Universidad de las Ciencias Informáticas

Facultad 7

Título: Métodos y herramientas libres en apoyo a la Gestión de la Configuración

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora: Elizabeth Paez Pernía

Tutor: Ing. Yanoksy Durañona Yero

Ciudad de La Habana, Junio de 2009

"Año del 50 aniversario del triunfo de la Revolución"

"Haz bien la Gestión de Configuración, u olvídate de avanzar en el proceso de desarrollo de Software"

DECLARACIÓN DE AUTORÍA

Declaro ser la única autora de este trabajo y autorizo 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 22 del mes de junio del año 2009.

_____	_____
Elizabeth Paez Pernía	Ing. Yanoksy Durañona Yero.
Autora	Tutor

DATOS DE CONTACTO

Ing. Yanoksy Durañona Yero

Profesor adiestrado, graduado como Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2007. Actualmente imparte asignaturas del Segundo Perfil de Imágenes en la Facultad #7. Se desempeña como subdirector de tecnologías y administrador de la configuración en el Polo de Imágenes.

AGRADECIMIENTOS

Agradezco a toda mi familia, en especial a mi Mami y Jorge por todo el amor y apoyo que siempre me han dado, por su constante preocupación, por quererme, por hacer de mí la persona que soy, por tantos buenos consejos y dedicación. Muchos besos los quiero mucho. A mi papito que me ha apoyado incondicionalmente en los momentos más difíciles y en los días más tristes dándome apoyo y fuerza diaria para poder seguir adelante; a su mamá y familia por estar ahí siempre para mí.

Quiero agradecer a todos los que de un modo u otro han influido en mi educación, dígase maestros, amigos, compañeros de aula tanto del preuniversitario como de la Universidad. Al tutor por apoyarme y a la estrella Aldebarán por matricularme en esta Universidad de Ciencias Informáticas y graduarme. A mis amistades sinceras que me han dado fuerzas para llegar aquí como Yudita y Clenda. En general a todos los que estuvieron durante estos 5 años.

Muchas Gracias

DEDICATORIA

A mi mami por ser mi razón de ser y de existir, va dedicado todo mi esfuerzo por ser este su mayor sueño.

A alguien muy especial para mi que ha pesar de estar muy lejos hoy, siempre está a mi lado en mis pensamientos, mi hermana Vivian.

En especial a mi papito, más conocidos por todo como Tarafa, que lo quiero mucho.

Besos Elizabeth

RESUMEN

La Universidad de las Ciencias Informáticas (UCI), tiene entre sus metas más importantes desarrollar la producción de software y de esta forma contribuir a la economía del país. En esta universidad existen gran número de proyectos productivos; los que se ven frenados por dificultades en la calidad del proceso productivo. Al no emplear adecuadamente las prácticas para la Gestión de la Configuración de Software. Lo que atenta contra la productividad, calidad, tiempo de desarrollo del producto y consecuentemente con la efectividad de los equipos de desarrollo, todo lo anterior influye negativamente en los resultados del producto final.

Con el presente trabajo se brinda una propuesta de estrategia para llevar a cabo la Gestión de Configuración de Software (GCS) en el Polo de Imágenes de la Facultad 7. En misma se definen procedimientos a aplicar en los diferentes proyectos incluidos en este polo, se hace necesario que los equipos de trabajo adopten un proceso de desarrollo de software, o sea un marco que defina las actividades necesarias. Lo que permite garantizar, técnica y administrativamente la creación de productos de una manera organizada, disciplinada y previsible. Lo que se alcanza mediante la puesta en práctica de la disciplina conocida como GCS. La estrategia propuesta es inicialmente aplicada en el proyecto alasRIS (Sistema de Información Radiológica).

Palabras Clave: Gestión de Configuración de Software, Polo de Imágenes.

Tabla de Contenidos

INTRODUCCIÓN

La producción de Software a nivel mundial crece cada día a un ritmo acelerado, aunque en muchos casos no llega a ser fructífera, los costos pueden llegar a ser muy elevados. Generalmente sucede en los casos en que no se aplican debidamente las metodologías y técnicas para la ingeniería y gestión de software.

En los inicios de la era informática, el gasto en computación era principalmente en hardware y el software era gratuito, o era incluido en el precio del hardware. La dinámica con que han evolucionado las nuevas tecnologías ha posibilitado en gran medida que los costos del hardware disminuyan en forma continua, mientras que el costo del software se ha transformado en la parte más importante del gasto en tecnología informática.

En los momentos actuales, el uso de las Tecnologías de la Información y las Comunicaciones (TIC) es uno de los componentes fundamentales del proceso de modernización de la sociedad a nivel universal y del cual no está exenta Cuba, insertada ya, con un protagonismo creciente y sólido, en dicho proceso. Como dijera el Comandante Fidel Castro “No hay más que asomarse a las puertas de la tecnología y la ciencia contemporáneas para preguntarnos si es posible vivir y conocer ese mundo del futuro sin un enorme caudal de preparación y conocimientos”.

El desarrollo de software podría constituir una importante fuente de recursos para el país, en la Cumbre Mundial sobre la Sociedad de la Informática en el año 2005 se plantea…” La Industria Cubana del Software (ICSW) está llamada a convertirse en una significativa fuente de ingresos nacional, como resultado del correcto aprovechamiento de las ventajas del considerable capital humano disponible. La Universidad de las Ciencias Informáticas (UCI) y el sistema de empresas cubanas vinculadas a este trabajo jugarán un papel importante en el desarrollo de la Industria Cubana del Software, y en la materialización de los proyectos asociados al programa cubano de informatización”.

La Industria Cubana de Software está encaminada, según las tendencias actuales, a la construcción de sistemas cada día más eficientes y complejos. De modo que se ha propuesto alcanzar un lugar relevante en el mercado mundial tanto en la rama de la medicina y la biotecnología que ya cuenta con grandes logros, como en la industria del software. Esta meta se encuentra limitada ya que el uso de la Informática

en Cuba ha estado marcado por el Sistema Operativo Windows; lo que constituye una gran desventaja para el país por ser un sistema privativo, que exige pago por su adquisición y licencias de uso.

Pero las esperanzas crecieron en el año 1991 con la creación del Sistema Operativo Linux, clasificado dentro de los sistemas de Software Libre. Linux difiere de Windows, principalmente por ser libre y resolver las cuestiones planteadas anteriormente. El gobierno cubano también ha hecho manifiesto su interés en este movimiento, considerándolo esencial para el desarrollo del país. Al respecto, en el año 2002 fue lanzada una estrategia guiada por el Ministerio de la Informática y las Comunicaciones (MIC) para favorecer la inserción en el país de las tecnologías libres.

Son varias y valiosas las acciones desarrolladas para la migración al software libre, y se ha adquirido una importante experiencia en muchos centros y profesionales del país. Aunque todavía se requiere mayor cohesión para enfrentar los altibajos y la poca consistencia de criterios existentes.

Existen ejemplos, como la Universidad de las Ciencias Informáticas, que acertadamente ha dedicado un por ciento de su matrícula al desarrollo de las tecnologías libres. Hoy en día constituye el pilar en el desarrollo de este campo en el país. La dirección del país confía que en el futuro, en la UCI se desarrollen una serie de polos científicos y grupos de desarrollo inmersos totalmente en la producción de software. Se necesita que la universidad trabaje cada día para lograr la excelencia productiva en esta nueva era tecnológica.

Como derivación lógica se encuentra el hecho de que el software requiere de mayor calidad y eficiencia por tanto; lo más estratégico y ventajoso será profundizar en aras de mejorar las prácticas de ingeniería y gestión de software. “El arte de coordinar el desarrollo de 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 es maximizar la productividad minimizando errores.” (Babich, 1986)

La Gestión de Configuración del Software es uno de los procesos clave para toda organización dedicada a la Producción de Software. Es una actividad de garantía de calidad que se aplica en todas las fases del

proceso de ingeniería de software. Es bien considerada como una disciplina de autoprotección que permite controlar y mantener la integridad de los productos durante su evolución.

En la Facultad 7, el Polo de Imágenes se encuentra inmerso en el desarrollo de diversos sistemas de elevada complejidad con alcance nacional e internacional que requieren gran madurez y excelencia. Para alcanzar el nivel antes mencionado surge la necesidad de llevar a cabo buenas prácticas de GCS, pues como bien plantea la primera ley de la Gestión de Configuración:

"La Gestión de la Configuración es el fundamento de un proyecto software, sin ella, no importa cuán talentoso sea el equipo, cuán grande sea el presupuesto, cuán robusto sean los procesos de desarrollo y prueba, o cuán superior sean las herramientas de desarrollo técnicamente, la disciplina del proyecto colapsará y se perderá la posibilidad de triunfo. Haz bien la Gestión de Configuración, u olvídate de avanzar en el proceso de desarrollo de Software" (Navarro, 2006)

Siempre que un proyecto se encuentra en ejecución los cambios son inevitables y a su vez provocan confusión e incertidumbre sobre todas las cosas cuando no se ha hecho correctamente un previo análisis o pronóstico. Los problemas más frustrantes son causados a menudo por la mala Gestión de Configuración del Software. Esta disciplina se ha considerado, en los últimos años, como la columna vertebral en el desarrollo de software. A pesar de esto, sus procesos son escasamente aplicados en la producción y no son incluidos en los planes de estudio de las carreras afines.

Actualmente en el Polo de Imágenes existen deficiencias para gestionar el engorroso proceso de desarrollo, el cual se encuentra limitado por la carencia del rol de Gestor de la Configuración. La ausencia de este rol provoca conflictos y en ocasiones pueden resultar diversos problemas como la no elaboración del Plan de Gestión de Configuración y por consiguiente las actividades definidas en él. A menudo estas inconsistencias ocurren en los peores momentos y un problema que tal vez se fijó resuelto de repente vuelve a aparecer, debido a que el proceso de Gestión de Cambios se realiza de modo informal, y en numerosas ocasiones sin dejar constancia escrita.

Otro problema para el Polo de Imágenes lo constituye la no definición de un estándar para el uso de las herramientas que se emplean en apoyo a las diversas actividades de la Gestión de Configuración. De modo que algunos proyectos emplean Visual SourceSafe que constituye una herramienta privativa y precisamente en los momentos actuales se aboga por el empleo de Software Libre. Muchas veces la comunicación entre los integrantes del proyecto es mínima, esto trae como consecuencia que los involucrados no dispongan de una versión adecuada o común del producto.

Los problemas previamente planteados traen consecuencias como: las insatisfacciones por parte de los clientes y en ocasiones no lograr alcanzar el producto esperado en el tiempo estimado. Por tal motivo se evidencia la necesidad de utilizar métodos y herramientas basadas en tecnologías libres para llevar a cabo la Gestión de Configuración de Software; proceso que se debe realizar desde el comienzo del desarrollo de software hasta que el producto queda liberado.

Como se ha evidenciado, el proceso de Gestión de Configuración, es determinante en el desarrollo de software por lo que se plantea como **problema científico** ¿Cómo mejorar la Gestión de Configuración del Software en el Polo de Imágenes?

En aras de solucionar el problema se enmarca como **objeto de estudio** los métodos y herramientas libres para apoyar la GCS. Se define como **campo de acción** la aplicación de métodos y herramientas libres dentro del Polo de Imágenes en vista a mejorar la GCS.

Se traza como **objetivo general** de la investigación: brindar una propuesta de los métodos y herramientas libres a utilizar en el Polo de Imágenes para perfeccionar la GCS.

Para dar cumplimiento al objetivo se plantean las siguientes Tareas a Desarrollar:

- ✓ Asimilar el estado actual de la Gestión de Configuración de Software tanto en la Universidad de las Ciencias Informáticas como en el mundo.

- ✓ Analizar las particularidades que presenta la Gestión de Configuración en el Polo de Imágenes.
- ✓ Definir las actividades a tener en cuenta en el proceso de Gestión de Configuración de Software.
- ✓ Desarrollar una búsqueda sobre las diferentes herramientas libres existentes que puedan aplicarse en las diferentes ramas de la Gestión Configuración de Software.
- ✓ Presentar un análisis valorativo acerca de las herramientas encontradas, clasificarlas según las características del Polo de Imágenes.
- ✓ Proponer el uso de herramientas que sean software libre, y factibles para llevar a cabo de forma eficiente y sin riesgos el proceso de GCS.
- ✓ Analizar el posible impacto que tendrá la implantación de estas herramientas en el desarrollo de aplicaciones en el Polo de Imágenes.
- ✓ Presentar una estrategia como propuesta a realizar en el Polo de Imágenes para llevar a cabo el proceso de Gestión de la Configuración de Software.
- ✓ Llevar a cabo la implantación de las herramientas, métodos y técnicas propuestas, en el Polo de Imágenes.

La tesis se encuentra estructurada en tres capítulos, donde el Capítulo 1, FUNDAMENTACIÓN TEÓRICA, está referido al marco teórico de la investigación. En dicho capítulo se realiza un análisis crítico y valorativo del estado del arte en el tema de GCS. En el Capítulo 2, SOLUCIÓN PROPUESTA PARA LA GCS EN EL POLO DE IMÁGENES, se propone una estrategia para llevar a cabo el proceso de GCS en el Polo de Imágenes. Dando paso al Capítulo 3 APLICACIÓN DE LA SOLUCIÓN PROPUESTA, en el cual tiene lugar la aplicación de dicha estrategia propuesta específicamente en el proyecto alasRIS, perteneciente al Polo de Imágenes. Se describen algunos resultados observados.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los fundamentos teóricos que sustentan la investigación. Se profundiza en las diversas definiciones y valoraciones sobre la GCS[1] en el mundo, haciendo referencia al estado del arte tanto nacional como internacional. Se trata la importancia del desarrollo de un producto software; éste a lo largo de todo su ciclo de vida transita por diferentes etapas, en las cuales como paso fundamental es llevar a cabo correctamente el proceso de GCS.

En este contexto, se presentan los principales conceptos a tener en cuenta, así como las tareas a desarrollar internamente dentro de este prolongado proceso. Se analizan algunas herramientas para el soporte a las actividades de GCS, y la integración de algunas de estas con los Entornos Integrados de Desarrollo, dichas herramientas sustentan la base para la propuesta de métodos y herramientas libres para llevar a cabo el proceso de GCS en el Polo de Imágenes.

1.1 Gestión de la Configuración de Software. Definición

La Gestión de Configuración de Software es la disciplina de la Ingeniería de Software que comprende las herramientas y técnicas (procesos o metodologías) que una organización utiliza para administrar las configuraciones de los componentes de software. Tiene como objetivo mantener la integridad de los componentes del producto de software, evaluar y controlar los cambios sobre ellos así como facilitar la visibilidad del producto a todo el equipo de proyecto.

En el desarrollo de software los cambios son inevitables como bien expresa la 1ª Ley de la Ingeniería de Sistemas: “Sin importar en qué momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a largo de todo el ciclo de vida”.(Pérez, 2005) Generalmente

[1] Gestión de Configuración de Software

todo proceso de desarrollo de software incluye un equipo de trabajo formado por varias personas, por lo que los cambios pueden estar dados por modificaciones o inconsistencias encontradas por una de estas o por un deseo del cliente. Por esta razón se hace preciso llevar a cabo un control y registro de todos los cambios efectuados, para ello la GCS está encaminada a facilitar la trazabilidad del producto tanto hacia adelante como hacia atrás. Se traza como meta, aumentar la calidad y la productividad del producto final, reduciendo al mínimo los errores que podrían provocar confusión e incertidumbre con el fin de evitar problemas que puedan acarrear una incorrecta sincronización y afectar a otros elementos del sistema o tareas realizadas por otros integrantes del proyecto.

En términos sencillos la autora Angélica de Antonio define la GCS como una: "… disciplina, cuya misión es controlar la evolución de un sistema software"(Antonio, 2001). Dicha definición a pesar de ser tan simple con pocas palabras abarca todo en cuanto a GCS se trata.

La GCS describe la estructura del producto e identifica los elementos que lo constituyen y que son tratados como entidades que pueden ser puestas bajo control de versiones en el proceso de GCS. Tiene que ver con la definición de la configuración así como la construcción, el etiquetado y recolección de versiones de los artefactos. A pesar de lo importante que es concentrase en el control de cambios y de versiones, se hace imprescindible llevar a cabo otras tareas; que no se tienen en cuenta en la anterior definición y que están encaminadas a una sólida transparencia en la gestión del proyecto.

Otra de las fuentes ineludibles para diseñar el ambiente de gestión de la calidad, proviene de la serie de normas del Instituto de Ingenieros Eléctricos (IEEE), que incluye definiciones y estándares de todos los términos de la Ingeniería de Software, entre los que se encuentran muchos relacionados con la GCS. Una de las definiciones más utilizada es la que establece: "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. (Appleton, 2000)(Ailyn Febles Estrada, 2005)

No es menos cierto que la GCS enmarca un conjunto de actividades diseñadas para administrar y controlar el cambio o el estado de la información que se encuentra compartida en un proyecto. Comprende factores como la identificación de la configuración, el control de la configuración, el estado de la contabilidad, gestión de la construcción, gestión de procesos y el trabajo en equipo. Se traza como propósito fundamental establecer y mantener la integridad teniendo en cuenta el control en los productos software a lo largo del ciclo de vida del proyecto. Sin embargo esta definición se considera incompleta, o sea no tiene en cuenta otros elementos que constituyen gran importancia para una definición de GCS, por ejemplo el control de procesos, así como el control de los esfuerzos realizados por los desarrolladores y la constancia en registros e informes.

En la Quinta Edición del libro Ingeniería de Software Un enfoque práctico, Roger Pressman brinda una definición más completa y concisa a tener en cuenta en el presente trabajo:"La gestión de configuración del software (GCS) es una actividad de autoprotección que se aplica durante el proceso del software. Como el cambio se puede producir en cualquier momento, las actividades de GCS sirven para identificar el cambio, controlar el cambio, garantizar que el cambio se implementa adecuadamente e informar del cambio a todos aquellos que puedan estar interesados". (Pressman, 2005)

La GCS es concebida como un elemento importante del proceso de desarrollo de software y de garantía en la calidad del sistema. Un buen proceso de GCS posibilita en gran medida a los equipos de desarrolladores, trabajar juntos en un proyecto de manera eficaz. Visto desde otras perspectivas la GCS garantiza en gran medida que no se realicen cambios incontrolados, posibilitando además que cada participante del proyecto en el desarrollo de un sistema disponga de la versión adecuada del producto que se maneja.

La Gestión de Configuración del Software implica la identificación de la configuración del software en puntos dados en el tiempo, el control sistemático de los cambios en la configuración y el mantenimiento de la integridad y trazabilidad de la configuración a través del ciclo de vida del software.(2007)

La GCS facilita el mantenimiento del sistema, aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un cambio, tanto evolutivo como

correctivo. Asimismo, permite controlar el sistema como producto global a lo largo de su desarrollo, obtener informes sobre el estado de desarrollo en que se encuentra y reducir el número de errores de adaptación del sistema, lo que se traduce en un aumento de calidad del producto, de la satisfacción del cliente y, en consecuencia, de mejora de la organización.()

En términos de mantener el equilibrio y la productividad en el desarrollo de un producto software no se debe confundir GCS con mantenimiento. Aunque pueda parecer común no lo es. Mantenimiento es el conjunto de actividades que comienzan una vez que el software se encuentra instalado en el cliente y está operativo. Las actividades de GCS se ejecutan desde el inicio hasta el fin del proyecto. Sin embargo la GCS se relaciona fuertemente con el problema de mantenimiento ya que sin una buena GCS, el mantenimiento de un software se puede convertir en una verdadera pesadilla.

1.2 Configuración de Software

Los elementos que componen toda la información producida como parte del proceso de ingeniería de software se denominan colectivamente C*onfiguración del Software*.(Pressman, 2005). Se puede separar en tres amplias categorías:

- Programas (tanto código fuente como ejecutable)
- Documentos que describen los programas (tanto técnicos como de usuarios)
- Estructuras de datos (contenidas en el programa o externas al mismo)

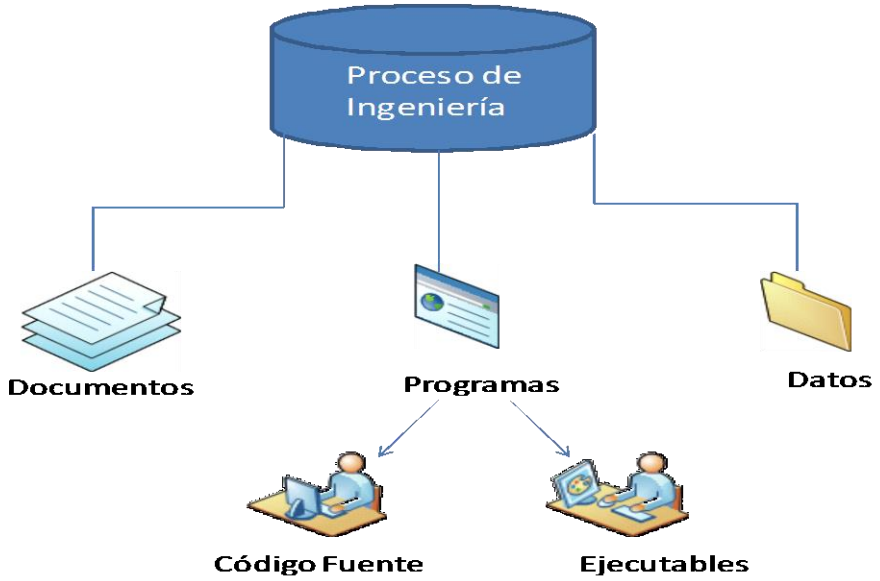


Fig 1.1 Configuración de Software

La configuración del software es el conjunto de todos aquellos elementos de la configuración que se producen durante el desarrollo. Los requisitos, diseño, e implementación que definen una versión particular de un sistema. A cada uno de los componentes de la configuración del software se le va a llamar Elemento de Configuración del Software (ECS). El ECS es la unidad de trabajo para la GCS. (Antonio, 2001)

Un elemento de configuración es cualquier producto de trabajo, tanto producto final como productos intermedios y tanto productos entregables al cliente como productos internos del proyecto, cuyo cambio pueda resultar crítico para el buen desarrollo del proyecto. (ITIL, 2008)

En los inicios de la ingeniería de software los ECS eran documentos de papel que se almacenaban físicamente. Esto trajo consigo muchos problemas dados por la dificultad de encontrar un elemento y en ocasiones no saber realmente cuál elemento había sufrido cambio, cuándo y por quién; la construcción de nuevas versiones consumía mucho tiempo y era proclive al error. En la actualidad, una buena práctica es conservar los ECS en una base de datos o depósito del proyecto ya que a medida que progresa el proceso de ingeniería de software el número de ECS crece rápidamente.

Un ECS debe ser un elemento que se pueda definir y controlar de forma separada, o sea, debe ser una unidad en sí mismo. Pressman considera un ECS como una sección individual de una gran especificación o cada caso de prueba de un gran conjunto de pruebas. De forma más realista, un ECS es un documento, un conjunto completo de casos de prueba o un componente de un programa dado.

Los ECS se organizan como objetos de configuración que han de ser catalogados en la base de datos del proyecto con un nombre único. Se pueden considerar como Elementos de Configuración del Software los siguientes componentes:

1. La especificación del sistema.
2. El plan del proyecto software.
3. La especificación de requisitos software.
4. Un prototipo, ejecutable o en papel.
5. El diseño preliminar.
6. El diseño detallado.
7. El código fuente.
8. Programas ejecutables.
9. El manual de usuario.
10. El manual de operación e instalación.
11. El plan de pruebas.
12. Los casos de prueba ejecutados y los resultados registrados.
13. Los estándares y procedimientos de ingeniería de software utilizados.
14, Los informes de problemas.
15. Las peticiones de mantenimiento.
16. Los productos hardware y software utilizados durante el desarrollo.
17. La documentación y manuales de los productos hardware y software utilizados durante el desarrollo.
18. Diseños de bases de datos.
19. Contenidos de bases de datos.

Los elementos citados con anterioridad pueden servir como ejemplo, aunque para cada proyecto en concreto es muy importante determinar: ¿qué se va a considerar como un ECS? En algunos casos existen organizaciones que toman como ECS los productos de hardware y software empleados durante el desarrollo bajo control de la configuración. Como estos productos son necesarios para desarrollar el software, deberán estar disponibles cuando se realicen cambios sobre su configuración.

1.3 Beneficios de la GCS

La necesidad de la Gestión de Configuración de Software está dada por el incremento de la complejidad de los sistemas de software y la integración entre diversas tecnologías debido a la dinámica del mercado. Teniendo en cuenta también la naturaleza cambiante del software, el nivel de personalización y los cambios en las reglas del negocio. Actualmente la dependencia de los sistemas de software en las empresas incrementa la presión para actualizar los mismos, lo cual conlleva a un incremento en la demanda. Es por ello que cada día se aboga más por la optimización en cuanto a tiempo y organización en el desarrollo de productos software, donde entra a jugar un papel fundamental la GCS.

Constituyendo esta una disciplina de la Ingeniería de Software que se preocupa de: identificar y documentar las características funcionales y físicas de los elementos de configuración, controlar los cambios a tales características y reportar el proceso de tales cambios y su estado de implantación.(Acevedo, 2004) Esta disciplina provee a las empresas dedicadas a la producción de software de los siguientes beneficios:

- Asegura la correcta configuración del software.
- Asegura que se construya el sistema correcto. Las auditorías se aseguran de que el software a entregar sea funcional y físicamente el que se espera.
- Proporciona la capacidad de controlar los cambios.
- Mejora la productividad de desarrollo de software. Contribuyendo a una mejora en la comunicación entre los integrantes de un equipo de desarrollo, lo cual evita duplicar esfuerzos.
- Disminuye los sobreesfuerzos causados por los problemas de integridad.
- Reduce los defectos ya que ayuda a identificar de manera precisa la versión sobre la que se realizarán los cambios.

- Agiliza la identificación de problemas y corrección de errores, de modo que mantiene el historial de problemas y cómo fueron resueltos.
- Garantiza que todo el equipo trabaja sobre una misma línea base de productos.
- Ayuda al proceso de desarrollo y por consiguiente se reduce la dependencia de las personas si se tiene por escrito todo el proceso realizado.
- Disminuye el costo de mantenimiento
- Mejora el aseguramiento de calidad. La información de la GCS ayuda a determinar las causas de los problemas y así evitar que ocurran nuevamente.

Una implementación efectiva de la GCS, permite un control eficiente del proceso de desarrollo durante todo el ciclo de vida del software mientras proporciona la máxima flexibilidad en la gestión del cambio de los productos y evoluciones del mismo. Para que el proceso sea efectivo y flexible es importante que su implantación forme parte del proceso de desarrollo software.

1.4 Proceso de Gestión de Configuración de Software

Como todo proceso, la GCS también puede ser sistematizada y automatizada. Un sistema de gestión de configuración incluye el sistema de almacenamiento, los procedimientos y las herramientas para acceder al sistema de gestión de configuración.

La GCS implica la identificación de la configuración del software en puntos dados en el tiempo, el control sistemático de los cambios en la configuración y el mantenimiento de la integridad y trazabilidad de la configuración a través del ciclo de vida del software. Acerca del extenso proceso de GCS pueden surgir diversas interrogantes, Pressman define algunas:

¿Cómo se identifican y gestionan las distintas versiones de un programa (y su documentación) de forma que se introduzcan cambios eficientemente?
¿Cómo controlar los cambios antes y después de distribuir el software al cliente?
¿Quién es responsable de aprobar y de asignar prioridades a los cambios?
¿Cómo garantizar que los cambios se han llevado a cabo adecuadamente?

¿Qué mecanismo se usa para avisar a otros de los cambios realizados?

Es de vital importancia plantearse el conjunto de cuestiones expuestas anteriormente ya que dan paso a identificar las actividades que engloban el extenso proceso de GCS, así como su trascendencia dentro del proceso de desarrollo de software. Si bien existen diversas herramientas que pueden hacer el proceso más sencillo, hay ciertas pautas a tener en cuenta para la GCS. Basado en la investigación realizada se encontraron diversos criterios de diferentes autores asociados a la GCS (Antonio, 2001), (Pressman, 2005),(Pablo, 2002), (IEEE, 1990) y por lo general todos coinciden en las siguientes actividades:

- Identificación de la Configuración
- Control de Cambios en la Configuración
- Generación de Informes de Estado
- Auditoría de la Configuración

Sin lugar a dudas este conjunto de actividades han sido definidas de una forma bastante homogénea, quedando claro que el éxito consiste en tener bien identificados cuales son los elementos que formarán parte del software y controlar los cambios que puedan sufrir los mismos. Sin embargo no se tiene en cuenta que a partir de estos cambios se pueden generar diferentes versiones, constituyendo un aspecto fundamental que resulta ser referido indistintamente por algunos autores, pero no lo incluyen como un proceso independiente.

Consecuentemente para evitar el caos lo más inteligente es llevar, en conjunto con las demás actividades, un control de las diferentes versiones que se van generando en un proyecto. Luego todos y cada uno de los integrantes de un equipo de desarrollo podrán trabajar sobre su propia versión y una vez terminada adquirir una versión común del producto.

Por otra parte Pressman define cinco importantes actividades; usualmente con mayor utilidad debido a la precisión y amplitud con que abarca cada rama incluida en el proceso de GCS e incluye el Control de Versiones como una tarea más del proceso de GCS. Por tal motivo resulta como propuesta elegida en la actual investigación las tareas definidas por el autor:

- Identificación de objetos en la Gestión de Configuración del Software.
- Control de Versiones
- Control de Cambios
- Auditoría de la Configuración
- Informes de Estado

Teniendo como preferencia acotar las actividades de la GCS descritas por Pressman, se describe además cómo se ejecutarán dichas actividades a realizar durante el desarrollo de un producto software.

1.4.1 Identificación de objetos en la GCS

La primera e insustituible etapa en el desarrollo de un software lo constituye la fase de identificación, que es un momento de gestación. En otras palabras el proceso de identificación de objetos o como también se le suele denominar IEC[2] de un software, es la cuna de su desarrollo, y está destinado a construir las bases del proyecto.

Lo primero y más importante antes de encaminarse en el desarrollo de un producto software es identificar cada uno de los elementos de configuración a gestionar y controlar durante el ciclo de vida del sistema. Gran variedad de autores concuerdan con que una de las tareas básicas que forma parte de la GCS es la Identificación de la Configuración de Software (Antonio, 2001), (Pressman, 2005), (IEEE, 1990). Se puede catalogar como la más importante porque permite identificar la estructura que va a tener un producto software, los elementos de configuración que se controlarán, las líneas base del proyecto, así como los componentes que contendrá especificándose su tipo y haciéndolos únicos.

