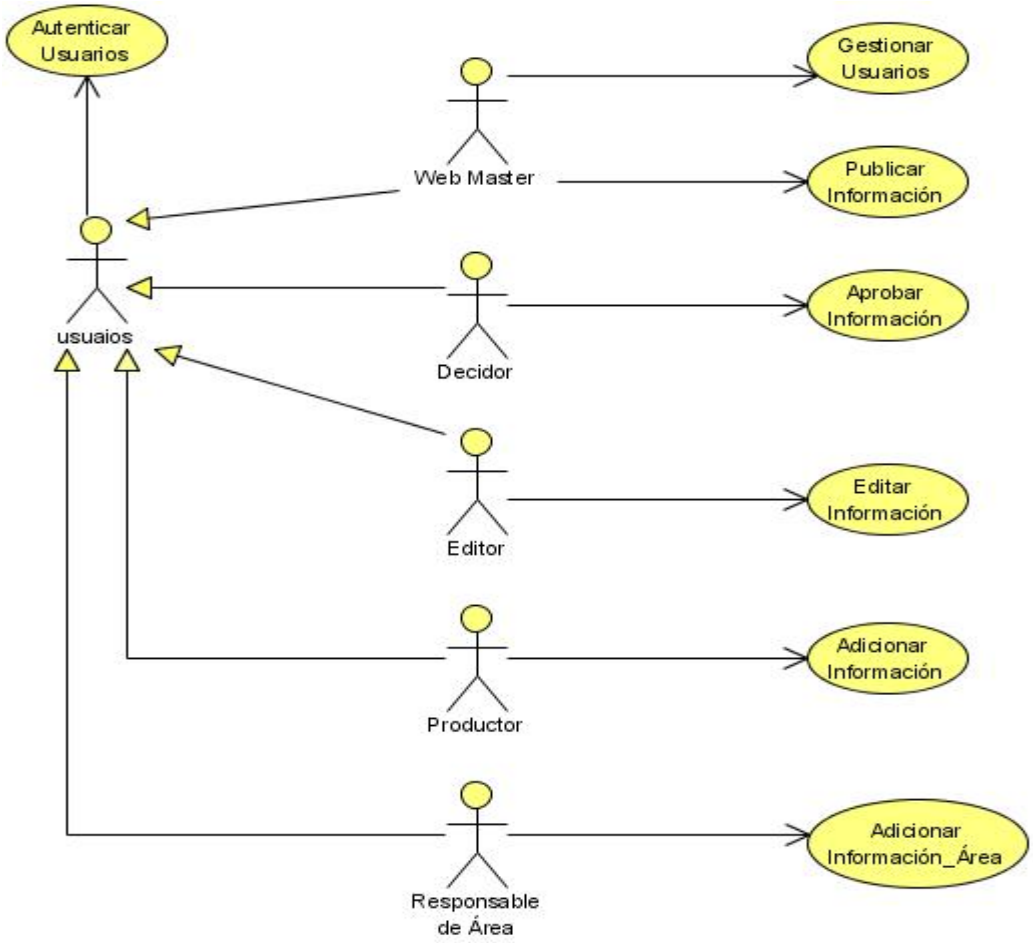

Fig. 2.6 Vista de Casos de Uso Arquitectónicamente Significativos.

2.3.2 Vista de Restricciones.

En esta vista se presentan las restricciones a la que está sujeto tanto el proceso de desarrollo como el producto desarrollado.

UML

Interfaz Web

Servicios Web

Tecnologías

Soporte

2.3.3 Vista de Calidad.

Describe los requerimientos no funcionales dentro de las categorías de:

Usabilidad.

Confiabilidad.

Portabilidad.

Seguridad.

2.3.4 Vista Lógica.

La vista de arquitectura del modelo de diseño o la lógica presenta los subsistemas e interfaces más importante para la arquitectura.

Arquitectura Sistema.

Descripción de la arquitectura vertical.

Los casos de usos arquitectónicamente significativos se agruparon en los siguientes módulos (se muestra los principales módulos que el sistema debe tener integrado).

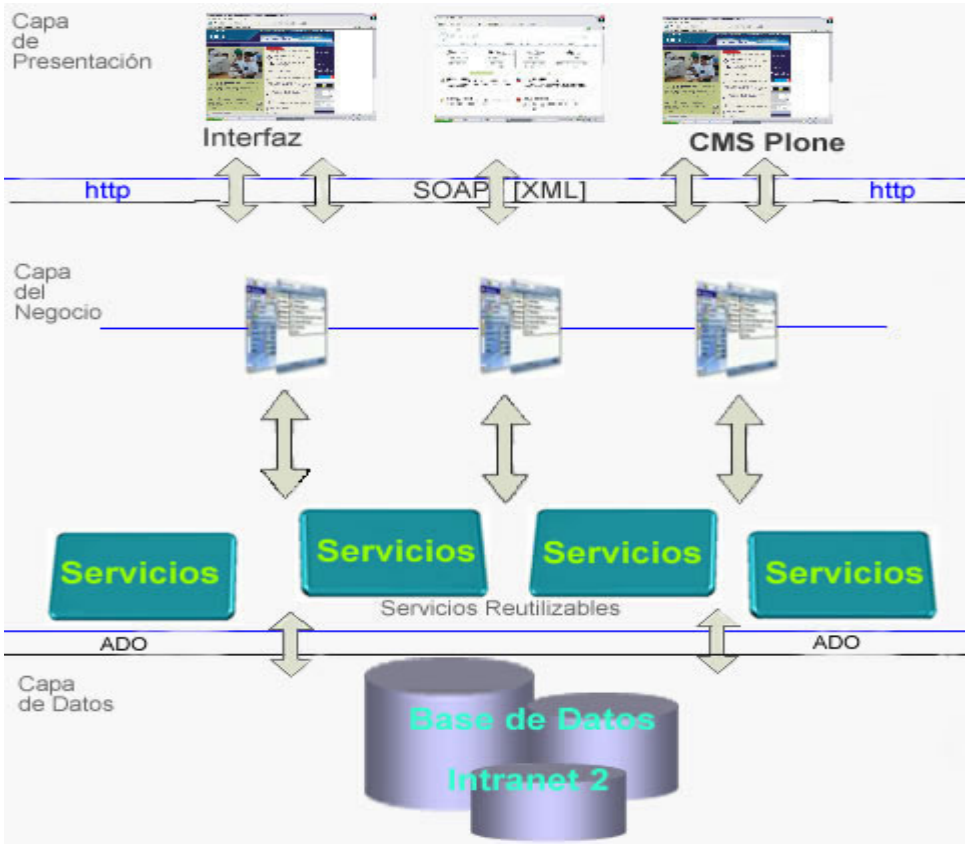

Fig. 2.7 Arquitectura Vertical del Sistema.

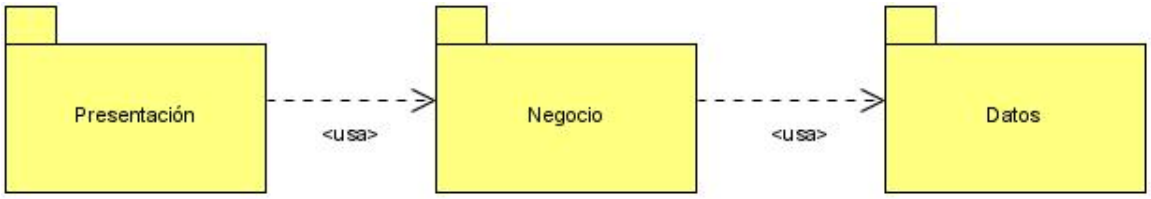

Fig. 2.8 Estructura del Sistema.

Descripción de la arquitectura horizontal.

Aquí se muestran los principales módulos que debe tener el sistema. Los casos de usos arquitectónicamente significativos se agruparon en los siguientes módulos:

Módulo Acceso: En este módulo se encuentra todo lo referente al acceso al sistema.

Caso de uso Contenido:

- Autenticar usuario.

Módulo Gestión de Usuario: En este módulo se encuentra todo lo referente a la gestión de perfil de los usuarios, entiéndase adicionar usuario, eliminar usuario y editar datos de usuario.

Caso de uso Contenido:

- Caso de Uso Gestionar Usuario.

Módulo Publicación: Este módulo engloba todo lo referente a las informaciones o artículos que se publicaran en el portal. Estas informaciones pueden ser noticias, avisos, efemérides, sitios Web, servicios, etc. Estos artículos pueden traer adjunto imágenes, sonidos, videos.

Caso de uso Contenido:

- Publicación Información.

Módulo Información: En este módulo se encapsula todo lo referente al proceso involucrado de las informaciones antes de su publicación, es decir, adicionar un artículo, editar un artículo y aprobar un artículo. Estos artículos pueden ser noticias, avisos, efemérides y sitios Web.

Caso de uso Contenido:

- Aprobar Información.
- Información.
- Adicionar Información.
- Adicionar Información _ Área.

2.3.5 Vista refinada de la arquitectura del sistema.

En el siguiente diagrama se muestra la arquitectura lógica del sistema, en la cual se representa como se interconectan mediante una interfaz los diferentes y principales subsistemas y paquetes del sistema.

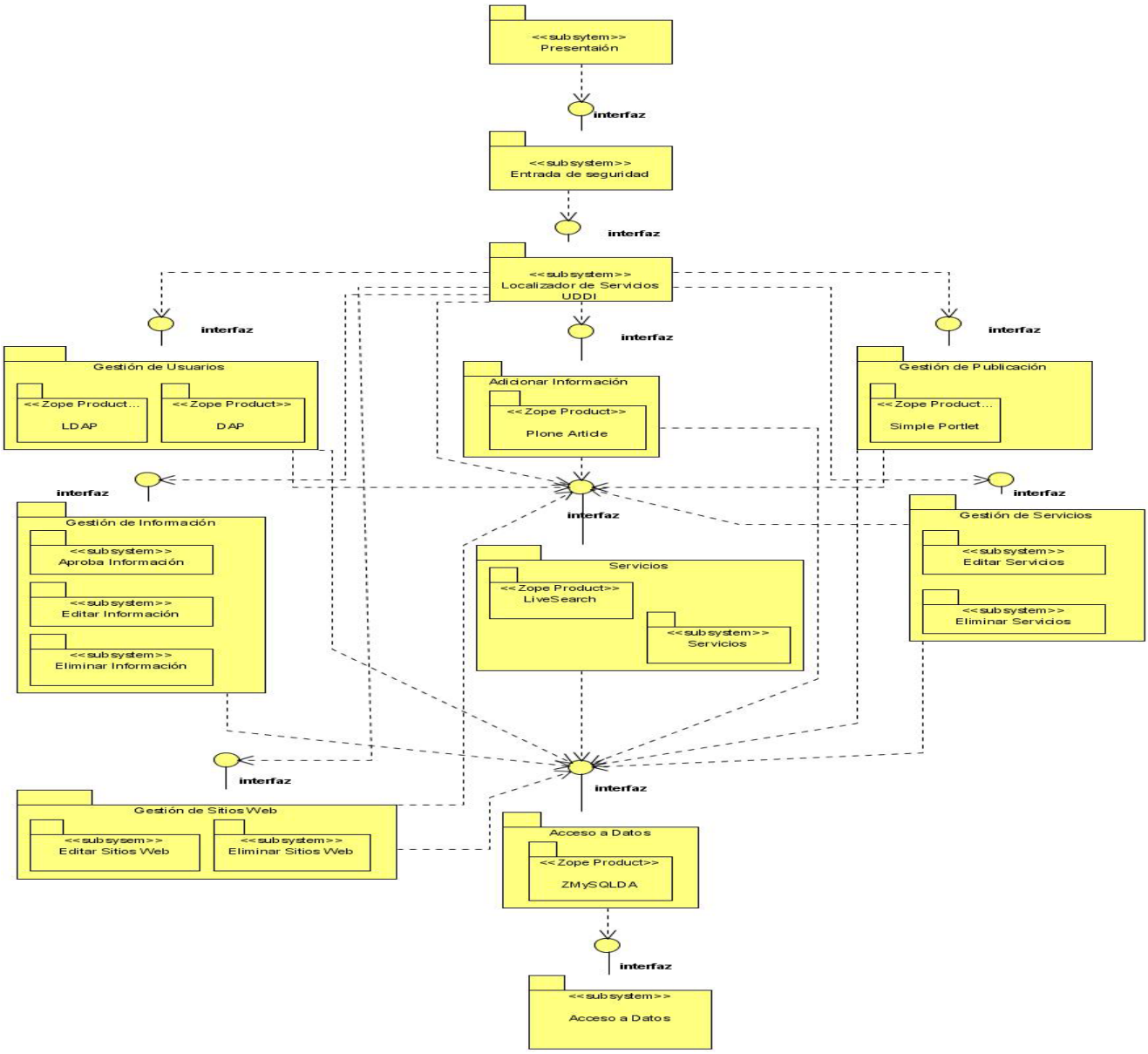

Fig. 2.9 Arquitectura lógica.

2.3.6 Vista de Procesos.

La vista de procesos muestra los principales procesos que el sistema debe tener integrado. Estos procesos se encuentran agrupados por módulos en dependencia de su funcionalidad.

2.3.7 Vista de Despliegue.

El modelo de despliegue define la arquitectura física del sistema mediante la representación de nodos interconectados, en los que se muestran la asignación de los componentes ejecutables a los nodos. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software.

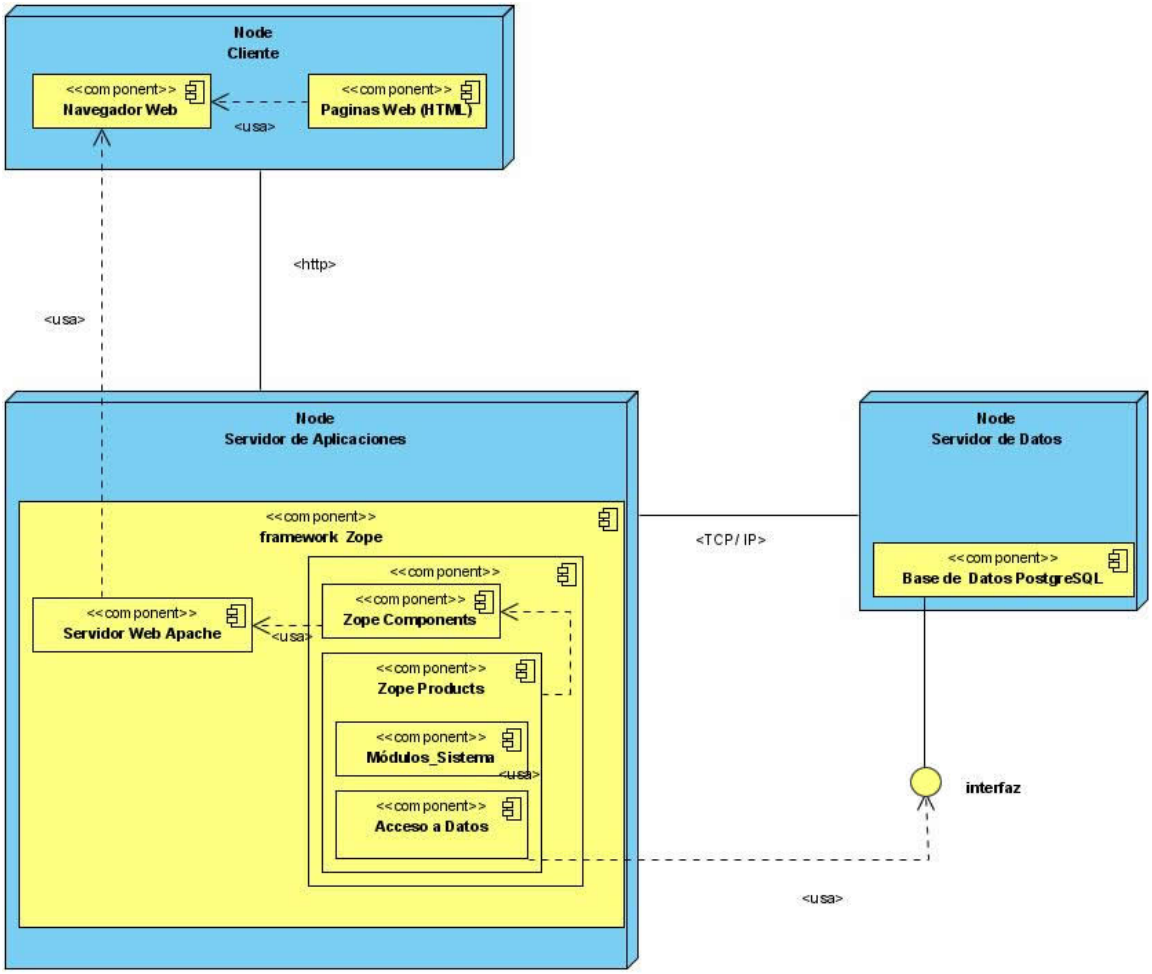

Fig. 2.10 Vista de Despliegue.

2.3.8 Vista de Implementación.

La vista de implementación describe cómo se implementan los componentes físicos mostrados en vista de distribución agrupándolos en subsistemas organizados en capas, ilustra, además las dependencias entre éstos. Básicamente, se describe el mapeo desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

Producto a que no se encuentra definida completamente la Arquitectura de Información de la Intranet 2, y que se va a utilizar a Zope / Plone para la construcción del sistema, esta vista queda omitida, pues, las funcionalidades básicas antes descriptas se realizarán mediante la utilización de los diferentes productos disponibles en Internet para Plone.

2.4 Conclusiones.

En terminología de Gestión de la Configuración se puede decir que con el desarrollo de este capítulo se ha permitido una mejor comprensión del contexto a gestionar, realizando una descripción del sistema, lo cual fue posible gracias a la información brindada por otros roles del proyecto, Analista del Sistema, Diseñador del Sistema, Diseñador de Bases de Datos, Arquitecto. Además se han expuesto algunos de los primeros elementos de configuración de este software.

Capítulo 3

Fundamentación de herramientas a utilizar para la Gestión de la Configuración de Intranet II.

Un proceso de software es el conjunto de actividades, métodos, prácticas y transformaciones que se utilizan para desarrollarlos y mantenerlos junto a sus productos asociados (planes, documentos de diseño, código, casos de prueba, manuales, etc.).
Un método de mejoramiento de procesos es un conjunto de procedimientos, herramientas y entrenamiento para incrementar la calidad del producto [32].
Como todo proceso, la Gestión de Configuración también puede ser sistematizada y automatizada, lo que se denomina un Sistema de Gestión de Configuración (SGC). Actualmente existen en el mercado diversas herramientas que permiten apoyar una o más actividades de la Gestión de Configuración. Luego, si esta se centra en el proceso de identificar y definir los elementos en el sistema, controlando el cambio de estos a lo largo de su ciclo de vida, registrando y reportando el estado de los elementos y las solicitudes de cambio, y verificando que los elementos estén completos y que sean los correctos, las tareas con mayor responsabilidad serian el control de versiones y la gestión de cambios. El siguiente capítulo describe un análisis de las herramientas mas utilizadas en la actualidad para el control de versiones y la gestión de cambios y así fundamentar la selección de este proyecto.

3.1 Comparación de Sistemas de Control de Versiones.

3.1.1 Licencia.

Primeramente se tendrá en cuenta el tipo de software que se trata (propietario o libre), ya que resulta de gran interés e importancia sobre todo para la sociedad cubana y los demás países del tercer mundo, el uso gratuito de un software y que sea de código abierto. Este mundo actual matizado por las tecnologías, dominadas por el mercado capitalista, hace más difícil la integración de los menos favorecidos con el mismo. Por tanto incentivar el uso de este tipo de herramientas a parte de crear

confianza en el movimiento de Software Libre, brinda nuevas oportunidades de desarrollo con las tecnologías.

CVS	GNU GPL (código abierto).
AccuRev	Propietario, licenciado nombrado-usuario.
Aegis	GNU GPL (código abierto).
Arch	GNU GPL (código abierto).
BitKeeper	Propietario, solamente licencia binaria. Pagar por licencia del uso, con una opción una licencia sin gastos reveladores del software de la abrir-fuente. Tenían una licencia gratuita, downloadable, que fue pensada para el desarrollo del software abierto de la fuente. Tenía una licencia problemática.
ClearCase	Propietario, soportado por la licencia flotante. Los precios son varios $k por licencia más honorario anual del mantenimiento.
CMSynergy	Tasa negociable con el vendedor. El servidor es típicamente áspero 20.000 libras británicas. Los clientes son 4.000 libras británicas. Costes por año, el 18%

	de la original.
Co-Op	Propietario, llave corta del texto. Ensayo completamente equipado de 30 días. Liberar a "observadores" (los miembros que no realizan cambios). $159 por el sitio de trabajo.
Darcs	GNU GPL (código abierto)
Mercurial	GNU GPL (código abierto)
Monotone	GNU GPL (código abierto)
OpenCM	GNU GPL (código abierto), pero moviéndose pronto al DEB o COMPLETO (también fuente abierta).
Perforce	A propietario, binario solamente, licencia comercial. Tasar empezar $800 por el asiento por el primer año y entonces $160 para la ayuda de continuación por los años subsecuentes. El último pago es opcional y requerido solamente para la ayuda, pues el producto se puede utilizar sin él. Liberar para los proyectos abiertos de la fuente (ninguna ayuda en este caso).
PureCM	Propietario, binario solamente, licencia comercial. El empezar

	$600 del precio para 5 usuarios.
Subversion	Licencia Apache/BSD-style. (código abierto)
Superversion	GNU GPL (código abierto)
svk	Licencia Perl (código abierto)
Vesta	GNU LGPL (código abierto)
Visual SourceSafe	El VSS envía con MSDN, y puede también ser comprado independiente o con otras herramientas.

Tabla 3.1 Licencias.

El software libre se ha transformado en una herramienta de vital importancia en la gestión de las empresas de alta tecnología, sobre todo en las más pequeñas, permitiendo una mejora de los procesos productivos y una reducción de costos considerable. Además de esto, la posibilidad de extender el software para uso dentro de la organización es algo que no se puede implementar con el modelo clásico de software propietario, sin considerar los beneficios que esto trae a toda la comunidad opensource [33].

3.1.2 Operaciones del repositorio.

A partir de aquí solo se analizaran las herramientas que a consideración del autor resultan de interés y beneficio para el proyecto (Software Libre).

Atomic Commits.

El soporte para atomic commits significa que si una operación en el repositorio se interrumpe en la mitad, el repositorio no será dejado en un estado inconsistente.

CVS	No. Los CVS commits no son atómicas.
Aegis	Los Commits son atómicos.
Arch	Si. Los Commits son atómicos
Darcs	Si. Los Commits son atómicos
Mercurial	Si.
Monotone	Si.
OpenCM	Si. Los Commits son atómicos
Subversion	Los Commits son atómicos
Superversion	Los Commits son atómicos
svk	Los Commits son atómicos
Vesta	Si. Los Commits son atómicos

Tabla 3.2 Atomic Commits.

Archivos o directorios se mueven o renombran.

El sistema permite el movimiento de un archivo o un directorio a una localización diferente mientras que conserva el historial del archivo.

CVS	No. No permite renombrar y un renombramiento manual puede dividir el historial en 2.
Aegis	Si. Soporta renombramiento.
Arch	Si. Soporta renombramiento.
Darcs	Si. Soporta renombramiento.

Mercurial	Si. Soporta renombramiento.
Monotone	Si. Soporta renombramiento.
OpenCM	Si. Soporta renombramiento.
Subversion	Si. Soporta renombramiento.
Superversion	No Soporta renombramiento.
svk	Si. Soporta renombramiento.
Vesta	Si. La unidad de checkout/checkin es un árbol del directorio. Los archivos y los directorios se pueden agregar, suprimir, y renombrar entre las versiones.

Tabla 3.3 Archivos y directorios.

Copias de Archivos y Directorios.

El sistema de control de versiones permite la copia de archivos o directorios a una localización diferente en el nivel del repositorio, mientras que conserva el historial.

CVS	No soporta copias.
Aegis	No soporta copias.
Arch	No soporta la estructura de copia de archivos y directorios.
Darcs	No soporta la estructura de copia de archivos y directorios.
Mercurial	Si soporta copias.
Monotone	Si. Soporta copias.

OpenCM	No soporta copias.
Subversion	Sí. Y es una operación muy barata (O (1)) eso también se utiliza para ramificar.
Superversion	No soporta copias.
svk	Si, igual a Subversion.
Vesta	Sí. Un nuevos paquete/rama se puede basar en cualquier versión existente sin afectar la última historia. (Esto es también una operación de O (1)).

Tabla 3.4 Copias.

Réplica del Repositorio Remoto.

El software permite reproducir un repositorio remoto para conseguir una copia funcionalmente equivalente en el sistema local. Eso se debe hacer sin ningún acceso especial al servidor remoto a excepción del acceso normal al repositorio.

CVS	Indirectamente, usando CVSup por John Polstra (que requiere el funcionamiento del daemon del cvsupd en el servidor)
Aegis	Si
Arch	Si.
Darcs	Si.
Mercurial	Si.

Monotone	Si.
OpenCM	No.
Subversion	Indirectamente, usando Chia-liang Kao's SVN::Mirror add-on o Shlomi Fish' SVN-Pusher utility.
Superversion	Si.
svk	Si.
Vesta	Si. La replica es una parte fundamental del diseño.

Tabla 3.5 Replica remota.