Una identificación de configuración eficaz es un prerrequisito para las restantes áreas o actividades de gestión de configuración porque todas utilizan los productos de la identificación de configuración. Si los elementos de configuración y su documentación de configuración asociada no están adecuadamente

[2] Identificación de los Elementos de Configuración.

identificados, es imposible controlar los cambios a los elementos de configuración, establecer registros e informes precisos y exactos, o validar la configuración mediante la auditoría.(Agustín)

Esta tarea consiste en identificar y asignar nombres significativos y consistentes a todos y cada uno de los elementos que forman parte del producto software, en cada fase de su desarrollo, es decir, a cada uno de los Elementos de Configuración del Software.(Antonio, 2001)

Muchos objetos son producidos en el curso del desarrollo del software, por lo que en la actividad de identificar los objetos en la GCS una forma de organizarlos es clasificándolos: están los *objetos básicos* que es una unidad de texto que va a ser creada por un ingeniero de software durante cualquier fase como, análisis, diseño, codificación o pruebas. Una sección de una especificación de requisitos puede constituir un ejemplo de objeto básico, como lo puede ser también un listado fuente de un módulo o un conjunto de casos de prueba. También pueden ser *objetos compuestos* que sería una colección de objetos básicos y de otros objetos compuestos. La Especificación de Diseño puede ser un ejemplo de los mismos. (Pressman, 2005)

El código fuente está lejos de ser el único tipo de objetos que podrían ser puestos bajo GCS, aunque tradicionalmente es el tipo más común. Los objetos pueden pertenecer al producto en sí o solamente para apoyar su desarrollo y mantenimiento. Es por ello que también pueden ser clasificados en *objetos físicos* por ejemplo elementos de hardware como las PC que puedan ser necesarias para llevar a cabo el desarrollo o los documentos e informes que se generan. Por otra parte se encuentran los *objetos electrónicos* que tienen una significativa importancia constituyendo así el eje central del producto, el código fuente. Teniendo en cuenta también los documentos digitales que se crean durante el ciclo de vida.

La autora Angélica de Antonio define algunas tareas que se deben efectuar para llevar acabo esta importante actividad:(Antonio, 2001)

- Establecimiento de una jerarquía preliminar del producto 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.

A continuación se describen en detalle cada una de estas actividades.

Establecimiento de una jerarquía preliminar del producto software

No es más que obtener una primera visión de la estructura que tendrá el sistema y los elementos que compondrán al mismo. En los primeros compases del proyecto se realiza de forma opcional aunque es muy importante ya que definir esta jerarquía constituye la base preliminar para otras actividades posteriores incluidas en la GCS.

Selección de los Elementos de Configuración.

En este contexto es de significativa importancia conocer que para controlar y manejar los ECS, cada uno tiene que ser nombrado separadamente y entonces organizarlos. El líder del proyecto debe ser responsable de identificar todos los ECS. Luego se le deben asignar etiquetas que los identifiquen como únicos recayendo esta responsabilidad sobre el Administrador de la Configuración. Seleccionar pocos ECS puede ocasionar la falta de suficiente visibilidad sobre el producto, es por ello que a la hora de seleccionar los ECS lo más aconsejable es separar en elementos de configuración distintos, las diferentes funcionalidades, lo que propiciaría minimizar el impacto de los cambios.

Por otro lado tener demasiados ECS puede provocar un número excesivo de especificaciones y documentos que pueden resultar inmanejables. De modo que se debe ser preciso en la selección adecuada de los ECS que se necesitarán en el proyecto. Para ello se pueden tener en cuenta los siguientes criterios:

Utilización múltiple: Número de elementos de su mismo nivel o niveles superiores que lo utilizan.

Criticidad: Gravedad del impacto de un fallo en dicho componente.

Número de personas implicadas en su mantenimiento.

Complejidad de su interfaz: Las interfaces de un ECS con otros deberían ser simples. Hay que minimizar el acoplamiento entre ECS.

Singularidad del componente con respecto al resto.

Reutilización: Si el componente se va a diseñar especialmente para ser reutilizado.

Tipo de tecnología: Si el componente incorpora nuevas tecnologías no utilizadas en otros componentes.

Definición de las relaciones en la configuración.

Los ECS constituyen objetos y como tal estos pueden estar relacionados con respecto al resto, y de esa forma ayudará a comprender donde se encuentra situado cada ECS dentro del proceso de desarrollo. De modo que ha continuación se muestran algunos ejemplos de relaciones existentes entre objetos.

- Equivalencia: Cuando un ECS se encuentra almacenado en lugares diferentes (disco maestro, copia de seguridad o en un diskette) pero todas las copias corresponden al mismo programa.
- Composición: Un ECS como "Especificación de diseño" está compuesto por otros ECS como el "Modelo de Datos" o el Diseño de modulo".
- Dependencia: Cualquier otro tipo de relaciones entre ECS.
- Derivación: A partir de qué se ha originado algo como por ejemplo el código objeto del código fuente.
- Sucesión: Los cambios desde una revisión a otra, donde puede ser muy útil un Grafo de Evolución. "el grafo de evolución describe la historia de los cambios de un objeto" (Pressman, 2005)
- Variante: Variación sobre un determinado elemento, con la misma funcionalidad.

Este conjunto de interrelaciones se pueden representar mediante un LIM[3] que describe las interdependencias entre los objetos de configuración y que permite construir automáticamente cualquier versión del sistema.

Definición de un Esquema de Identificación.

Una función importante en la actividad de identificar los objetos de la configuración es establecer un Esquema de Identificación con el objetivo de etiquetar cada ECS, siendo el líder de configuración el responsable. Dicho esquema debe proporcionar información útil como:

- Número o código del ECS.
- Nombre

[3] Lenguaje de Interconexión de Módulos

- Descripción
- Autor
- Fecha de creación
- Identificación del proyecto al que pertenece el ECS.
- Identificación de la línea base a la que pertenece.
- Tipo de ECS (documento, programa, elemento físico, etc.).
- Localización

Definición y Establecimiento de Líneas Base.

Una vez identificados los productos que estarán bajo GCS, habrá que incluirlos bajo una línea base para que a partir de ese momento, cualquier modificación que se vaya a realizar sobre dicho producto tenga que seguir los procedimientos diseñados para ello. Como concepto clave se hace necesario tener presente que es una Línea Base:

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. [Estándar IEEE 610.12-1990]. Para que un ECS pase a formar parte de la línea base no sólo tiene que estar identificado como elemento a incluir en la configuración del software sino que tendrá que cumplir con las condiciones mínimas, es decir que el producto esté acabado y haya sido formalmente aprobado.

Uno de los primeros pasos para poder efectuar la actividad de Identificación de la Configuración, por lo tanto, consiste en definir cuáles van a ser los hitos, dentro del proceso de desarrollo, en los que se va a establecer una línea base. Se puede definir una línea base como un punto de referencia en el proceso de desarrollo del software que queda marcado por la aprobación de uno o varios ECS, mediante una revisión técnica formal.(Antonio, 2001)

Las líneas base deben ser establecidas al concluir determinadas fases del ciclo de vida con el objetivo de identificar los resultados de las tareas realizadas durante la fase y asegurar que se ha completado

correctamente la fase. La línea base constituye la base oficial para el trabajo subsiguiente. Existen diversos tipos de línea base como por ejemplo:

- Línea Base Funcional: Al final de la fase de análisis del proyecto se deben haber definido el Plan de análisis, los requerimientos del sistema, el Plan de calidad, el Plan de GCS, Plan de pruebas.

- Línea Base de Distribución o Asignación de funciones: Con la culminación de la fase de diseño es importante la arquitectura del sistema, las interfaces de subsistema, el Plan de pruebas de sistema.

- Línea Base de Diseño Preliminar: Se requiere un diseño detallado, así como el Diseño de subsistemas y el Plan de pruebas de integración.

- Línea Base de Producto: Al final de la codificación del producto se deben tener como ECS el Código fuente, objeto y ejecutable, los resultados de pruebas de integración y versión preliminar de los manuales.
- Línea Base de Operación: Con la implantación del sistema pues se esperan los resultados de pruebas de sistema y la documentación de usuario.

Fig. 1.2 Líneas base

Las líneas base constituyen el elemento central siendo a su vez la parte fundamental del desarrollo de software. Las mismas marcan el final de cada fase del ciclo de vida del software. Estas pueden estar establecidas de dos formas: *físicamente (*etiquetando cada ECS y almacenándolos en un archivo del proyecto) o *lógicamente* (publicando un documento de identificación de la configuración, que identifica el estado actual del producto en dicho punto del proceso de desarrollo).

Definición y Establecimiento de bibliotecas de software.
El desarrollo, uso y mantenimiento de software requiere estar soportado por las bibliotecas que van a contener cada proyecto en particular. Una biblioteca de software es una colección controlada de software y/o documentación relacionada cuyo objetivo es ayudar en el desarrollo, uso o mantenimiento del software.(Antonio, 2001)

Las bibliotecas están encaminadas a viabilizar las tareas de GCS, como el engorroso proceso de control de cambios y la contabilidad de estado. La autora Angélica de Antonio define diferentes tipos de bibliotecas de software:(Antonio, 2001)

- Biblioteca de trabajo: La cual se establece al inicio del proyecto, comprende el área de trabajo donde los analistas y diseñadores elaboran los documentos del proyecto y donde los programadores desarrollan el software. Cuando se han completado las revisiones o pruebas y el elemento de configuración en cuestión ha sido revisado y aprobado, se inicia la transferencia del ECS a la Biblioteca de Soporte al Proyecto. En esta biblioteca el control de cambios es informal.

- Biblioteca de Integración: Es de esta biblioteca de donde se toman los ECS para su integración en ECS de nivel superior del sistema.

- Biblioteca de Soporte al Proyecto: Su función es almacenar los ECS aprobados y transferidos desde la Biblioteca de Trabajo o desde la Biblioteca de Integración. Si un ECS pasa a formar parte de esta biblioteca se encuentra sujeto a un control de cambios interno y semiformal.

- Biblioteca de Producción: Compuesta por la Biblioteca de trabajo, la de integración y la Biblioteca de Soporte al Proyecto.

- Biblioteca Maestra: En ella se almacenan los ECS liberados para su entrega al cliente o distribución en el mercado. Los ECS en ella se encuentran sujetos a un control de cambios formal y estricto. Normalmente esta biblioteca tiene fuertes restricciones de acceso para escritura, aunque no los tiene para lectura. Se almacenan los Releases del sistema.

- Repositorio de Software: Entidad en la que se archivan los ECS de un proyecto tras su cierre. Se transfieren a él desde la Biblioteca Maestra. Opcionalmente se puede identificar un segmento especial en el que se almacenarán los elementos reutilizables. Todo lo que se almacena en el repositorio debe estar adecuadamente identificado y catalogado, para facilitar su recuperación en caso de necesidad. Se supone que es un almacenamiento a largo plazo, por lo que puede ser de recuperación lenta. Es central y común a todos los proyectos, mientras que la biblioteca de Producción y la Maestra son individuales para cada proyecto.

- Biblioteca de Backup: También debe estar adecuadamente identificada, aunque su contenido no está sujeto a Gestión de Configuración (las copias contenidas en ella no están catalogadas en los registros de Gestión de Configuración).

  Resalta como conclusión que a la hora de realizar la actividad de identificación de objetos en la GCS se debe prestar gran importancia en la selección de los ECS. Si se omite algún elemento significativo se pueden producir graves errores. La asignación adecuada de los atributos a cada elemento también es un paso importante, así como la identificación de las relaciones entre los ECS. Tener todos estos pasos en claro brinda una mayor información y visualiza las modificaciones que pueda sufrir algún ECS durante todo el desarrollo del software.

1.4.2 Control de Versiones

Partiendo de la definición de Pressman, el cual considera que el "control de versiones combina procedimientos y herramientas para gestionar las versiones de los objetos de configuración creados durante el proceso del software" (Pressman, 2005) se puede entender que el control de versiones consiste en la administración de múltiples revisiones de una misma unidad de información. Pressman cita a Clemm en su descripción sobre el control de versiones:

"La gestión de configuración permite a un usuario especificar configuraciones alternativas del sistema de software mediante la selección de las versiones adecuadas. Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar y construir una configuración describiendo el conjunto de atributos deseados".

Cuando el autor emplea el término atributo, se refiere a especificar un número para cada versión asociado a cada objeto. La representación de las diferentes versiones de un sistema pueden ser graficadas mediante un grafo de evolución. Cada nodo del grafo representa un objeto compuesto o lo mismo una versión completa del software. Las versiones están conformadas por una colección de ECS (por ejemplo código fuente, documentos, datos) que estas a su vez pueden estar compuestas por diferentes variantes.

Se define **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.(Antonio, 2001)

Las distintas versiones que van apareciendo en el tiempo, según se va avanzando en el desarrollo de un elemento, se les suele llamar también **Revisiones Técnicas Formales (RTF).** Tienen como objetivo conseguir un trabajo técnico de una calidad uniforme detectando errores durante el proceso, con el fin de hacer más manejable el trabajo técnico, señalar las necesidades de mejora, verificar que durante el proceso se están alcanzando los requisitos, los atributos de calidad, el uso de estándares definidos, evaluando además, los artefactos generados por cada disciplina o flujo de trabajo que una vez aprobado formarán parte de la línea base del proyecto.

La Gestión de Configuración debe permitir también especificar y gestionar distintas **Variantes** de los elementos de configuración. Las variantes son versiones de un elemento de configuración que coexisten en un determinado momento y que se diferencian entre sí en ciertas características. (Antonio, 2001)

Siempre que exista un equipo de desarrollo con varios integrantes cualquier modificación, revisión o cambio mínimo conlleva a disponer de diferentes versiones, que en un entorno determinado pueden derivar grandes conflictos. Para evitar estos conflictos generalmente se emplean los controladores de versiones o en algunas literaturas le suelen denominar sistemas de control de versiones. Estos facilitan a los usuarios la administración de los ECS y el saber en cada momento en qué fase se encuentran. Se encargan de identificar, comparar y revertir cambios en la información, permitiendo el manejo de múltiples revisiones.

Los sistemas de control de versiones se suelen integrar a los entornos de desarrollo y realizan administración de versiones del código fuente. Cada modificación de uno de los archivos del programa va generando una revisión del mismo, y periódicamente se crean líneas base de todo el proyecto. De este modo, un equipo de desarrollo puede trabajar en paralelo, compartiendo versiones de archivos de código fuente y actualizándolos periódicamente según se van creando o modificando los archivos que conforman el proyecto.(2007)

La actividad más corriente del control de versiones está basada en controlar las distintas versiones que se van generando del código fuente, pero esto no quiere decir que no sea aplicable a otros ámbitos como por ejemplo documentos, archivos de texto, imágenes, o sea todo lo que constituya un ECS. En dependencia de la herramienta que se determine emplear en un proyecto se obtendrá de la misma un registro histórico de todas las acciones realizadas sobre cada ECS o conjunto de ellos, normalmente pudiendo extraer o volver a un estado anterior del producto.

Los sistemas de control de versiones emplean para su funcionamiento algún mecanismo de almacenamiento de los datos y la información asociada (metadatos). Esta base de datos o “almacén”, suele denominarse **Repositorio** (servidor que guarda el conjunto de información gestionada por el sistema). Este repositorio contendrá el historial de versiones de todos los elementos gestionados.

La modificación en una versión, por pequeña que sea, conduce a la creación de una nueva versión; esto trae como consecuencia que el número de revisiones se incremente en mayor medida hasta llegar al punto en que se haga imposible almacenarlas todas. Por tanto se deja a consideración del usuario que decida cuando crear una nueva versión. Los desarrolladores pueden modificar un objeto tantas veces como lo necesite y puede mantener el mismo identificador de versión, él es quien decide luego cuando debe ser congelada. A partir de ese momento se almacena la versión y no podrá ser modificada, para realizarle cambios entonces se debe crear una nueva versión. La forma en que se actualiza un ECS es la siguiente:

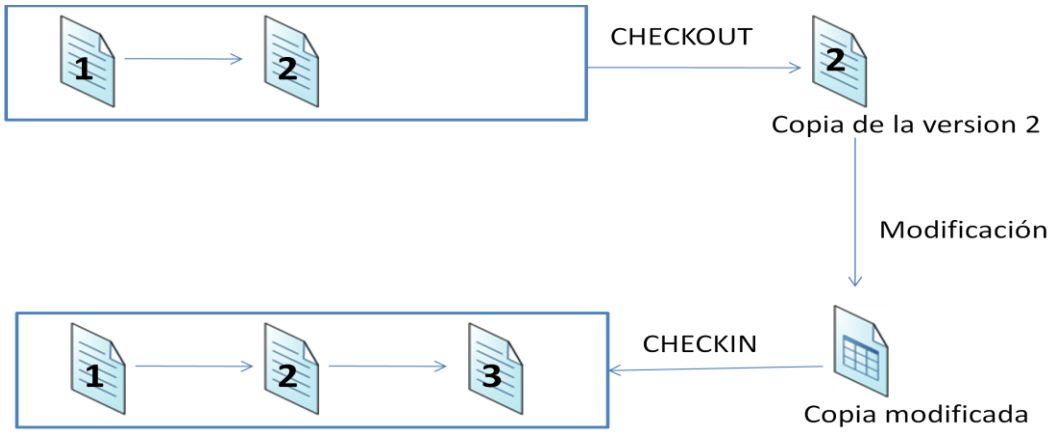


Fig. 1.3 Control de versiones de un elemento de configuración.

Las revisiones permanecen almacenadas en el repositorio gestionado por la herramienta. Cuando un desarrollador necesita trabajar en alguno de los objetos encontrados en el repositorio, accede a ellos mediante la operación de CHECKOUT, de esta forma solicita una copia de la última versión guardada para descargarla en un directorio de trabajo local (puesto de trabajo del desarrollador) con el objetivo de trabajar sobre la versión del objeto y realizarle modificaciones.

La copia solicitada por el desarrollador a partir de ese momento queda fuera del alcance de la herramienta, lo que permite que el usuario pueda trabajar libremente sobre ella, o sea hacerle todas las modificaciones. Luego una vez terminado el cambio, se inserta nuevamente en el repositorio, mediante la

operación CHECKIN, que almacenará la copia como una nueva versión sucesora de la que se había extraído anteriormente.

1.4.3 Control de Cambios

El control de cambios de la configuración implica el control de las liberaciones y de los cambios a los productos software durante todo el ciclo de vida del producto. “Es quizás la parte más visible de la gestión de configuración. Es el proceso de gestionar la preparación, la justificación, la evaluación, la coordinación, la disposición y la implementación de los cambios y desviaciones de ingeniería propuestos a los elementos de configuración afectados que se encuentran en la documentación de la línea base actual”.(Agustín)

En un gran proyecto de Ingeniería de Software, el cambio incontrolado lleva rápidamente al caos. Para estos proyectos, el control de cambios combina los procedimientos humanos y las herramientas automáticas para proporcionar un mecanismo para el control del cambio. (Pressman, 2005). Pressman también cita a James Bach el cual considera que:

“El control de cambio es vital. Pero las fuerzas que lo hacen necesario también lo hacen molesto. Nos preocupamos por el cambio porque una diminuta perturbación en el código puede crear un gran fallo en el producto. Pero también puede reparar un gran fallo o habilitar excelentes capacidades nuevas. Nos preocupamos por el cambio porque un desarrollador pícaro puede hacer fracasar el proyecto; sin embargo las brillantes ideas nacidas en la mente de estos pícaros, y un pesado proceso de control de cambio pueden disuadirle de hacer un trabajo creativo”.

Existen varios niveles de control de cambios. Cuando un ECS aún no ha pasado a formar parte de una línea base el desarrollador puede efectuar cualquier cambio justificado sobre el mismo, por tanto se le denomina Control de Cambios **Informal** porque todavía no se ha tenido en cuenta en la línea base del proyecto. Este proceso no conlleva a realizar gran cantidad de acciones, solo se establece en las bibliotecas, espacios de trabajo o en ficheros individuales de cada desarrollador.

Sin embargo cuando un cambio implica afectar la línea base del proyecto y este aún no se encuentra en manos del cliente, o sea el desarrollo no ha concluido entonces el cambio se efectúa a **Nivel de Proyecto**. Si el desarrollador desea realizar algún cambio; como el ECS ya ha pasado por la Revisión Técnica Formal, pues debe recibir la aprobación del Líder del Proyecto si es un cambio local. En caso de tener el cambio algún impacto sobre los ECS debe ser aprobado por el CCC. [4]

Una vez que el producto se comienza a comercializar o se encuentra distribuido y se hace necesario realizar algún cambio, se lleva a cabo el Control de Cambios **Formal**. Producto de que los elementos de configuración ya están transferidos a la Biblioteca Maestra, de modo que cada cambio deberá ser aprobado por el CCC.

Todo proyecto que desee administrar adecuadamente los cambios al software, se debe asegurar de que los cambios que se propongan se evalúen cuidadosamente, que las personas indicadas tomen decisiones sobre esos cambios, que los cambios se comuniquen oportunamente a todos los afectados y que el proyecto incorpore los cambios de una forma disciplinada. Para conseguir esto es necesario contar con un proceso formal de cambios.

Precisamente esa es la meta del proceso de control de cambios en la configuración, establecer los mecanismos que ayudarán a asegurar la producción de un software de calidad. También garantiza que cada versión del software contenga todos los elementos necesarios, y que todos los elementos de una versión trabajen juntos correctamente. En la figura se muestra un proceso genérico de control de cambio.

[4] Comité de Control de Cambios: persona o grupo encargado de tomar las decisiones finales acerca del estado y la prioridad de las peticiones de cambio.

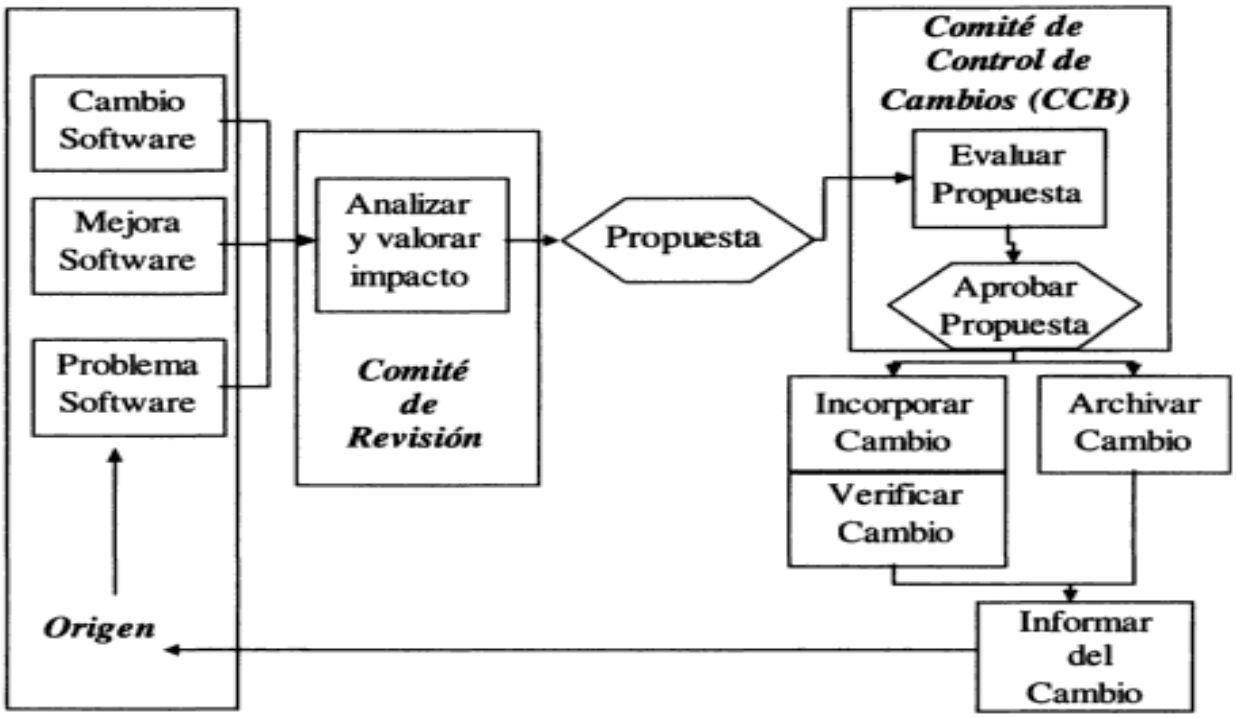


Fig 1.4 Proceso de control de cambio.

Generalmente el origen de la propuesta de cambio está dada por la necesidad de corregir un defecto, un cambio de software debido a cualquier circunstancia, una mejora para perfeccionar el software o simplemente incluirle alguna funcionalidad en vista producirle mejoras al sistema. Esta propuesta debe ser presentada al CCC a través de los mecanismos correspondientes para que un comité de revisión formado por varias personas en función de calcular la magnitud de la propuesta y que analice y valore algún impacto global sobre otras funciones del sistema o sobre otros objetos de la configuración.

Por lo tanto es de necesaria importancia estructurar formas o definir etapas típicas para llevar a cabo un proceso formal, o sea pasos a seguir para hacer un cambio sobre una línea base.

1. Iniciación del Cambio: Cuando se ha detectado un problema o un cambio en los requisitos pues se presenta una solicitud de cambio. Se presentan en un informe de cambios a la autoridad de control y de cambios (persona o grupo que toma la decisión final del estado y la prioridad del cambio).

2. Se clasifica y registra la solicitud de cambio.
3. La solicitud de cambio puede ser aprobada o rechazada, de ello se suele responsabilizar el Comité de Control de Cambios.
4. De ser aprobada la solicitud de cambio se analizan los posibles efectos secundarios y la repercusión que pueda tener sobre otras funciones del sistema, obteniendo como resultado un Informe de Cambio.
5. Dicho informe se presenta al Comité de Control de Cambio y si se considera beneficioso el cambio se genera una Orden de Cambio de Ingeniería (OCI) donde se describe en detalle el cambio a realizar. A partir de ese momento se da de baja en la Biblioteca de Soporte al Proyecto.
6. Se comienza a realizar el cambio dándole un seguimiento y control.
7. Una vez finalizado el cambio, este se certifica mediante una revisión verificando si se ha efectuado correctamente el cambio, si se ha corregido el problema detectado y si se han satisfecho los requisitos modificados. Luego se devuelve el objeto a la Biblioteca de Soporte al Proyecto.
8. Se notifica el resultado del cambio al originador.

Llevar un adecuado control de los cambios constituye la actividad de GCS de mayor importancia, tiene como objetivo proporcionar un mecanismo riguroso para controlar los cambios permitiendo a su vez mantener la integridad del producto. Una petición de cambio puede aparecer en cualquier momento durante el proceso de desarrollo del software y puede ser solicitada por todos los *miembros del proyecto.* Estos serán los encargados de realizar los cambios siempre y cuando informen el estado de los cambios que están realizando.

El *jefe de proyecto* tiene como responsabilidad asegurar que los procedimientos de Control de Cambios se efectúen correctamente. Será la persona encargada de informar el estado de los cambios y también puede participar en la evaluación de los mismos, determinando los impactos sobre los ECS, así como el coste y efecto sobre la programación del proyecto.

Los cambios no solo afectan el código fuente sino otra serie de elementos a tener en cuenta como:

- Requerimientos
- Especificaciones

- documentos de diseño
- código fuente
- código objeto
- planes de prueba
- casos de prueba, configuraciones de prueba y resultados
- herramientas de mantenimiento y desarrollo
- manuales de usuario
- manuales de mantenimiento

El Control de Cambios es un largo y cauteloso proceso, cada proyecto en su comienzo debe definir de forma precisa cuál será el proceso de gestión de cambios que va a adoptar. Para ello Angélica de Antonio define tres aspectos importantes que se pueden tomar como premisa.(Antonio, 2001)

- Definir políticas a nivel organizativo que promuevan las actividades de Control de Cambios y que reflejarán la filosofía y las metas de la organización en cuanto a las actividades de GCS.
- Los estándares que se van a adoptar y a los que será necesario ajustarse.
- Los procedimientos que se van a utilizar para poner en práctica las políticas de GCS.

Manejar correctamente los cambios en el proceso de GCS es lo más aconsejable. Todos los procedimientos que se realizan en el Control de Cambios garantizan que el proyecto se mantenga estable, ya que solamente un elemento podrá ser modificado por un único desarrollador. En caso de ser solicitado ese elemento que se encuentra prestado, para modificarse, pues el desarrollador deberá reservar el cambio para cuando este sea dado de alta nuevamente.

Se puede decir que tener una política y un procedimiento que proteja y sostenga el código fuente de las aplicaciones en funcionamiento, es lo mínimo necesario para avanzar consistentemente en la salida del caos a la productividad. Si se incorporan cambios al software de forma descontrolada, pues provocaría caos en los proyectos, retrasos en la entrega de los productos y problemas de calidad. Sin una adecuada gestión de cambios no se comunican oportunamente a todos los involucrados, ni se incorporan los cambios de una forma disciplinada. Por lo que se considera que puede resumirse en la siguiente frase: *“El*

arte del progreso está en mantener el orden en medio del cambio y mantener el cambio en medio del orden". [Alfred North Whitehead]

1.4.4 Auditorías a la Configuración

Una auditoría es una verificación independiente de un trabajo o del resultado de un trabajo o grupo de trabajos para evaluar su conformidad respecto especificaciones, estándares, acuerdos contractuales u otros criterios. La auditoría de la Configuración es la forma de comprobar que efectivamente el producto que se está construyendo es lo que pretende ser. (Antonio, 2001)

Las auditorías a la configuración consisten en validar la completitud del software, mantener la consistencia entre los elementos, así como asegurar que los cambios se han realizado correctamente. Las actividades del proceso de GCS que se han venido definiendo, identificación de los objetos de la GCS, el control de versiones y el control de cambios, tienen un sentido común, beneficiar al equipo de desarrollo de software a controlar los cambios y a conservar el orden, pero surge una interrogante. ¿Cómo asegurar que esos cambios se han implementado correctamente?

Pressman considera dos posibles respuestas: en primer lugar, la revisión técnica formal y en segundo lugar la auditoría de la configuración del software. La RTF se centra en la corrección técnica del ECS que ha sido modificado, sin embargo la Auditoría de Configuración del Software complementa la RTF al revisar características que no se tienen en cuenta en la revisión, ya que no es suficiente solo con evaluar y autorizar el cambio, sino que es preciso asegurarse de que se efectué de forma correcta.

Esta tarea es de vital importancia teniendo en cuenta que garantiza que los ECS auditados estén completos, que siguen las normas de GCS y que la línea base corresponda con la descripción de sus elementos. Dentro del proceso de GCS es la actividad más costosa debido a que requiere ser llevada a cabo por personal experimentado y calificado, con un amplio conocimiento del proceso y sobre todo debe ser personal ajeno al equipo de desarrollo técnico.