Propagar cambios a los Repositorios Padres

Puede el sistema propagar cambios de un repositorio a otro.

CVS	No.
Aegis	Si.
Arch	Si.
Darcs	Si.
Mercurial	Si.
Monotone	Si.
OpenCM	No.
Subversion	Sí, usando cualquier SVN de Chia-Ling Kao.
Superversion	No.

svk	Si.
Vesta	Si.

Tabla 3.6 Cambios a padres.

Permisos del Repositorio.

Es posible definir permisos para el acceso a diferentes partes de un repositorio remoto o es accesible abiertamente para todos.

CVS	Limitado. “pre-confiar las escrituras del gancho” puede ser utilizado poniendo varios sistemas de los permisos en ejecución.
Aegis	Sí. La tutela confía en el sistema de los permisos del UNIX para poner los permisos en ejecución para los archivos en el repositorio.
Arch	Sí. Es posible definir permisos en el acceso a diversas piezas de un repositorio alejado basado en los sistemas del permiso del protocolo subyacente.
Darcs	No.
Mercurial	Sí. Es posible trabarse abajo de los repositorios, de los subdirectorios, o de los archivos usando los ganchos.
Monotone	Sí. Es posible restringir cambios entrantes de ciertas fuentes que se realizarán solamente en ciertas

	piezas del repositorio.
OpenCM	Los permisos se definen sobre una base del por-rama.
Subversion	Sí. Las ayudas de servicio WebDAV-basadas que definen los permisos del HTTP para los varios directorios del depósito.
Superversion	No.
svk	Iguales que Subversion
Vesta	Sí. Los permisos de acceso para cada paquete (la unidad de la comprobación/del registro) pueden ser diferentes. Los permisos de acceso para una rama pueden ser diferentes del paquete de la base.

Tabla 3.7 Permisos.

Soportes de cambios.

Puede el repositorio permitir soportes de cambios. Estos son una manera de agrupar un número de modificaciones que sean relevantes para cada uno en un paquete atómico, que puede ser cancelado o propagado según se necesite.

CVS	No. Los cambios son archivo-específicos.
Aegis	Si soporta sistemas de cambios.
Arch	Si soporta sistemas de cambios.

Darcs	Si soporta sistemas de cambios.
Mercurial	Si soporta sistemas de cambios.
Monotone	Si soporta sistemas de cambios.
OpenCM	Si soporta sistemas de cambios.
Subversion	Soporte parcial. Hay changeset implícito que se genera en cada uno confía.
Superversion	Soporte parcial. Los cambios se agrupan en changesets, pero no se pueden cancelar individualmente todavía.
svk	Igual a Subversion.
Vesta	No exactamente. Vesta utiliza un concepto relacionado de las configuraciones en lugar de otro, que algo tiene características similares.

Tabla 3.8 Soporte de cambios.

Siguiendo el historial de la línea-sabia del archivo.

Si el sistema de control de versiones tiene una opción para seguir el historial del archivo línea por línea. Es decir: ¿para cada línea muestra en qué revisión fue cambiada más eficientemente y por quién?

CVS	Si.
Aegis	Si.
Arch	No en la línea de comando

	cliente, pero ViewARCH, una interfaz web para el arco, lo tiene.
Darcs	Si.
Mercurial	Si
Monotone	Si.
OpenCM	Se desconoce, probablemente no.
Subversion	Si.
Superversion	No.
svk	Si
Vesta	No, pero sería fácil poner una herramienta en ejecución que hizo esto, como el depósito de Vesta proporciona el acceso directo del filesystem a todas las versiones.

Tabla 3.9 Historial.

3.1.3 Mejoras.

Capacidad de trabajar solo en un directorio del Repositorio.

Puede el sistema de control de versiones comprobar solamente un directorio del repositorio. O restringir los registros a un directorio solamente.

CVS	Si.
Aegis	No. Todos los cambios se realizan en el repositorio.
Arch	Es posible enviar solamente a cierto directorio. Sin embargo, se

	debe comprobar si se puede adquirir del repositorio entero en su totalidad.
Darcs	Es posible confiar solamente cierto directorio. Sin embargo, uno debe comprobar hacia fuera el depósito entero en su totalidad.
Mercurial	Es posible confiar cambios solamente en un subconjunto del árbol. Hay planes para las comprobaciones parciales.
Monotone	Es posible confiar cambios solamente en un subconjunto del árbol. Sin embargo, uno debe extraer el árbol entero para trabajar en él.
OpenCM	No. Todos los cambios se realizan a un proyecto como unidad.
Subversion	Si.
Superversion	No.
svk	Si.
Vesta	Sí y no. La unidad de la comprobación/del registro (llamada paquete) es un árbol del directorio. La mayoría de los proyectos utilizan más de uno. Una vez que esté creado, un paquete deba ser out/in

	comprobado como unidad.

Tabla 3.10 Capacidad de trabajo.

Seguimiento de cambios Uncommited.

Si el software tiene la habilidad de seguir los cambios en la copia de trabajo que no ha sido confiada al repositorio.

CVS	Si. Usando el diff de los cvs.
Aegis	Si. Usando aediff.
Arch	Si, usando"tla changes".
Darcs	Si, usando "darcs whatsnew".
Mercurial	Si, usando hg diff.
Monotone	Si. En una manera similar a CVS.
OpenCM	Si, usando cm diff.
Subversion	Si, usando svn diff.
Superversion	Sí. Los cambios locales se detectan y se demuestran inmediatamente. Los cambios se pueden recoger en un buffer local intermediario antes de ser confiado al repositorio.
svk	Si. Usando svk diff
Vesta	Sí. Las fotos inmutables intermedias se pueden tomar durante una comprobación activa (con advance). Estas versiones intermedias pueden ser tratadas

	justas como versiones llegadas: pueden ser replegadas a otros repositorios y ser utilizadas como la base para las ramas.

Tabla 3.11 Seguimiento de cambios.

3.1.4 Estado técnico.

Documentación.

Qué tan Buena es la documentación del sistema. Cuan fácil es comenzar a usarla.

CVS	Excelente. Hay muchas clases particulares y recursos online y un libro también. La línea de comando cliente también proporciona un sistema de ayuda comprensivo online.
Aegis	Medio. La documentación se da en varios documentos grandes del troff, ésa es solamente usable como pdf del not-so-PDFish. Es muy duro conseguirla comenzado a usarla con los recursos online. El contenido está es de buena calidad, pero de otra manera no es muy accesible.
Arch	Medio. Hay dos clases particulares en línea y una documentación comprensiva. La

	línea de comando cliente también provee una página de la referencia. Sin embargo, parte de la documentación es anticuada o incompleta.
Darcs	Bueno. El manual contiene una breve referencia preceptoral y sólida. Cada submandato puede imprimir su uso. Porque fijar comando es pequeño y el modelo es simple, muchos usuarios encuentran fácil comenzar.
Mercurial	Muy bueno. Hay una descripción y una clase particular en el Web site, y una ayuda integrada para cada comando.
Monotone	Bueno. Hay una descripción y una clase particular escrita en Texi. El cliente provee la documentación para cada comando.
OpenCM	Bien documentado.
Subversion	Muy bueno. Hay un libro en línea libre y algunas clases particulares y recursos online. El libro se escribe en DocBook/XML y así que es convertible a diversos formatos. La línea comando cliente también proporciona un buen sistema de ayuda en línea que se pueda utilizar como

	referencia.
Superversion	Bastante pobre. Hay dos clases particulares, pero no hay referencia. Sin embargo instalarlo y comenzar con el GUI resulta muy fácil.
svk	Relativamente pobres, pero puede mejorar. Hay trabajos en libros así como el Wiki y algunos artículos y clases particulares externos.
Vesta	Absolutamente a fondo (HTML, páginas del hombre, papeles publicados).

Tabla 3.12 Documentación.

Facilidad de despliegue.

Cuan fácil es desplegar el sistema. ¿Qué son las dependencias y como pueden ser eliminadas?

CVS	Bueno. Fuera de ser el estándar de hecho, CVS está disponible en la mayoría de los sistemas y es fácil de desplegar.
Aegis	La tutela binaria se debe instalar como SUID-raíz, y así que requiere privilegios de la raíz a la instalación. Él también no es muy portable a los sistemas Win32. Con excepción de ese, la tutela apoya un proceso fácil del autoconf o de la instalación de

	RPM/apt-based.
Arch	Excelente. Un servicio del arco no es nada sino un filesystem-espacio recibido por cualquiera de sus protocolos apoyados (ftp, SFTP, WebDAV, etc.). Escrito en C, y es portable el cliente del arco a través de sistemas del UNIX (y en Win32 solamente con una capa de la emulación del UNIX).
Darcs	Muy bueno, los darcs requieren pocas bibliotecas externas, no obstante es necesario el recopilador de Glasgow Haskell si no se puede encontrar un binario. Para comenzar a trabajar, apenas “init de los darcs”.
Mercurial	Excelente. Los paquetes binarios están disponibles para todas las plataformas populares. El edificio de la fuente requiere solamente el Python 2.3 (o más adelante) y el recopilador de la CA.
Monotone	Excelente. Es posible copiar o compilar el ejecutable a la máquina del usuario, sin ninguna configuración o dependencias externas.
OpenCM	Muy bueno. Instalar la RPM, o construirla de tarball e instalar la

	escritura del init.
Subversion	Los servicios de subversión requieren la instalación de un módulo de Apache 2 (si se desea utilizar el HTTP como el protocolo subyacente) o su propio servidor propietario. El cliente requiere solamente la lógica Subversión-específica y la biblioteca de neón de WebDAV (para el HTTP). La instalación de los componentes es absolutamente directa, pero requerirá un cierto trabajo, subversión no viene preembalado para su sistema.
Superversion	Si Java 1.4 está instalado, el despliegue de Superversion toma generalmente dos tecleos.
svk	Además de instalar subversión, los usuarios requieren instalar los atascamientos del Perl de subversión y algunos módulos de CPAN.
Vesta	Más o menos. Hay una guía detallada de la instalación para fijarla para arriba que usa un kit binario. Los paquetes de RPMs y de Debian se han lanzado recientemente. No hay dependencias en el otro software.

Tabla 3.13 Facilidad de Despliegue.

Paquete de comandos.

Que es un paquete de comandos. Cuán compatible es con los comandos del CVS.

CVS	Un sistema de comando simple que incluye tres comandos más de uso general (los cvs confía, actualización de los cvs y comprobación de los cvs) y otros.
Aegis	Un sistema de comando complejo que implica muchas operaciones apenas para conseguir un comienzo. No es compatible con CVS (no obstante la ayuda para tales operaciones básicas es contemplada) observar que la tutela es un sistema de gerencia de la configuración del software y no apenas un sistema de control simple de la versión, que puede justificar esta complejidad adicional.
Arch	Muchos comandos son compatibles con CVS o BitKeeper. Sin embargo, hay muchos otros comandos para él hacia diversas aplicaciones.
Darcs	El sistema de comando es bastante compacto y los comandos de la base son fáciles

	de entender. Sigue a CVS en algunos lugares, pero puesto que el modelo es diferente la mayoría de los comandos son únicos.
Mercurial	Intenta seguir las convenciones de CVS, pero se desvía donde hay un diverso diseño.
Monotone	Intenta seguir las convenciones de CVS, pero se desvía donde hay un diverso diseño.
OpenCM	A CVS-como el sistema de comandos que es familiar a los usuarios existentes de CVS.
Subversion	A CVS como el sistema de comando que es fácil de conseguir.
Superversion	Hay poca necesidad de memorizar un sistema de comandos porque todas las acciones ocurren en un GUI. Una parte de la terminología utilizada se pide prestada de CVS.
svk	A CVS-como el sistema de comando que es fácil de conseguir.
Vesta	El sistema de comandos no tiene relación con CVS. La mayor parte del tiempo, los usuarios utilizan cerca de 5 comandos. Pocos

	necesitan saber más de 20 comandos.

Tabla 3.14 Paquete de comandos.

Soporte de red.

Cuan buena es la integración de la red en el sistema. Compatibilidad con infraestructuras y protocolos existentes.

CVS	Bueno. CVS utiliza un protocolo propietario con las varias variaciones para su protocolo del cliente/ servidor. Este protocolo puede ser tunneled sobre una conexión SSH al cifrado de la ayuda.
Aegis	Pobre.
Arch	Excelente. El arco puede utilizar una multiplicidad de los protocolos para su servicio, que no es nada solamente un servidor alejado mudo del filesystem. Los protocolos actualmente apoyados incluyen ftp, SFTP, WebDAV (HTTP excesivo del acceso alejado del archivo), tan bien como cualquier protocolo alejado del filesystem (NFS, SMB).
Darcs	Bueno.
Mercurial	Excelente. HTTP de las

	aplicaciones o ssh.
Monotone	Bueno. Utiliza un protocolo de encargo llamado “netsync”.
OpenCM	Bueno. Utiliza su propio protocolo propietario del cliente/ servidor.
Subversion	Muy bueno. El servicio de subversión puede utilizar WebDAV+DeltaV (que sea HTTP o HTTPS basado) como su protocolo underylying, o su propio protocolo propietario que se pueda acanalar sobre una conexión de SSH.
Superversion	Bueno. La ayuda de la red basada en el RMI se integra seamlessly. El cifrado y el hacer un túnel del HTTP se planean para el futuro cercano.
svk	Muy bueno. El svk utiliza SVN
Vesta	El establecimiento de una red es inherente al sistema. El depósito exporta una interfaz del NFS y otra del RPC.