Debe ser efectuada de forma periódica en aras de asegurar la integridad de las líneas base y para ser llevada a cabo es preciso contener un plan documentado de GCS, ya que constituye la base para la auditoría. En cuanto a las auditorías, se suelen distinguir tres tipos de auditorías de configuración: (Antonio, 2001)

- Auditoría Funcional: Su objetivo es comprobar que se han completado todas las pruebas necesarias para el ECS auditado, y que, teniendo en cuenta los resultados obtenidos en dichas pruebas, se puede afirmar que el Elemento de Configuración satisface los requisitos que se impusieron sobre él.

- Auditoría Física: Se encarga de verificar la adecuación, completitud y precisión de la documentación que constituye las líneas base de diseño y de producto. Se trata de asegurar que representa el software que se ha codificado y probado. Tras la auditoría física se establece la línea base de producto. Tiene lugar inmediatamente después de haberse superado la auditoría Funcional.

- Revisión Formal de Certificación: Su objetivo es certificar que los ECS se comportan correctamente en su entorno operativo.

Requiere de disciplina la sistematicidad con que se ejecute esta actividad de auditar la configuración, es una labor que debe ser llevada a cabo durante todo el proceso de desarrollo; pues es cuando más se necesita para conocer realmente los errores existentes. No tendría sentido al finalizar el proyecto, ya cuando los problemas no tengan solución.

Para esta actividad se definen tres tipos de actividades:

- Revisiones de fase: Se realizan al finalizar cada fase del desarrollo teniendo como objetivo examinar los productos de dicha fase. Las revisiones propias de la GCS son aquellas en las que se establecerán las líneas base. El objetivo de esta revisión es descubrir problemas, no comprobar que todo está bien. Hay que ser capaz de desenmascarar los problemas ocultos y sutiles, no sólo los que son obvios.

- Revisiones de cambios: Se realizan para comprobar que los cambios aprobados sobre una línea base se han realizado correctamente.

- Auditorías: Se realizan al final del proceso de desarrollo de software y su objetivo es examinar el producto en su conjunto.

Con el objetivo de lograr el correcto comportamiento del producto y que el mismo se emplee satisfactoriamente en su entorno operativo, es imprescindible verificar que la configuración del software se corresponda en cada fase, o sea que exista correspondencia y trazabilidad entre los ECS y la línea base del proyecto. Es por ello la significativa atención que se le debe brindar a esta actividad.

1.4.5 Informes de Estado

La generación de Informes de Estado de la Configuración; también se suele denominar contabilidad de estado, se efectúa con intención de mantener a los gestores y profesionales al tanto de los cambios importantes o sea proporciona información sobre cada cambio a aquellos que deben estar informados.

Esta actividad implica el registro e información del proceso del cambio. Su meta es mantener un continuo registro del estado y la historia de todos los elementos de la línea base y de los cambios propuestos a ellos. Incluye los informes de traza de todos los cambios a la línea base durante todo el ciclo de vida del software.(Agustín)

Contribuye en el enriquecimiento de la comunicación entre los participantes en un proyecto. Esto se va a conseguir registrando toda la información necesaria acerca de lo que va ocurriendo y generando los informes necesarios. Esta tarea implica, por tanto, la realización de tres actividades básicas:(Antonio, 2001)

- Captura de la información.
- Almacenamiento de la información.
- Generación de informes.

En la actividad de generación de informes puede surgir como duda: ¿de dónde proviene la información necesaria para poder realizar esta tarea? Pues de las actividades realizadas en el proceso de GCS. Es importante destacar que los productos en esta actividad pueden diferenciarse en dos categorías:

- Registros
- Informes

De los registros existe una gran variedad de diversos tipos, por ejemplo:

Registro de elementos de configuración: Contiene toda la información relativa a los diferentes Elementos de Configuración de Software.

Registro de Líneas base: Contiene toda la información relativa a cada línea base del proyecto (Nombre, Fecha establecimiento, ECS que la componen).

Registro de Solicitudes de cambios: El tipo de información que se suele mantener acerca de cada solicitud de cambio es la recogida a través del formulario de Solicitud de Cambio, incluyendo:

- Código de solicitud.
- Información acerca del solicitante.
- Descripción del cambio.
- la documentación que apoya la petición de cambio, por ejemplo, una referencia a un Informe de Incidencia.
- la resolución o disposición acerca del cambio (aprobado, aplazado, denegado).

Registro de Cambios: El tipo de información que se suele mantener acerca de cada cambio es la recogida a través del Informe de Cambio, la Orden de Cambio, el proceso de Gestión de Problemas, etc.
Registro de Incidencias: El tipo de información que se suele mantener acerca de cada incidencia es la recogida a través del Informe de Incidencia.

Los informes pueden ser clasificados en:

Informe de estado de los cambios: Es un resumen del estado en que se encuentran todas las solicitudes de cambio registradas durante un determinado período de tiempo.

Inventario de elementos de configuración: Ofrece visibilidad sobre el contenido de las bibliotecas de proyecto.

Informe de incidencias: Es un resumen del estado en que se encuentran todas las incidencias originadas durante un determinado período de tiempo y las acciones a las que han dado lugar.

Informe de modificaciones: Resumen de las modificaciones que se han efectuado en el producto software durante un determinado período de tiempo.

Informe de diferencias entre versiones: Resumen de las diferencias entre las sucesivas versiones de un ECS.

La generación de informes es tarea importante, la misma permite mantener la continuidad del proyecto y evita el derrumbe del proyecto en caso de faltar algún jefe de proyecto, o sea es la manera de tener constancia escrita de todo lo que se ha realizado en el proyecto. Si la información no se tiene registrada puede traer graves consecuencias ya que se duplica el trabajo o se repite el trabajo ya realizado. En ocasiones puede ayudar a encontrar causas de fallo como también evita que se cometan los mismos errores.

1.5 Herramientas de Soporte a las actividades de GCS

En los últimos años, en la rama de la informática se han percibido grandes avances en disciplinas de apoyo al desarrollo de software, tal es el caso de la Ingeniería de Software y la Administración de Configuraciones. Estas disciplinas brindan grandes aportes al control de los proyectos y de los productos de software a lo largo de su ciclo de vida.

Actualmente la realización de dichas disciplinas se sustenta del conjunto de herramientas que existen en el mercado y que han surgido para apoyar las diferentes actividades comprendidas en la GCS. Las mismas están encaminadas a facilitar el trabajo, de una manera sencilla y segura garantizando buenos

resultados. En aras de lograr la eficiencia en un proyecto lo más corriente es que todo el personal se encuentre involucrado con las herramientas a emplear, ya que la no incorporación de parte del equipo de desarrollo en los procesos implementados por dichas herramientas puede conllevar al fracaso de la actividad de gestión del software y en general la administración del proyecto.

Es un hecho que hoy en día las propuestas más robustas son expuestas por soluciones privativas, sus licencias alcanzan altos precios en el mercado por lo que son casi inalcanzables para Cuba. Por tal motivo la opción es profundizar en el estudio y selección de herramientas libres, esta meta constituye el pilar en el campo del desarrollo de software nacional. La introducción paulatina del software libre en Cuba resulta muy importante para avanzar en la informatización de la sociedad.

Las herramientas automatizadas de SCM[5] utilizables hoy en día, y aquellas que todavía están en desarrollo, son mucho más versátiles que sus precursoras. Sin embargo, surge la pregunta ¿no existe alguna herramienta que lo haga todo? La respuesta es ¡todavía no! Y ello es debido, en gran parte, al hecho de que dentro del entorno en que funciona la GCS, se está evolucionando todavía.

1.5.1 Herramientas para el Control de Versiones.

Un sistema de control de versiones es el responsable de mantener las trazas de varias revisiones de la misma unidad de información. Se utiliza normalmente en proyectos de desarrollo de software para gestionar el código del proyecto, pero también puede usarse para gestionar versiones de documentación de cualquier tipo de proyecto.(Palomar, 2008)

A la hora de desarrollar un proyecto de software entre varias personas, uno de los servicios más útiles y en ocasiones imprescindible es mantener un buen sistema de control de versiones; permitiendo manejar el código fuente con una total libertad y transparencia entre los desarrolladores. Un sistema de control de versiones debe proporcionar mecanismos de almacenamiento de los elementos de configuración como por ejemplo: archivos de texto, imágenes, documentación, código fuente, etc. Debe brindar la posibilidad de realizar cambios sobre los elementos almacenados (modificaciones parciales, añadir, borrar, renombrar

[5] Software Configuration Management por sus siglas en inglés.

o mover elementos) y debe facilitar un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos que en un momento determinado se pueda volver o extraer un estado anterior del producto.

Por esa razón ofrece una mayor flexibilidad una solución que integre herramientas que viabilicen el proceso de GCS. Existen actualmente en el mercado una gran variedad de herramientas para realizar el control de versiones. Ver Anexo 1. A continuación se presentan ejemplos de alguna de estas herramientas por ser de las más difundidas y utilizadas hoy en día, así como una pequeña descripción de las mismas. Estas herramientas se pueden clasificar como:

Software privativo

- IBM Rational ClearCase
- Plastic SCM (Software Configuration Manager)
- VSS (Visual SourceSafe)
- AccuRev
- Bitkeeper
- Perforce
- Team Foundation Server (TFS)

También existe otro conjunto de herramientas que satisfacen los mismos objetivos pero en este caso pertenecen a la comunidad de Software Libre como por ejemplo:

Software Libre

- CVS (Concurrent Versions System)
- SVN (Subversion)
- Darcs
- Monotone
- Mercurial
- OpenCM
- Arch

- Bazaar
- Git

El sistema de control de versiones mantiene un repositorio con los ficheros, cada grupo de trabajo busca una forma habitual de trabajo: mantener una copia local y modificarla. Luego actualizarla en el repositorio, tiene como gran ventaja que no se necesita acceso continuo al repositorio. En otros casos dependiendo del sistema de control de versiones es posible trabajar directamente contra el repositorio, lo que hace que el trabajo sea más transparente aunque el bloqueo de ficheros puede ser un inconveniente. Es preciso tener en cuenta a la hora de elegir una herramienta, la forma de trabajo y las características del equipo de desarrollo, pues existen dos formas distintas de ver a los sistemas de control de versiones.

Distribuidos: La característica más importante de un SCM distribuido, como su nombre lo indica, es que no existe un punto central de desarrollo sino que los repositorios están distribuidos y descentralizados en diversas computadoras, que pueden o no ser independientes entre sí, y técnicamente no hay ninguno más importante que otro. Esto trae bastantes beneficios al momento de desarrollar, dado que el modo de trabajo suele ser que cada desarrollador tenga su repositorio propio sobre el cual trabaje de forma independiente, y periódicamente se pongan en común los trabajos de todos en algún repositorio convenido a tal efecto.(2006)

En mayo del año 2007 Linus Torvalds realizó una presentación sobre Git y desde entonces el interés por los sistemas de control de versiones distribuidos ha ido aumentando considerablemente. Estos sistemas brindan la ventaja a los clientes de poder comunicarse con otros y mantener sus propias ramas locales sin la necesidad de acceder a un repositorio central. Sólo se trabaja con copias. Las operaciones normales como commits, vista de históricos, sincronización y diferencias son más rápidas porque no hay necesidad de comunicar con un sistema central. Cada copia de trabajo es vista como un backup remoto proporcionando así un sistema natural de seguridad contra pérdida de datos. Así como crear y borrar nuevas ramas son operaciones simples y rápidas.

Los principales desarrollos de este tipo de sistemas son Mercurial (Apr, 2005), Bazaar (Mar, 2005), Darcs (Nov, 2004), Monotone (Apr, 2003), Arch, Codeville y el propio Git ya mencionado anteriormente.

Centralizados: Los SCM centralizados se basan en un repositorio único central, de este otro modo va a existir un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos) y los desarrolladores se conectarán para guardar sus cambios. Al haber un sólo repositorio, éste debe ser el encargado de manejar las ramas, quedando todos dentro de éste. Es decir, en un mismo repositorio existen dos (o más) caminos evolutivos distintos del mismo programa. La forma en que se implementa esto, varía de SCM en SCM, pero el concepto general se mantiene. (2006)

Los sistemas centralizados al ser naturalmente cliente-servidor, necesitan de un servidor que probablemente esté prendido todo el tiempo y con conexión permanente y este a su vez debe ser configurado, con algún sistema de autenticación y permisos. O sea la configuración puede ser un poco más compleja. Otro problema que puede acarrear por su naturaleza cliente-servidor está dada porque generalmente todas las operaciones son online, o sea requieren de conexión con el servidor.

Sin embargo facilitan las tareas administrativas a cambio de reducir la potencia y flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable. Este repositorio es quién se encarga de manejar las distintas variantes de implementación de un mismo programa. Algunos ejemplos de estos lo constituyen CVS y Subversion. Para trabajar con este tipo de herramientas centralizada existen dos modelos o soluciones:

Bloquear-Modificar-Desbloquear

Este tipo de modelo es el más simple y limitado, experimenta un desarrollo en serie por lo que consiste en que cada vez que un usuario desee modificar un archivo, este archivo se "bloquea" y no puede ser modificado por nadie más hasta que este usuario termine de editarlo, o sea que un sólo programador puede modificar un fichero a la vez. La primera impresión es que el desarrollo en serie da mayor seguridad ya que sólo un programador puede hacer cambios en un fichero en un momento dado. Se trata de una percepción errónea, heredada de años de realizar desarrollo de software sin usar sistemas de control de versiones adecuados.

Con este modelo en serie descrito se producen múltiples problemas administrativos, pérdida de productividad ya que mientras un desarrollador tiene que esperar a que otro termine con el fichero que

éste desea, y también produce desestabilización del producto. El estilo de trabajo Bloquear-Modificar-Desbloquear es el estilo predeterminado para una base de datos de Visual SourceSafe (VSS), siendo esta una herramienta representativa de este modelo aunque este sistema de control de versiones admite los dos estilos, o sea el de Bloquear-Modificar-Desbloquear que también se le conoce como modelo de desprotección exclusiva y el modelo Copiar-Modificar-Unir o modelo de varias protecciones que se muestra a continuación.

Copiar-Modificar-Unir

Este modelo soluciona el error del modelo anterior referido al bloqueo de ficheros y la programación en serie. Permite el desarrollo en paralelo donde cada programador puede acceder al repositorio y crear una copia de trabajo personal de determinado elemento o del proyecto, hacerle cambios tantas veces como sea necesario sin comprometer la línea base del proyecto. Esta operación la podrán hacer simultáneamente varios desarrolladores cada uno en su espacio de trabajo pudiendo desarrollar en paralelo sobre un mismo fichero e integrando los cambios. Para ello el sistema de gestión de configuración debe permitir que existan múltiples líneas de desarrollo abiertas al mismo tiempo.

Los cambios realizados son almacenados, y pueden evolucionar hasta que se decide que deben juntarse de nuevo en la línea principal, creando una nueva versión del producto. De esta manera se garantiza no sólo una mayor estabilidad en el proyecto, sino una mejor coordinación de las diferentes actividades. Existen herramientas capaces de ejecutar estas acciones como Subversion, CVS, Plastic SCM, etc.

Caracterización de las herramientas

Cada una de las herramientas citadas con anterioridad cuenta con un conjunto de potencialidades y características que se muestran a continuación.

IBM *Rational ClearCase* Es una solución líder en la industria y que proporciona un sofisticado Control de Versiones. Ofrece una gestión fiable, ampliable y flexible de los activos de software para equipos de desarrollo de gran tamaño y tamaño medio. IBM Rational ClearCase proporciona una gestión del ciclo de vida y control de los activos de desarrollo de software. Con un control integrado de versiones, una gestión del espacio de trabajo automatizado, un soporte de desarrollo en paralelo, una gestión de línea base, así como gestión de builds y releases. Proporciona las funciones necesarias para crear, actualizar, ofrecer,

reutilizar y mantener los activos más importantes del negocio.() Posee interfaces locales, remotas y web que permiten un acceso prácticamente a todos los sitios y en cualquier momento.

Se integra sin fisuras con Rational ClearQuest para obtener una solución completa de GCS. Escalable con un soporte extendido de diversas opciones de plataforma y sistemas operativos apropiados como AIX, HP-UX, Linux, Sun Solaris, Microsoft Windows. Está íntimamente integrada a los IDE[6] más populares, brindando a los desarrolladores un acceso inmediato a las capacidades de gestión de configuración sin abandonar su ambiente de desarrollo preferido. Sin embargo constituye un SCM privativo, el precio con los impuestos incluidos es de €5,153.88. En el caso de los precios que incluyen los impuestos aplicables, impuesto abarca el IVA[7] u otros impuestos aplicables propios de cada país.

Plastic SCM es otro ejemplo de herramienta privativa y que satisface dos actividades básicas de la GCS: Control de Cambios y Control de Versiones. Posibilita una correcta gestión de ramas, y con ello habilita la implementación de desarrollo en paralelo orientado a tareas, ayudando a que su equipo trabaje más rápido y pueda controlar mejor las entregas. Sistema de control de versiones de primer nivel con características avanzadas, es centralizado y emplea el modelo *Copiar-Modificar-Unir.* Plastic SCM se ha desarrollado con tecnología .NET por lo que funciona perfectamente con cualquier versión de Windows incluyendo W2K, W2003, XP y Vista, pero además es totalmente compatible con Mono por lo cual se puede utilizar en Linux, MacOS y Solaris. Proporciona integraciones con los entornos de desarrollo más utilizados como son Eclipse, JDeveloper y Visual Studio.

La aplicación Plastic SCM pertenece a la empresa española Códice Software, es uno de los primeros productos dedicados a la administración de la configuración del software de colaboración. Plastic es lanzada al mercado en noviembre de 2006 para luego ser galardonada con el premio europeo a la innovación AETICAL 2007. Esta herramienta es una de las que más evolución tiene en el mercado ya que ofrece un potente control de versiones para código y cualquier otro activo digital involucrado en el

[6] Integrated Development Environment (por sus siglas en inglés) significa Entorno de Desarrollo de Software, es un entorno de programación.

[7] Impuesto al Valor Agregado: es un impuesto indirecto al valor agregado sobre el consumo.

desarrollo de software; sin dejar de mencionar que es un software privativo y que se distribuye bajo licencia.

Es una potente herramienta por lo que sus licencias tienen precios muy altos, el costo de licencia es por desarrollador, con lo cual una empresa debe comprar tantas licencias como desarrolladores vayan a interactuar con la herramienta. De 1 a 20 licencias por cada desarrollador tiene un precio de $649 y en euros 525, para un número de licencias de 21 a 50 por cada integrante son $599 y 485 euros, de 51 a 100 serían $549 y así sucesivamente a medida que aumenta el número de desarrolladores disminuye el precio de las licencias.

Visual SourceSafe sistema de control de versiones en el nivel de archivos. Surge a principio de los años noventa como solución a la GCS, herramienta creada por la compañía One Tree Software. Fue liberada en su primera versión 3.1, luego Microsoft adquirió los derechos sobre SourceSafe que inicialmente era un programa de 16 bit y lo liberó en 1995 como un programa de 32 bit en su versión 4.0. Esta herramienta para el control de versiones forma parte de Microsoft Visual Studio aunque en la actualidad esta siendo sustituida por el Team Foundation Server, sistema por excelencia para las nuevas tecnologías de Microsoft.

VSS se ha diseñado para controlar los problemas de seguimiento y portabilidad que implica mantener una base de control de código fuente, como una base de código de software, en varios sistemas operativos. Para las personas que desarrollan programas bajo el sistema operativo Windows, resulta una herramienta útil ya que se integra fuertemente con el IDE de Visual Studio permitiendo un manejo relativamente simple de versiones.

Esta herramienta permite compartir archivos entre proyectos de forma rápida y eficaz. Para los desarrolladores, Visual SourceSafe aloja código reutilizable u orientado a objetos. Asimismo, facilita el seguimiento de las aplicaciones que utilizan módulos de código concretos. Permite a los equipos de desarrollo proteger y llevar un registro de manera automática de su código fuente, documentación, archivos binarios, y todos los demás tipos de archivo a medida que cambian a través del ciclo de vida de software.

Microsoft Corp. ofrece diferentes tipos de licencia que pueden ser vendidas como producto empaquetado (Caja), licencia en CD-Rom y documentación en un paquete o licencias para software preinstalado en una PC nueva. Microsoft también hace oferta de licencias para distintos organismos como por ejemplo organismos e instituciones gubernamentales que tiene un monto de $3,376.04, licencia simple abierta para colegios, academias, bibliotecas y museos públicos de $665.96 y licencias libres de negocio para usuarios particulares o empresariales en $4,442.81 siempre garantizando un buen soporte técnico.

AccuRev avanzada herramienta para la Gestión de Configuración de Software. Permite a los equipos de desarrollo de software mejorar la productividad, trazabilidad, reusabilidad y calidad de software, así como gestionar eficazmente los ficheros que están en producción y el mantenimiento de las entregas de software. Como herramienta de GCS dispone de múltiples sistemas de integración para utilizarlo en combinación con otras herramientas (modelaje, IDEs, control de incidencias, etc.). Esto permite escoger el mejor ciclo de vida para sus proyectos permitiendo una trazabilidad completa. Esta herramienta brinda mejoras para la escalabilidad y el proceso de visualización personalizada.

AccuRev puede manejar centenares de millares de archivos en un solo espacio de trabajo. Los precios varían según la configuración. Un solo usuario, con licencia perpetua es de $ 1.495. AccuRev, ha sido seleccionado como finalista de los premios Jolt del Dr. Dobb del 19th Annual Jolt Product Excellence Awards, Premios a la Excelencia en la categoría de Gestión de Cambio y Configuración. Es el único software de GCS que proporciona un sistema de optimización de procesos de colaboración. Por lo que hoy en día las organizaciones mundiales de desarrollo de aplicaciones se incluyen como usuarios de AccuRev, como por ejemplo grandes compañías: SanDisk, T-Mobile, SunGard, Sony, etc., creciendo su uso a más del 90% anual durante los últimos tres años.

Resalta la potencialidad de las herramientas citadas con anterioridad, pero cada una tiene un valor adquisitivo elevado para Cuba; por lo que se hace imprescindible auxiliarse de herramientas libres de apoyo a la GCS en el desarrollo de software nacional. Ejemplo de ellas lo constituyen:

Concurrent Versions System Sistema de Control de Versiones de GNU[8] y Centralizado más conocido por sus siglas en inglés CVS. Fue liberado en 1986 y si bien el CVS puede ser una tecnología más antigua, es todavía muy útil para cualquier diseñador o programador para hacer copias de seguridad y compartir archivos. CVS es un software de código fuente abierto que ejecuta en la mayoría de las plataformas UNIX existentes: Linux, Mac OS X, FreeBSD así como Microsoft Windows. Combina una arquitectura cliente-servidor lo que permite a los desarrolladores funcionar como un solo equipo aunque estén dispersos en la geografía. Funciona en base a un conjunto de comandos para administrar repositorios, módulos y archivos. CVS se ha hecho popular en el mundo del software libre. Sus desarrolladores difunden el sistema bajo la licencia GPL.

Subversion (SVN) sistema de control de versiones libre y de código abierto con gran proyección en la actualidad. El servidor de Subversion gestiona un repositorio centralizado con la política Copiar-Modificar-Unir. Fue diseñado para reemplazar a CVS, no sólo sustituirlo sino con la intención de mejorarlo ya que SVN mantiene las ideas fundamentales de CVS pero suple sus carencias y evita sus errores. Originalmente Subversion es patrocinado y organizado por CollabNet, que reunió un grupo de desarrolladores dedicados a tiempo completo en el proyecto Subversion, que luego de catorce meses de codificación es concluido el 31de agosto del año 2001. Aunque fue dirigido por CollabNet es un verdadero proyecto de código abierto ya que la licencia copyright de CollabNet es completamente compatible con las Directrices de Software Libre de Debian, por lo que su descarga es completamente libre y bajo una licencia de tipo Apache/BSD [9].

Subversion está construido sobre una capa de portabilidad llamada APR (la biblioteca Apache Portable Runtime), lo cual significa que Subversion debería funcionar en cualquier Sistema Operativo donde lo haga el servidor httpd Apache: Windows, Linux, todos los sabores de BSD, Mac OS X, Netware y otros.(Ben Collins-Sussman, 2004). Subversion proporciona versionado de directorios, implementa un

[8] El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre: el sistema GNU.

[9] licencia de software otorgada principalmente para los sistemas BSD (pertenece al grupo de licencias de software libre.

sistema de ficheros versionado “virtual” que sigue los cambios sobre árboles de directorios completos a través del tiempo. Brinda un verdadero historial de versiones y permite añadir, borrar, copiar, y renombrar ficheros y directorios. Y cada fichero nuevo añadido comienza con un historial nuevo, limpio y completamente personal.

Subversion puede conectarse al servidor HTTP Apache como un módulo de extensión. Esto proporciona a esta herramienta una gran ventaja en estabilidad e interoperabilidad, y acceso instantáneo a las características existentes que ofrece este servidor como autenticación, autorización y compresión de la conexión. Constituye un excelente sistema de control de versiones por lo que es rápidamente acogido por la comunidad de desarrolladores de software libre.

Después de la hegemonía del CVS se creía que había poco que hacer con el tema de control de versiones. El conocidísimo Subversion, llegó para reemplazarlo, simplemente reimplementa el concepto de los sistemas controladores de versiones clásicos. Pero no siempre un control de versiones centralizado es la respuesta a todos los problemas. Aquí es donde entran los controles de versiones descentralizados, o más conocidos como Distribuidos. Estos sistemas no solo representan una importante innovación al mundo de la ingeniería del software sino que cambian la manera de hacer las cosas.(Mallorca, 2006) Es por ello que surgen otras herramientas libres con diferentes características.

Darcs avanzado sistema de control de versiones distribuido. Permite controlar qué cambios se le hacen a un conjunto de ficheros. Normalmente se aplican a programas, pero no necesariamente. Originalmente desarrollado por David Roundy, y se basa en un álgebra de parches. La interfaz de Darcs es bastante sencilla, y sobre todo interactiva; a diferencia de otros sistemas de control de versiones este tiene una interfaz amigable. Es fácil de aprender y utilizar. Darcs es muy potente y a la vez sencillo todo lo contrario a Arch y Bazar que también son potentes pero complicados. Tiene independencia de un servidor central, cada repositorio es una rama por sí misma y existe una simetría entre ellos, es decir todos siguen los mismos esquemas.

En Darcs un repositorio es un conjunto de parches o cambios, entonces, como los parches se pueden llevar de un repositorio a otro es posible hacer que dos repositorios sean la colección de exactamente los

mismos parches. Brinda la posibilidad de renombrar ficheros manteniendo el historial. Tiene varios métodos de acceso: local, ssh, http y ftp por lo que da la posibilidad de hacer commits locales (sin conexión). Su última versión estable funciona para varias plataformas incluyendo Windows, Mac OS X, y sus hermanos de FreeBSD, Solaris y AIX, una docena de sabores de Linux. Con estas pruebas, de Darcs se espera que pronto escale alto para tener un buen desempeño en los proyectos más importantes. Podría resultar una herramienta muy útil para muchos equipos de desarrollo de código abierto.

Git software de sistema de control de versiones distribuido, con funcionalidad plena, diseñado por Linus Torvalds, siempre pensando en la eficiencia y confiabilidad de mantenimiento de versiones de aplicaciones con una gran cantidad de archivos de código fuente. Es empleado por el grupo de programación del núcleo del Sistema Operativo Linux. Git abre un mundo de posibilidades y ventajas, aumentando la velocidad al trabajar casi siempre en local, o sea que cada directorio de trabajo local es un repositorio completo independiente de acceso a un servidor o a la red por lo que se reduce la necesidad de estar siempre conectado.

Se aumenta la fiabilidad del sistema al tener todos los nodos una copia completa del proyecto. Esta herramienta ofrece una gestión eficiente para proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.

Lo esencial en la descentralización del control de versiones es que no va existir un único repositorio sino que todos estarán al mismo nivel, cada uno será una rama distinta. Con tal propósito se logrará una gran independencia entre los desarrolladores y sin embargo no limitará para nada los métodos y técnicas de los sistemas centralizados, con la teoría de los parches se puede asegurar que en cualquier momento se podrá centralizar todo el trabajo en una rama de un repositorio que se considere central.

1.5.2 Herramientas de seguimiento de tareas y errores

En la GCS se concibe como actividad fundamental controlar la versión del fichero con el que se esté trabajando, sin embargo la idea de la GCS va más allá, ya que trata de abarcar todo lo relacionado con las versiones: como el control de versiones de cada fichero, el control de cambios que se realizan ya sea por problemas o mejoras, y el control del proceso de construcción (build) de las versiones.

La administración de cambios y la configuración del software son parte de las disciplinas a las que algunos desarrolladores en sus inicios adhieren a regañadientes, y a veces sólo forzados por una política directiva. Pero cuando un proyecto toma mayor madurez, las reglas y restricciones propias de la administración de los cambios pasan a ser imprescindibles. No existe actualmente ningún software que haga conjuntamente el control de versiones y el control de cambios. No, al menos, ninguno que se esté adoptando ampliamente o que se esté destacando especialmente. Por el momento se deben seleccionar herramientas separadas para llevar a cabo todo este proceso.

Existe un gran número de herramientas que aparecen como soluciones independientes o como parte de una suite que integra todo el proceso de desarrollo de software. Pero, en general son soluciones de compañías que debido a su elevado precio se han limitado al mercado de las grandes empresas de desarrollo de software. Incluso el bloqueo económico impuesto a Cuba constituye además del elevado precio, un freno que imposibilita el uso de las mismas en la Industria Cubana del Software. Esto motiva la búsqueda de soluciones alternativas; encontrando algunas herramientas que han tenido éxito dentro del mundo Open Source o de proyectos libres.

Estas en su mayoría han surgido como sistemas para el seguimiento de errores (Bug Tracking System) pero con el tiempo han evolucionado ofreciendo soluciones más completas y adaptables a las necesidades de control de cambios de muchos proyectos. El uso de alguna de estas herramientas ayuda en gran medida en la automatización del proceso de solicitud de cambios, hacer cumplir el flujo de proceso de cambios o capturar las decisiones del Comité de Control de Cambios. Existen diversos paquetes de herramientas para el seguimiento de tareas y errores, desde software libre a soluciones profesionales con características avanzadas. Ver (Anexo 2).

1.5.3 Herramientas para la Gestión de Proyectos.

Las actividades de Gestión de Configuración se presentan en dos dimensiones: una dimensión de soporte al grupo de desarrollo y a sus necesidades de manipulación del producto y una dimensión administrativa, que considera las necesidades de gestión y reporte del cambio. Aunque uno de los principales problemas en todos los proyectos de desarrollo de software es el “versionaje” ya que permite llevar un control de los

cambios en el código fuente, como por ejemplo saber en cada momento quién y cuándo lo hizo, por qué lo hizo, y qué cambio a razón de qué.