Tabla 3.15 Soporte de red.

Portabilidad.

Cuán portable es el sistema de control de versiones para diferentes sistemas operativos, arquitecturas de sistemas y otros tipos.

CVS	Bueno. El cliente trabaja en UNIX, el OS de Windows y del Mac. El servidor funciona en UNIXes y en Windows con una capa de la emulación del UNIX.
Aegis	Medio. La fuente es portable a través de todo el UNIXes, pero del trabajo de la versión de Windows que usa, solamente el cygwin.
Arch	Bueno. La fuente es portable a través de todo el UNIXes, pero requiere una capa de la emulación del UNIX en Windows. Necesidad de verificar.
Darcs	Muy bueno. Soporta mucho OS X de UNIXes, del Mac, y Windows, y se escribe en una lengua portable.
Mercurial	Excelente. Funcionamientos en todas las plataformas apoyadas por Python. Los repositorios son portables a través de arquitecturas de la CPU y de convenciones endian.
Monotone	Excelente. Ejecutable es portable a través de todo el UNIXes y Win32.
OpenCM	Bueno. Portable a través de todos los sistemas del UNIX.

Subversion	Excelente. Los clientes y los servidores trabajan en el UNIX, OS X. de Windows y de Mac.
Superversion	Excelente. Los clientes y los servidores trabajan en cualquier plataforma compatible de Java 1.4. Hay ayuda oficial para Windows, Linux y OS/2.
svk	Bueno. Los clientes requieren subversión y sus atascamientos del Perl.
Vesta	Bueno. Debe ser portable a cualquier sistema del UNIX. Funciona actualmente en el UNIX y Linux de Digital/Compaq/HP Tru64 en diversas arquitecturas de la CPU. Los puertos a Solaris y a FreeBSD se planean pero no han comenzado todavía.

Tabla 3.16 Portabilidad.

3.1.5 Interfaces de usuarios.

Interfaz web.

Si el sistema tiene una interfaz basada en WWW que se pueda utilizar para hojear el árbol y las diferentes revisiones de los archivos, etc.

CVS	Si. CVSweb, ViewVC, Chora, y wwCVS.
Aegis	Si.

Arch	Hay ViewARCH, y ArchZoom que son trabajos en marcha.
Darcs	darcs.cgi se incluye en la distribución.
Mercurial	Si, la interfaz web es un componente liado.
Monotone	No.
OpenCM	No.
Subversion	Si. ViewVC, SVN_Web, ViewSVN, mod_svn_view, Chora, Trac, SVN_RaWeb_Light, SVN Browser, Insurrección y perl_svn. Aparte de estos, el servicio de Apache de subversión proporciona una interfaz web rudimentaria.
Superversion	No.
svk	Si. Iguales que subversión.
Vesta	Si. Vestaweb.

Tabla 3.17 Interfaz web.

Disponibilidad de Interfaces Gráficas de usuarios

Cual es la disponibilidad de las interfaces gráficas de usuario para el sistema. Cuántos clientes GUI están presentes para esto.

CVS	Muy bueno. Hay mucho GUIs disponible: WinCVS, Cervisia (para KDE), TortoiseCVS (plug-in

	del explorador de Windows).
Aegis	Hay tkaegis.
Arch	Hay tlator, Octopy, y ArchWay y posiblemente otros en el desarrollo.
Darcs	No se habla de ninguno.Hay un interfaz gráfico modesto a algunos comandos en la distribución, pero no se está desarrollando actualmente.
Mercurial	La extensión de los registros (hgct) hace los committing más fácil. Algunas herramientas de tercera persona de IDEs y del GUI (e.g. eric3, meld) han integrado la ayuda de Mercurial .
Monotone	No hay GUIs disponible.
OpenCM	No hay GUIs disponible.
Subversion	Muy bueno. Hay mucho GUIs disponible: RapidSVN (cruz-plataforma), TortoiseSVN (plug-in del explorador de Windows), Jsvn (Java), etc. La mayor parte todavía están en el desarrollo.

Superversion	Está integrado.
svk	No hay GUIs disponible.
Vesta	No hay GUIs disponible, pero el depósito tiene A.c. ++ API, y no es duro escribir uno. (Por lo menos tres diversos unos proyecto-específicos han sido escritos por los usuarios en Compaq y Intel).

Tabla 3.18 Disponibilidad de Interfaces.

Después de este análisis se puede concluir que Subversion resulta una de las mejores opciones para el control de cambios en un proyecto. Siendo así solo quedaría seleccionar de las interfaces gráficas que brinda este software la que se va a utilizar.

3.2 Subversion y Trac.

Subversión.

Su desarrollo comenzó en el año 2000 como proyecto de código abierto esponsorizado por CollabNet. El líder del equipo de desarrollo fue Karl Fogel, autor de Open Source Development with CVS (Concurrent Versions System), y fundador de Cyclic Software (compañía de desarrollo y soporte comercial para CVS, hoy adquirida por SourceGear). La versión 1.0 fue publicada en febrero del 2004.

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual tuvo el merito de ser el primer sistema usado por el movimiento de código abierto para que los programadores colaboraran remotamente mediante el envío de parches, ya que posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD. Permite hacer lo que quieras con el código fuente (incluso forks y productos propietarios sin entregar el código) la única restricción es que les reconozcas su trabajo. Se lo conoce también como **svn** por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión

independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Clientes:

Existen varias interfaces a Subversion, incluyendo programas individuales como interfaces que lo integran en entornos de desarrollo.

- TortoiseSVN. Provee integración con el explorador de Windows. Es la interfaz más popular en este sistema operativo.
- Subclipse. "Plugin" que integra Subversion al entorno Eclipse.
- Subversive. "Plugin" alternativo para Eclipse.
- ViewVC. Interfaz web, que también trabaja delante de CVS.
- Trac: Herramienta open source de interfaz grafica, impresionante para la gestión de proyectos.

Para mac, pueden emplearse los interfaces SvnX, RapidSVN y Zigversion.

Características:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- El creado de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante (O (1)) y no lineal (O (n)) como en CVS.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido, mediante Apache, sobre WebDAV/ DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.).

También seria bueno conocer sus falencias, por tanto:

- El manejo de cambio de nombres de archivos no es completo. Lo maneja como la suma de una operación de copia y una de borrado.

- No resuelve el problema de aplicar repetidamente parches entre ramas, no facilita el llevar la cuenta de qué cambios se han trasladado. Esto se resuelve siendo cuidadoso con los mensajes de commit.

Trac.

Trac es una herramienta open source de uso libre y de interfaz web, simple y minimalista que integra herramientas para comunicación, gestión, seguimiento de proyecto; y gestión de la configuración (control de versiones). Trac:

Pre-requisitos para trabajar con Trac:

- Python 2.3 ó superior.
- Subversion 1.0 so superior (se recomienda 1.1).
- Plantillas Clearsilver 0.93 o superior.
- Base de datos SQLite o Postgre SQL.

Ventajas:

- Brinda una vista del código svn y maneja las diferencias de manera impresionante.
- Puede verse el histórico del proyecto (timeline).
- Gestión de tickets (algo parecido a un bugzilla).
- Previsiones de características del proyecto (milesones).
- Un wiki supercompleto integrado en el resto del sistema trac.
- El buscador, busca tanto por logs de cambios svn, como entre las páginas del wiki o los tickets.
- Es auto documentado, cualquier cosa que se quiera hacer/configurar, el buscador te dirá como hacerlo.

3.3 Conclusiones.

Con lo expuesto en este capítulo se fundamenta el por qué de la utilización de Trac y Subversion, en el proyecto Intranet, como propuesta de herramientas para el control de versiones y la gestión de cambios.

Capítulo 4

Propuesta de Gestión de la Configuración del Software para la Intranet II.

Solucionar el gran problema de tratar de administrar los cambios en sistemas informáticos, resulta para sus profesionales una tarea difícil. Con el presente capítulo se pretende aportar la solución a este. Para ello primeramente se refieren los problemas más comunes en un desarrollo del software que conllevan a gestionar la configuración del mismo, a continuación se explicarán los elementos para la solución de la gestión de configuraciones según la experiencia de S. Dart y N. Bounds y se mencionarán las tres claves para el éxito de la gestión. Una de estas claves es el plan, para el cual existen estándares, y antes de presentar el plan de gestión de la configuración del proyecto intranet, objetivo de este trabajo, se analizan tres estándares para planes de gestión y se decide cual aplicar en este caso.

4.1 Problemas típicos en un desarrollo de software.

Existen tres problemas típicos que sirven para darse cuenta de la necesidad de la gestión de configuración.

- El problema del doble mantenimiento.
- El problema de los datos compartidos.
- Los problemas de actualización simultánea.

4.1.1 El problema del doble mantenimiento.

Cuando hay dos copias del mismo software ambas necesitan ser mantenidas. Por lo tanto cuando se encuentra y soluciona un error, alguien debe recordar que debe solucionarse el mismo error en la otra copia. El primer principio de la gestión de configuraciones es evitar las múltiples copias de una misma información. En el verdadero mundo del desarrollo, ni el mayor de los esfuerzos puede asegurar que dos copias de una información idéntica lo sigan siendo para siempre.

4.1.2 El problema de los datos compartidos.

Este problema surge cuando más de una persona simultáneamente accede y modifica los mismos datos. Estos datos pueden ser el código del sistema. Es decir cuando los programadores están modificando una copia del programa, los cambios hechos por un programador pueden interferir en los cambios de otro.

Cuando se realiza una modificación, esta puede cambiar el comportamiento del programa de tal forma que todos los miembros del grupo necesiten entenderla. De cualquier forma, cualquiera sea la razón, siempre se van a encontrar problemas cuando los programadores trabajen juntos en una parte del código del programa.

Muchas veces los cambios introducidos por un programador serán una sorpresa inesperada para otro de los programadores.

4.1.3 Los problemas de actualización simultánea.

Según el problema mencionado anteriormente no se puede trabajar sobre una misma copia del programa, y por el primero, doble mantenimiento, no conviene tener varias copias del mismo código, una solución simple es dividir el código en un número de archivos llamados módulos. Cuando un programador quiere hacer una modificación de un módulo, generará una copia del mismo, de esta forma no interfiere en el trabajo de los demás a medida que realiza sus modificaciones, y una vez testada la cargará nuevamente en el espacio común. El problema aquí surge cuando dos personas están actualizando el mismo módulo, ya que los cambios de uno de ellos pueden ser sobrescritos por los cambios del otro.

4.2 Elementos de la solución para la gestión de configuraciones.

Solucionar los problemas planteados en la sección anterior es el objetivo de la gestión de configuraciones.

Basado en la experiencia de Nadine M.Bounds y Sunsan Dart existen diez elementos que son las claves para resolver las necesidades de la gestión de configuraciones dentro de la organización. Estas son: planeamiento, procesos, recursos humanos, cultura, productos, automatización, gestión, el plan de gestión de configuraciones, el sistema de gestión de configuraciones y la adopción de una estrategia para la gestión de configuraciones. Los primeros siete elementos se relacionan con el problema en sí y la búsqueda de una solución. Los últimos tres son el resultado de los siete primeros. Esto significa que el plan, el sistema y la adopción de una estrategia para la gestión de configuraciones son el foco de la solución de la gestión de configuración.

Figura 4.1 Elementos de la solución de la gestión de configuración.

- Planeamiento: definición de todas las características que deben ser documentadas en el plan de gestión de configuración, todos los procedimientos y las políticas.
- Procesos: descripción del proceso actual y el nivel de control que se alcanzará cuando el nuevo proceso sea implementado.
- Recursos humanos: está relacionado a todos los roles, responsabilidades y tareas que la gente desempeña durante la implementación del proceso de gestión de configuración.
- Cultura: esto es entender el tipo de cultura que existe dentro de la organización y encontrar una solución para la gestión de configuración que sea compatible con la cultura de la empresa.
- Productos: Involucra determinar qué productos y partes de los mismos estarán bajo el control de la gestión de configuración.
- Automatización: buscar los requerimientos de la funcionalidad de un sistema de gestión de configuraciones automatizado.
- Gestión: resolución gerencial de las decisiones asociadas con la solución para la gestión de configuraciones.
- Plan de gestión de configuración: es el plan que será implementado para lograr las necesidades de la gestión de configuración. Engloba todos los procedimientos, políticas, cronogramas, responsabilidades, etc.
- Sistema de gestión de configuración: es la(s) herramienta(s) elegida(s) para el proceso de gestión de configuración.
- Adopción de la estrategia para la gestión de configuración: es la estrategia utilizada para asistir a la organización en la adopción del proceso y del sistema.

4.3 Estándares.

El gestor para elaborar el plan debe guiarse por estándares, ya sean externos (ej. IEEE) o existentes de la propia empresa. Ya que un estándar:

- Reúne las mejores prácticas. Evita la repetición de errores pasados.
- Proporciona un marco para el análisis de calidad -involucra verificar la conformidad con estándares.
- Proporciona continuidad. El personal nuevo puede entender a la organización entendiendo a los estándares aplicados.

4.3.1 Los estándares deben definir:

- Cómo identificar objetos.
- Cómo son controlados los cambios.
- De qué forma se gestionan las versiones nuevas.

4.3.2 Comparación de tres planes estándares.

Susan Dart y Nadine Bounds, en el artículo Configuration Management Plans: The Beginning to your CM Solution realizan una evaluación y comparación entre tres planes estándar. Estos son:

- IEEE Standard for Software Configuration Management Plans (IEEE Std 828-1990).
- NASA Software Configuration Management Plan Data Item Description (NASASfw-DID-04).
- DoD Software Development Plan Data Item Description (DID) asociado con DoD-STD-2167A (Di-MCCS-90030A).

Los estándares se compararon según seis criterios:

a. Facilidad de uso
b. Completo
c. Flexibilidad
d. Consistencia
e. Corrección
f. Conexión con el ciclo de vida

a. Facilidad de uso.

Por facilidad de uso se entiende el esfuerzo requerido para entender y aprender a usar el estándar y para utilizarlo en sus sucesivos proyectos. Un estándar que sea fácil de utilizar debe ser sencillo tanto para el usuario inexperto que por primera vez va a escribir el plan y no se encuentra familiarizado con los conceptos de gestión de configuración, y debe ser también fácil de utilizar para el usuario experto que sólo lo use como una guía para su plan. Este criterio también implica que el estándar esté bien organizado y escrito, y que la información específica pueda ser localizada fácil y rápidamente.
Los atributos mínimos que debe cumplir un estándar para ser considerado fácil de usar son:

- El usuario debería obtener un entendimiento progresivo de cómo escribir el plan.
- El estándar debería está formateado y estructurado para que la información sea fácilmente accesible.
- Las palabras claves deberían estar escritas de forma distintiva y las secciones y párrafos deberían estar escritos alrededor de ellas.
- El usuario no debería tener que leer muchas veces un párrafo para lograr entenderlo.

Los atributos adicionales que debe cumplir un estándar para ser considerado bueno son:

- El estándar es fácil de leer, es claro y conciso y está bien escrito gramaticalmente.
- El estándar asume que el usuario no tiene ningún conocimiento previo o bien el mismo es muy escaso en el desarrollo de planes.
- Las definiciones de los términos utilizados se encuentran agrupadas y ubicadas en un lugar lógico.
- El material está muy bien organizado, de forma tal que el usuario puede localizar muy fácilmente la información. Le lleva como máximo tres minutos.
- Toda la información relacionada a un tópico se encuentra en un solo lugar.
- Los temas tratados son claros para el usuario.
- Las secciones del estándar son cortas.
- El estándar satisface los requerimientos del usuario.

Hay un atributo adicional que debería tener un estándar para ser excelente y es que al usuario le tome no más de un minuto localizar la información que necesita.
Los tres estándares están bien formateados y estructurados. Cada sección está muy bien encabezada. Los tres son fáciles de leer, claros, concisos y bien organizados, mantienen toda la información relativa a un tópico en un solo lugar para que sea más fácil para el usuario.
La mayor desventaja encontrada en este tópico fue que tanto el estándar de la NASA como el de DoD fueron escritos asumiendo que no eran los únicos documentos a utilizar. El estándar de la IEEE por el contrario está pensando para ser el único documento. Esto se ve también en el tamaño de los documentos ya que el de la IEEE tiene 15 páginas mientras que el de la NASA tiene 9 y el Dod sólo 4. El estándar de la IEEE es extremadamente fácil de usar. No sólo alcanza el nivel para ser excelente, sino que en algunas áreas aporta aún valor adicional. El estándar hace que para el usuario sea fácil de

distinguir entre aquellos ítems que son obligatorios que estén en el plan y los que son opcionales, esto lo hace utilizando las palabras “shall” y “required”. Otra ventaja adicional es que para los ítems “required” trae ejemplos. Finalmente el estándar mapea unas actividades con otras. A pesar de estas ventajas, también tiene algunas desventajas como en la definición de algunos términos hace referencia a otros estándares, en lugar de definirlos en el mismo documento. También el estándar podría haber provisto una guía rápida para el usuario experto para no tener que recorrer por todo el documento sin necesidad. Sobre el estándar de la NASA se puede decir que es fácil de usar y medianamente completo. Este estándar se podría haber evaluado como bueno sino fuera por dos omisiones: la primera es que no se puede utilizar como único documento, y por lo tanto falla en el aspecto de satisfacer todas las necesidades del usuario. Si el usuario es experto el uso de este estándar es una muy buena referencia pero si el usuario es nuevo será muy complejo y probablemente imposible. La segunda omisión es que no hay una lista de definición de términos. Sus ventajas son que está bien organizado, con toda la información relativa a un tópico en el mismo lugar; que es fácil de leer y relativamente corto y está formateado de tal forma que resulta muy sencillo encontrar los componentes claves. En cuanto al estándar DoD, fue valuado como medio, ya que es medianamente fácil de usar pero no es muy completo. Sólo provee información de muy alto nivel sobre los componentes claves del plan. Este estándar es una excelente referencia para los usuarios expertos. Tampoco es útil al igual que el de la NASA como único documento. Otra desventaja es que no contiene ninguna tabla de contenidos o índice. El resultado es que el estándar no está bien organizado, no tiene una definición de los términos, consecuentemente un usuario no informado deberá buscar otros documentos para poder completar su plan. Finalmente no es fácil de usar debido a que no tiene toda la información de un mismo tópico en un mismo lugar. Como aspectos positivos de este estándar se puede decir que está bien estructurado y las secciones son cortas, lo que hace que no sea confuso.

b. Completo.

Un estándar es completo si el usuario no necesita buscar información en otros documentos sobre el desarrollo del plan de gestión de configuraciones.

Los atributos mínimos que debe cumplir el estándar para considerarlo completo son los siguientes:

- Proveer una descripción de los siguientes componentes del plan:
  - Introducción al documento y propósito.
  - Organización y responsabilidades de la gestión de configuraciones.
  - Identificación de la configuración.
- Identificación y descripción de las líneas bases.

- Procedimientos para realizar cambios cuando ya se ha establecido la línea base.
- Debe ser claro, conciso, no ambiguo y simple.

Para que el estándar sea considerado bueno en cuanto a este ítem, debe cumplir los siguientes atributos:

- Descripción del esquema de identificación.
- Como se llega a las versiones y release.
- Descripción de librerías.
- Mecanismos de backup y recuperación – Políticas.
- Revisiones de documentos.
- El estándar no debe contener información que no sea referente a la creación del plan de gestión de configuraciones.
- Toda la información que figure debe ser necesaria y suficiente.
- Están definidas todas las interfaces externas.
- Cada componente está descrito en el estándar con la profundidad adecuada.

Para que el estándar sea considerado excelente en este punto debería cubrir los siguientes ítems.

- Proveer una descripción de los siguientes elementos del plan:
  - Relación entre la gestión de configuraciones y el ciclo de vida.
  - Cómo puede un cambio afectar al ciclo de vida.
  - Uso de herramientas automatizadas para la gestión de configuraciones.
- El estándar provee ejemplos de planes para poder asistir al usuario

En este punto uno de los estándares fue valuado como bueno, otro medio, y el otro como malo. Esto se debe especialmente a que la información relevante no se encuentra en algunos de los estándares, otra razón es que dos de los ellos contienen información incompleta de los tópicos que tratan.
El de la IEEE es bueno y está muy cerca de recibir un excelente. Algunos puntos en los que sobresale son, los componentes del plan descriptos están en su adecuada profundidad. Además, trata sobre todos los componentes necesarios para poder realizar el plan. Uno de los puntos por lo que no fue valuado como excelente es porque no tiene ejemplos suficientes. Por otro lado tampoco trata en detalle el tema de herramientas automatizadas.

El de la NASA fue evaluado medio, ya que no trata sobre varios de los componentes del plan que se necesitan para que pueda ser bueno.

Por ejemplo, no trata sobre las librerías, las versiones, los release, backups, recuperación. Además el nivel de detalle en que trata los componentes que si están descriptos es muy bajo. Como las explicaciones son de muy bajo detalle este estándar sólo puede ser efectivo cuando es usado por expertos como una guía rápida.

El DoD no alcanza los requerimientos mínimos ya que no trata sobre los componentes que necesitaría para poder ser considerado completo. Por ejemplo, no trata sobre la identificación y descripción de líneas base. Como tiene muy poco detalle, este estándar sólo puede ser efectivo al igual que el de la NASA para ser usado por expertos. Un aspecto positivo es que contiene un buen ejemplo de un diagrama de control de configuración. Esto puede ser muy útil especialmente para usuarios no expertos.

c. Flexibilidad.

Un estándar es considerado flexible si puede ser adaptado a las necesidades de cualquier proyecto.

Los requerimientos mínimos para que el estándar sea considerado flexible son:

- Los componentes que no se aplican a un proyecto en particular pueden ser eliminados fácilmente del plan.
- Es posible reorganizar los componentes dentro del plan para que se acomoden al proyecto.

Los atributos adicionales que debe tener un plan para ser considerado bueno son:

- El estándar puede ser adaptado, aunque quizás no muy fácilmente, a cualquier dominio de aplicación.
- El estándar puede ser adaptado, aunque quizás no muy fácilmente, a cualquier ciclo de vida.
- El estándar puede ser adaptado, aunque quizás no muy fácilmente, a cualquier software, hardware.
- El estándar da información sobre contratistas y vendedores.

Los atributos adicionales para que el plan sea considerado excelente son:

- El estándar puede ser adaptado a cualquier dominio de aplicación.
- El estándar puede ser adaptado a cualquier ciclo de vida.

- El estándar puede ser adaptado a cualquier software, hardware.
- El estándar puede adaptarse, fácil, completa y consistentemente. Esto requiere que esté organizado de tal forma que sea fácil de usar y que no contenga información redundante.

El de la IEEE fue clasificado como excelente. Fue especialmente escrito para ser flexible y trata explícitamente la flexibilidad como parte del estándar.
Contiene además instrucciones específicas que impiden que el usuario pueda eliminar algún componente del plan sin especificar primero porque lo desea eliminar. Este estándar no contiene puntos negativos en este criterio.
Tanto el estándar de la NASA como el DoD fueron valuados como medios, pero pudieron haber sido buenos ya que sólo les falta información sobre los contratistas y vendedores. La razón más importante por la cual estos estándares no son flexibles es porque no son muy completos.

d. Consistencia.
Un estándar es considerado consistente si su estructura se mantiene uniforme en todo el documento.
Los atributos mínimos que debe cumplir para ser clasificado consistente son:

- Se utiliza un formato estandarizado.
- Los nombres están estandarizados.
- El mismo componente está referenciado a lo largo de todo el documento de forma única, y sus características están definidas sólo una vez.

Los requerimientos adicionales para que sea evaluado como bueno son:

- Se utiliza un formato estructurado.
- Los nombres únicos de los componentes son conocidos y usados en la industria.
- Todo el estándar es uniforme.
- Es consistente con otros estándares del ciclo de vida del proyecto.

Para que sea evaluado como excelente debe ser consistente con todos los estándares de gestión de configuración conocidos en la industria.

Los tres estándares fueron calificados como excelentes. Los tres son muy consistentes en sí mismos, pero lo más importante es que lo son con el resto de los estándares conocidos. La única característica

que se puede agregar es que el de la IEEE contiene una sección sobre el criterio de consistencia para el plan. Ninguno de los tres estándares tiene algún aspecto negativo en este criterio.

e. Corrección.

Un estándar es considerado correcto si sólo contiene información válida.

Los atributos mínimos que debe cumplir para ser considerado correcto son:

- No provee información incorrecta.
- No tiene informaciones contradictorias.

Los atributos adicionales para que sea evaluado bueno son:

- Contiene sólo la información necesaria para explicar claramente el tópico. Si el usuario tiene que buscar en otro documento hace que el mismo no sea efectivo. Información sobre temas que no estén relacionados o excesiva información sobre el tópico, hace que el mismo sea difícil de leer.
- Los temas que se tratan tienen sentido tanto técnica como prácticamente.