A su vez para la Gestión de Proyectos también se hace necesaria la planificación de las tareas en el desarrollo de software y planificación de las actividades, de manera que se hace necesario manejar una descripción del producto en términos de los servicios que ofrece, así como de las actividades administrativas. Para ello actualmente existen alternativas de software libre, que unidas a las herramientas para la GCS citadas anteriormente permiten mantener un ordenamiento de la documentación que acompaña al proyecto, seguimiento de defectos, registro de mejoras pendientes**,** acceso interactivo y foros, calendarios, o alguna plataforma para la comunicación entre los integrantes de un proyecto. Existen diversas herramientas con esta finalidad y se muestran a continuación algunos ejemplos:

DotProyect herramienta creada por dotmarketing.org en el año 2000, con el fin de construir una herramienta para la Gestión de proyectos. Está construido por aplicaciones de Código abierto y es mantenida por un pequeño grupo de voluntarios dedicados al software libre. Programada en PHP[10], y utiliza MySQL como gestor de base de datos. Es una aplicación basada en web, para administración de proyectos, es multiusuario, soporta varios lenguajes. Incluye módulos para compañías, proyectos, tareas (con gráficas de Gantt), foros, archivos, calendario, contactos, tickets de soporte.

Trac sistema web libre para la gestión de proyectos software y seguimiento de errores. Trac es un sistema web centrado en un wiki, con sistemas montados alrededor de él: tags, blog, sistema de tickets/incidente y gran cantidad de plugins, extensiones y macros. Está pensado para la organización de proyectos de desarrollo software de forma principalmente colaborativa. Soporte configurable para workflow en sistema de ticket. Mejoras en sistema de ayuda y documentación, en sistema de usuarios y sesiones. Trac permite enlazar información entre una base de datos de errores de software, un sistema de control de versiones y el contenido de una wiki. También sirve como interfaz web de un sistema de control de versiones como Subversion, Git, Mercurial, Baazar o Darcs.

[10] PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor.

Jira es una aplicación diseñada para mejorar el proceso de seguimiento a los defectos y errores, es decir mantener la trazabilidad de la información, así como administrar y manejar proyectos de desarrollo de software. Contiene una interfaz gráfica versátil, fácil de entender por usuarios tanto administrativos como técnicos. Soporta casi cualquier plataforma de hardware, sistemas operativos y plataforma de base de datos. Grafica los procesos de trabajo en flujos modificables. Maneja errores, características, tareas, mejoras o cualquier tipo de defecto. La misma contiene opciones de notificación altamente configurables. Más de 3.500 organizaciones en 55 países alrededor del mundo utilizan JIRA como su administrador de proyectos. Empresas del tipo tecnológico como NOKIA y AT&T Labs, Gobiernos regionales (Quebec y Yukon), y hasta universidades se benefician actualmente de las increíbles prestaciones de JIRA.

Mantis excelente herramienta en la categoría de Código Abierto. Es una de las soluciones más completas hoy en día para la gestión de tareas entre un equipo de trabajo. Mantis es un sistema de control de fallos de programación que funciona a través del navegador, está programado en php y puede funcionar con varias bases de datos: MySQL, PostgreSQL, MS SQL, etc. La interfaz de Mantis muestra los errores detectados, a quién se le ha asignado la tarea de corregirlos y el estado en que se encuentran.
Este sistema de seguimiento funciona a través de cuentas de usuario desde las que se informa de los fallos detectados mediante formularios, una vez aceptada la tarea, los informadores reciben correos con cada variación en el estado de ésta. Por sus características existen empresas de desarrollo de software con más de 300 empleados que la utilizan constantemente para encaminar problemas, testear soluciones, registro histórico de alteraciones y gestionar equipos remotamente, ya que es muy fácil de instalar y muy flexible en su configuración.

Redmine moderna herramienta para la gestión de proyectos software con interface web. Es un sistema multi-plataforma, programado con Ruby on Rails, es una herramienta Open Source con licencia GPL. Posee una interfaz sencilla y muy configurable. Brinda unas funcionalidades asombrosas como foros, envío automático de e-mail a los desarrolladores cada vez que se les asigna una tarea o ante cualquier evento relacionado con el proyecto. En Redmine también se pueden dar de alta los "bugs" o errores que se encuentren, asignándoselos al desarrollador correspondiente y al hito para el que se considere que debe estar corregido dicho error.

Posibilita subir ficheros y documentos, bien al proyecto, como adjuntos a las tareas y errores. También brinda la posibilidad de definir nuevos tipos de tareas y errores, con campos personalizados, todo ello fácilmente a través de la interface web. Se puede ver a través de Redmine los cambios en el repositorio. Entiende CVS, Subversion y algunos de los sistemas de Control de Versiones más conocidos. Contiene control de acceso por roles, Multi-proyectos, gráficos Gantt, gestión de avisos, documentos y ficheros, wiki, registro de tiempos, campos personalizables, multi-lenguaje. Es muy similar a Trac, pero con una administración e interfaz web más amigable, aunque con menos tiempo en marcha y menos plugins disponibles.

Conclusiones

A lo largo del presente capítulo se evidencia como los productos software evolucionan desde su concepción y la captura de requisitos inicial, hasta la puesta en producción del mismo. Se realizan una serie de cambios, tanto en el código como en la documentación asociada. Por lo que la Gestión de Configuración del Software es una disciplina encargada del control de la evolución de los productos de software cuyo objetivo es mantener la integridad y la trazabilidad a lo largo del ciclo de vida.

Durante el desarrollo de software se pueden realizar varias actividades como: planificación, modelado, construcción, revisiones, actividades relacionadas con la GCS, entre otras. Estas pueden ser realizadas de manera manual pero atentaría contra la productividad y eficiencia. Sin embargo el desarrollo acelerado de la industria del software a nivel mundial ha llevado a que el proceso de desarrollo sea apoyado por un grupo de herramientas. Existe un gran número de las mismas que garantizan diversas actividades de la GCS que han sido explicadas en el presente capítulo, lo que es importante para posteriormente seleccionar las que se necesitan realmente.

CAPÍTULO 2. SOLUCIÓN PROPUESTA PARA LA GCS EN EL POLO DE IMÁGENES.

En el presente capítulo se documenta la solución conceptual brindada, y como estrategia propuesta, cinco actividades que deben implantarse para realizar de forma eficiente la Gestión de Configuración específicamente en el Polo de Imágenes perteneciente a la Facultad 7.

Se proponen actividades, que una vez estudiado el tema, la autora considera de necesaria importancia su aplicación en el proceso de GCS del proyecto, así como los procedimientos a seguir en cada una de dichas actividades. Teniendo siempre en cuenta quien va a ejecutarlas, pues se definen roles y responsables para las mismas.

Es una realidad que el proceso de desarrollo de un producto software se torna extenso y a la vez complicado, razón por la que hoy en día el uso de herramientas de software en apoyo a este proceso cobra una mayor importancia. Basado en el análisis crítico de cada una de las herramientas pues se seleccionan las que se adecuan al Polo de Imágenes según las características que presenta.

2.1 Propuesta para la estructura del equipo de GCS

Un aspecto primordial en el proceso de GCS es la organización del equipo de desarrollo, según Pressman dentro de un proceso de software bien definido y avanzado, apropiado para los productos que se van a construir y que satisfaga las demandas del mercado, el factor más importante que va a contribuir al éxito real del proyecto de software son las personas que van a intervenir en el proceso. (Pressman, 2005)

La falta de coordinación en la GCS puede ser una de las principales causas que llevan a un proyecto al caos, por tanto se hace sustancial como vía de solución centralizar este proceso, de forma tal que las responsabilidades queden bien definidas, así como el trabajo correspondiente a cada uno de los integrantes; y en un futuro se podrá tener un mejor control de los mismos. Un punto a tener presente es la forma en que se encuentra estructurado actualmente el Polo Productivo de Imágenes que es la siguiente:

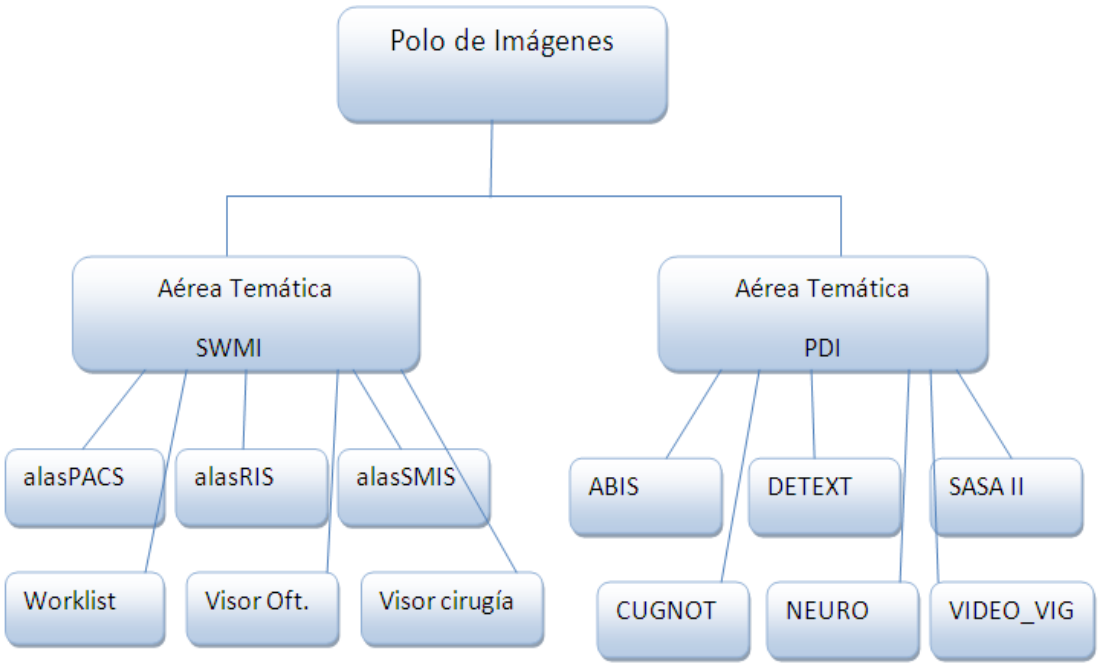


Fig. 2.1 Estructura actual del Polo de Imágenes.

Términos empleados:

SWMI: Software Médico Imageneológico.

PDI: Procesamiento Digital de Imágenes.

alasPACS: Sistema de almacenamiento y transmisión de imágenes PACS(Picture Archiving and Communication System).

alasRIS: Sistema de Información Radiológica.

alasSMIS: Almacenamiento de Estudios Médicos.

Visor Oft. Visor Oftalmológico.

ABIS: Sistema de identificación balística.

NEURO: Proyecto integrado por los subproyectos DIANA, Dislexia, Discalculia.

CUGNOT: Suite para la identificación de los vehículos.

VIDEO_VIG: Video Vigilancia.

DETEX: Detección y extracción de texto en videos.

SASA II: Servicio Autónomo de Sanidad Agropecuaria.

El Polo de Imágenes se encuentra subdividido en dos Áreas Temáticas una especializada en software médico imageneológico para la salud y otra en el procesamiento digital de imágenes respectivamente. Estas a su vez se encuentran integradas por un grupo de proyectos por separado. Es por ello que dada las peculiaridades que presenta pues se elabora la siguiente propuesta de una estructura organizativa para el proceso de GCS:

Fig.2.2 Estructura del equipo de GCS.

Se define en la solución propuesta un Administrador de Configuración Principal, en la bibliografía se suele nombrar Líder de Configuración, pero no existe tal diferencia entre una forma u otra, lo esencial es que este rol es el principal responsable de organizar y controlar la Administración de la Configuración, proceso de conservar las versiones relevantes del proyecto.

Como estrategia para llevar a cabo la GCS en cada Área Temática se propone un equipo de GCS para cada una de estas, el cual quedará conformado por:

- Gestor de Configuración de Área Temática.

- Jefe de Área Temática.
- Líderes de cada Proyecto.
- Arquitectos Principales de cada Proyecto.

Siendo el Gestor de Configuración de Área Temática el principal líder del equipo encargado de llevar a cabo la GCS. Dicho equipo será encargado de los asuntos asociados a la GCS y que en ocasiones funcionará como un CCC[11]. De igual forma cada rol debe responder por las siguientes actividades:

Administrador de Configuración Principal

- Encargado de controlar y dirigir el proceso de GCS a nivel de Polo Productivo.
- Discutir los planes de AC con el equipo de desarrollo antes de implantarlos.
- Mantener el documento PACS[12] que es el documento que describe cómo deben administrarse los artefactos del proyecto en cuestión.
- Definir, instalar y mantener herramientas de AC especificadas.
- Adquirir y respaldar las herramientas de configuración usadas.
- Realizar un resumen de configuración al menos una vez a la semana.
- Ejecutar controles internos al proceso de GCS, en cada área temática.

Gestor de Configuración de Área Temática

- En situaciones excepcionales es el encargado de la función Administrador de Configuración Principal.
- Conocer todos los medios relevantes de acceso a los documentos durante la vida del proyecto.
- Asegurar que los documentos se archiven de acuerdo con las políticas establecidas.
- En conjunto con el líder de proyecto debe ser responsable de identificar todos los EC.
- Define la Estructura que tendrá el Repositorio, así como las políticas de acceso para cada integrante del proyecto.

[11] Comité de Control de Cambios.

[12] Plan de Administración de la Configuración del Software.

- Encargado de revisar y verificar que la línea base del proyecto, se encuentre actualizada en el período de tiempo establecido.
- Inspecciona que cada integrante del equipo de desarrollo utiliza las herramientas que han sido seleccionadas para el desarrollo.

Desarrollador

- Son los principales involucrados en el desarrollo del sistema software, por tanto contribuyen a obtener productos con calidad pero para ello deben lograr el cumplimiento de las reglas definidas por el Administrador de Configuración Principal ya que están encaminadas a crear cada vez mejores y eficientes productos.
- Debe respetar la línea principal de desarrollo pues de otra forma se vería afectado todo el equipo de desarrollo.
- Realizar cada envió al repositorio con comentarios como una forma de constar con una detallada documentación del proyecto, lo cual permite mantener informado a cada integrante del proyecto.
- Enviar actualizaciones de la copia de trabajo frecuentemente al repositorio.

2.2 Análisis de las herramientas para la GCS

En la GCS quizás el ejemplo más claro es la evolución del código fuente, lo mismo a lo largo del desarrollo y posteriormente en su mantenimiento. Generalmente las modificaciones al software se realizan sobre el código fuente y a partir de ahí entonces se valida la documentación asociada. De modo que lo más óptimo sería entonces llevar un eficiente control de las diferentes versiones que se van generando en el desarrollo de un producto.

Mientras más grande o extenso en el tiempo, es un proyecto, resulta más difícil compartir la información entre todos los participantes y la necesidad de tener mecanismos o herramientas para almacenarla y acceder a ella es innegable. Es por ello que un punto clave en la estrategia propuesta, es la definición de las herramientas que den soporte y ayuden a automatizar una tarea tan importante como el Control de Versiones en el proceso de GCS.

2.2.1 ¿Por qué emplear herramientas de Software Libre?

En la actualidad las tecnologías de la información han tenido un enorme auge, constituyendo una de las bases más importantes del desarrollo. Este no solo implica lo material sino otra serie de aspectos importantes relacionado con los programas, las políticas educativas, en la informatización del sector de la salud, en la capacitación y superación a los profesionales. Es por ello que cada día se incrementa más la necesidad de que los países de Latinoamérica se integren a sus avances y beneficios.

Aunque se presenta como desventaja, que hoy en día, el software privativo mantiene un alto grado de aceptación y popularidad a nivel mundial, por sus potencialidades sin embargo es claro que al ser la empresa proveedora la única que dispone del código fuente, en caso de algún cliente presentar insatisfacciones con algún producto del cual ha adquirido una licencia de uso, pues lógicamente pone al usuario en una crítica dependencia del proveedor, esto sin contar con los exuberantes precios que presentan en el mercado. En caso de presentar defecto algún producto, el cliente debe aceptar las condiciones de la empresa productora y en el mejor de los casos si dicha empresa reconoce el error, pues esperar a que la misma acceda a repararlo.

Sin embargo no sucede lo mismo con herramientas libres que al ser Open Source, el usuario tiene libertades en cuanto al estudio y entendimiento de como operan estas, ya sea en aras de mejorarla y adaptarla a sus necesidades como adicionarle funcionalidades; siempre y cuando comparta y publique con los demás usuarios de la comunidad de Software Libre. Este término surge hace ya más de treinta años, cuando un gran número de programadores de diversas universidades decidieron desarrollar herramientas de forma cooperativa y abierta. Siempre con la convicción de distribuir libremente su código fuente; con lo que se ha logrado la construcción de productos software de gran envergadura y con excelentes cualidades técnicas.

Con el surgimiento de Internet el movimiento de Software Libre ha acrecentado su desarrollo involucrando cada día más personas de todo el mundo, incluyendo empresas que tradicionalmente han utilizado el modelo propietario para el desarrollo y la comercialización de sus productos, ejemplo de ello lo constituye

IBM que basa todos sus productos orientados a la Web en el servidor libre Apache. El Software Libre es una cuestión de la libertad que tienen los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Precisamente, se refiere a cuatro tipos de libertades para los usuarios:

- La libertad de ejecutar el programa, para cualquier propósito (libertad 0).
- La libertad de estudiar cómo trabaja el programa, y adaptarlo a sus necesidades (libertad 1).
- La libertad de redistribuir copias para que pueda ayudar al prójimo (libertad 2).
- La libertad de mejorar el programa y publicar sus mejoras, y versiones modificadas en general, para que se beneficie toda la comunidad (libertad 3).

Para las cuales una condición necesaria es el acceso al código fuente. La comunidad del Software Libre está comprometida con la proposición de que los elementos ejecutables primordiales de la tecnología puedan ser producidos sin que existan relaciones de propiedad discriminatorias.

Durante la XI Convención Internacional Informática, realizada el 13 de mayo del 2005, se debatió sobre el desarrollo y la utilización del Software Libre, como una opción necesaria para Cuba, dentro del programa de informatización de la sociedad cubana. Este es un tema fuertemente tratado y analizado por una cuestión de viabilizar una solución contra el criminal bloqueo impuesto a Cuba. Para el país la adquisición de productos de grandes empresas de software es limitado. El uso de Software Libre permite desarrollar productos y aplicaciones de software propio, es por este motivo que la perspectiva del Software Libre se debe insertar paulatinamente en todos los sectores dedicados a la producción de software y en general en el ámbito informático cubano.

En los momentos actuales, la Universidad de las Ciencias Informáticas es la institución líder y de mayor avance, en la estrategia cubana para la migración al Software Libre. Actualmente tres grupos de trabajo se ocupan de los temas técnicos, de capacitación y legales que rigen de manera organizada el desarrollo y despliegue del Software Libre en todo el país, afirmó Héctor Rodríguez, decano de la Facultad 10, quien se encuentra al frente del Grupo Nacional para la Migración al Software Libre. En dicha universidad se desarrolló un sistema operativo propio basado en la distribución de GNU/Linux conocida como Gentoo, y

este fue nombrado Nova; al cual ya se le han efectuado pruebas piloto de migración en el ámbito estatal y educativo.

2.2.2 Comparación entre algunas herramientas existentes para la GCS

Existe un gran número de herramientas dedicadas a la GCS que permiten gestionar las versiones en todo el proceso de desarrollo de software. Por lo que se decidió realizar un repaso de cuatro sistemas para el control de versiones en específico por ser de los más conocidos y extendidos en todo el mundo. Para ello se toman como ejemplo Microsoft Visual SourceSafe, Concurrente Version Systems (CVS), Subversion y Git, según se muestra en la tabla.

Herramientas SCM	Tipo de Licencia	Sistema Operativo soportado	IDE[13] soportado	Protocolos de Red
Visual SourceSafe (Centralizado)	Privativo (Microsoft)	Windows	- Microsoft Visual Studio .NET	http, https
CVS (Centralizado)	GPL	Unix, Windows, Linux, Mac OS X	-NetBeans -Eclipse	pserver, ssh
Subversion (Centralizado)	Apache/BSD	Unix, Windows, Linux, Mac OS X	- Eclipse -JDeveloper -NetBeans -Visual Studio.NET	Protocolo propio(svn), http, https, ssh, y SSL (usa WebDAV)
Git (Distribuido)	GPL	Windows, Linux Mac OS X	- NetBeans -Microsoft Visual Studio .NET -Eclipse	http, ftp, ssh, rsync

[13] Entorno Integrado de Desarrollo del inglés: Integrated Development Environment.

Una de las herramientas CASE más difundida y utilizada en el mercado para el control de versiones es Microsoft Visual SourceSafe; es un sistema de control de versiones en el nivel de archivos, que permite a muchos tipos de organizaciones trabajar en distintas versiones de un proyecto al mismo tiempo. Esta funcionalidad es especialmente ventajosa en un entorno de desarrollo de software, donde se usa para mantener versiones de código paralelas, por tanto es evidente que la herramienta admite el desarrollo en paralelo. De esta forma posibilita que todos los miembros del equipo terminen las distintas partes y versiones de un proyecto al mismo tiempo, aumentando la eficiencia.

Visual SourceSafe aloja código reutilizable u orientado a objetos. Asimismo, facilita el seguimiento de las aplicaciones que utilizan módulos de código concretos. Permite compartir archivos entre proyectos de forma rápida y eficaz. Se puede integrar Visual SourceSafe al entorno de desarrollo integrado de Visual Studio.NET como proveedor de control de código fuente, lo cual es muy importante para el Polo de Imágenes, ya que emplea este entorno de desarrollo, pero su principal freno es que constituye un software privativo, es una herramienta desarrollada por la corporación Microsoft, por lo que está limitado solamente a las plataformas de Microsoft, lo cual lo pone en desventaja con otros sistemas que si soportan como sistema operativo Linux, o sea que en un futuro cuando se lleve a cabo la migración a Software Libre su uso será imposible.

Sin embargo el CVS es un sistema de control de versiones de GNU y se difunde libremente bajo la licencia GPL, Es una aplicación cliente-servidor y está disponible para los sistemas operativos más difundidos, en cualquiera de sus versiones. Es extremadamente sencillo de instalar y utilizar. La información la almacena en un único servidor central (Centralizado), de modo que cada uno de los integrantes del equipo tendrá una copia completa del desarrollo, la cual se descarga directamente en el equipo del cliente, o sea, los usuarios trabajan sobre una copia de trabajo local. Por lo tanto, la red entre el cliente y el servidor debe estar preparada para realizar la sincronización a petición del usuario (*update* y *commit*), y de mantener la integridad y consistencia de los ficheros. Los archivos del control de versiones se almacenarán en el sistema de ficheros del servidor.

Inicialmente fue concebido para el sistema operativo Unix, pero actualmente existen varias interfaces gráficas que ocultan dichos comandos brindando al usuario interfaces amigables, facilitando así su

utilización. Entre ellas se pueden mencionar: TortoiseCVS (se integra con el windows explorer), Cliente CVS incorporado en el IDE Eclipse, Cervisia para Linux. A pesar de las funcionalidades que brinda tiene como limitación que los archivos en el Repositorio sobre la plataforma CVS no pueden ser renombrados, estos deben ser agregados con otro nombre y luego eliminados.

El protocolo CVS no provee tampoco una manera de que los directorios puedan ser eliminados o renombrados, cada archivo en cada subdirectorio debe ser eliminado y re-agregado con el nuevo nombre. CVS es incapaz de registrar los cambios en la estructura de directorios, pues no es posible mover, renombrar, ni copiar. Dichas operaciones se consiguen eliminando y añadiendo, pero con estas acciones se pierde el historial de cambios. Estos defectos se deben a que CVS usa internamente el sistema de almacenamiento de RCS[14], que solo registra cambios de contenido en ficheros individuales.

En CVS es necesario interrumpir el acceso al repositorio para crear copias de seguridad. No permite lo que se conoce como "conjuntos de cambios". Si un desarrollador sube un conjunto de cambios, estos van subiendo uno a uno, quizás al mismo tiempo que otro desarrollador realiza la misma operación. Como no es una operación atómica, no se puede asegurar el estado del repositorio tras su commit o transacción, es decir, no se asegura que el conjunto de cambios sea consistente. Se almacenan los ficheros binarios enteros (no sus diferencias entre versiones). Esto consume espacio en disco y ancho de banda.

Como gran desventaja se tiene el hecho de que para CVS el código fuente es difícil de mantener. CVS comenzó como un conjunto de scripts shell que usaban RCS e implementaban algoritmos desarrollados entre los años 60 y 80. El resultado del desarrollo experimentado es producto de sucesiones de parches, y no tiene un diseño fácil de entender o mejorar, lo que hace difícil su evolución.

La principal causa por la que ha quedado de un lado CVS está dada por el surgimiento en el año 2000, del hoy afamado Subversion. De modo que éste último constituye un excelente sistema de control de versiones. Se dice que fue diseñado para reemplazar al popular CVS pero realmente manteniendo sus ideas básicas, y basándose en la mejora de sus deficiencias y fallos. Es un software libre y se distribuye

[14] Revision Control System

bajo la licencia de tipo Apache/BSD. Subversion en una de las herramientas más utilizadas actualmente por su robustez y por brindar las siguientes características:

- Utiliza un sistema con repositorio centralizado.
- Implementa un sistema de ficheros versionado virtual que sigue los cambios sobre árboles de directorios completos a través del tiempo.
- Mantiene un verdadero historial de versiones, las cuales no sólo son de archivos, sino también de directorios y almacena versiones de los metadatos asociados a los directorios a través de copias y renombrados.
- Atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Brinda la posibilidad de elegir el protocolo de red. Tiene disponible un servidor de Subversion independiente y más ligero, el cual tiene un protocolo propio (svn) que puede estar encaminado fácilmente a través de un túnel SSH. Además que dicho protocolo propio, puede trabajar sobre http o https mediante las extensiones WebDAV[15] (más conocido como DAV). Por su interoperabilidad con WebDAV es posible acceder al repositorio con cualquier software que soporte dicho protocolo ("Web Folders" de Windows XP, Photoshop, etc).
- La capacidad de funcionar con un protocolo tan universal como el http permite compartir y versionar ficheros a través de la web. Cualquier infraestructura de red actual soporta dicho protocolo y universaliza las posibilidades de acceso. Si se desea puede utilizarse a través de Internet.
- Permite conectarse al servidor http Apache como un módulo de extensión. Proporcionando a Subversion una gran ventaja en estabilidad e interoperabilidad, y acceso instantáneo a las características existentes que ofrece este servidor como la autenticación, autorización, compresión de la conexión, etc.
- Ramificación y etiquetado eficientes ya que el coste de ramificación y etiquetado no necesita ser proporcional al tamaño del proyecto. Subversion crea ramas y etiquetas simplemente copiando el proyecto, usando un mecanismo similar al enlace duro. De este modo estas operaciones toman solamente una cantidad de tiempo pequeña y constante.

[15] Protocolo que amplía las posibilidades del HTTP/1.1 añadiendo nuevos métodos y cabeceras.

- Le da un mejor uso al ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.
- Subversion no tiene un equipaje histórico; está implementado como una colección de bibliotecas compartidas en C con APIs bien definidas y documentadas. Esto lo hace extremadamente fácil de mantener y reutilizable por otras aplicaciones y lenguajes.

Además de todas las funcionalidades que presenta se decide emplear para el Polo de Imágenes este sistema de control de versiones tanto por su potencial como por su soporte por parte de la comunidad. El mismo permite controlar en todo momento qué parte del código, el documento o la base de datos fue modificado, en qué momento y por quién. Con Subversion es posible poner a disposición de varias personas los ficheros relacionados con el proyecto, así como los cambios que se realicen serán guardados de forma tal que se pueda acceder a una versión anterior en cualquier momento.

La organización que mantiene Subversion hace que la coordinación del trabajo en equipo sea fácil e intuitiva. Esta herramienta deja mucho en que pensar, montar un repositorio de Subversion es fácil, rápido, seguro y se integra perfectamente con la plataforma que desea desarrollar, para ello contiene una serie de clientes que lo integran en entornos de desarrollo como:

- TortoiseSVN: interfaz más popular en el sistema operativo Windows. Cliente de Subversion integrado al shell de Windows. Disponible en 28 idiomas diferentes.
- Subclipse: plugin que integra Subversion al entorno de desarrollo Eclipse y como plugin alternativo Subversive.
- VisualSVN: plugin para Visual Studio con licencia comercial.
- AnkSVN: plugin para Visual Studio Open Source.
- RapidSVN: plugin para interacción con Linux.
- SvnX, y Zigversion: interfaces que pueden emplearse para Mac.
- Easyeclipse: es un paquete basado en eclipse, es una plataforma de desarrollo, con algunos plugins de código abierto.
- Sventon: Interfaz web.
- Versions: Interfaz de escritorio para Mac OS X.

De los múltiples sistemas de control de versiones existentes, uno de los más utilizados es Subversion, también conocido como SVN, que se ha popularizado bastante y sobretodo dentro de la comunidad de desarrolladores de software libre.

Por otro lado el mundo de los desarrolladores se está enfocando en uno de los nuevos sistemas de control de código distribuido: Git por ser extremadamente rápido y eficiente con el espacio. Es software libre y creado por Linus Torvalds para manipular el código fuente del kernel de Linux. Ha evolucionado de forma tal que algunos cálculos recientes estiman que para finales del 2008, un 80% de los proyectos y desarrolladores de Rails migrarían a Git. Se ha convertido en un sistema muy potente y fácil de usar. Contiene fuerte incidencia en la no linealidad de los cambios, por ende rapidez en la gestión de ramificaciones y mezclado de diferentes versiones.

A diferencia de los sistemas centralizados como SVN o CVS, este es un sistema de control distribuido lo que permite clonar el repositorio central, completo con todas sus ramas de trabajo e historia, y trabajar sus propios cambios en su repositorio clon, independientemente del central. Luego los cambios pueden mezclarse con el repositorio central para integrarlos a esa base de código. Los sistemas de control distribuido se prestan óptimamente al trabajo fuera de línea ya que mantienen una copia local del repositorio, por lo cual no se depende totalmente de la red, y los cambios que se hagan fuera de línea pueden mezclarse para integrarlos cuando se desee a otros repositorios remotos, sean centralizados o de otros desarrolladores.

Sin embargo hoy por hoy Git adolece de un gran problema y está relacionado con la facilidad de uso, muchas cosas pueden resultar sencillas de hacer con Git, pero hay que saber hacerlo. Dado que es un proyecto tan joven comienzan a vislumbrar algunos proyectos para su mejora como el TortoiseGit que aún es muy reciente pero prometedor, y una GUI[16] que rompe con todo lo conocido en el software libre con la idea de lograr una interfaz de usuario más amigable. Git está programado pensando en la eficiencia y confiabilidad del control de versiones.

[16] Interfaz gráfica de usuario

Muchas organizaciones dedicadas a la producción de software requieren de sistemas para la administración de proyectos y de seguimiento de errores (tracking) para administrar los tickets de los proyectos y llevar un seguimiento de los tiempos y porcentaje de las tareas de cada versión. Entre los más populares se encuentran Jira, Trac y Redmine.