Para que sea considerado excelente debe haber una evidencia sobre que el estándar es correcto. Por ejemplo que haya sido exitosamente aplicado en algún caso de estudio.

El de la IEEE fue evaluado excelente. No sólo toda la información provista es correcta sino también en un nivel de detalle adecuado. Hay unos pocos tópicos que se hubieran podido explicar con mayor detalle, pero de todas formas la información provista es suficiente.

Tanto el estándar de la NASA como el DoD, fueron caracterizados como medios, ya que no contienen la información necesaria para poder realizar el plan. Si hubieran pasado este punto, serían evaluados como excelentes en este atributo.

f. Conexión con el ciclo de vida.

El estándar es considerado como conectado al ciclo de vida si se refiere a como conectar la gestión de configuración al ciclo de vida.

Se considera que está conectado al ciclo de vida y es bueno si el estándar trata como el plan se amolda y evoluciona a través de todo el ciclo de vida.

Es excelente si trata como las actividades individuales del plan se amoldan al ciclo de vida.

Ninguno de los tres cumple con este atributo. Sólo el de la IEEE trata sobre el ciclo de vida pero no su conexión o evolución a través del mismo.

Conclusiones de la comparación.

Criterio	IEEE	NASA	DoD
Facilidad de uso	**3**	**1**	**1**
Completo	**2**	**1**	**0**
Flexibilidad	**3**	**1**	**1**
Consistencia	**3**	**3**	**3**
Correcto	**3**	**1**	**1**
Conexión con el ciclo de vida	**0**	**0**	**0**

Tabla 4.1 Resultados de la comparación.

0- No satisface el mínimo de requerimientos.
1- Satisface los requerimientos.
2- Satisface los requerimientos y es considerado bueno.
3- Satisface los requerimientos y es considerado excelente.

Como se puede observar el estándar de la IEEE es el mejor calificado, y es por lo tanto el más recomendado.
Algunos puntos para remarcar son, que el estándar de la IEEE está escrito tanto para usuarios principiantes como para expertos, mientras que el de la NASA y DoD sólo pueden ser utilizados con éxito por usuarios experimentados. El de la IEEE es el único que no necesita el uso de documentación adicional. Por lo tanto los usuarios expertos pueden utilizar cualquiera de los tres que les resultará igualmente efectivo ya que éstos sólo necesitan una guía rápida, mientras que un usuario nuevo deberá utilizar el de la IEEE. Por otro lado cualquiera de estos resulta adecuado para un usuario que

quiere repasar los conceptos generales de la gestión de configuración, ya que todos los componentes del plan descriptos en los estándares son correctos.

4.4 Plan de Gestión de Configuración.

1. Introducción del Plan.

El Plan de Gestión de Configuración es un documento que se debe producir en el comienzo de cada proyecto y que define las políticas, estándares y procedimientos que se van a utilizar para gestionar la configuración en el transcurso de dicho proyecto.

1.1. Propósito.

El principal objetivo de este plan es coordinar, organizar y controlar el desarrollo de las tareas de la Gestión de Configuración del Software del proyecto Intranet II. Además de establecer y documentar los requisitos, políticas, estándares y procedimientos para la gestión de la configuración de los elementos del software en dicho proyecto.

1.2. Alcance.

Se puede definir el alcance de este plan como todo lo referente al proyecto Intranet II. Donde se incluye una lista de los elementos de configuración del software seleccionados inicialmente en dicho proyecto:

- Diagrama Conceptual Modelo del dominio.
- Requerimientos Funcionales.
- Requerimientos no Funcionales.
- Diagrama de Casos de uso del Sistema.
- Diagramas de Colaboración.
- Diagramas de Clases Web.
- Diagrama de Clases Persistentes.
- Diagramas Entidad Relación.
- Vista de Casos de Uso.
- Vista de Restricciones.
- Vista de Calidad.
- Vista Lógica.
- Descripción de la arquitectura (Vertical, horizontal).
- Vista Refinada.

- Vista de Procesos.
- Vista de Despliegue.
- Vista de Implementación.
- Diagramas de Clases Persistentes (Noticias, Facultad).
- Diagramas Entidad de Relación (Noticias, Facultad).

1.3. Definiciones, Acrónimos y Abreviaturas.

CCB……Comité de Control de Cambios.

DD……...Defectos Descubiertos.

DR……Defectos Resueltos.

ECS……Elementos de Configuración del Software.

GCS……Gestión de Configuración del Software.

IEEE……Institute of Electrical and Electronics Engineers.

LB……Línea Base.

MD……...Momento del ciclo de vida en que son Detectados los defectos.

MR……Momento del ciclo de vida en que son Resueltos los defectos.

1.4. Referencias.

Estándar de la IEEE (Anexos 3).

1.5. Resumen.

El plan de gestión de la configuración del software del proyecto Intranet II recoge los elementos necesarios para el cumplimiento de sus objetivos. Puntos como la organización, las responsabilidades, las herramientas asignadas, la identificación de la configuración, el control de versiones, el control y la gestión de cambios y configuración, estado de la configuración, el entrenamiento y recursos e hitos constituyen las temáticas abordadas que de una forma u otra permiten coordinar, organizar y controlar

las tareas de la gestión de configuración del proyecto Intranet II así como su contribución al buen desarrollo del mismo.

2. Gestión de Configuración de Software.

2.1. Organización, Responsabilidades e interfaz.

Línea Accesos: Luis Manuel Teijón Acosta.

Línea Gestión de Usuario: Osmani Bayard Matos.

Línea Información: Karel Rodríguez Varona

Línea Publicación: Liermes Ferriol Mena.

2.2. Herramientas, entorno e Infraestructura.

La ubicación física de los servidores y las estaciones de trabajo, van a estar distribuido en el laboratorio de producción del proyecto Intranet II, ubicado actualmente en la Universidad de Ciencias Informáticas (UCI), docente 3, 1er Piso, Laboratorio 102, Boyeros, Ciudad Habana, Cuba.

El parque tecnológico a utilizar estará distribuido de la siguiente manera:

Servidores.

Sistema operativo que utilizan los servidores es Linux, cualquier distribución (Debian, Ubuntu, KDE, etc.), como servidor de bases de datos, PostgreSQL, Servidor Web, Apache, y como servidor de aplicaciones, Zope. El repositorio del proyecto será montado con Subversion.

PCs del proyecto.

Utilizaran como sistema operativo Windows o Linux, en el caso de Linux, cualquier distribución. Además de tener instalado Plone, Apache 2.0, Trac y navegadores web.

3. Programa de Gestión de Configuración.

3.1. Identificación de la Configuración.

3.1.1 Métodos de Identificación.

Cada ECS del proyecto deberá tener:

- nombre (descriptivo).
- ID (numero, letra, ambos, no ambiguo).
- tipo (código, documento, etc.).
- estado (elaboración, finalizado, revisado, aceptado).

3.1.2 Línea base del proyecto.

La línea base provee un estándar oficial en el que se basan los siguientes trabajos. Todos los elementos que estén sometidos a una aceptación por parte del cliente, son candidatos obligados a pertenecer a una línea base, estos serán incluidos en la línea que corresponda junto a cualquier otro elemento cuya modificación influya en el desarrollo posterior del proyecto. Como en este momento no ha concluido la identificación de la configuración, no se han establecido las líneas bases, por tanto, quedaran establecidas las primeras líneas bases luego de haber superados los procesos de evaluación, verificación, aprobación y aceptación.

Política: al concluir cada una de las fases del proyecto, se definirán nuevas líneas bases, siempre recordando que los elementos de una línea base pueden ser modificados únicamente a través de un proceso formal de control de cambios. Las líneas bases son autorizadas por el jefe de proyecto, en este caso, por la Ing. Dunia Suárez Ferreiro.

3.1.3 Definición de bibliotecas.

Las bibliotecas constituyen el centro de almacenamiento o acumulación de información de ingeniería del software, que permite a un equipo de trabajo gestionar los cambios de manera eficiente. La ubicación física de estas será el IP de la PC en la que será montado el repositorio para el proyecto. Además se delimitara el alcance de cada usuario sobre esta (Nivel de acceso).
Biblioteca = Repositorio = Depósito de ECS = Base de Datos del Proyecto.

3.1.4 Relaciones entre ECS.

Es de gran importancia para el proyecto definir la relación entre los diferentes elementos de configuración, para cuando surja un cambio conocer todos los ECS que se ven afectados de una forma u otra, ya sea directa o indirectamente.

Para una mayor visión sobre estas relaciones, se propone elaborar un nuevo documento, llamado Relaciones de ECS, el cual será almacenado en el repositorio como un elemento más de gran importancia para el desarrollo de la Intranet.

Relaciones de ECS	
Elemento de Configuración del Software.	**Se relaciona con:**
ECS 1.	Lista de elementos con los que se ve directamente relacionados. Cualquier cambio en ECS 1 induce a modificar esta lista.

Tabla 4.2 Relaciones de ECS.

3.2. Control de versiones. Control de configuración. Control y gestión de cambios.

El control de versiones y la gestión de cambio en todas las líneas se desarrollarán bajo las mismas herramientas y políticas de trabajo.

Recordar que un sistema de control de versiones es *un software que administra el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados*. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como una novela, o el código fuente de un programa. Normalmente consiste en una copia maestra en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local. Esto permite compartir los cambios sobre un mismo conjunto de ficheros. Además, el repositorio guarda registro de los cambios realizados por cada usuario, y permite volver a un estado anterior en caso de necesidad.

Proceso Control Versiones y Gestión Cambios:

El proceso de Control de Versiones dentro de la línea se realizará en su totalidad mediante la herramienta el "Subversion", la cual será como el depósito de toda la información que se genere y se

guarde, esta herramienta permite llevar una traza de los cambios efectuados sobre determinados documentos; fechas de cambios particulares o totales, comentarios sobre los cambios hechos, usuario que actuó sobre el, etc. Para el proceso de Gestión de Cambios se utilizará como interfaz la herramienta denominada "trac", la cual brinda un ambiente más amigable para trabajar. Esta implementa mediante los "ticket" un método para la organización, seguimiento y cumplimiento de las tareas asignadas a cada estudiante.

Todos los artefactos [Glosario de Términos] que se generen deben estar bajo el control de versiones.

Estructura y flujo de trabajo en el Repositorio.

El repositorio estará organizado de la siguiente manera (Cuatro ramas fundamentales):

- Trunk: rama principal donde se encontrarán los últimos cambios efectuados en todo el proyecto.
- Branches: rama en la que cada miembro del proyecto realizará las tareas correspondientes a su línea de trabajo.
- Tags: rama donde estarán las actividades que tiene que realizar cada persona que se encuentra en el proyecto.
- Unstable: rama donde cada usuario podrá publicar los cambios realizados al Branches de otro usuario, para luego saber si va a ser admitida o no la modificación en el Trunk.

Recordar que las ramas son útiles principalmente cuando un equipo de desarrolladores tiene permisos para enviar cambios a un mismo repositorio. Si se tiene que hacer un gran desarrollo experimental, es mejor crear una rama, hacer los cambios en ella, y reintegrarlo al tronco cuando se haya acabado el desarrollo. Esto evita interferencias.

El trabajo en el repositorio.

La rama denominada tronco ***(trunk)***, será donde se almacene la línea base del proyecto, por lo que para poder modificarlo (a), deberán realizarse diferentes pasos. Cada desarrollador tiene una carpeta ***branches***, donde tendrá las tareas realizadas hasta el momento. Este ***branches*** solamente podrá ser modificado por el desarrollador al cual pertenece. En caso de que algún desarrollador necesite llevar a cabo algún cambio en otro ***branches***, lo lleva a cabo en una carpeta denominada ***unstable***, la cual se utiliza solamente para realizar los cambios por desarrolladores, carpeta en la cual varía mucho la información y es muy inestable. Luego el Jefe de Línea y el responsable de la Gestión de

Configuración de la Línea verán si son pertinentes estos cambios o no, para luego de ser revisadas por el Jefe de Línea ser subidos al tronco. En el ***unstable*** no se va a llevar a cabo el proceso de Gestión de Cambios, porque es una carpeta de trabajo inestable.

Si se desea realizar algún cambio en el tronco, remitirse al proceso de Gestión de Cambios.

3.2.1 Proceso de solicitud y aceptación de cambios

- Desarrolladores.
- Entre líneas.
- Cliente.