Estas herramientas tienen como propósito dar seguimiento y manejar los defectos y errores que emergen durante la ejecución de cualquier proyecto, tarea que se ha convertido en estos días, en una tarea crítica y fundamental dentro de las organizaciones. Jira es una de las herramientas diseñada para mejorar el proceso de dar seguimiento a los defectos y errores, es decir mantener la trazabilidad de la información. Posee una interfaz gráfica versátil, fácil de entender para usuarios tanto administrativos como técnicos. Soporta casi cualquier plataforma de hardware, sistema operativo y plataforma de base de datos.

También grafica los procesos de trabajo en flujos modificables. En resumen, JIRA es una herramienta muy potente que te permite no sólo introducir defectos sino también gestionarlos. Sin embargo JIRA, no es una herramienta gratuita por tanto exige pago de licencia y es un poco cara, por lo que se descartaría como herramienta a emplear en el Polo de Imágenes.

Para la gestión de proyectos existen otras herramientas que prestan las mismas funcionalidades o tal vez mejores y que son Open Source, evidentemente es lo que se desea emplear, una herramienta gratuita que permita gestionar las tareas y errores vía web; lo cual permite que sea accesible por todos desde el navegador. Herramientas gratuitas y sencillas de usar como Trac o Redmine.

Trac permite enlazar información entre una base de datos de errores de software, un sistema de control de versiones y el contenido de una wiki. Esta herramienta hace mucho hincapié en la gestión de bugs y tareas pero no ofrece mucha herramienta visual, como un diagrama Gantt, calendario o registros de información referente a las tareas, o sea queda un poco corta en sus funcionalidades. Aunque tiene una multitud de plugins que hacen de él un gestor de proyectos software a tener en cuenta. Sin embargo el hecho de no tener todas sus funcionalidades integradas en la herramienta y de no admitir múltiples proyectos constituye una gran desventaja frente a Redmine. En este caso no sería recomendable emplear Trac, puesto que es mejor emplear una herramienta que haga un trabajo y no que de más trabajo.

Estas herramientas están pensadas para poner las distintas versiones o hitos, asociarle las tareas a realizar y los errores; llevar bien el control de cuánto se ha avanzado en el proyecto y cuánto queda, de ahí su gran importancia. Hoy en día Redmine se ha convertido en un competidor fuerte para el veterano Trac, son herramientas similares pero Redmine presenta mejoras sobre este. Con esta joven herramienta toda la administración se hace a través de la web y es bastante sencilla. Permite crear múltiples proyectos, configurar los repositorios de fuentes, dar de alta a los usuarios e incluso estos pueden autenticarse.

Este proceso de autenticación con Trac es un poco más complicado ya que se necesita o bien un comando que viene con Apache para dar de alta a un usuario, o ya viene con uno por defecto desde su versión 0.11 pero hay que habilitarlo en un fichero de configuración. Otra de las sorpresas agradables de Redmine es que soporta directamente varios sistemas de control de versiones, incluido CVS, a diferencia de Trac que sólo soporta Subversion, salvo que se le instalen más plugins.

A través de Redmine se puede navegar por el código, línea de tiempo, bugs y mucho más, además de soportar varios proyectos con subproyectos. Incluso en la página de acceso se presentan todas las tareas asignadas por rol en lugar de ir visitando proyecto por proyecto como en Trac. Redmine tiene foro por proyecto, brinda la posibilidad de subir documentos por proyecto, genera un gráfico de Gannt con la evolución del proyecto además presenta estadísticas de horas gastadas por proyecto, por componente, por persona, por tipo de tarea, etc. Se enlaza con Subversion permitiendo una GCS más completa por lo que hay que pensar en ella como herramienta a emplear en el Polo de Imágenes para los proyectos que lo integran.

2.3 Herramientas propuestas

Se propone como sistemas controlador de versiones por sus potencialidades y robustez montar un Repositorio Subversion. El mismo puede ser servido mediante cualquier servicio Web (Apache, IIS (Internet Information Server)) aunque Subversion incluye Svnserve que es un servidor ligero y autónomo que utiliza un protocolo propio sobre una conexión TCP/IP ordinaria. Es ideal para las instalaciones más pequeñas, o para organizaciones donde no se pueda utilizar un servidor completo Apache. En la mayoría

de los casos Svnserve es más fácil de instalar y se ejecuta más rápido que el servidor basado en Apache, aunque no tiene alguna de las características más avanzadas.

La configuración más flexible de todas las instalaciones de servidor posibles para Subversion es la basada en Apache. Aunque es un poco más complicada de preparar, ofrece beneficios que otros servidores no pueden dar y que son muy necesarios. Por ejemplo el servidor de Subversion basado en Apache utiliza el protocolo WebDAV que se utiliza por muchos otros programas. Se podría montar dicho repositorio como una carpeta web en el explorador de Windows y luego acceder a ella como cualquier otra carpeta en su sistema de ficheros. Puede utilizar cualquier mecanismo de autentificación que Apache soporte, incluyendo SSPI y LDAP. Dado que Apache es muy estable y seguro, automáticamente obtendrá la misma seguridad para el repositorio.

En este caso se propone Apache Web Server, por ser uno de los servidores más rápido que existe y principalmente por ser libre. Es el software más usado mundialmente para crear servidores http. Apache permite lograr que los repositorios Subversion del servidor estén accesibles desde el exterior**,** ya que es mucho más flexible a la hora de aplicar políticas de acceso y sistemas de autenticación de usuarios, potencialidad que viene como anillo al dedo cuando se trata de un equipo de desarrollo tan amplio como el que presenta actualmente el Polo de Imágenes.

Se considera Subversion como herramienta a regir el control de versiones teniendo en cuenta que la misma pertenece a la comunidad de software libre, por lo que su elección es un buen paso en camino a la política que quiere adoptar la Universidad de Ciencias Informáticas: la migración a software libre. Dicha herramienta ofrece magníficas prestaciones y que son muy convenientes, específicamente para el Polo de Imágenes, como la excelente integración que presenta con el entorno de desarrollo que actualmente se emplea: Visual Studio.NET 2005 en algunos proyectos; y en otros Visual Studio.NET 2008.

Para la integración con Visual Studio.NET se empleará el AnkhSVN que es una herramienta que se integra al IDE Microsoft Visual Studio 2005 y 2008, permitiendo el manejo gráfico del Repositorio y posibilitando el fácil acceso a la mayoría de los comandos de Subversion. La última versión estable es AnkhSVN 2.0 liberada bajo la licencia Apache.

Subversion al ser multiplataforma garantiza posteriormente la migración de plataforma o ambiente de desarrollo. En el caso del Polo de Imágenes se encuentra instalado como ambiente de desarrollo Windows, por lo que se requiere de un cliente como TortoiseSNV para la integración con Subversion. TortoiseSNV es un cliente gratuito de código abierto. El mismo posee numerosas características que lo hacen un buen cliente de Subversion, como su excelente integración con el shell de Windows Explorer, permitiendo manejar de forma gráfica el repositorio de SVN. Contiene varios iconos sobre impresionados, así el estado de cada carpeta y fichero versionado se indica por esos pequeños iconos y de esa forma, permite ver fácilmente el estado en el que se encuentra una copia de trabajo.

TortoiseSVN posibilita un fácil acceso a los comandos de Subversion, todos los comandos están disponibles desde el menú contextual del explorador, éste añade su propio submenú ahí. Se basa en el sistema de control de revisiones Subversion, un sistema general que puede ser utilizado para manejar cualquier colección de ficheros, incluyendo código fuente. Con un cliente como TortoiseSVN ni siquiera es obligado usar el Explorador de Windows, los menús contextuales de TortoiseSVN también funcionan en otros administradores de archivos, y en la ventana Fichero/Abrir que es común a la mayoría de las aplicaciones estándar de Windows.

Subversion también cuenta con varios clientes que le permiten su integración con otros entornos de desarrollo. Para Linux está diseñado RapidSVN, cliente gráfico que permite manipular los repositorios de Subversion y que es liberado bajo la licencia GPL. Enteramente escrito en el lenguaje de programación C++. Proporciona una interfaz fácil de usar para las características de Subversion por lo que es una de las alternativas más conocidas, es muy intuitiva y fácil de utilizar.

Por otro lado “La administración de Proyectos no es sólo fijar objetivos y preparar reportes de seguimiento. Es también dirigir personas. Para ello se requiere de habilidades diarias. Aunque puede dar por seguro que en muchas ocasiones, esas habilidades pueden constituir la diferencia entre el éxito y el fracaso”. [Tempus Handbook]

En cada proyecto es necesario darle un seguimiento y control a cada una de las actividades, así como los recursos humanos y materiales que intervienen en el desarrollo, es lo que normalmente se denomina

gestión de proyectos. Tiene como objetivos estimar el esfuerzo a realizar para desarrollar el sistema y planificar las tareas de dicho desarrollo, para ello también se pueden emplear herramientas que optimicen esta labor.

En función de las características que presenta el Polo de Imágenes, dada la magnitud de la cantidad de desarrolladores con que cuenta y de proyectos que se encuentran en desarrollo; lo más aconsejable es emplear herramientas software que sustenten el cumplimiento de la gestión de sus proyectos. Con este fin se propone emplear como herramienta Redmine por las buenas características que presenta. Esta herramienta presenta una interfaz web magnifica e incluso es muy configurable y con grandes utilidades como: poder definir las funciones y permisos para cada usuario, y estos a su vez pueden tener un papel diferente en cada proyecto. Con dicha herramienta también se pueden definir campos propios y personalizarlos para los proyectos y los usuarios.

La misma ofrece informes para revisar los estados de la configuración y que están disponibles en diferentes formatos como: texto, fecha, enteros, listas desplegables, casillas de verificación. Para el almacenamiento de la información Redmine funciona con diferentes gestores de base de datos como MySQL, PostgreSQL o SQLite. Todos estos puntos característicos que enriquecen la herramienta conllevan a exponerla como solución en la estrategia propuesta para apoyar la gestión de proyectos en el Polo de Imágenes.

2.4 Caracterización del proceso de GCS a aplicar

Desarrollar software implica dentro de sus primeros pasos describir la estructura que tendrá el producto esperado, así como identificar los elementos versionables que serán tratados en el proceso de manejo de configuraciones y construir, recolectar y etiquetar artefactos con el objetivo de mantener la rastreabilidad entre las versiones.

Es por ello que basado en las definiciones de los diversos autores que escriben sobre el tema (Babich, 1986) (Antonio, 2001), (Pressman, 2005), los cuales proponen diferentes tareas a ejecutar en el proceso de GCS, se formula una estrategia a tener en cuenta en el proceso de GCS en el Polo de Imágenes, para lo cual se puntualizan cinco importantes actividades a desarrollar:

- Identificar los Elementos de la Configuración del Software.
- Controlar las Versiones de la Configuración del Software.
- Controlar los Cambios.
- Auditar y Controlar internamente el proceso de GCS.
- Generar Informes de Estado.

Resulta entonces de extrema importancia la selección de estas actividades, por la trascendencia de su cumplimiento en la GCS, se podría decir literalmente que son decisivas en el éxito de un proyecto. Todas las actividades que se especificarán a continuación están definidas para cada uno de los proyectos que integran el Polo de Imágenes, pues cada proyecto está dedicado a la producción de un único software; y estas actividades son tenidas en cuenta para la creación de un producto software.

2.4.1 Identificación de los Elementos de Configuración de Software

En virtud de desarrollar la actividad de identificación de objetos en la GCS y para poder llevarla a cabo de forma exitosa, se tiene en cuenta a la autora Angélica de Antonio que define un conjunto de tareas incluidas dentro de esta actividad:(Antonio, 2001)

De modo que existen tareas dentro de la Identificación de los Elementos de Configuración (IEC) que urge realizarlas desde etapas tempranas del desarrollo de un software. Sin embargo hay otro conjunto de actividades que no necesariamente tienen que tener un orden, sino que se pueden efectuar de forma independiente a lo largo de todo el ciclo de vida. A manera ilustrativa se presenta el siguiente gráfico que describe los procesos de esta actividad:

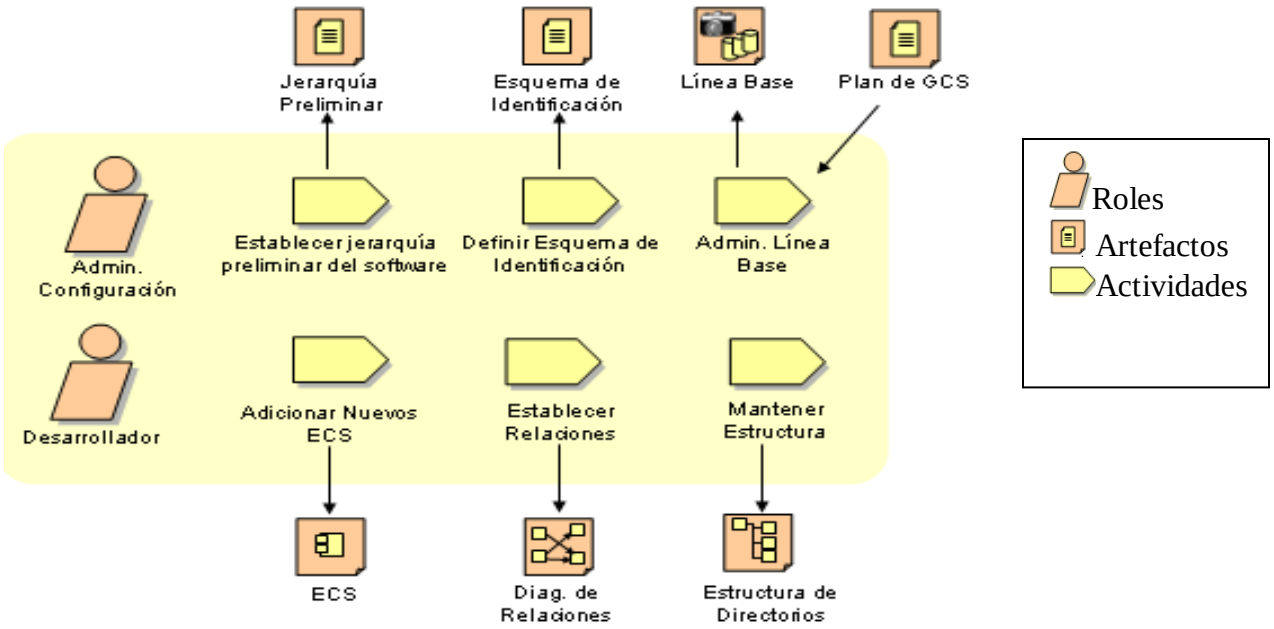


Fig. 2.3 Actividades, Roles y Artefactos en la Identificación de la Configuración.

Un papel importante es el que debe jugar cada rol en el proceso de IEC, en la gráfica se evidencian solo dos roles teniendo en cuenta que esta solo delimita el proceso para un solo proyecto. En el caso particular del Polo de Imágenes que abarca tantos proyectos pues la misma es aplicable para cada uno de los proyectos que lo integran. La propuesta de los roles para llevar a cabo el proceso de GCS se explica con anterioridad en el epígrafe 2.2.

2.5.1.1 Actividad de IEC

Para la IEC es importante que se establezca en la organización un método único y que sea conocido por todas las partes involucradas, para que todos los integrantes del proyecto se comprometan con sus objetivos y actúen uniformemente. Esta función tiene como objetivo organizar todos los ECS, lo que facilitará su localización en el Repositorio del proyecto y una mejor organización del producto, para ello se proponen las siguientes actividades:

Establecer la jerarquía preliminar del software

El primer punto es establecer la jerarquía preliminar del software, que tipos de elementos formarán parte de los controlables del proyecto actual, así como las posibles relaciones a establecerse entre ellos. Esta actividad tendrá como salida el documento de jerarquía preliminar, el cual queda compuesto por un conjunto de ECS y las relaciones posibles entre estos.

Seleccionar los Elementos de Configuración de Software

Uno de los requisitos vitales de esta actividad y que de no ser así dificultaría el trabajo de las fases subsiguientes, es que esta tarea debe ser realizada por personas que posean un gran conocimiento sobre la arquitectura del sistema. Es por ello que deben participar junto con el Administrador de la Configuración Principal, el Líder de Proyecto y el Arquitecto Principal en representación de los desarrolladores, ya que este rol en la práctica es el que construye la mayoría de los ECS.

En la actividad debe regir cierto nivel de formalidad en cuanto a los ECS que estarán bajo control de configuración, evitando así que se intenten adicionar nuevos ECS que no formen parte de los que se identificaron en la jerarquía preliminar del proyecto. Es por ello que todos deben ser analizados por el Administrador de Configuración Principal para su aprobación.

En aras de fomentar este clima de identificación de los elementos de configuración, se deben nombrar, almacenar y notificar los elementos que se deseen controlar en el proceso de desarrollo del software. Particularmente para el caso del Polo de Imágenes debido a la gran cantidad de información existente y al crecimiento elevado de la misma en el tiempo; para que se cumpla exitosamente esta actividad se propone lo siguiente:

Nombrar y Etiquetar los ECS

Cada objeto o elemento de configuración debe tener un conjunto de características distintivas que lo identifiquen de manera exclusiva y se propone tener en cuenta los siguientes:

- Identificador (número, letra, ambos. No ambiguo).

- Nombre (descriptivo).
- Tipo (documento, código, producto).
- Localización.
- Fecha.
- Versión (mayor, menor, revisión).
- Estado (Ej. Para un documento En elaboración, finalizado, revisado, aceptado).
- Proyecto/producto.

Con el objetivo de organizar los ECS y de optimizar la búsqueda de la localización que tienen en el repositorio se propone clasificarlos en categorías según al tipo que pertenezcan. Por ejemplo:

BD- Base de Datos.
DOC- Documentación.
REQ- Fichero de Requerimientos.
CUS- Casos de Uso del Sistema.
BIN- Ejecutables.
PLN- Plan de Proyecto.
COM- Componentes.
APL- Aplicaciones.

En la práctica cuando múltiples desarrolladores interactúan con el sistema controlador de versiones se generan un número elevado de versiones de ECS, las cuales no todas tiene por qué ser importantes o corresponder con un hito del proyecto. Por tanto se hace necesario que los desarrolladores usualmente distingan las versiones mediante el uso de etiquetas. Estas deben tener un formato estándar que se define en los momentos iniciales de la IEC.

Notificar los ECS.
El Gestor de Configuración de cada Área Temática tendrá como tarea informar los elementos identificados una vez terminado este proceso a todos los desarrolladores para cada proyecto. Luego el Gestor de Configuración Principal será el encargado de mantener documentado para futuras actualizaciones, ya sea

la adición de nuevos elementos, modificación, el estado de los mismos o eliminar los que no se utilicen o que carezcan de validez.

Definir las relaciones entre los ECS en la configuración

Teniendo en cuenta que un ECS es la información que se crea como parte del proceso de Ingeniería de Software, que lo mismo puede ser un documento, un conjunto completo de casos de prueba o un componente de un programa, es importante considerar los ECS como objetos de configuración y que cada objeto se encuentra relacionado con otros.

Para la automatización del proceso de GCS se propone apoyarse en una base de datos de administración de configuración CMDB[17], que contendrá detalles relevantes de cada elemento de configuración, cuya principal característica, además de almacenar información sobre los distintos elementos de configuración, tenga la capacidad de almacenar información sobre las relaciones entre los distintos ECS y sobre los artefactos de procesos relacionados con estos.

Los ECS deben ser organizados como objetos de configuración que deben ser catalogados en la base de datos de cada proyecto con un nombre único, para su identificación. Cada ECS contendrá entonces un nombre, atributos, y su conexión con objetos mediante relaciones. De modo que si se realiza algún cambio sobre un objeto las interrelaciones señalarán a que otros objetos afectará.

Con el objetivo de mantener y proporcionar información precisa a todos los procesos que lo necesiten sobre los ECS, y sus relaciones en un modelo lógico, se propone crear una base de datos relacional para implementar un espejo del subconjunto de la información almacenada en la herramienta controladora de versiones. Todo el almacenamiento en dicha herramienta controladora de versiones será desde el punto de vista físico y en la base de datos lógico. Desde dicha base de datos relacional, solamente se tendrá una referencia al camino en el repositorio real de control de versiones en donde se encuentran los ECS para su recuperación.

[17] Repositorio de información donde se relacionan todos los componentes de un sistema de información, ya sean hardware, software, o documentación.

Para ello se propone el uso del gestor de base de datos MYSQL y de la herramienta para modelar bases de datos, Db Designer, producto de ser una herramienta de software libre que es muy utilizada debido al gran número de funcionalidades que presenta. Dicha herramienta proporciona un sistema visual de diseño de bases de datos ODBC[18] que integra diseño de bases de datos, modelado, creación y mantenimiento en un único entorno sin fisuras. Está disponible para Linux y Windows.

Establecimiento de la Línea Base para cada proyecto.

Según se plantea en el capítulo anterior una línea base provee un punto estable, es una foto de los artefactos del sistema en un momento dado. Es un conjunto de versiones establecidas de archivos y directorios creados al alcanzar ciertos hitos en el desarrollo de un proyecto. Su establecimiento es fundamental para la GCS constituyendo la base oficial para el trabajo subsiguiente. Provee un estándar a partir del cual se realizarán los cambios correspondientes a solicitudes de cambio aprobadas, logrando que los cambios sean controlados y así no poner en riesgo trabajos en curso.

Primeramente se deben identificar claramente los productos de cada fase del ciclo de vida, aunque la línea base puede ser establecida en diferentes hitos del desarrollo de software y puede ser creada para un sistema completo o para subsistemas, pero en el caso del Polo de Imágenes que se encuentra subdividido por proyectos, se propone establecer líneas base solo a nivel de proyectos.

Lo más aconsejable es crear líneas base cada vez que concluya un hito del proyecto. Generalmente las más utilizadas son la funcional, la de desarrollo y de producto. La línea base funcional corresponde a la especificación de requisitos software y del sistema que han sido ya revisados. La de desarrollo representa la evolución de la configuración del software en determinados momentos seleccionados del ciclo de vida. La línea base del producto corresponde al producto finalizado y entregado para su integración en el sistema.

[18] Open DataBase Connectivity, es un estándar de acceso a bases de datos que utilizan los sistemas Microsoft.

Cada línea base que se realice en un proyecto junto con el nivel de autorización requerido para aprobación se recomienda que queden bien identificadas y documentadas en el Plan de Gestión de la Configuración.

Definir las prácticas de la Línea Base

En cada Área Temática establecer las líneas base de cada proyecto será tarea del Líder de Proyecto en conjunto con el Arquitecto Principal y llevada a la práctica por el Gestor de Configuración Principal. Teniendo como actividades fundamentales:

- Establecer la línea base a nivel de proyecto, alcanzando los objetivos para cada una de las fases definidas según la metodología empleada.
- Definir que ECS formarán parte de la línea base en cada hito en el desarrollo.

Luego a la hora de gestionar los cambios sobre la línea base podrá ser analizada por el Líder de Proyecto y sus integrantes, siempre que los cambios sobre la línea base tengan impacto dentro del ámbito del proyecto. En otro caso pues debe elevarse la solicitud de cambio a una comisión de gestión de cambios a nivel de proyecto, en este caso el equipo de GCS que conforman el Comité de Control de Cambios, puesto que debe efectuarse el cambio mediante un proceso formal.

Si se procede a un proceso formal de cambio, el Gestor de Configuración Principal será el encargado de retirar el ECS sujeto a cambio e integrará el mismo una vez actualizado, siempre que haya sido revisado y aprobado por el CCC. No constituye una buena práctica eliminar del Repositorio el ECS que se retire de la línea base, lo más aconsejable es almacenarlos en una Biblioteca de ECS, pues en algún momento será necesario consultar dicho ECS y para ello existe otro procedimiento:

1. Realizar solicitud al Gestor de Configuración Principal o al Arquitecto Principal del proyecto, de la versión del ECS necesaria y con ellos argumentar la necesidad de su empleo.
2. De ser aprobada la solicitud se obtendrá una copia del ECS de la Biblioteca de ECS y será entregada al solicitante de la misma.
3. El desarrollador, al finalizar, debe eliminar de su espacio de trabajo el ECS.

Definición y Establecimiento de las Bibliotecas de Software

En el capítulo anterior quedó bien fundamentada la importancia del establecimiento de las bibliotecas de software, actividad fundamental en la IEC ya que: una biblioteca de software es una colección controlada de software y/o documentación relacionada cuyo objetivo es ayudar en el desarrollo, uso o mantenimiento del software. (Antonio, 2001)

En la realidad en cada proyecto se definen las bibliotecas como un concepto interno, de modo que para los desarrolladores el contacto solo será en su espacio de trabajo y por lo tanto la dirección del proyecto define la estructura que tomará el Repositorio. No solo definirlas es lo importante sino hay que tener en cuenta otros procedimientos como:

- El establecimiento las Bibliotecas de Software.
- Como introducir los elementos en la biblioteca.
- Controlar el acceso a la biblioteca.

Se considera necesario que todas las bibliotecas implementadas estén respaldadas por la herramienta utilizada para la GCS con un alto soporte en cuanto a la transferencia de ficheros, acceso y asignación de permisos. Lo más esencial es que cada proyecto contenga una entrada o directorio raíz, según la herramienta recomendada (Subversion) se propone que cada Área Temática contenga un directorio raíz, como se muestra en la figura:

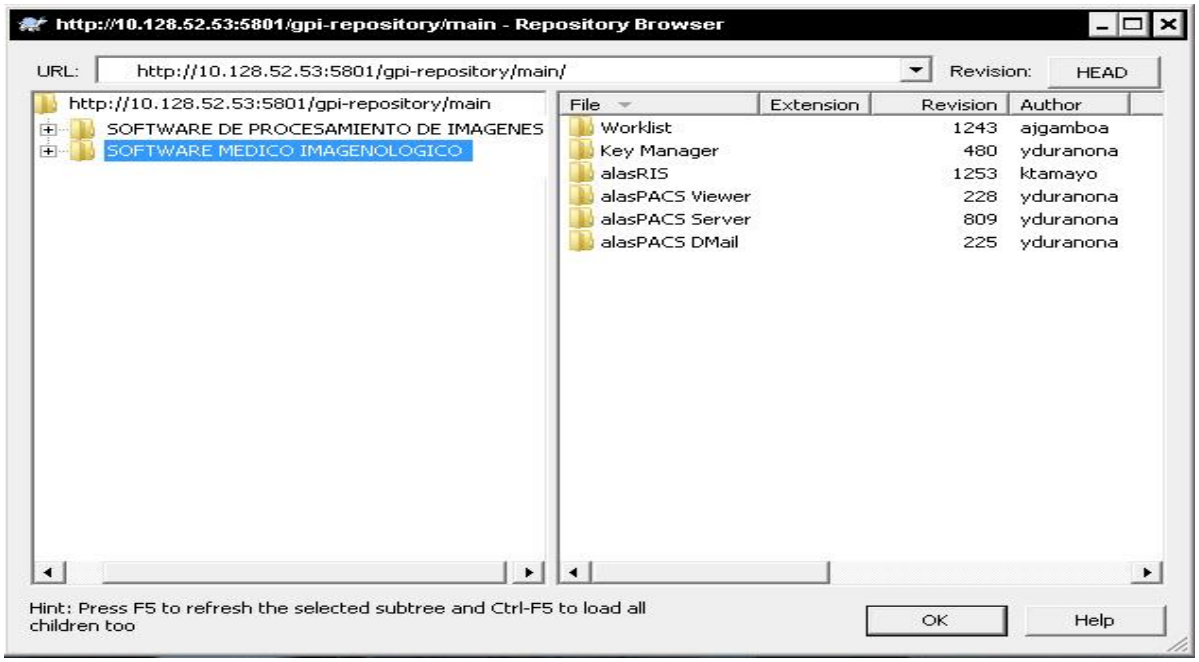


Fig. 2.4 Estructura del Repositorio para cada Área Temática.

Luego para cada Área Temática existirá una Biblioteca de trabajo para cada proyecto que comprenda el área de trabajo donde los analistas y diseñadores elaboren los documentos del proyecto y donde los programadores desarrollen el software, es decir, donde se realice la codificación y pruebas unitarias.

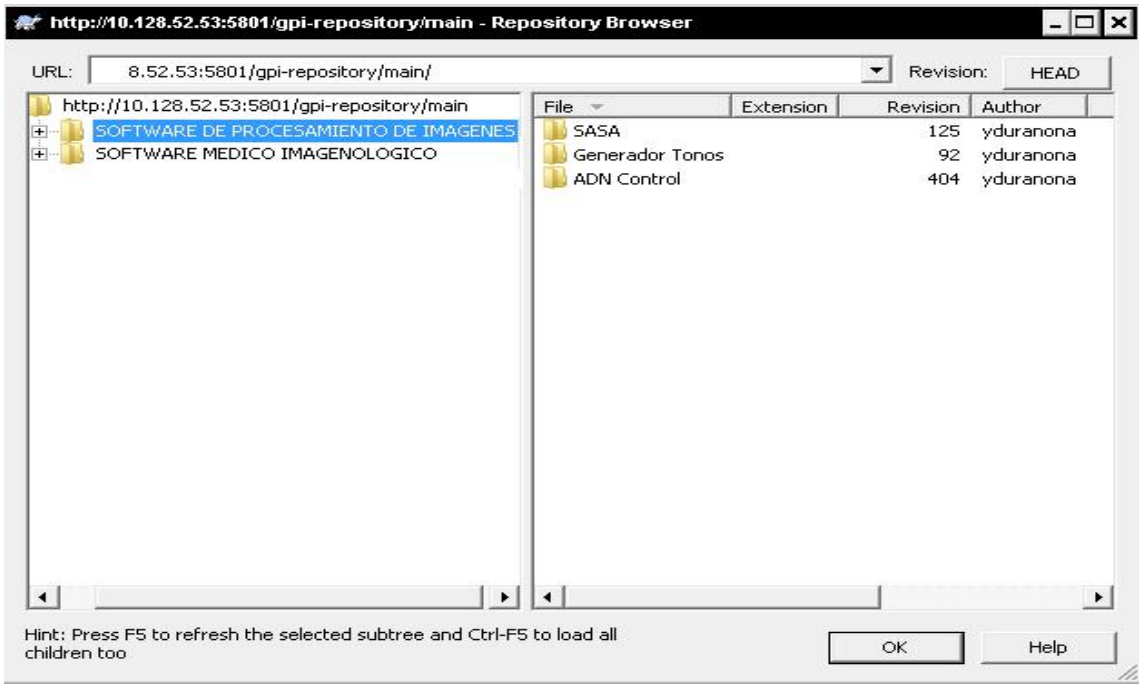


Fig. 2.5 Estructura dentro de cada Área Temática.

Una propuesta de cómo puede funcionar en su interior, es establecer un espacio para la Línea Base, un espacio para Desarrolladores, espacio para la Gestión de Proyectos y otro espacio para Entregables (*releases*).

En las bibliotecas de trabajo, se almacenan los ECS que se estén desarrollando por los integrantes del proyecto, por lo que los mismos no están aún en estado estable. Así el movimiento de una versión de un ECS de una biblioteca a otra depende del nivel de estabilidad, de madurez o de acabado que presente la versión del ECS. Una vez terminada su implementación debe ser probada para tener la seguridad que los cambios se realizaron de una manera correcta y luego su movimiento a la Biblioteca de Integración donde se toman los ECS para su integración en ECS de nivel superior del sistema.

En caso de resultar exitosa la prueba la versión del ECS puede moverse a la Biblioteca de Soporte al Proyecto, que en esta biblioteca se almacenan los ECS aprobados y transferidos desde la Biblioteca de Trabajo o desde la Biblioteca de Integración. Cuando un elemento pasa a esta biblioteca, se encuentra sujeto a un Control de Cambios interno y semiformal.

Cuando se tiene la seguridad que la versión de un ECS se encuentra liberada para la entrega al cliente o su distribución, se mueve a una Biblioteca Maestra donde se encontrarán sujetos a un Control de Cambios formal y estricto. En esta biblioteca se almacenan los Releases del sistema.

2.4.2 Control de Versiones

En el Polo de Imágenes cada proyecto contiene un numeroso equipo de involucrados por lo que es imprescindible tener controlada la versión sobre la cual se esté trabajando en cada uno, ya que el desconocimiento o confusión sobre las versiones que constituyen el estado último de los productos pueden destrozar la planificación, y en ocasiones todo el trabajo realizado por los equipos. En virtud de evitar este descontrol se considera un elemento clave en la solución propuesta, el uso de una herramienta controladora de versiones, que posibilite guardar todas las versiones para permitir la gestión eficaz de las liberaciones de los productos y permitir que los desarrolladores regresen a versiones anteriores siempre que sea necesario.

Por lo que es conveniente guardar versiones antiguas ya que en ocasiones estas versiones pueden seguir formando parte del sistema, es por ello que se recomienda mantenerlas y así poder reutilizar elementos que tal vez aparezcan en ellas y que en algún momento fueron eliminados. Para crear, identificar y almacenar nuevas versiones generalmente se emplean dichas herramientas de control de versiones pero el problema que tiene esta opción es que el número de revisiones crece desmesuradamente, y puede llegar a ser imposible almacenarlas todas. Es por ello que otra opción puede ser permitir a los desarrolladores que decidan cuando crear una nueva versión, así este puede modificar un objeto cuantas veces desee en la misma versión y luego subirla al repositorio.

Hoy en día se hace imprescindible poner el código fuente bajo control y sobre todo si se trata de un equipo de desarrollo tan amplio como el del Polo de Imágenes. Por ello se hace tanto hincapié en el empleo de una herramienta controladora de versiones en este caso se recomienda Subversion ya que garantiza la comunicación entre los desarrolladores, el manejo de los lanzamientos, administración de fallos, estabilidad entre el código y los esfuerzos de desarrollo experimental y autorización en los cambios de los desarrolladores.

Lo más inteligente y eficiente es centralizar el almacenamiento de todos los componentes de un mismo sistema en un repositorio común, de esta forma se obtiene una mejor organización y control del trabajo en equipo a lo largo de todo el ciclo de vida del software.

El control de versiones no solo incluye el uso de herramientas, también combina procedimientos que regulan como llevar a cabo esta actividad y tiene como responsables al Líder de Proyecto y al equipo de GCS. El núcleo del control de versiones es la gestión de cambios, y para ello se deben tener en cuenta procedimientos como:

- Cuando se recibe notificación del cambio o del problema a solucionar, y si el mismo afecta ECS en estado estable o liberado se debe traspasar la solicitud al Proceso de Gestión de Cambios (sección 2.4.3).
- De proceder el cambio se bloquea la última versión en el Repositorio y se le facilita al desarrollador para que realice la modificación. Si no proceder se informa al solicitante y se cancela la solicitud.
- Luego de realizado el cambio, el ECS debe ser probado por el equipo de calidad interna del Polo de Imágenes, con el objetivo de revisar que el ECS cumpla los requisitos establecidos. De no ser así se debe regresar al paso 2.
- Actualizar la versión del ECS nueva en el Repositorio.
- Notificar a todo el equipo de desarrollo de la nueva versión del ECS.
- Almacenar la versión antigua en la biblioteca de software del proyecto.

2.4.3 Control de Cambios

Indudablemente durante el desarrollo de un proyecto por muy completa que sea la definición de los requisitos, aparecerá la necesidad de ciertos cambios, motivados por ajustes en las necesidades del cliente, por la propia tecnología, por modificación o aparición de normativas, por reemplazo de los promotores o responsables del proyecto, o muchas otras razones. No esperarlos es negar la realidad y no estar preparados conlleva rápidamente al caos. Es por ello que se propone un grupo de actividades a tener en cuenta para el Control de Cambios que combine procedimientos fuertemente aliados a las herramientas automáticas para proporcionar mejores resultados.

1. Definir el proceso de cambio.
2. Establecer las normas, procedimientos y formularios del control de cambios.
3. Mantener las líneas base.
4. Procesar los cambios.
5. Controlar la liberación de los productos software.

El uso de herramientas puede ayudar a que el proceso de solicitud de cambios operé más eficientemente; la automatización puede apoyar para iniciar solicitudes de cambio, hacer cumplir el flujo del proceso de cambios, capturar las decisiones del comité del CCC y obtener reportes acerca de la información del proceso. Sin embargo, hay que tener bien claro que una herramienta no es un proceso. Emplear una herramienta para administrar los cambios de software nunca reemplazará a un proceso bien definido, que describa el contenido y la forma de procesar las solicitudes de cambio.

Es por ello que como primer aspecto necesario a tener en cuenta al comienzo de cada proyecto es establecer de forma precisa cual será el proceso de Control de Cambios que se va a efectuar. Teniendo en cuenta la definición de Angélica de Antonio para este proceso de Control de Cambios, existen tres tipos fundamentales y son los que se proponen tener en cuenta: Control de Cambios informal, Control de Cambios semi-formal y el Control de Cambios formal, antes vistos en el capítulo 1.

Control de Cambios Informal

Cuando un cambio no es de gran impacto sobre la línea base del proyecto o aún el ECS no ha pasado a formar parte de ella. Al ser así se podrá realizar cualquier cambio justificado sobre el ECS y en este caso no es necesario generar una solicitud de cambio, ni una orden de cambio aunque sí, el cambio debe evaluarse y mantenerse su seguimiento.

Control de Cambios Semi-formal o a nivel de Proyecto

Una vez que el ECS pasa la RTF y se convierte en una línea base del proyecto, el proceso de Control de Cambios toma un carácter más serio y es tenido en cuenta a nivel de proyecto. Si el cambio es local puede ser aprobado por el Líder de Proyecto, pero si el cambio tiene algún impacto sobre otros ECS la solicitud de cambio entonces si debe ser evaluada por el CCC. En el marco de este contexto se tiene

como resultado de la investigación y estudios realizados tener en cuenta los pasos que define Pressman para este procedimiento, por concretar un proceso formal, ver (Anexo 3).

Cuando se reconoce la necesidad de un cambio por parte de algún integrante de los proyectos que existen, se presenta un informe del cual se comentará más adelante para ser evaluado por la ACC[19], en el caso del Polo de Imágenes esta debe ser constituida por el Líder del Proyecto y el equipo de Gestión de Configuración. Por cada cambio aprobado se genera una OCI[20] que describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y de auditoría.

El objeto sometido al cambio es "dado de baja" de la base de datos para que se efectúe el cambio y que se le apliquen las actividades de calidad. Luego el objeto es "dado de alta" y se usan los mecanismos de control de versiones. En este procedimiento de baja y alta existen dos elementos importantes: el control de acceso y control de sincronización, el de acceso gobierna los derechos de los desarrolladores del software a acceder y modificar objetos de configuración concretos y el de sincronización asegura que los cambios en paralelo, realizados por personas diferentes, no se sobrescriban mutuamente.

Control de Cambios Formal

Una vez distribuido el producto de software a los clientes se instituye el control de cambio formal, y a partir de ese momento cada cambio deberá ser aprobado por el Comité de Control de Cambios. Dado que actualmente en el Polo de Imágenes existen productos en etapa de despliegue y comercialización con hospitales tanto en Cuba como en Venezuela, productos que por su gran envergadura requieren de un funcionamiento eficaz. Para mantener la integridad de dichos productos es preciso establecer procedimientos adecuados para la planificación y gestión de los cambios como por ejemplo los siguientes pasos:

1. Realizar la solicitud de cambio, ya sea provocada por un problema detectado o por algún cambio en los requisitos, requerido por el cliente. Debe efectuarse mediante la plantilla de solicitud de cambio.

[19] Autoridad de Control de Cambios

[20] Orden de Cambio de Ingeniería

2. Registrar la solicitud de cambio.
3. Aprobación o rechazo inicial de la solicitud de cambio. El responsable de esta tarea será el Comité de Control de Cambios.
4. Evaluar la solicitud del cambio para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y el coste estimado del cambio. Si la solicitud es aceptada se genera un Informe de Cambio.
5. El Informe de Cambio se presenta al Comité de Control de Cambios y si es considerado beneficioso se genera una OCI, que luego es asignada a alguno de los desarrolladores de software para que se encargue de llevar a cabo el cambio.
6. Se realiza el cambio y una vez finalizado se certifica mediante una RTF para comprobar que se ha efectuado correctamente. Luego el objeto pasa a la Biblioteca de Soporte del Proyecto.
7. Informar del cambio al originador.

Proporcionar un mecanismo riguroso para controlar los cambios también requiere de mantener una constancia escrita de cómo, quien y cuando se desea realizar el cambio. Es por ello que se propone hacer uso de las plantillas estándar que propuso Calidad Interna de la Infraestructura Productiva para todos los proyectos de la Universidad de las Ciencias Informáticas. Es importante contar con estas para así obtener un registro de incidencia de todos los cambios que deban ser realizados sobre alguna versión del producto, en aras de corregir un error encontrado o mejorar alguna funcionalidad.

Para evitar que los cambios se apoderen de la marcha de los proyectos en el Polo de Imágenes y conduzcan al fracaso, es importante adoptar y cumplir cada procedimiento descrito para la planificación y gestión del control de los cambios. Importante también mantener informado de los mismos al equipo de desarrollo en todo momento.

2.4.4 Auditoría y Control Interno del proceso de GCS

Las auditorías a la configuración en ocasiones se consideran dentro de la garantía de calidad en conjunto con las actividades de validación y verificación, es que en la práctica es un punto de intersección entre todas ellas. Controlar los cambios en el software ayuda a incorporar el orden en su desarrollo, sin

embargo el éxito del proceso de GCS depende de comprobar que los cambios hayan sido implementados correctamente, de lo cual se encargan las auditorías a la configuración.

En vísperas a una mejor organización en cuanto a las tareas que se deben realizar en esta actividad de control interno en el Polo de Imágenes, se propone implementar las auditorías enfocadas a controlar el proceso de GCS en cada proyecto. Estas auditorías tendrán como objetivo asegurarse de que los cambios se realizan correctamente, y no solo basta con evaluar y autorizar el cambio sino que hay que asegurarse que se han realizado correctamente.

Requieren de personal experimentado, y con un gran conocimiento del proceso de desarrollo. Sin embargo, debe ser realizada por personal ajeno al equipo de desarrollo técnico para mantener la objetividad de la auditoría. (Antonio, 2001) Según lo planteado por la autora lo más convenientes entonces será ceder la tarea de auditar el proceso al equipo de calidad interna del Polo con apoyo del Administrador de la Configuración de cada Área Temática en el empleo del paquete Exploración del Repositorio para la interacción y navegación en el mismo.

Existen dos tipos de auditorias: auditoría funcional y auditoría física. Según el proyecto en cuestión las auditorías a ejecutar deberán ser bien planeadas y llevadas a cabo de forma estricta cumpliendo las fases del proceso de auditoría: planeación, ejecución e informe.

Para comprobar que se han completado todas las pruebas necesarias para el ECS auditado, y que satisfacen los requisitos impuestos sobre él se deben realizar las auditorías funcionales. Medio por el cual en cada proyecto se asegura que los desarrolladores han hecho su trabajo de forma que se satisfacen todas las responsabilidades externas. También garantizaría que la línea base de cada proyecto corresponde con la descripción de sus elementos.

Se propone efectuar también de forma periódica auditorías físicas, que tienen lugar inmediatamente después de haberse superado la auditoría funcional, tienen como propósito determinar si las especificaciones de diseño, producto y otros documentos referenciados representan el software luego de codificado y probado para uno o varios ECS específicos, teniendo en cuenta las siguientes actividades:

- Verificar la adecuación, integridad y precisión de los elementos físicos de documentación que constituyen la Línea Base.
- Determinar una lista de requerimientos funcionales, no funcionales, interfaz, desempeño, etc. que serán revisados durante el proceso de auditoría física.
- Seleccionar una lista de módulos o componentes que deben ser auditados. Los módulos se obtienen a través de la línea base entregada por el equipo de desarrollo y permite definir que módulos están relacionados durante el proceso de desarrollo del proyecto.
- Elegir los elementos de configuración de mayor prioridad.
- Comprobar que los ECS correctos se han incorporado en una construcción específica y que la documentación está actualizada y es consistente con la versión que se ha construido.
- Como resultado del proceso de auditoría física se obtiene el Informe de Auditoría Física y el Registro de no Conformidades.

2.4.5 Generación de Informes de Estado

Uno de los elementos importantes en el desarrollo de software es la comunicación entre sus integrantes, sin embargo una de las formas que viabiliza este proceso es mediante la generación de informes de estado. Esta actividad precisamente consiste en informar sobre el estado de los ECS y su finalidad es mantener a los desarrolladores y profesionales alertas ante los cambios importantes. Deben ser elaborados por el Gestor de Configuración o Líder de cada Proyecto, y a pesar de que pueden ser definidos en un inicio del proyecto, los diferentes tipos de informes que se deseen generar pueden ir determinándose a lo largo del desarrollo en dependencia de las situaciones que se vayan presentando y en la experiencia que se vaya adquiriendo.

Existen dos grandes grupos de categoría que generan los informes: los registros e informes, como una forma de mantener constancia escrita de todo lo ocurrido en cada proyecto durante su evolución. Es por ello que no se pueden dejar de tener en cuenta en cada proyecto del Polo de Imágenes los siguientes.

- Registro para almacenar toda la información que contiene cada ECS.

- Documento que registre la información relativa a cada línea base del proyecto lo mismo cuando sean creadas o actualizadas. Debe contener identificador de la línea base, fecha de establecimiento, ECS que la componen y las versiones.
- Registro de resumen acerca de cada solicitud de cambio obtenida del formulario de solicitud de cambio. Solo con los cambios registrados en un período de tiempo.
- Registro sobre el estado en que se encuentran cada uno de los ECS existentes, los mismos pueden estar en (Desarrollo, Prueba, Estable, Liberado).

Conclusiones

En el presente capítulo se mostró como en la solución propuesta para el desarrollo del proceso de GCS, en el Polo de Imágenes, se brindan una serie de técnicas y procedimientos sólidos encaminados a proporcionar mejorías en dicho proceso. Los mismos cumplen con los objetivos propuestos inicialmente, dada la necesidad de organizar, administrar y centralizar el proceso de GCS en el Polo de Imágenes. Para ello se afirma la utilización de dos potentes herramientas Subversion y Redmine como estrategia para automatizar diversas actividades de la GCS, lo que ha desempeñado un papel primordial y de vital importancia en el momento de su aplicación.

CAPÍTULO 3. APLICACIÓN DE LA SOLUCIÓN PROPUESTA

El presente capítulo describe el proceso de aplicación de la solución propuesta para el Polo de Imágenes. Aunque por sus características este cuenta con dos Áreas Temáticas y cada una de ellas formada por varios proyectos; por lo que resulta sumamente complejo, sino imposible llevar a cabo la GCS a nivel de polo.

Por este motivo se decide proceder a la aplicación de la propuesta para la GCS de forma gradual comenzando por el proyecto alasRIS: Sistema de Información Radiológica, encargado de gestionar de forma eficiente toda la información radiológica de pacientes en los centros hospitalarios o clínicas radiológicas, del Área Temática Software Médico Imageneológico. El sistema alasRIS fue desarrollado en el seno de la facultad # 7 de la Universidad de las Ciencias Informáticas, por un grupo de estudiantes y profesores.

3.1 Antecedentes del proceso de GCS en el proyecto alasRIS

El proyecto alasRIS surge como sucesor de Cassandra XWeb, el cual se dedicaba exclusivamente a la gestión de los informes de radiología; por lo que alasRIS, que ya se encuentra en su primera versión, la cual está numerada como 2.0 y la misma está siendo desplegada en la República Bolivariana de Venezuela.

Recientemente se están dando los primeros pasos para comenzar una nueva versión del proyecto con las mismas perspectivas, con el objetivo de ganar en madurez y eficiencia en su desarrollo, de modo que este nuevo proyecto constituirá la base de ejemplo para la aplicación de la solución propuesta para llevar a cabo la GCS.

En los inicios del desarrollo el proyecto se empleaba como sistema de control de versiones Microsoft Visual SourceSafe, pero dada las limitantes del mismo por ser un software privativo, se optó como estrategia migrar a un software que fuera libre y potente como Subversion, tanto para los artefactos de construcción del producto como para los artefactos de documentación asociados al proceso de desarrollo.

La gestión de los cambios al sistema era un tanto ineficiente por lo que se han establecido los pasos fundamentales para llevar a cabo un procedimiento formal, así como el establecimiento de un equipo oficial de control de cambios, equipo tradicionalmente conocido como Comité de Control de Cambios.

El proyecto alasRIS también se encuentra inmerso en el programa de mejoras propuesto por la Dirección de Calidad de la Infraestructura Productiva; por lo que se generarán una serie de nuevos artefactos que antes no se concebían en los proyectos productivos de la Universidad y que constituyen elementos de importancia notoria. Esto posibilita en gran medida que se generen más Informes de Estado que antes no se realizaban cabalmente y que en algún momento conllevaron una desinformación parcial del equipo de proyecto sobre las condiciones reales del mismo.

El conjunto de situaciones expuestas dan lugar a la concepción y aplicación de estrategias concretas por parte de la dirección del proyecto para llevar a cabo la GCS en el mismo, y la puesta en práctica de la solución propuesta en el presente trabajo para la realización de la GCS de forma cabal y lo más eficiente posible.

3.2 Aplicación de la solución propuesta en el proyecto alasRIS

Se tiene en cuenta como primer paso en la aplicación de la solución propuesta, definir las responsabilidades a cada uno de los miembros del proyecto, según los roles establecidos para enfrentar el proceso de GCS en el proyecto alasRIS. Existe un designado para el rol de Gestor de la Configuración Principal en el Polo de Imágenes, el cual tendrá como responsabilidad administrar el repositorio, así como poner en práctica todas las actividades que comprende la GCS.

El Líder del Proyecto alasRIS debe actuar como gestor de la configuración en su proyecto específicamente asumiendo el papel de presidente del Comité de Control de Cambios. Para llevar el

proceso de Control de Cambios se estableció dicho comité internamente para el proyecto alasRIS quedando conformado por el jefe de proyecto como máximo responsable en conjunto con el Arquitecto Principal y algún responsable del aseguramiento de la calidad. Se definió emplear la planilla estándar de la Infraestructura Productiva para la solicitud de cambios. Ver (Anexo 4).

El proceso de GCS debe estar soportado por herramientas CASE[21], bien para apoyar alguna de las tareas del gestor, o bien integrando algunas o todas las tareas que se deben llevar a cabo. Es por ello que como otro paso fundamental se tiene la idea de emplear herramientas que posibiliten en gran medida almacenar un historial de las versiones y cambios que se van generando en el desarrollo del sistema. Dando lugar a la instalación de un Repositorio Subversion que almacena tanto el código fuente como la documentación del proyecto. Se definió por cuestiones de optimización confeccionar un repositorio para el código fuente y otro para la documentación, facilitando también de este modo la migración de la documentación a una herramienta de gestión documental en el caso de su utilización en el futuro.

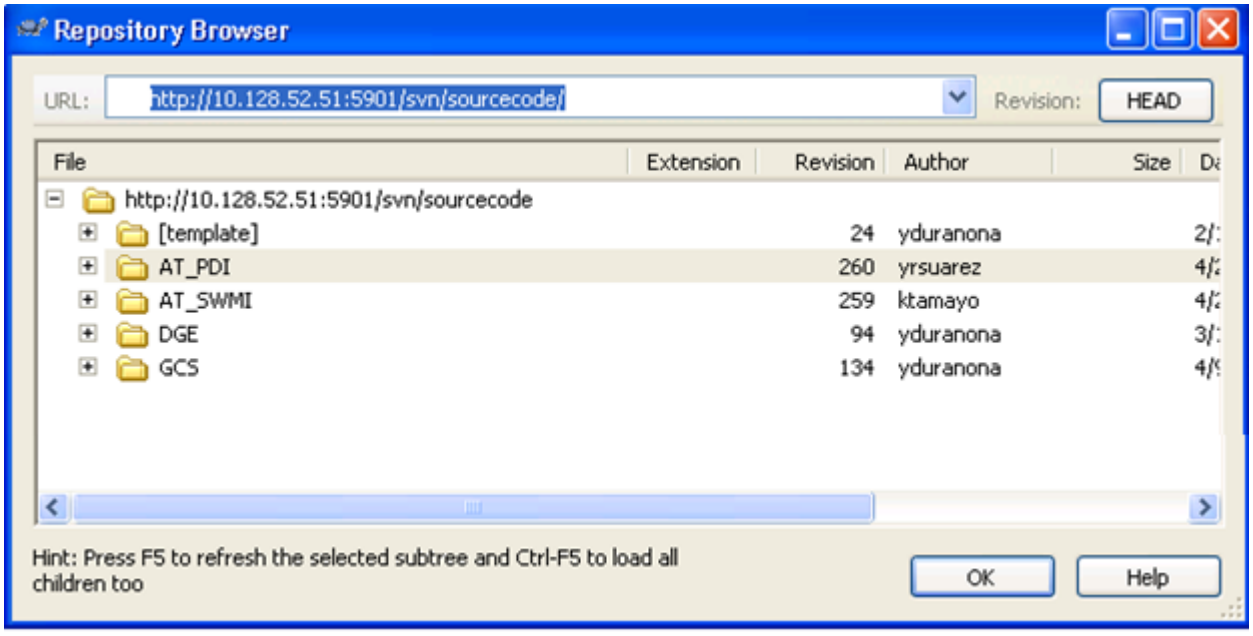


Fig 3.1 Estructura del Repositiro Subversion para el código fuente.

[21] Computer Aided Software Engineering

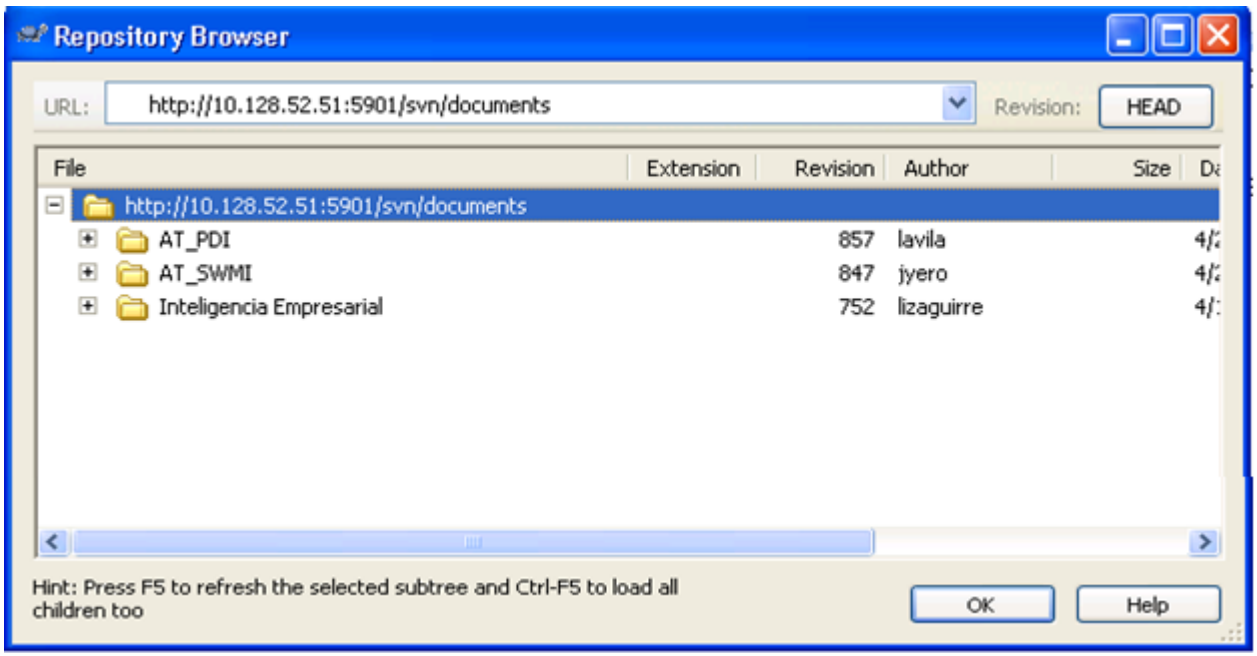


Fig 3.2 Estructura del Repositiro Subversion para la documentación.

Subversion es un sistema de archivo (filesystem versionado) con un conjunto organizado jerárquicamente de directorios y archivos, y sus contenidos, por supuesto. Es por ello que se pueden efectuar operaciones como: crear un directorio, copiar un archivo, mover un archivo, borrar un archivo, etc. Por lo tanto se creó un directorio branches, tags y trunk en la raíz del repositorio.

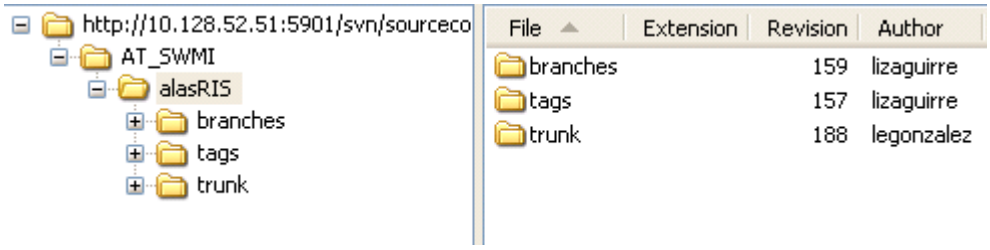


Fig 3.3 Organización jerárquica de directorios y archivos.

Trunk: Es la línea principal de desarrollo. Directorio donde se lleva a cabo la mayor parte de desarrollo del proyecto.

Branches: Es una revisión que surge a partir de una versión de la línea evolutiva principal (*trunk*) y evoluciona independientemente. Las ramas también son conocidas como "líneas de desarrollo". Es una copia del proyecto, bajo el control de versiones, pero aislado, de forma que los cambios realizados en esta rama no afecten al resto del proyecto y viceversa; aunque existe una flexibilidad bidireccional, o sea que existe la posibilidad de integrar cambios realizados en el branch hacia el trunk y de igual forma en sentido opuesto.

Las ramas o branches, permiten aislar diferentes líneas de desarrollo de sí mismas. Por ejemplo, una rama puede ser utilizada para un desarrollo experimental que sería demasiado inestable para la rama principal. O al contrario, una rama puede ser utilizada como sitio para estabilizar una versión para lanzamiento. Una rama de desarrollo en SVN, también es una copia de un directorio, pero en este caso la intención es modificarla, para conseguir un producto final diferente y alternativo al original. Muy útiles para personalizar una aplicación y que a su vez permiten implementar desarrollo en paralelo.

Tags: Una etiqueta para SVN es una copia de un directorio que se hace con el objetivo de obtener una ´foto´ del mismo, y sin intención de modificar, o sea es una colección particular de ficheros en una revisión específica. Las etiquetas (tags) deben tener un nombre fácil de recordar o significativo que identifique una versión específica. Los tags son utilizados para delimitar los hitos o momentos relevantes en el ciclo de vida del sistema. Por ejemplo, un tag es hecho para cada lanzamiento público, de forma que cada integrante pueda obtener, directamente desde el sistema de control de versiones, el conjunto exacto de ficheros/revisiones que componen el lanzamiento.

Acerca de la instalación de las herramientas

Para el Repositorio Subversion y el Redmine se instaló un servidor con:

- Sistema operativo GNU/Linux Debian 5.0 (Lenny).
- Servidor Web Apache2 version 2.2.9.
    - Protocolo de acceso a svn es http.
    - Módulo mod_ldap para autenticación del Subversion con el dominio UCI.
    - Repos – Style 2.0 para el estilo del repositorio (http).

  - Módulo mod_rails (Passenger) 2.1.2 para la interpretación de RoR[22].
- Subversion 1.5, versión 1.5.6 (rama testing o squeeze).
- Redmine 0.8.3.
- Ruby 1.8.7.

Para los clientes en MS Windows XP.

- TortoiseSVN 1.6.0 (para garantizar que ante una migración a Subversion 1.6.0 en el futuro el cambio sea transparente a los usuarios).
- AnkhSVN 2.1. como herramienta para integrar Subversion con Visual Studio. NET.

3.3 Plan de Gestión de la Configuración de Software

El Plan de GCS generalmente se estable a través de una norma del estándar IEEE Std. 828-1998 o más actual IEEE Std. 828-2005 pero en este caso se ha empleado la plantilla interna estándar que ha sido elaborada por la Dirección de Calidad de la Infraestructura Productiva para todos los proyectos de la Universidad de las Ciencias Informáticas. Basado en la plantilla se elabora este documento con el objetivo de describir la GCS en el proyecto alasRIS.

1. Introducción

El Plan de Gestión de Configuración permite mantener un informe detallado de todos cambios que se realizan en el proyecto, así como los responsables y los procedimientos para el control de cambios y de versiones del sistema que se aplicarán durante todo el ciclo de desarrollo de software en el proyecto alasRIS.

1.1 Propósito

En función de lograr una adecuada Gestión de Configuración es importante definir, establecer y documentar los requisitos, estándares, políticas y procedimientos. Pues todo ello queda contemplado en el

[22] Ruby on Rails

Plan de GCS que su propósito no es más que documentar todas las especificaciones a tener en cuenta por parte del equipo de desarrollo.

1.2 Alcance

Proceso de Gestión de Configuración de Software del proyecto alasRIS.

1.3 Definiciones, acrónimos y abreviaturas

GCS: Gestión de Configuración del Software.
ECS: Elementos de Configuración del Software
IEC: Informe de Estado de la Configuración.
ACC: Autoridad de Control de Cambios.
CC: Control de cambios.
CCC: Comité de Control de Cambios.