Proceso de cambios por desarrolladores:

- Cada desarrollador tiene acceso a publicar ticket.
- El desarrollador que desee llevar a cabo el cambio ya sea por defecto o mejora deberá acceder al ticket a través del Trac.
- El desarrollador llenará todos los datos que se especifican en el ticket: ***tipo, prioridad***.
- En caso de otros aspectos como: descripción del pedido, listado de nuevos requerimientos que desean ser agregados, condiciones de software bajo las que fue detectado el problema y descripción del cambio propuesto por parte del creador, deberán ser publicados en la región de comentarios**.**
- No es necesario al llenar los ticket poner nombre del desarrollador ni fecha de creación, ya que esto queda registrado automáticamente al publicar el ticket.
- Una vez publicado el ticket por el desarrollador, el CCB tendrá la tarea de acceder al ticket de cada desarrollador para ver cuales cambios están propuestos a realizar.
- Una vez aceptado o rechazado el cambio, el CCB deberá acceder a la plantilla De Gestión de Cambios entre Desarrolladores que se encuentra en el trac para ser llenada.
- Una vez llenada dicha plantilla, será adjuntada por el CCB en el ticket correspondiente para que el desarrollador que pidió la autorización del cambio tenga acceso a esta plantilla para que sepa si lo efectúa de inmediato o no.
- En caso de que fuera aceptado el cambio, informar a todos los miembros de la línea de dicho cambio.
- En caso de no aceptarlo, el CCB de la línea deberá exponer sus criterios en dicha plantilla.

Proceso de cambios entre líneas:

- Cada desarrollador tiene acceso a publicar ticket.
- El desarrollador que desea llevar a cabo el cambio ya sea por defecto o mejora deberá acceder al ticket a través del Trac.
- El desarrollador llenará todos los datos que se especifican en el ticket como son: ***tipo, prioridad***.
- En caso de otros aspectos como: descripción del pedido, listado de nuevos requerimientos que desean ser agregados, condiciones de software bajo las que fue detectado el problema y descripción del cambio propuesto por parte del creador**,** deberán ser publicados en la región de comentarios.
- No es necesario al llenar los ticket poner nombre del desarrollador ni fecha de creación, ya que esto queda registrado automáticamente al publicar el ticket.
- Una vez publicado el ticket por el desarrollador, el CCB tendrá la tarea de acceder al ticket de cada desarrollador para ver cuales cambios están propuestos a realizar.
- Como el cambio es entre líneas, el CCB estará integrado por cada línea involucrada.
- Una vez aceptado o rechazado el cambio, el CCB deberá acceder a la plantilla de Gestión de Cambios entre Líneas que se encuentra en el trac para ser llenada.
- Una vez llenada dicha plantilla, será adjuntada por el CCB en el ticket correspondiente para que el desarrollador que pidió la autorización del cambio tenga acceso a esta plantilla para que sepa si lo efectúa de inmediato o no.
- En caso de que fuera aceptado el cambio, informar a todos los miembros de las líneas involucradas de dicho cambio.
- En caso de no aceptarlo, el CCB de ambas líneas deberá exponer sus criterios en dicha plantilla.

Proceso de cambios por el cliente.

- El CCB tiene en sus manos la plantilla de Gestión de Cambios para Clientes.
- El CCB deberá entregar (por correo electrónico o personalmente) la plantilla al cliente.
- El cliente la llenara con todos los datos que se especifican en la misma.
- Se le entregara al CCB para que vean si el cambio afecta en algo el trabajo que se venía desarrollando (si se aprueba o no).
- Una vez aceptado o rechazado el cambio se le comunicara al cliente de inmediato.
- En caso de que fuera aceptado, informar a todos los miembros del proyecto de dicho cambio.

- En caso de no aceptarlo el CCB deberá exponer sus criterios.

Nota Importante:

Cada una de estas plantillas después de ser aprobados o no los cambios, deben ser almacenadas en algún sitio por el CCB para su posterior apoyo en caso de necesitarse.

3.2.2 Comité de Control de Cambios (CCB).

Los que integraran este comité serán:

Línea	**Jefe de línea**	**Responsable de GCS.**
Línea Accesos	Luis Manuel Teijón Acosta	Raudel González Echenique
Línea Gestión de Usuario	Osmani Bayard Matos	Javier Piloto Rodríguez
Línea Información	Karel Rodríguez Varona	Yordanis Cabreja Núñez
Línea Publicación	Liermes Ferriol Mena	Liermes Ferriol Mena

Tabla 4.3 Comité de Control de Cambios.

3.3. Estado de la configuración.

3.3.1 Almacén del proyecto y procedimiento de liberación.

El repositorio estará montado en la máquina designada por el proyecto. Serán almacenado todos los ECS definidos, junto a todos los elementos que de una forma u otra puedan influir en el desarrollo de las líneas bases. Además cualquier documento o artefacto que resulte de interés para el proyecto en general.

Solo podrán acceder y modificar los elementos almacenados, los miembros del proyecto según el privilegio predefinido por las directivas del mismo. En base a esto se establecen las siguientes medidas:

- Cada usuario debe autenticarse para poder acceder al repositorio.

- Cada modificación realizada en el repositorio será registrada.

Nota: La herramienta que permitirá establecer el repositorio, Subversion, será configurada en base a lo anterior.

En el Control de Cambios.

¿Qué hacer cuando dos usuarios intentan modificar el mismo fichero?

Existen dos estrategias:

- *Bloqueos*: el usuario bloquea el fichero durante su edición, evitando el acceso concurrente de otros usuarios. Existen varios problemas: el usuario que acapara ficheros, el interbloqueo entre usuarios que necesitan varios ficheros, y la falta de concurrencia.

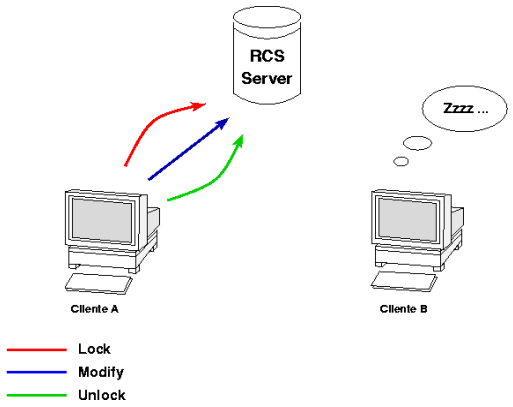

Fig. 4.1 Esquema Lock - modify- unlock.

- *Merge* (fusión de cambios): los ficheros se acceden concurrentemente. Los cambios realizados sobre un mismo fichero son fusionados inteligentemente por el sistema. El único problema es el intento de fusión de cambios incompatibles, que ha de solucionarse manualmente.

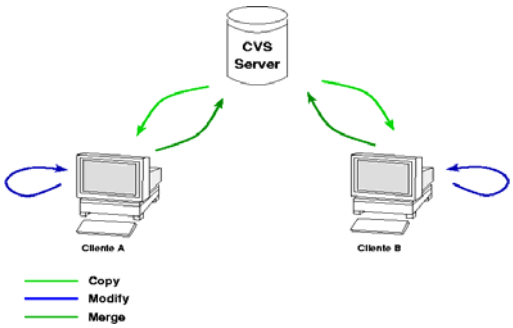

Fig. 4.2 Esquema copy - modify- merge.

En el proyecto se utilizara la segunda variante, ya que la herramienta propuesta para el control de versiones, Subversion, trabaja por este método.

3.3.2 Reportes y auditorias.

Los reportes basados en los defectos o en las solicitudes de cambio pueden aportar indicadores de calidad útiles, además de alertar a los gestores y a los desarrolladores sobre algunas áreas particularmente críticas del desarrollo.

Reporte basado en defectos	
Grado critico	*[alto, medio, bajo]*
Distribución	*[Contar los defectos por diferentes categorías (el dueño, prioridad, estado de arreglo)]*
Antigüedad *(basado tiempo)*	
Tiempo	*[Que tiempo se mantiene abierto un defecto]*
MD vs. MR	*[Momentos del ciclo de vida en el que son detectados defectos contra el momento en el que fueron resueltos]*
Combinación de tiempo y conteo	
Estadísticas	*Número de defectos acumulados y resueltos en un tiempo*
Razón DD/DR	*Razón de defectos descubiertos contra los resueltos*
Brecha de Calidad	*"brecha de calidad" defectos abiertos contra defectos cerrados*
Promedio	*Promedio de defectos resueltos en un tiempo*

Tabla 4.4 Propuesta de reporte.

4. Hitos.

Siendo los sucesos o acontecimientos que sirven como puntos de referencia, se marcará un hito al final de cada fase de desarrollo del proyecto (Inicio, Elaboración, Construcción, Prueba y Transición). Agregando a esto cada entregable, que constituirán por si mismos un hito. En cada uno de estos momentos, se podrán efectuar actualizaciones y cambios sobre este Plan de Gestión de la Configuración en caso de ser necesario.

5. Entrenamiento y Recursos.

Comandos principales en Subversión.

- Actualizar la copia local.
  - **svn update**: Actualiza la copia local de trabajo desde los datos del repositorio.
- Hacer cambios.
  - **svn copy**: Realiza una copia de un fichero dentro de un repositorio (no entre repositorios).
  - **svn add**: Añade ficheros y directorios a la copia local que se copiarán en el repositorio central después.
  - **svn delete**: Elimina ficheros de un repositorio que no han sido modificados o versionados.
  - **svn move**: Mueve un fichero o directorio dentro de un repositorio (no entre repositorios).
- Enviar los cambios al repositorio.
  - **svn commit**: Envía los cambios desde la copia local al repositorio central e indica al usuario que dé una descripción de los cambios realizados.
- Examinar los cambios.
  - **svn diff**: Muestra diferencias entre 2 rutas.
    - Examinar cambios locales.
    - Comparar la copia local con la del repositorio.
    - Comparar los repositorios.
  - **svn status**: Imprime el estado de los ficheros y directorios de su copia local de trabajo.
  - **svn revert**: Deshacer todas las modificaciones locales.
- Fusionar los cambios con otros.
  - **svn merge**: Aplica las diferencias entre dos fuentes a una ruta de su copia local de trabajo.

- **svn resolved**: Resuelve el estado “conflictivo” en ficheros o directorios de la copia de trabajo local.

- Otros comandos
  - **svn checkout**: Obtiene una copia local de trabajo de un repositorio.
  - **svn help**: Obtiene información de ayuda acerca de un comando.
  - **svn list**: Muestra las entradas de un repositorio.
  - **svn log**: Muestra los mensajes de los informes de cambios.
  - **svnadmin**: Administración de repositorios y del sistema Subversión.

4.5 Conclusiones.

Al concluir este capítulo se dota al proyecto Intranet de un plan de gestión de configuración, que resulta de gran importancia para el buen desempeño del mismo para garantizar el control del trabajo con los elementos de configuración, el registro y la evaluación de los cambios sobre ellos, logrando una mayor visibilidad sobre el producto. Como fundamento de la guía para desarrollar el mismo, se muestra un análisis del estándar brindado por IEEE. De modo general se pretende legar un plan que permita un mayor rendimiento y calidad en el producto final a lograr por el proyecto, la Intranet II de la UCI.

Conclusiones.

Durante el proceso de construcción de un software, los cambios son inevitables, provocan confusión e incertidumbre, sobre todo cuando no se han analizado o pronosticado correctamente. La finalidad de la Gestión y Configuración del Software es el conocer la estructura de procesos y herramientas para aplicar dentro de la construcción del software que favorece el control de los cambios.

La gestión y configuración del software identifica, controla audita e informa de las modificaciones que invariablemente se dan al desarrollar el software y una vez que ha sido distribuido a los clientes. Esta se organiza de tal forma que sea posible un control organizado de los cambios.

Además está compuesta por un conjunto de objetos interrelacionados, denominados elementos de configuración del software, que son provocados para las actividades del desarrollo, de la ingeniería del software.

Las líneas base permiten guiarse para desarrollar los cambios, los objetos forman un asociación que refleja las variantes y relaciones creadas, el control de versiones son procedimientos y herramientas que ayudan a gestionar el uso de los objetos. El control de cambios es una actividad procedimental que asegura la calidad en los cambios del los ECS.

No realizar una adecuada gestión de la configuración a un proyecto puede causar grandes problemas y resultados poco eficientes. Una de las clave para el éxito en la solución de estos posibles problemas radica en la elaboración de un Plan de gestión. Los estándares proveen una excelente asistencia a los usuarios para la realización de estos planes de gestión de configuración, los cuales brindan un marco dentro del cual escribir el plan y sirve como una guía general.

Con este plan de gestión de configuraciones se agrega al proyecto un gran valor, ya que provee principalmente la documentación de los procesos de gestión de configuraciones. Se establece además una política a ejercer para el buen desarrollo del control de versiones, la administración de cambios. Permite saber dónde, qué y cuales son los pasos a seguir.

Gestión de la Configuración, control y la calidad del software.

Recomendaciones.

- Dar seguimiento a este plan, profundizar en su contenido producto de actualizaciones obtenidas de estudios o situación del proyecto Intranet.
- Realizar análisis y actualizaciones de las herramientas aquí expuestas que puedan mejorar el funcionamiento del proyecto.
- Estudiar efectividad del plan sobre el proyecto.