1.4 Referencias

Código	Título
[1]	Documento Plan Desarrollo de Software.
[2]	Documento Arquitectura
[3]	Documento Plan Gestión de Requerimiento

1.5 Resumen

En este plan se puntualizan diversos procedimientos que describen como van a ser realizadas las diferentes actividades que comprende el proceso de Gestión de Configuración de Software, además de las herramientas que se emplean con el fin de facilitar alguna de estas actividades. También abarca temas relacionados con la organización de estas tareas y las responsabilidades de cada persona incluida en el proceso.

2. Gestión de Configuración de Software

2.1 Organización de la Gestión de Configuración de Software

El proyecto cuenta con un gestor de configuración que se encarga de mantener la Gestión de Configuración, así como los métodos y las herramientas necesarias para sostener un soporte tecnológico eficiente. En cuanto a las herramientas empleadas tanto para el control de versiones como el control de cambios, es importante la capacitación de los miembros del equipo de desarrollo.

Para ello existe un equipo que es el encargado de gestionar la configuración dentro del proyecto alasRIS y ayudar en estas cuestiones, el mismo está subordinado al Jefe de Proyecto. El proyecto también cuenta con un CCC que está integrado por el presidente que sería el Líder del Proyecto y por un conjunto de representantes de los roles que definen de forma crucial el desarrollo, para así lograr de forma más eficiente el proceso de gestión de cambios.

2.2 Responsabilidades

Gestor de la Configuración

- Crear, revisar y mantener el Plan de Gestión de Configuración.
- Organizar y controlar la Administración de la Configuración.
- Crear las líneas base del proyecto.
- Establecer el Proceso de Control de Cambios.
- Definir las actividades de identificación de la configuración, la frecuencia de los reportes e informes del estado de la configuración y las políticas para el procedimiento del control de cambio.
- Seleccionar las herramientas a usar para el control de versiones y de cambios, así como definir la estructura del repositorio.

Líder de Proyecto:

- Identificar los ECS principales asociados a la gestión del proyecto.
- Planificar las tareas a ejecutar en el desarrollo del sistema.

- Asignar responsabilidades.
- Establecer un cronograma.
- Controlar la ejecución del plan de gestión de configuración.
- Establecer las políticas para el control de las versiones y de cambio. Velar por su cumplimiento.
- Aprobar las tecnologías a usar en el desarrollo del proyecto.

Cualquier Rol:

- Crear áreas de trabajo de desarrollo.
- Actualizar las Peticiones de Cambio.
- Efectuar los procedimientos de control de cambio guiados por las pautas definidas en el proyecto.
- Solicitar cambio.
- Entregar cambio.
- Cumplir con las políticas para el manejo del control de versiones.

3. Actividades de Gestión de Configuración de Software

3.1 Identificación de la Configuración

3.1.1 Especificación de la identificación

Esquema para etiquetado y numerado de documentos y directorios.

La nomenclatura de los documentos del expediente de proyecto que se tiene en cuenta está definida por la facultad 7 de la siguiente forma. Para ello se toma como ejemplo el documento Plan de Desarrollo de Software del proyecto alasRIS.

SWMIRIS – SW-DC-XXX alas RIS_Plan de Desarrollo de Software_vx.x.doc

La primera parte del nombre se compone por un código único que identifica el documento dentro del proyecto.

SWMIRIS: Estas siglas se refieren al nombre de la línea de desarrollo en este caso (**SWMI** –*Software Médico Imageneológico*) y al nombre del proyecto (RIS – *Sistema de Información Radiológica*) en ese orden.

SW: Estas siglas se refieren al tipo de proyecto, cuyos tipos deben ser:

Tipos de Proyectos	Código
Aplicaciones Informáticas	SW
Suministros	SU
Capacitación	CP
Instalación y Montaje	IM
Soporte Técnico	ST
Administración del Proyecto	GE

DC: Estas siglas se refieren al tipo de documento, cuyos tipos se enuncian a continuación.

Tipos de Documentos	Código
Documento Rector	DC
Doumento Entregable	DE
Documento Interno	DI
Comunicaciones	CC

XXX: Se refiere a un número consecutivo que se le asignará al documento. La secuencia de este número será por el momento de creación, y la serie será interna del proyecto. Luego de este código le sigue un espacio y a continuación se escribe el nombre del módulo, el cual en este caso, respetando la marca alas, se escribe de la siguiente forma: *alas* un espacio *RIS.* Luego le sigue un guión bajo "_" y por último el nombre del documento separado por espacios *Plan de Desarrollo de Software.*

En otro caso para la enumeración de las versiones se utilizan tres cifras decimales lo cual permite indicar la importancia de los cambios realizados. Por ejemplo Versiones X.Y.Z.
X: indica la versión mayor del documento. Si empieza con un cero significa que el documento aún no está listo o no cumple con los requerimientos mínimos. Cada cambio en esta cifra denota una reescritura o la incompatibilidad con versiones mayores anteriores.
Y: indica la versión menor del documento. Denota cambios en el contenido o en la funcionalidad del documento pero no lo suficientemente importantes como para decir que ya no es el mismo. Cuando se estrena una versión mayor se deja la versión menor a cero pero aún así se incluye de modo que la segunda versión mayor sería la 2.0.
Z: indica la segunda versión menor. Indica que el documento se ha corregido pero que no se ha añadido ni eliminado nada relevante. Cuando se estrena una versión menor, es decir, cuando la segunda versión menor es igual a cero; suele omitirse.

Además para llevar el control de versiones se explotan las funcionalidades que brinda la herramienta utilizada, Subversión. La misma almacena la revisión del documento, el usuario y la fecha y la hora entre otros datos de la versión actual.

Cómo identificar las relaciones
Para identificar las relaciones entre objetos se crea una jerarquía de ECS donde un elemento u objeto puede ser “parte de” un objeto compuesto.

Cómo identificar las versiones y los entregables
La identificación de versiones estará dada por la unión de varios valores: major (principal), minor (inferior), release (liberación) y opcionalmente un quantifier (cuantificador). Se constará con un modelo de numeración para los prototipos no funcionales y otro para el resto de las liberaciones (releases).

Prototipos No Funcionales
Principal: Siempre será 0(cero), ej. 0.01. Siempre será incluido en el número de la versión.
Inferior: Siempre será un número de dos dígitos utilizándose un tercero en caso necesario. Los cambios en este número indicarán cambios en el prototipo. Siempre será incluido en el número de la versión.

Otras Liberaciones

Principal: primer número en la cadena de caracteres del número de versión (ej. 1.2.3). Los cambios en este número indicarán cambios significativos en el producto y/o las funcionalidades del usuario final. Siempre se incluirá en el número de la versión.

Inferior: segundo número en la cadena de caracteres del número de versión (ej. 1.2.3). Los cambios en este numero indicarán cambios en el código base (o arquitectura base) y/o en las funcionalidades del usuario final. Este número siempre se incluirá en el número de la versión.

Liberación: tercer número en la cadena de caracteres del número de versión (ej. 1.2.3). Los cambios en este número indicarán típicamente resolución de errores (bug fixes). Este número será omitido en el caso en que su valor sea igual a cero (ej. 1.2 tiene número de liberación 0).

Cuantificadores

En ocasiones se adjuntará una cadena al final del número de versión atendiendo a necesidades específicas. Las cadenas a utilizar comúnmente serán:

aX: Indica una versión alfa (alpha release); donde X es un entero que indica el número específico de versión alfa (ej. 1.2.3a5 indica que es la quinta versión alfa del producto). Esta denominación (alfa) será utilizada para identificar a los prototipos funcionales (versiones dirigidas a las pruebas internas solamente) resultantes en el desarrollo de un producto.

bX: Indica una versión beta (beta release); donde X es un entero que indica el número específico de versión beta (ej. 1.2.3b3 indica que es la tercera versión beta del producto). Esta denominación (beta) será utilizada para identificar los productos que con todas o prácticamente todas las funcionalidades terminadas pero sin estabilidad comprobada y que serán sometidos a pruebas por parte de los grupos de pruebas de las entidades de QA1 y clientes específicos previo acuerdo con los mismos.

rcX: Indica una versión candidata a liberación (release candidate); donde X es un entero que indica el número específico de candidato a liberación (ej. 1.2.3rc4 indica que es la cuarta versión candidata a

liberación del producto). Esta denominación (release candidate) será utilizada para identificar los productos terminados y con un grado de estabilidad aceptable que será puesto en manos de los usuarios con el fin de encontrar posibles errores y/o vulnerabilidades que no hallan sido detectados en las etapas alfa y beta.

rtm: Define una versión completada y lista para su distribución y/o comercialización (ready to manufacture). Cuando una versión release candidate ya no requiere la realización de más cambios su denominación cambiará de rcX a rtm (ej. 1.2.3rc4 pasaría a 1.2.3rtm lo cual indica que la versión 1.2.3 fue culminada tras la cuarta etapa como candidata a liberación). La denominación rtm será utilizada solamente en los tags en el repositorio de código fuente, modelos y documentación; en el producto de software será omitido (ej. 1.2.3).

rV: Denota una revisión del repositorio de código fuente, indica a partir de cuál revisión del repositorio se construyeron los binarios del producto de software (ej. 1.2.3b3r2541 indica que la tercera beta a sido generada a partir de la revisión 2541 del repositorio). Este cuantificador puede utilizarse en la forma demostrada en como sufijo de los números minor y release; y de los cuantificadores alfa, beta y release candidate. Puede ser omitido y en caso de utilizarse solo será en los tags en el repositorio de código fuente, modelos y documentación; no así en el producto de software.

3.1.2 Identificación para el formulario de control de cambios

Formulario definido para el proceso de solicitud de cambio en el proyecto alasRIS.

Elemento	No	Pedido de cambio	Aspecto correspondiente	Necesidad / Mejora	Importancia	Complejidad
					Alta / Baja	Alta / Baja

Criterio	Observación
Necesidad	Errores en la interpretación procesos del negocio, e impediría el correcto funcionamiento de la aplicación para la consecución del fin.

3.1.3 Líneas base del proyecto

Las líneas base seleccionadas con sus respectivos ECS son las siguientes:

Línea Base de Planificación del Proyecto

- Documento Visión
- Acta de Inicio de Proyecto Nacional
- Acta de Terminación del Proyecto Nacional
- Plan de Desarrollo de Software
- Proyecto Técnico
- Diario de Actividades
- Plan de Mitigación de Riesgos
- Roles y responsabilidades
- Plan de capacitación
- Ambiente de desarrollo
- Documento Visión

Línea Base de Requerimientos del Sistema

- Salidas del Sistema
- Modelo de Negocio
- Reglas de Negocio
- Diccionario de Datos
- Evaluación de áreas de la organización
- Especificación de Requisitos de Software
- Plan de Gestión de Requisitos
- Requisitos Rechazados
- Roles y Responsabilidades de Administración de Requisitos
- Criterios para validar requisitos del cliente
- Criterios para validar requisitos del producto

- Especificación de casos de uso
- Acta de aceptación

Línea Base de Arquitectura y Diseño

- Arquitectura de software
- Arquitectura de Información
- Modelo de diseño
- Informe del Levantamiento de Información para la Arquitectura de Información

Línea Base de Implementación y Prueba

- Manual de usuario
- Código fuente
- Registro de Prueba Unitaria o Integración
- Plan de Pruebas
- Casos de Pruebas
- No Conformidades

Línea base de aseguramiento de la calidad del producto

- Glosario de Términos
- Plan de Aseguramiento de la Calidad
- Plan de mediciones
- Lista de chequeo
- Plan de Gestión de Configuración
- Pedido de Cambio
- Solicitud de Cambio
- Formulario de Solicitud de Cambios
- Registro de Revisiones y Cambios

3.1.4 Bibliotecas

Se estableció un servidor SUBVERSION con dos directorios uno para los ECS relacionados con el código fuente (sourcecode) y otro para los relacionados con la documentación (documents).Dentro de estos directorios se definen como:

Biblioteca de trabajo

Trunk: Se encuentra en “sourcecode/AT_SWMI/alasRIS” es el área de trabajo de los programadores.

Expediente: Se encuentra en “documents/AT_SWMI/Proyectos/alasRIS/” es el área de trabajo de los diseñadores y los analistas, O sea el Expediente del Proyecto.

Biblioteca de Integración

Branch: Se encuentra en “sourcecode/AT_SWMI/alasRIS” es donde se toman los ECS para su integración.

Para las políticas de acceso se definen tipos de usuario según los roles que ocupe cada uno de estos en el proceso de desarrollo del sistema. El acceso al repositorio Subversion se realiza mediante el Módulo mod_ldap para autenticación con el dominio UCI.

3.2 Control de la Configuración

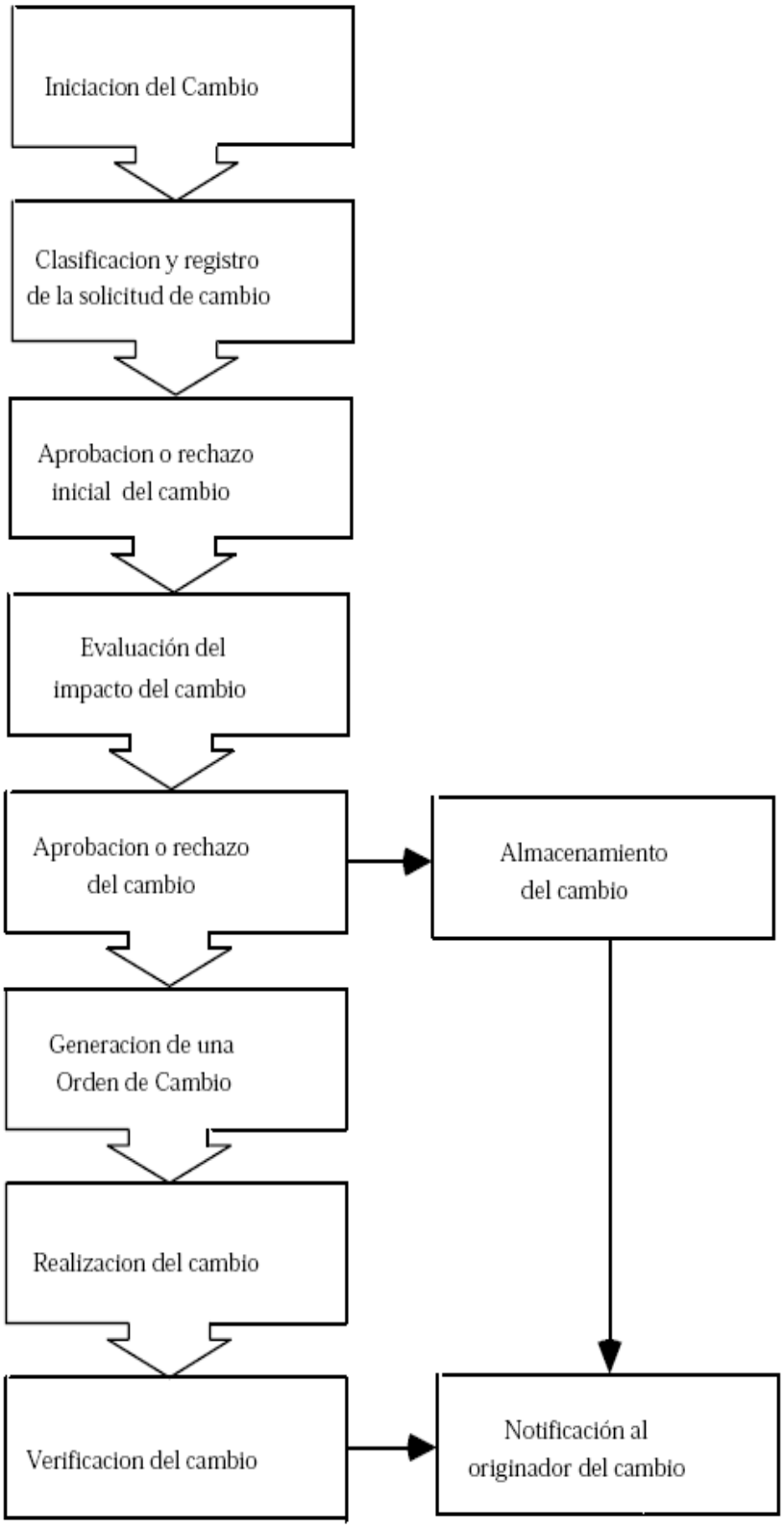
3.2.1 Procedimientos para cambiar una línea base

Se establecerá una nueva línea base siempre que ocurran cambios importantes en los documentos comprendidos dentro de esta o también dada una indicación del Jefe de Proyecto u Organismo Superior. Siendo esta creada una vez que haya pasado por una revisión técnica formal, o sea que los cambios requeridos en la documentación hayan sido revisados y aprobados. Las líneas base deberán ser verificadas por el Comité de Control de Cambios y será responsabilidad del mismo autorizarla.

Responsables: Gestor de la Configuración

Entrada: Los elementos de configuración que deben ser cambiados, además del formulario de Solicitud de Cambio.

3.2.3 Procedimiento para procesar pedidos de cambios y su aceptación


Iniciacion del Cambio
Clasificacion y registro de la solicitud de cambio
Aprobacion o rechazo inicial del cambio
Evaluación del impacto del cambio
Aprobacion o rechazo del cambio
Almacenamiento del cambio
Generacion de una Orden de Cambio
Realizacion del cambio
Verificacion del cambio
Notificación al originador del cambio

3.2.3 Comité de Control de Cambios

Se conformó un Comité de Control de Cambios integrado por Gestor de Configuración del Área Temática, el Jefe del Proyecto, Arquitecto Principal y el Analista Principal. Actualmente de estos roles son responsables las siguientes personas:

- Jefe del Proyecto: Alejandro Arias Naranjo
- Arquitecto Principal: Karel Eddy Tamayo Peña
- Analista Principal: Leodan Vega Izaguirre
- Gestor de Configuración: Yanoksy Durañona Yero

3.2.4 Revisión de documentos

Se genera una copia de los documentos y se procede a su revisión marcando los posibles errores que deban ser reparados. Estos documentos serán revisados por el Líder de Proyecto junto con el encargado de aseguramiento de la Calidad que serán encargados de corregirlos. En caso de los documentos revisados no presentar problemas se declaran satisfactorios y se archivan en el expediente del proyecto. Los documentos como plantillas de peticiones de cambio o solicitudes de cambio serán almacenados por el Gestor de Configuración siendo este el encargado de las trazas de todo el proceso de gestión de los cambios al sistema. Los documentos que presenten problemas se retornan para su reelaboración continuando el flujo inicial.

3.2.5 Herramientas automatizadas para el Control de Cambios

En el proyecto para llevar a cabo el Control de Cambios se hace uso de dos herramientas integradas, el Subversion, para controlar las versiones tanto del código fuente como de la documentación que se va generando en el transcurso del proyecto, y el Redmine para reforzar el trabajo con las versiones y hacer seguimiento de los cambios.

Subversion: es la herramienta de Gestión de Cambios (Software Control Management- SCM) más efectiva del mercado. Fue seleccionada por ser una herramienta de código abierto, multiplataforma (Win32, Linux,

Mac, etc), para el control de versiones de ficheros electrónicos, como son el software o la documentación. Se basa en un repositorio central que actúa como un servidor de ficheros, con la capacidad de recordar todos los cambios que se hacen tanto en sus directorios como en sus ficheros.
El repositorio incrementa un número global de revisión con cada conjunto de cambios enviados (commit) al mismo. Es posible copiar y renombrar ficheros; crear una rama del proyecto es tan fácil como copiar un directorio. También se puede pedir una salida con las diferencias entre dos revisiones arbitrarias, o que recupere algún sub-árbol de la revisión N.

Redmine: es una herramienta utilizada para la administración de proyectos, y muy conveniente por ser desarrollada bajo licencia GNU–GPL (Licencia Pública General) Open Source. Permite algunas funcionalidades básicas para la gestión de proyectos. Posee un sistema de ticket que posibilita automatizar el procedimiento de solicitud de cambios y controlar la asignación de tareas y todos los sucesos registrados en el proyecto.

3.3 Estado de la Configuración

3.3.1 Almacenamiento, manipulación y entregables del proyecto

Los entregables del proyecto son almacenados en el Repositorio con su consecuente control de versiones y periódicamente se realizan salvas (backup) al servidor.

3.3.2 Reportes

Mensualmente se generarán reportes, en los cuales se verán plasmando los cambios que se hayan realizado, además del estado de la configuración del proyecto. De esta tarea serán encargados el gestor de la configuración en conjunto con el líder de proyecto que este a su vez almacenará cada reporte ya sea de un ECS o una auditoría.

3.3.3 Proceso de entregas

La entrega no se realiza directamente al cliente, para ello se destina un equipo del proyecto para la instalación y puesta a punto del sistema.

3.4 Auditorías a la configuración

3.4.1 Número de auditorías a realizar y cuándo serán llevadas a cabo

Las auditorías físicas en el proyecto se establecerán siempre que concluya el desarrollo de una línea base o hitos del proyecto, y también concluido el producto final. En estas auditorías en el caso de concluir con una línea base se chequea si una línea base constituye un ejecutable en el cumplimiento de los requerimientos definidos en los inicios. Se revisará la documentación y actualización de los elementos de configuración que componen la línea base. Una vez concluido el producto se revisará el cumplimiento del producto y si satisface todas las funcionalidades, el manual de usuario y la realización adecuada de las pruebas al producto y la documentación asociada.

Para las auditorias internas del proyecto se estableció realizar una auditoría interna mensual con el objetivo de verificar la gestión de las versiones, las salvas, que se encuentre actualizada toda la documentación, el cumplimiento de las tareas acorde con el cronograma del proyecto y que la ejecución de los cambios con las respectivas solicitudes se encuentren archivados. En estas auditorías debe estar presente algún representante del grupo de aseguramiento de la calidad del proyecto, el encargado de la Gestión de Configuración en conjunto con el Líder de proyecto.

4. Hitos

Los hitos del proyecto están establecidos por los que define la metodología RUP para los cuales se de deben cumplir determinados ECS en cada fase como se muestra en la tabla:

Fases	Hitos
Inicio	Documentos de Captura de los Requisitos
Elaboración	Documentos de Diseño y Arquitectura de Software
Construcción	Versión entregable del RIS con toda la documentación asociada y el manual de usuario.
Transición	Revisar las No Conformidades

5. Entrenamiento

Como acciones de entrenamiento para llevar a cabo la preparación del personal del proyecto se ha destinado una carpeta en el Repositorio, con documentación necesaria para cuestiones relacionadas con el proyecto como se muestra en la figura.

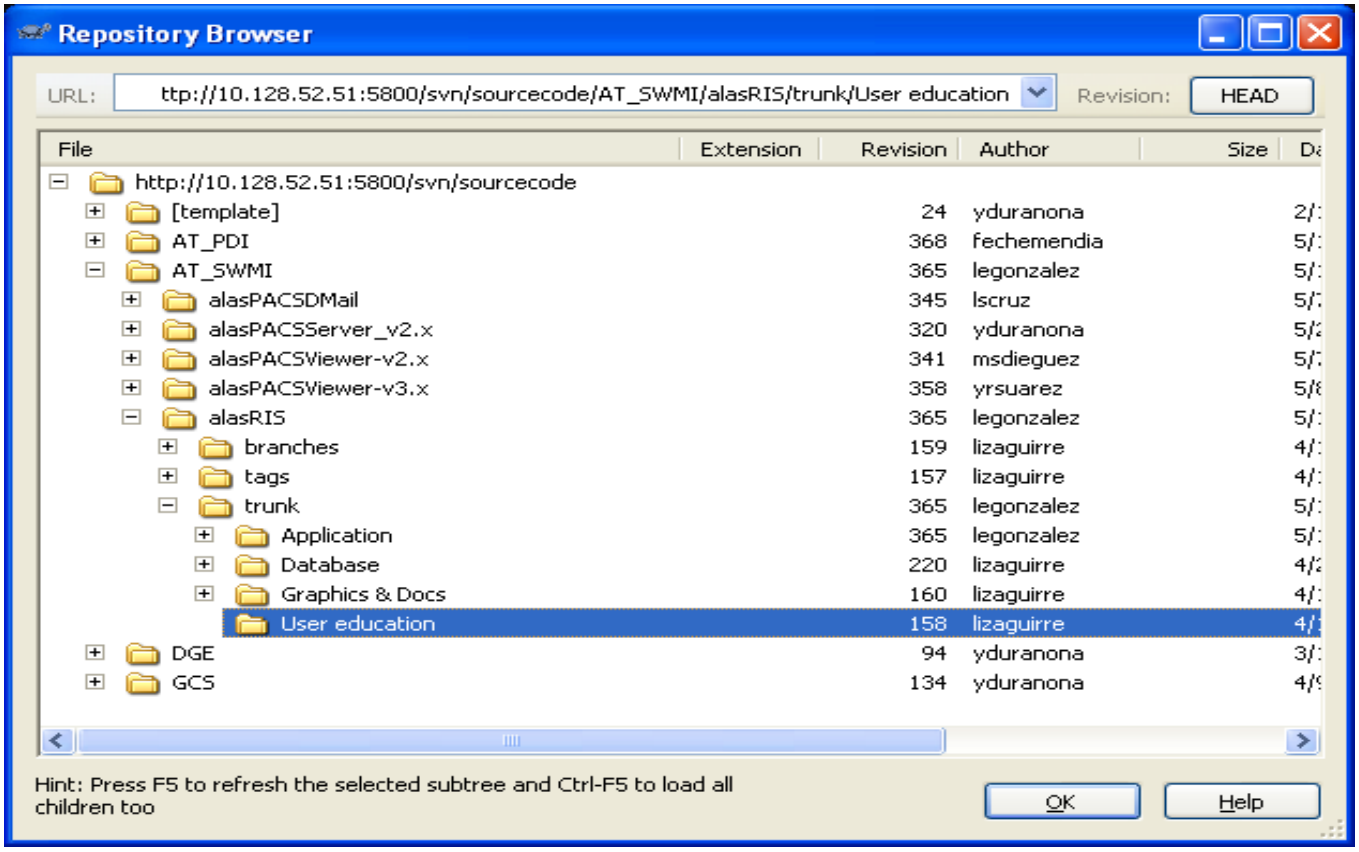


Fig 3.4 Capacitación a los integrantes del proyecto.

3.4 Validación del proceso

Para incidir positivamente en el desarrollo de software de los proyectos que integran el Polo de Imágenes, y de forma paulatina lograr establecer parámetros de excelencia, es sumamente importante implantar

modelos de procesos tomando en consideración las mejores prácticas que existen internacionalmente para la GCS. Así como la incorporación de herramientas en su apoyo, y adaptarlas creativamente a las características concretas de los proyectos. Para esto entre otros aspectos, es necesario validar la solución antes propuesta teniendo en cuenta la opinión de diferentes integrantes del proyecto alasRIS, en el cual se ha comenzado a aplicar la misma.

Para facilitar este proceso se encuestaron el líder del proyecto, el analista principal, el arquitecto principal y a algunos desarrolladores. En la encuesta, se plantearon las siguientes: Ver (Anexo 5)

- ¿Cómo considera usted la aplicación del proceso de GCS en el proyecto?
- ¿Cree que ha servido de ayuda en el desarrollo del proyecto? ¿Por qué?
- ¿Qué opina acerca del procedimiento para la Solicitud de Cambio?
- ¿Cómo evaluaría usted el funcionamiento de la herramienta Subversion? Argumente.
- ¿Qué opina sobre la herramienta propuesta Redmine? Justifique su respuesta.

El proyecto alasRIS cuenta con un total de 7 integrantes de los cuales 5 llenaron la encuesta, y el 100 % de éstos concuerdan en el criterio de la importancia que tiene implementar cada uno de los pasos a tener en cuenta en la GCS, debido a que se ha posibilitado en gran medida simplificar y hacer más ameno el trabajo. Esto es solo posible mediante el empleo de las herramientas para las cuales coinciden mayoritariamente en evaluar las herramientas propuestas, Subversion y Redmine respectivamente como muy útiles en el proceso de desarrollo de software. A continuación se expone un resumen de los diferentes criterios emitidos por los encuestados respecto a las herramientas propuestas.

Muchos se ajustan al criterio de que el uso de estas dos herramientas para la gestión de proyectos ha tenido un impacto muy positivo para el equipo, y con resultados palpables, pues Subversion como sistema de control de versiones, posibilita a todo el equipo trabajar sobre la misma versión de código, de una manera muy organizada y práctica, permitiendo hacer un correcto seguimiento del historial de versiones.

Presenta grandes ventajas frente a otras herramientas de su tipo existentes y está dado principalmente por la facilidad de uso, posibilita hacer copias de seguridad, llevar un seguimiento de los distintos componentes de la solución para el caso en que haya que realizar un retorno a una versión anterior. Los

que habían trabajado antes con Visual Source Safe hoy alegan que Subversion presenta mayores potencialidades frente a este.

Con Subversión la integración con el IDE de VisualStudio mediante plugins es excelente y el modo en que este opera es muy interesante y eficiente, pues cuando realiza las transacciones solo envía las diferencias lo que lo hace mucho más rápido y operativo. Mediante el cliente TortoiseSVN se integra perfectamente al Shell del sistema operativo por lo que no necesita ninguna otra herramienta complicada para acceder al repositorio facilitando aún más su modo de empleo. Hasta el momento para las actividades que se llevan a cabo en el proyecto alasRIS no se han encontrado desventajas sobre este.

En cuanto a Redmine como herramienta para la gestión del proyecto ha sido valorada como muy útil; constituyendo la plataforma central para la comunicación entre los integrantes del proyecto, sobre todo en lo concerniente a la asignación de tareas y el control del cumplimiento de las mismas, ya que permite al equipo de desarrollo saber que debe hacer en cada momento, sobre todo mediante la vinculación RSS que se puede realizar con un cliente como el Microsoft Office Outlook o un navegador como Mozilla Firefox hoy utilizado por todos.

Mantiene todo el tiempo al equipo informado dejando la trazabilidad de todo el proceso mediante un historial. Es sobre la Web y resulta para los integrantes del proyecto alasRIS muy fácil de usar, es una herramienta extensible y personalizable aunque uno de sus miembros recomienda que sería mucho mejor si permitiera que un miembro estuviese en más de un rol; además de que los tracker sean configurables completamente. Pero es evidente que Redmine ha sido un enorme paso de avance y mejora en el proceso en cuanto a la organización dentro del proyecto.

Conclusiones

En el capítulo se han desarrollado los objetivos planteados para el presente trabajo. Se han tomado todos los pasos descritos en la solución propuesta en el mejoramiento de la realización del proceso de Gestión de Configuración. Para ello un paso esencial lo constituyó el empleo de un sistema de control de versiones, la instalación de la herramienta Subversion tanto para el proyecto alasRIS como para todos los

otros proyectos que integran el Polo de Imágenes y que paulatinamente se irán acogiendo al proceso de GCS.

Fue importante también el establecimiento de los roles logrando una mayor organización en el proceso de desarrollo y sus involucrados, así como la mejora del seguimiento de las tareas asignadas a cada uno de estos. La generación de informes del estado de la configuración aumentó el nivel informativo del proyecto durante su desarrollo y también se espera que así sea en un futuro. Llevar a cabo auditorías internas a la configuración posibilitó en gran medida el control y detección de errores en el proceso permitiendo mejoras desde etapas tempranas del desarrollo. Además de proporcionar un eficiente control de las tareas.

CONCLUSIONES GENERALES

Una vez concluido el estudio realizado referente a la Gestión de Configuración de Software se demostró que para desarrollar y mantener software efectivamente es imprescindible disponer de los procedimientos, herramientas y recursos necesarios que permitan administrar los cambios y las configuraciones de software.

En el desarrollo del presente trabajo se lograron detallar las etapas por las que transita la GCS dando paso a adquirir y enriquecer los conocimientos de GCS. Se elaboró una estrategia para la GCS que ayuda a llevar un proceso de desarrollo de software mejor controlado en el Polo de Imágenes.

Además de caracterizar el conjunto de herramientas existentes en la actualidad que tienen como objetivo garantizar el soporte y automatizar los procedimientos de la solución que se propuso. Para lo cual se sostuvo la proposición del uso de la herramienta controladora de versiones, Subversion para llevar a cabo tareas relacionadas con subprocesos de la GCS en el Polo de Imágenes.

Se establecieron procedimientos capaces de estandarizar el proceso de GCS en dicho Polo Productivo, donde se definió un esquema para establecer los roles involucrados en el proceso de mejora de la aplicación de la GCS.

Se efectuó una propuesta para la estructuración de las bibliotecas de software, al igual que para el almacenamiento y recuperación de las líneas base, logrando alcanzar gran madurez y organización en todo este proceso.

Se establecieron mecanismos formales para el Control de los Cambios, y un Comité de Control de Cambios.

RECOMENDACIONES

Para darle continuidad a este trabajo en aras de perfeccionar el proceso de desarrollo de software se recomienda:

- Extender el empleo de la solución propuesta, para llevar a cabo el proceso de GCS, definida en el presente trabajo a todos los proyectos del Polo de Imágenes, así como su aplicación a todos los proyectos con características similares de la Universidad de las Ciencias Informáticas.

- Hacer extensible este trabajo y que de pie a la elaboración de una estrategia genérica para llevar a cabo el proceso de GCS en la Facultad 7, de forma que pueda ser aplicada a todos los proyectos productivos y así lograr una homogeneidad en el proceso de GCS.

- Evaluar la solución que se describe en este trabajo, mediante las valoraciones de expertos en el tema de GCS.

- Valorar el proceso de GCS con la utilización de métricas para la GCS una vez transcurrido un tiempo de aplicación de la solución propuesta.

- Potenciar las herramientas seleccionadas con mejores y nuevas funcionalidades.

- Profundizar el estudio en cuanto a las herramientas basadas en el desarrollo distribuido, enfocado en las tendencias actuales que cada día giran más hacia el desarrollo distribuido geográficamente.

- Incluir este tema en el plan de cursos optativos para brindar una mayor capacitación en cuestiones propias de la ingeniería y gestión de software.

Referencias Bibliográficas

Acevedo, Rodolfo Villarroel. 2004. Mejoramiento del Proceso de Gestión de Configuración de Software. [Online] 4 2004. [Cited: 1 20, 2009.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/rodolfo.pdf.

Agustín, Gonzalo Cuevas. *Gestión del Proceso Software.* Disponible en: http://books.google.com.cu/books?id=PI2HokV7HtIC&pg=PA195&lpg=PA195&dq=gestion+de+configuracion+del+software&source=bl&ots=yQD8DHycjE&sig=RBzzFpq9RFPmRLA_Z-BH2MazbQE&hl=es&ei=-GAdSrmjLcKLtgeo8OiIDQ&sa=X&oi=book_result&ct=result&resnum=1#PPA : Editorial Centro de Estudios Ramón Areces, S.A.

Ailyn Febles Estrada, Sofía Álvarez Cárdenas. 2005. LA GESTIÓN DE CONFIGURACIÓN Y EL DESARROLLO DE SOFTWARE EN LAS UNIVERSIDADES.UNA EXPERIENCIA PRÁCTICA. [Online] 2005. [Cited: 3 26, 2009.] http://www.dict.uh.cu/Revistas/Educ_Sup/012005/Art06.pdf.

Antonio, Angelica. 2001. *La Gestión de la Configuración del Software.* Chile : s.n., 2001.

Appleton, Brad. 2000. The ACME Project Assembling Configuration Management Environmentsfor Software Development. *SCM Definitions.* [Online] 2000. [Cited: 01 15, 2009.] http://www.cmcrossroads.com/bradapp/acme/scm-defs.html.

Babich, W. 1986. *Software Configuration Management.* s.l. : Addison Wesley, 1986.

Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. 2004. *Control de versiones con Subversion.* s.l. : Ed. O'Reilly Media disponible en: http://svnbook.red-bean.com/, 2004.

Gestión de la Configuración. [Online] [Cited: 02 20, 2009.] http://www.csi.map.es/csi/metrica3/gescon.pdf.

2007. Hhista Internacional S.A. [Online] 02 2007. [Cited: 01 14, 2009.] http://www.histaintl.com/soluciones/configuracion/configuracion.php.

IEEE. 1990. "IEEE Standard for Software Configuration Management ". s.l. : IEEE Computer Society, 1990.

ITIL. 2008. Guía Práctica de Gestión de Configuración. [Online] 2008. [Cited: 3 23, 2009.] https://www.inteco.es/file/1000219454.

2006. LugFI, Grupo de usuarios de GNU/Linux de la Facultad de Ingeniería de la Universidad de Buenos Aires, Argentina. *Introducción a los sistemas de control de versiones.* [Online] 9 22, 2006. [Cited: 3 10, 2009.] http://www.lug.fi.uba.ar/documentos/scms/index.php.

Mallorca, Usuarios de GNU/Linux de. 2006. BULMA: Darcs: control de versiones redux. *Bulma.* [Online] 08 25, 2006. http://bulmalug.net/pdf.phtml?nIdNoticia=2335.

Navarro, José Angel Franco. 2006. *Entorno Unificado para la Gestión de Configuración de Software.* La Habana, Cuba: CUJAE. : s.n., 2006.

Pablo, Gervás. 2002. *Estándares y Gestión de Configuración. .* 2002.

Palomar, David. 2008. Sistemas de Control de Versiones Distribuidos. *Adama Consulting.* [Online] 5 17, 2008. [Cited: 3 25, 2009.] http://adamaconsulting.blogspot.com/2008/05/sistemas-de-control-de-versiones.html.

Pérez, Ing. Melvin. 2005. Fundamentos de Administración de Configuración de Software (SCM), v2.1. [Online] 2005. [Cited: 03 10, 2009.] http://delfin.mxl.uabc.mx/~angelica/SCM%20Fundamentals.ppt.

Pressman, Roger. 2005. *Ingeniería de Software, Un enfoque práctico, 5ta. Edición.* s.l. : Mc GrawHill., 2005.

Rational ClearCase. *IBM.* [Online] http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=Z012568L96063G61.

Bibliografía

Acevedo, Rodolfo Villarroel. 2004. Mejoramiento del Proceso de Gestión de Configuración de Software. [Online] 4 2004. [Cited: 1 20, 2009.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/rodolfo.pdf.

Agustín, Gonzalo Cuevas. *Gestión del Proceso Software.* Disponible en: http://books.google.com.cu/books?id=PI2HokV7HtIC&pg=PA195&lpg=PA195&dq=gestion+de+configuracion+del+software&source=bl&ots=yQD8DHycjE&sig=RBzzFpq9RFPmRLA_Z-BH2MazbQE&hl=es&ei=-GAdSrmjLcKLtgeo8OiIDQ&sa=X&oi=book_result&ct=result&resnum=1#PPA : Editorial Centro de Estudios Ramón Areces, S.A.

Ailyn Febles Estrada, Isabel Pérez. Argentine Symposium on Software Engineering (2005). Métricas para la Gestión de la Configuración de Software, un Planteamiento Formal. [Online] Argentine Symposium on Software Engineering (2005). [Cited: 3 20, 2009.] http://www.frcu.utn.edu.ar/deptos/depto_3/34JAIIO/34JAIIO/asse/asse22.pdf.

Ailyn Febles Estrada, Sofía Álvarez Cárdenas. 2005. LA GESTIÓN DE CONFIGURACIÓN Y EL DESARROLLO DE SOFTWARE EN LAS UNIVERSIDADES.UNA EXPERIENCIA PRÁCTICA. [Online] 2005. [Cited: 3 26, 2009.] http://www.dict.uh.cu/Revistas/Educ_Sup/012005/Art06.pdf.

Andreas Back, Ola Bodin. Febrero 2006. *Agile SoftwareConfiguration Management.* Febrero 2006.

Antonio, Angelica. 2001. *La Gestión de la Configuración del Software.* Chile : s.n., 2001.

Appleton, Brad. 2000. The ACME Project Assembling Configuration Management Environmentsfor Software Development. *SCM Definitions.* [Online] 2000. [Cited: 01 15, 2009.] http://www.cmcrossroads.com/bradapp/acme/scm-defs.html.

Babich, W. 1986. *Software Configuration Management.* s.l. : Addison Wesley, 1986.

Barreiro, Enrique. tema 8 – gestión de configuración y mantenimiento del software. [Online] [Cited: 3 14, 2009.] http://trevinca.ei.uvigo.es/~ebalonso/asignaturas/esx/transparencias/esxTransp8.ppt.

Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. 2004. *Control de versiones con Subversion.* s.l. : Ed. O'Reilly Media disponible en: http://svnbook.red-bean.com/, 2004.

—. Marzo, 2004. *Control de versiones con Subversion. Revision 2078.* Chicago : Disponible en http://osl.uca.es/jornadas/cd/Contenidos/Manuales/svn-book.pdf, Marzo, 2004.

Booch, Rumbaugh, Jacobson. 2000. *El proceso unificado de desarrollo de software.* Madrid : 1ra. Edición, Pearson Educación- Addison Wesley Disponible en: http://bibliodoc.uci.cu/pdf/reg00060.pdf, 2000.

Brad Appleton, Stephen P. Berczuk,Ralph Cabrera,Robert Orenstein. Streamed Lines: Branching Patterns for Parallel Software Development. [Online] [Cited: 2 10, 2009.] http://www.cmcrossroads.com/bradapp/acme/branching/.

Burrows, Clive. 1999. Configuration Management. Coming of Age in the Year 2000. [Online] Croostalk Magazine, 1999. [Cited: 3 10, 2009.] http://www.stsc.hill.af.mil/crosstalk/1999/03/burrows.asp.

Carrasco, Luis de Salvador. *Gestión de Proyectos de Software.* [Online] [Cited: 2 10, 2009.] http://www.luisdesalvador.com/Oposicion/T016_019_GestionProyectoSoftw.pdf.

Castagnetto, Jesús M. Sistemas para el Control de Versiones de Código. *Slideshare.* [Online] Facultad de Ciencias y Filosofía, y Dirección Universitaria de Información, Universidad Peruana Cayetano Heredia . [Cited: 3 30, 2009.] http://www.slideshare.net/jesus.castagnetto/sistemas-para-el-control-de-versiones-de-cdigo-presentation.

Cuenca, Iliana Burguán Edwin. Gestión del Cambio del Software. *Slideshare.* [Online] [Cited: 2 20, 2009.] http://www.slideshare.net/imburguan/gestin-del-cambio-del-software.

Dra. Sofía Álvarez Cárdenas, Ing. Humberto Fernández Yanes. 2002. Sitio para la implementación de las mejores prácticas de software. [Online] 2002. [Cited: 3 15, 2009.] http://espejos.unesco.org.uy/simplac2002/Ponencias/ambientes%20digitales/AD072.doc.

Gadea, Ivan. 2009. Git: ¡Pobre Perforce! [Online] Febrero 7, 2009. http://blog.ivangadea.com/2009/02/07/git-pobre-perforce/.

Garcerant, Iván. 2008. Prácticas: Gestión de Configuración. *Tecnología y Synergix.* [Online] 6 2, 2008. [Cited: 4 25, 2009.] http://synergix.wordpress.com/2008/06/02/practicas-gestion-de-configuracion/.

Gerónimo. 2008. GERONET, Informática e Ingeniería. [Online] Julio 2, 2008. [Cited: 1 25, 2009.] http://www.geronet.com.ar/?p=90.

2009. Gestión de Configuración de Software. [Online] Mayo 2009. [Cited: 1 30, 2009.] www.fing.edu.uy/inco/cursos/gestsoft/ppts/gs13.ppt.

Gestión de configuración.(Control de versiones, configuración y cambios). *Asignatura: Entornos de programación.* [Online] [Cited: 2 20, 2009.] http://lml.ls.fi.upm.es/ep/versiones.html.

Gestión de la Configuración. [Online] [Cited: 02 20, 2009.] http://www.csi.map.es/csi/metrica3/gescon.pdf.

GISC. 2000. Unidades temáticas de Ingeniería del Software.Gestión de la configuración del software. [Online] 2000. [Cited: 2 15, 2009.] http://serdis.dis.ulpgc.es/~a013775/asignaturas/ii-is2/Apuntes/UT10.%20Gesti%C3%B3n%20de%20la%20configuraci%C3%B3n%20del%20software.pdf.

GNU., Licencia de documentación libre de. 2009. Wikipedia, La enciclopedia libre. [Online] 2009. [Cited: 3 5, 2009.] http://es.wikipedia.org/wiki/CVS.

Gracia, Joaquin. 2003. IngenieroSoftware. *Control de código fuente.* [Online] 8 23, 2003. [Cited: 3 25, 2009.] http://www.ingenierosoftware.com/equipos/controlcodigo.php.

2008. GUÍA PRÁCTICA DE GESTIÓN DE CONFIGURACIÓN. [Online] Diciembre 2008. [Cited: 3 30, 2009.] https://www.inteco.es/file/1000219454.

2009. Herramientas del Software Libre. Sistema de control de versiones. *Wikilibros.* [Online] 4 1, 2009. [Cited: 9 10, 2007.] http://es.wikibooks.org/wiki/Herramientas_del_Software_Libre/Sistema_de_control_de_versiones.

2007. Hhista Internacional S.A. [Online] 02 2007. [Cited: 01 14, 2009.] http://www.histaintl.com/soluciones/configuracion/configuracion.php.

IEEE. 1990. "IEEE Standard for Software Configuration Management ". s.l. : IEEE Computer Society, 1990.

Inc, AccuRev. 2008. AccuRev 4.7 Software Configuration Management. [Online] 2008. http://www.norsys.se/websites/webb_filer/filbank/AccuRev_data_sheet4.7.pdf.

Introducción a Visual SourceSafe. *msdn Library* . [Online] Microsoft Corporation. http://msdn.microsoft.com/es-es/library/3h0544kx(VS.80).aspx.

ITIL. 2008. Guía Práctica de Gestión de Configuración. [Online] 2008. [Cited: 3 23, 2009.] https://www.inteco.es/file/1000219454.

Klagenfurt. 2007. Gestion de versiones con CVS y Subversion. [Online] Septiembre 2007. http://macprogramadores.org/tutoriales/tutoriales/cvssvn.pdf.

Krill, Paul. 2008. AccuRev scales software change management. *InfoWorld.* [Online] September 25, 2008. http://www.infoworld.com/d/developer-world/accurev-scales-software-change-management-075.

Lic. Ailyn Febles Estrada, Ing. Isabel Pérez Estévez. Medir el proceso de control de configuración, ¿una utopía para la Industria Nacional de Software? La Habana, Cuba : Instituto Superior Politécnico "José Antonio Echeverría".

Luaces, Pablo Santos. Febrero 2007. Plastic SCM Platform. Desarrollo Paralelo. s.l. : Códice Software, Febrero 2007.

2006. LugFI, Grupo de usuarios de GNU/Linux de la Facultad de Ingeniería de la Universidad de Buenos Aires, Argentina. *Introducción a los sistemas de control de versiones.* [Online] 9 22, 2006. [Cited: 3 10, 2009.] http://www.lug.fi.uba.ar/documentos/scms/index.php.

Mallorca, Usuarios de GNU/Linux de. 2006. BULMA: Darcs: control de versiones redux. *Bulma.* [Online] 08 25, 2006. http://bulmalug.net/pdf.phtml?nIdNoticia=2335.

Manchado, Esteban. Una introducción a Darcs. [Online] http://www.demiurgo.org/charlas/darcs.pdf.

Marcello Visconti, Hernán Astudillo. Evolución de Software. [Online] [Cited: 3 26, 2009.] http://images.gsb.com.ar/pdf/16-EvoluciondeSoftware.pdf1226350031.pdf.

Matas, Miguel. 2008. Gestión de la Configuración. *Miguel Matas Blog.* [Online] 04 06, 2008. [Cited: 3 2, 2009.] http://www.miguelmatas.es/blog/2008/04/06/gestion-de-la-configuracion/.

Julio 2007 . MPS.BR - Mejora de Proceso del Software Brasileño. *Guía de Implementación – Parte 2: Nivel F.* s.l. : SOFTEX Excelence in Software, Julio 2007 .

1997. Nasa Software Configurations Management GuideBook. [Online] 1997.

Navarro, Antonio. Gestión de la configuración software. [Online] [Cited: 2 25, 2009.] http://www.fdi.ucm.es/profesor/anavarro/10._Gestion_de_la_configuracion_software.pdf.

Navarro, José Angel Franco. 2006. *Entorno Unificado para la Gestión de Configuración de Software.* La Habana, Cuba: CUJAE. : s.n., 2006.

2007 . Opiniones sobre: Gestión de Configuración y Calidad. [Online] Publicado por Grupo 3 IS III Madrid, 2007 . [Cited: 2 25, 2009.] http://ingenieriadesoftwareiii.blogspot.com/2007/10/opiniones-sobre-gestin-de-configuracin.html.

Ossa, Gustavo Londoño. 2008. Redmine. Manual de Usuario. *Gustavo Lodoño.* [Online] Julio 2008. http://gustavolondonho.blogspot.com/2008/07/manual-de-usuario-redmine.html.

P., Daniel Cohen. 2006. Introducción al Control de Vesriones usando Subversion. *Open PUC.* [Online] Agosto 2006. http://openpuc.cl/index.php?option=com_docman&task=doc_view&gid=12&Itemid=59.

Pablo, Gervás. 2002. *Estándares y Gestión de Configuración.* . 2002.

Palomar, David. 2008. Sistemas de Control de Versiones Distribuidos. *Adama Consulting.* [Online] 5 17, 2008. [Cited: 3 25, 2009.] http://adamaconsulting.blogspot.com/2008/05/sistemas-de-control-de-versiones.html.

Pérez, Ing. Melvin. 2005. Fundamentos de Administración de Configuración de Software (SCM), v2.1. [Online] 2005. [Cited: 03 10, 2009.] http://delfin.mxl.uabc.mx/~angelica/SCM%20Fundamentals.ppt.

Plana, Jorge de las Peñas. 2007. Sistema evolutivo de gestión de la configuración del software. *Biblos- e Archivo.* [Online] Febrero 2007. [Cited: 3 10, 2009.] http://digitool-uam.greendata.es/view/action/singleViewer.do?dvs=1243461126887~951&locale=en_US&search_terms=000001245&adjacency=N&application=DIGITOOL-3&frameId=1&usePid1=true&usePid2=true.

2006 . Plastic, Sistema para gestión de la configuración desarrollado con Scrum . *Navegapolis.net.* [Online] 09 13, 2006 . http://www.navegapolis.net/content/view/436/.

Pressman, Roger. 2005. *Ingeniería de Software, Un enfoque práctico, 5ta. Edición.* s.l. : Mc GrawHill., 2005.

Ramirez, Ignacio. 2002. Sistemas de control de versiones y CVS. [Online] Octubre 15, 2002. http://iie.fing.edu.uy/~nacho/blandos/archivos/cvs.pdf.

Rational ClearCase. [Online] Grupo soluciones GSInnova. http://www.rational.com.ar/herramientas/clearcase.html.

Rational ClearCase. *IBM.* [Online] http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=Z012568L96063G61.

2009. Redmine. [Online] Mayo 2009. http://www.chuidiang.com/chuwiki/index.php?title=Redmine.

Revision Control Systems. [Online] [Cited: 3 12, 2009.] http://migo.sixbit.org/papers/Revision_Control_Systems/slide-index.html.

Rodolfo Villarroel, Marcello Visconti. Un Modelo de Madurez para el Proceso de Gestión de Configuración de Software. [Online] [Cited: 3 10, 2009.] http://www.inf.utfsm.cl/~visconti/papers/paperscm1999.pdf.

Ruiz, Nicolás. 2006. Herramientas para colaborar en la red:. [Online] 1 12, 2006. [Cited: 3 10, 2009.] http://www.saber.ula.ve/bitstream/123456789/13355/1/scm-nicolas-ruiz.pdf.

Sanz, Marcos López. Curso 2007/2008. Gestión de Proyectos III:Gestión de la configuración. [Online] Curso 2007/2008. [Cited: 2 10, 2009.] http://www.kybele.etsii.urjc.es/docencia/IS3/2007-2008/Material/%5BIS3-2007-08%5DGestionProyectos.III.pdf.

Software configuration management. *Wikipedia. La enciclopedia libre.* [Online] http://es.wikipedia.org/wiki/Software_configuration_management.

Sommerville. Julio 2005. *Ingeniería del Software.* . s.l. : 7ª Edición, Addison-Wesley. Disponible en: http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/index.html, Julio 2005.

Tuya, Javier. 2008. Gestión de la Configuración del. [Online] 2008. [Cited: 3 20, 2009.] http://www.di.uniovi.es/~tuya/is/descarga/gestion/CalidadSCM.x2.pdf.

UCI, Msc.Maypher Román Durán Asesor Dpto. IGSW. La Gestión de Configuración de Software. *Curso de Gestión de Proyectos.*

2009. Wikipedia, La enciclopedia libre. *Software configuration management.* [Online] 1 17, 2009. [Cited: 3 2, 2009.] http://es.wikipedia.org/wiki/Software_Configuration_Management.

ANEXO 1

En la siguiente tabla se muestran los diversos sistemas de control de versiones existentes hoy en día, tanto libres como privativos.

Software	Proveedor	Licencia	Sistemas Operativos soportados
AccuRev	AccuRev, Inc.	Privativa	Unix, Windows, Mac OS X
Aldon	Aldon	Privativa	Linux,Windows
Alienbrain	Avid Technology Inc.	Privativa	Linux,Windows, Mac OS X
AVS-Advanced Versioning System	CA, Inc.	GPL	Linux,Windows,
AllChange	Intasoft Ltd	Privativa	Windows
Bazaar	Canonical Ltd.	GPL	Unix, Windows, Mac OS X
BitKeeper	BitMover Inc.	Privativa	Unix, Windows, Mac OS X
ClearCase	IBM Rational	Privativa	Unix, Windows, AIX
Code Co-op	Reliable Software	Privativa	Windows
Codeville	Ross Cohen	BSD	Unix, Windows, Mac OS X
CVS	The CVS Team	GPL	Unix, Windows, Mac OS X
CVSNT	March Hare Software y community members.	GPL Libre Comercial	Unix, Windows, Mac OS X
Darcs	David Roundy	GPL	Unix, Windows, Mac OS X
DesingSync	ENOVIA MatrixOne	Privativa	Unix,Linux, Windows, HPUX

Git	Junio Hamano	GPL	POSIX[23], Windows, Mac OS X
GNU Arch	Andy Tai	GPL	Unix, Windows, Mac OS X
Mercurial	Matt Mackall	GPL	Unix, Windows, Mac OS X
Monotone	Nathaniel Smith, Graydon Hoare	GPL	Unix, Windows, Mac OS X
Perforce	Perforce Software Inc.	Privativa	Unix, Windows, Mac OS X
PlasticSCM	Codice Software	Privativa	Unix, Windows, Mac OS X
Star Team	Borland	Privativa	Windows
SubversionSVN	CollabNet, Inc.	Apache/BSD	Unix, Windows, Mac OS X
Surround SCM	Seapine Software	Privativa	Unix, Windows, Mac OS X
Team Foundation Server	Microsoft	Privativa	Windows
Telelogic Synergy	Telelogic (IBM)	Privativa	Unix, Windows, Linux
Visual SourceSafe	Microsoft	Privativa	Windows

[23] Portable Operating System Interface for Unix

ANEXO 2

Grupo de herramientas de apoyo a la Gestión de Proyecto, tanto libres como privativas.

Software privativo	Software libre
Rational ClearQuest	Codeville
Team Foundation Server	Bugzero
Aldon	Bugzilla
AceProject	JitterBug
ADT Web	ArX
Agility	
Bug/Defect Tracking Expert	
AllChange	
Libre (BSD)	
PureCM	
Star Team	
Surround SCM	
Telelogic Synergy	

ANEXO 3

Pasos a seguir para llevar a cabo el proceso de Control de Cambios. Tomado del libro de Ingeniería de Software, Un enfoque práctico del autor Roger Pressman. Capítulo 9, página 157.

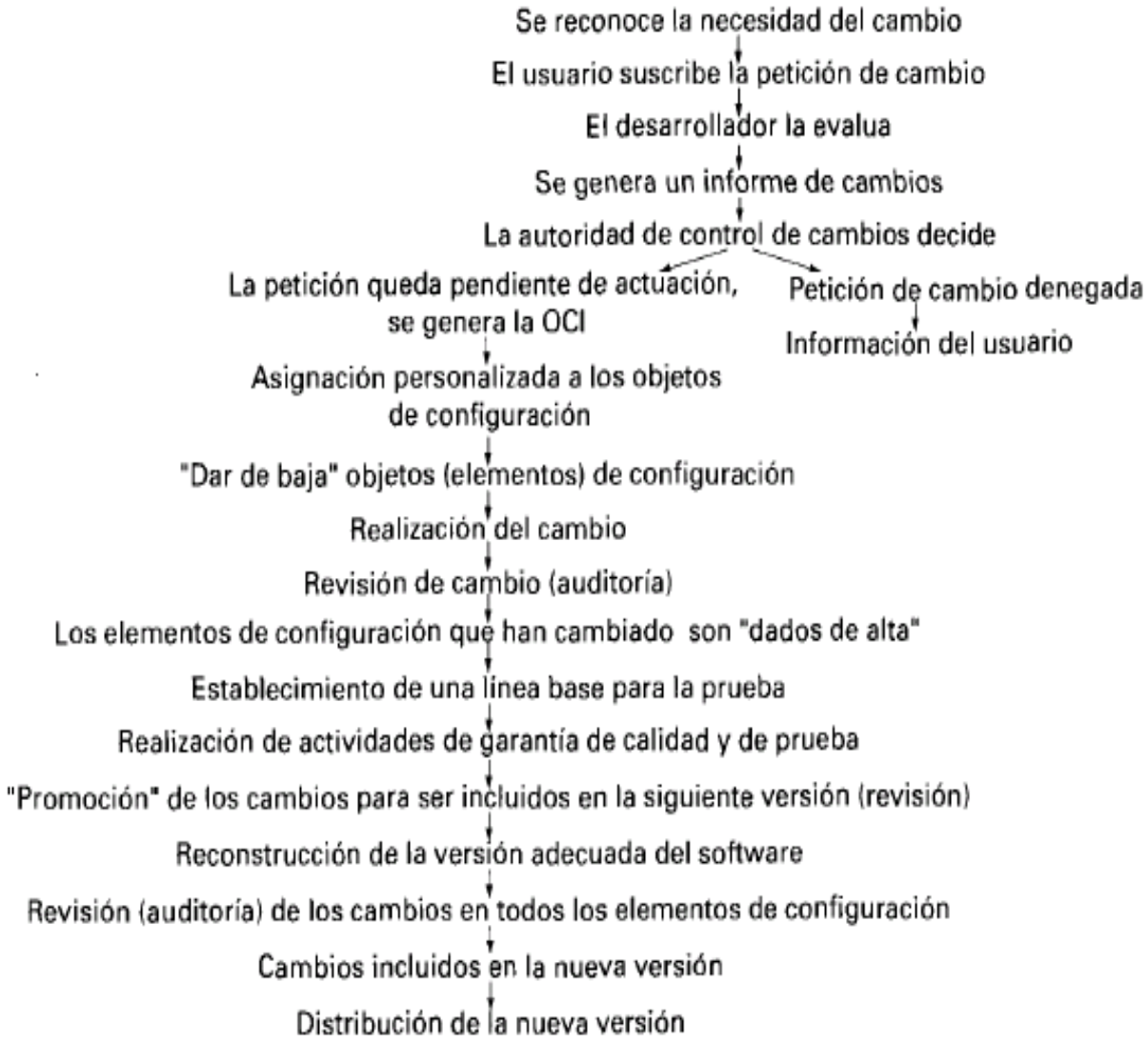


FIGURA 9.5. El proceso de control de cambios.

ANEXO 4

INFRAESTRUCTURA PRODUCTIVA

SOLICITUD DE CAMBIO
V1.0

<Nombre del proyecto>
<Nombre del producto>
<Versión>

Elemento	No	Pedido de cambio	Aspecto correspondiente	Necesidad / Mejora	Importancia	Complejidad
					Alta / Baja	Alta / Baja

Criterio	Observación
Necesidad	Errores en la interpretación procesos del negocio, e impediría el correcto funcionamiento de la aplicación para la consecución del fin.
Mejora	Problemas de interpretación y redacción de la solución.

ANEXO 5

Encuesta elaborada a integrantes del proyecto alasRIS para evaluar el nivel de aceptación del proceso de GCS.

Con el objetivo de evaluar el grado de aceptación por parte de los integrantes del proyecto alasRIS del proceso de Gestión de Configuración de Software un a vez puesto en marcha, es necesario dar respuesta al siguiente cuestionario.

1. ¿Cómo considera usted la aplicación del proceso de GCS en el proyecto? ¿Por qué?

   ___ Muy Útil
   ___ Útil
   ___ Innecesario
2. ¿Cree que ha servido de ayuda en el desarrollo del proyecto? ¿Por qué?

   ___ Si
   ___ No
3. ¿Qué opina acerca del procedimiento para la Solicitud de Cambio? ¿Por qué?

   ___ Muy Útil
   ___ Útil
   ___ Innecesario

4. ¿Cómo evaluaría usted el funcionamiento de la herramienta Subversion? Justifique su respuesta.

   ___ Muy Útil
   ___ Útil
   ___ Innecesario

5. ¿Qué opina sobre la herramienta propuesta Redmine? Justifique su respuesta.

   ___ Muy Útil
   ___ Útil
   ___Innecesario