Referencia bibliográfica.

[1] Anónimo. Intranet, Mayo 2007 [Disponible en: http://es.wikipedia.org/wiki/Intranet].

[2] Imacom, Intranet [Disponible en: http://www.imacom.cl/otros.html].

[3] Imacom, Intranet [Disponible en: http://www.imacom.cl/otros.html].

[4] Anónimo, Intranet Corporativa, Septiembre 2006 [Disponible en: http://es.wikipedia.org/wiki/Intranet_Corporativa].

[5] Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

[6] Babich, W, Software Configuration Management, Adison-Wesley, 1986.

[7] Rational 2003 Rational Unified Procces, IBM, 2003.

[8] Ramsés Delgado y Emilio Glez., ConfigCase 3.0 Herramienta de apoyo a la gestión de configuración. Propuesta arquitectónica, Instituto Superior Politécnico “José Antonio Echeverría”, Ciudad de la Habana, 2006,135 pg.

[9] Ramsés Delgado y Emilio Glez., ConfigCase 3.0 Herramienta de apoyo a la gestión de configuración. Propuesta arquitectónica, Instituto Superior Politécnico “José Antonio Echeverría”, Ciudad de la Habana, 2006,135 pg.

[10] Pressman, R. S. Ingeniería de Software. Un enfoque práctico. Madrid, 2001. 614 p.

[11] Bersoft, E.H., Henderson, V.D., Siegel, S. G., Software Configuration Management, Prentice-Hall, 1980.

[12] Ramsés Delgado y Emilio Glez., ConfigCase 3.0 Herramienta de apoyo a la gestión de configuración. Propuesta arquitectónica, Instituto Superior Politécnico “José Antonio Echeverría”, Ciudad de la Habana, 2006,135 pg.

[13] Pressman, R. S. Ingeniería de Software. Un enfoque práctico. Madrid, 2001. 614 p.

[14] Ramsés Delgado y Emilio Glez., ConfigCase 3.0 Herramienta de apoyo a la gestión de configuración. Propuesta arquitectónica, Instituto Superior Politécnico “José Antonio Echeverría”, Ciudad de la Habana, 2006,135 pg.

[15] Pressman, R. S. Ingeniería de Software. Un enfoque práctico. Madrid, 2001. 614 p.

[16] Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

[17] Febles, Ailyn, MConfig.PM, Modelo de referencia para la Gestión de Configuración en la pequeña y mediana empresa de software, La Habana, Cuba, CUJAE, 2004.

[18] Ramsés Delgado y Emilio Glez., ConfigCase 3.0 Herramienta de apoyo a la gestión de configuración. Propuesta arquitectónica, Instituto Superior Politécnico “José Antonio Echeverría”, Ciudad de la Habana, 2006,135 pg.

[19] Ramsés Delgado y Emilio Glez., ConfigCase 3.0 Herramienta de apoyo a la gestión de configuración. Propuesta arquitectónica, Instituto Superior Politécnico “José Antonio Echeverría”, Ciudad de la Habana, 2006,135 pg.

[20] Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

[21] Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

[22] Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

[23] Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

[24] Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

[25] D. d. I. Software, "Conferencia 5," *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis*, 2006.

[26] D. d. I. Software, "Conferencia 5," *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis*, 2006.

[27] D. d. I. Software, "Conferencia 5," *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis*, 2006.

[28] D. d. I. Software, "Conferencia 5," *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis*, 2006.

[29] D. d. I. Software, "Conferencia 5," *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis*, 2006.

[30] D. d. I. Software, "Conferencia 5," *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis*, 2006.

[31] D. d. I. Software, “Conferencia-_Diseño”, 2007, [Disponible en: http://teleformacion.uci.cu/course/view.php?id=43].

[32] Pressman, R. S. Ingeniería de Software. Un enfoque práctico. Madrid, 2001. 614 p.

[33] Max Zeledón Collins, Stallman y el software libre, 2005 [Disponible en: http://www.fernandoflores.cl/node/137].

Bibliografía.

A

Anónimo, Configuración del Software [Disponible en: http://html.rincondelvago.com/configuracion-de-software.html].

Anónimo, Intranet Corporativa, Septiembre 2006 [Disponible en: http://es.wikipedia.org/wiki/Intranet_Corporativa].

Anónimo. Intranet, Mayo 2007 [Disponible en: http://es.wikipedia.org/wiki/Intranet].

B

Babich, W, Software Configuration Management, Adison-Wesley, 1986

Bersoft, E.H., Henderson, V.D., Siegel, S. G., Software Configuration Management, Prentice-Hall, 1980

Bounds, N. – Dart, S. Configuration Management Plans: The Beginning to your CM Solution. 1998. URL: http://www.sei.cmu.edu/legacy/scm/papers/CM_Plans.

D

D. d. I. Software, "Conferencia 5," *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis*, 2006.

D. d. I. Software, "Conferencia-_Diseño", 2007, [Disponible en: http://teleformacion.uci.cu/course/view.php?id=43].

Delgado Ramsés y Glez Emilio., ConfigCase 3.0 Herramienta de apoyo a la gestión de configuración. Propuesta arquitectónica, Instituto Superior Politécnico "José Antonio Echeverría", Ciudad de la Habana, 2006,135 pg.

Diccionario de la lengua española © 2005 Espasa-Calpe S.A., Madrid.

E

Escuela Universitaria de Informática, Evaluación Organización y Gestión de Proyectos Informáticos [Disponible en: http://www.upv.es/~jmontesa/eog-ind.html].

F

Febles, Ailyn, MConfig.PM, Modelo de referencia para la Gestión de Configuración en la pequeña y mediana empresa de software, La Habana, Cuba, CUJAE, 2004.

I

Imacom, Intranet [Disponible en: http://www.imacom.cl/otros.html].

M

Ministerio de Administraciones Públicas Metodología MÉTRICA Versión 3, Gestión de Configuración [Disponible en: http://www.csi.map.es/csi/metrica3/gescon.pdf].

N

Navarro Antonio, Ingeniería del Software. Gestión de la Configuración [Disponible en: http://www.fdi.ucm.es/profesor/anavarro/10._Gestion_de_la_configuracion_software.pdf].

O

Olivera Ángel, Criterios para construir una Intranet Corporativa, 2007 [Disponible en: http://www.arearh.com/software/intranet.htm].

P

Pressman, R. S. Ingeniería de Software. Un enfoque práctico. Madrid, 2001. 614 p.

R

Rational 2003 Rational Unified Procces, IBM, 2003.

T

Tinoco Gaviria Hermilson, Control de versiones sobre bases de datos SQL Server 2000/2005 [Disponible en: http://www.dotnetcr.com/Libreria.aspx?art=95&tag=Control-de-versiones-sobre-bases-de-datos-SQL-Server-2000-2005].

Tutorial del sistema de control de versiones Subversion. Alejandro Ramírez.

Z

Zeledón Collins Max, Stallman y el software libre, 2005 [Disponible en: http://www.fernandoflores.cl/node/137].

Anexos.

Anexo 1 Estándar de la IEEE.

IEEE Std. 828-1998	
1. Introducción	1.1 Propósito 1.2 Alcance 1.3 Definición de términos clave 1.4 Referencias
2. Gestión de la GCS	2.1 Organización 2.2 Responsabilidades GCS 2.3 Políticas, directivas y procedimientos aplicables
3. Actividades de la GCS.	3.1 Identificación de la configuración 3.1.1 Identificación de ECSs 3.1.2 Nombrado de ECSs 3.1.3 Adquisición de ECSs 3.2 Control de la configuración 3.2.1 Petición de cambios 3.2.2 Evaluación de cambios 3.2.3 Aprobación o desaprobación de cambios 3.2.4 Implementación de cambios 3.3 Contabilidad de estado de configuración 3.4 Auditorias y revisiones de la configuración 3.5 Control de interfaz 3.6 Control de la subcontratación/compra
4. Planificaciones de la GCS	
5. Recursos de la GCS	
6. Mantenimiento del plan de GCS	

Glosario de Términos.

Actualización: ("sync") integra los cambios que han sido hechos en el repositorio (por ejemplo por otras personas) en la copia de trabajo local.

Cambio: ("change", "diff", "delta") representa una modificación específica a un documento bajo control de versiones. La granularidad de la modificación considerada un cambio varía entre diferentes sistemas de control de versiones.

Check-out: crea una copia de trabajo local desde el repositorio. Se puede especificar una revisión específica, y por defecto se suele obtener la última.

Cliente: persona que utiliza con asiduidad los servicios de un profesional o empresa. Es el último destinatario del producto desarrollado.

Código fuente un conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por humanos.

Commit: sucede cuando una copia de los cambios hechos a una copia local es escrita o integrada sobre repositorio.

Configuración del Software: el arte de coordinar el desarrollo del software para minimizar la confusión; identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación. La meta es maximizar la productividad minimizando los errores.

Congestionar: obstruir o entorpecer el paso, la circulación o el movimiento de algo.

Ejecutable: fichero de un programa informático que tiene las órdenes para activar y utilizar dicho programa.

Estándar: que tiene el tamaño, la forma o cualquier otra característica que sigue al modelo. Se aplica a lo que se produce en serie. Que sigue una tendencia muy extendida. Aquello que se considera modelo.

Etiqueta: ("tag", "release") se refiere a una "toma" en el tiempo importante, consistente a lo largo de muchos ficheros. Estos ficheros en un instante del tiempo pueden ser colectivamente etiquetados con un nombre fácil de identificar por el usuario, con significado, o con un número de revisión.

Exportación: ("export") es similar a un check-out, salvo porque crea un árbol de directorios limpio sin los metadatos de control de versiones presentes en la copia de trabajo. Se utiliza a menudo de forma previa a la publicación de los contenidos.

Gestión: conjunto de trámites que se llevan a cabo para resolver un asunto. Dirección, administración de una empresa, negocio, etc.

Importación: ("import") es la acción de copia un árbol de directorios local (que no es en ese momento una copia de trabajo) en el repositorio por primera vez.

Integración: ("merge") une dos conjuntos de cambios sobre un fichero o un conjunto de ficheros en una revisión unificada de dicho fichero o ficheros.

Interfaz: colección de operaciones que son usadas para especificar un servicio de una clase o un componente.

Interfaz gráfica: tipo de interfaz que permite a los usuarios comunicarse con un programa mediante funcionalidades gráficas. Normalmente incluyen una combinación de gráficos, barras de menús e íconos.

Intranet: es una red de ordenadores. Importante medio de difusión de información interna a nivel de grupo de trabajo. Creada para mayor seguridad para poder compartir archivos, carpetas y recursos. Excelente opción de bajo costo para las empresas.

Licencia: contrato entre el desarrollador de un software sometido a propiedad intelectual y a derechos de autor y el usuario, en el cual se definen con precisión los derechos y deberes de ambas partes. Es el desarrollador, o aquél a quien éste haya cedido los derechos de explotación, quien elige la licencia según la cual distribuye el software.

Línea base: una revisión aprobada de un documento o fichero fuente, a partir del cual se pueden realizar cambios subsiguientes.

Página web: documento situado en una red informática, al que se accede mediante enlaces de hipertexto. Es una fuente de información adaptada para la World Wide Web y accesible mediante un navegador de Internet. Las páginas web pueden ser cargadas de un ordenador local o remoto, llamado Servidor Web.

Proveedor (es): persona o empresa que provee o abastece de todo lo necesario para un fin a grandes grupos, asociaciones, comunidades, etc.

Rama: ("branch") un conjunto de ficheros bajo control de versiones puede ser branched o forked en un momento de tiempo de forma que, desde ese momento en adelante, dos copias de esos ficheros puedan ser desarrolladas a diferentes velocidades o de diferentes formas, de modo independiente.

Red de ordenadores: una red de computadoras, también llamada red informática. Es un conjunto de computadoras y/o dispositivos conectados por enlaces de un medio físico (medios guiados) ó inalámbricos (medios no guiados) y que comparten información (archivos), recursos (CD-ROM, impresoras, etc.) y servicios (e-mail, chat, juegos), etc.

Repositorio: Almacén, es el lugar en el que se almacenan los datos actualizados e históricos, a menudo en un servidor. A veces se le denomina depósito (e.g. with SVK, AccuRev and Perforce).

.

Rol: papel que desempeña una persona o grupo en cualquier actividad.

Revisión: ("versión") es una versión dentro de una cadena de cambios.

Servidor: una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes.

Software: término genérico que se aplica a los componentes no físicos de un sistema informático, como p. ej. Los programas, sistemas operativos, etc., que permiten a este ejecutar sus tareas. Esto incluye aplicaciones informáticas tales como un procesador de textos, que permite al usuario realizar una tarea, y software de sistema como un sistema operativo, que permite al resto de programas

funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de aplicaciones.

Software Libre: Software libre (en inglés free software) es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente.