UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.
FACULTAD 3 TURISMO Y NEGOCIOS.

Diseño del Proceso de Gestión de Configuración del Software para el Sistema de Gestión de Inventario y Almacenes (SIGIA).

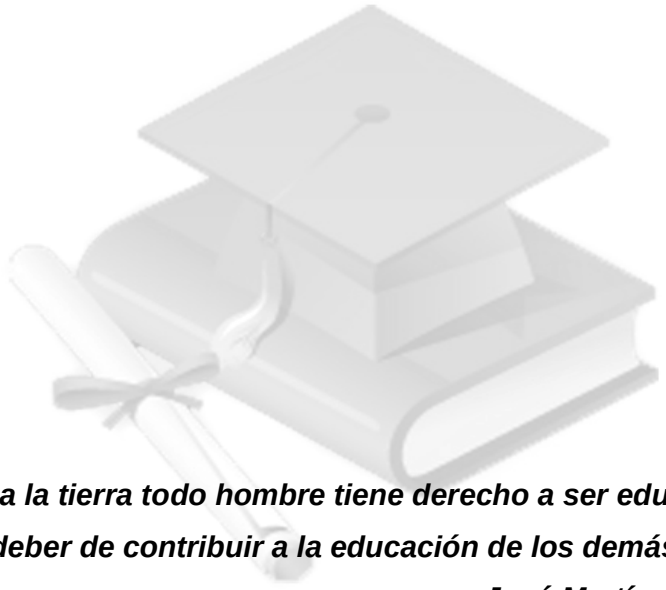
TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Autor: Daimi Lamorú Marciel.

Tutor: Héctor García Llanes.

Consultante: Febe Angel Ciudad Ricardo.

"C.Habana, junio del 2007".



“Al venir a la tierra todo hombre tiene derecho a ser educado y después en pago; el deber de contribuir a la educación de los demás.”

José Martí.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

DATOS DE CONTACTO

Datos del tutor: Graduado en la Especialidad de Ingeniería Industrial.

Categoría docente: Adiestrado.

Años de graduado: 2 años

AGRADECIMIENTOS

A la **Revolución Cubana,** por lograr que las universidades de nuestro país se pintaran de obreros, de campesinos, de personas humildes. Por darnos el derecho a todos a tener una educación gratuita y de una alta calidad. Por hacer de nosotros jóvenes integrales, hombres y mujeres de ciencia, de futuro, comprometidos y concientes del momento que estamos viviendo.

A la **UCI** por ser nuestra segunda casa durante estos cinco años.

A mis **padres**, por el cariño, el amor, el orgullo, la confianza, la honestidad, el ejemplo, que me han transmitido durante toda la vida, principalmente en los momentos más difíciles.

A **Eduardo** por ser compañero, amigo, cómplice, novio.

A mi **tutor**. Gracias por ser paciente y soportarme.

Al profesor **Febes** por su atención y ayuda. Eternamente gracias.

A todos mis compañeros de clases, de cuarto, por reír, criticar, charlar y apoyarnos.

A todos aquellos que se acercaron a preguntar, a indagar por el estado de la tesis. Gracias a todos.

DEDICATORIA

A toda mi familia.

A mis tíos que siempre han confiado en mí.

A mis amigos del PRE-universitario Jorge, Raumel, Idalia, Ramón, Yaimara por la confianza que depositaron en mí.

A mi hermanita Bárbara para que siga mi ejemplo.

A mi madre por el tiempo que estuve lejos.

De manera muy especial a mi padre a quien va dedicada por completo este trabajo.

RESUMEN

La Gestión de Configuración de Software (GCS) es una de las actividades claves para que una organización de desarrollo pueda alcanzar el Nivel de Maduración 2 establecido por el Software Engineering Institute (SEI).

Desafortunadamente, la necesidad de implementar estas actividades surge como consecuencia de la aparición de problemas en la calidad de los productos ya desarrollados o en etapas avanzadas de desarrollo, o por dificultades para mantener el ritmo de producción, puesto que en estas circunstancias, el personal asignado a tareas de desarrollo debe también atender pedidos de cambios o mejoramiento de productos finalizados.

Otro factor que contribuye a detectar esta necesidad, es la aparición gradual de programadores, analistas o especialistas, pertenecientes a los grupos de desarrollo, que, con el tiempo, se van convirtiendo en “insustituibles”, pues, ante la falta de una documentación completa y coherente, son los únicos que se encuentran en capacidad de implementar cambios en los productos entregados o en etapa avanzada de implementación.

Estas dificultades se corrigen mediante la adopción de las técnicas de Gestión de Configuración que se encuentran ampliamente desarrolladas en los libros y artículos relacionados con la Ingeniería de Software.

En tal sentido, en el presente trabajo se efectúa una propuesta del Proceso de GCS para el Sistema de Gestión de Inventario y Almacenes (SIGIA), ofreciendo entre sus resultados el Plan de GCS, el procedimiento para controlar los cambios y la estructura del repositorio. Para el control de la integridad y consistencia del código, así como de cada uno de los elementos del software, se propone “Subversion y Trac” como herramientas de apoyo a la GCS.

Además se proponen un conjunto de métricas fácilmente recolectables que pueden ser calculadas en el Proceso de Gestión de Configuración

PALABRAS CLAVE

Calidad, cambios, integridad, gestión, métricas, proceso.

TABLA DE CONTENIDOS

Introducción

Los métodos de desarrollo de software están evolucionando desde formas artesanales a la producción industrial en gran escala. La industria de software cubana no está ajena a esos cambios. Para incidir positivamente en su desarrollo y lograr establecer en ella parámetros de excelencia es imprescindible implantar modelos de procesos tomando en consideración las mejores prácticas internacionales y adaptándolas creativamente a las condiciones concretas del país. Para esto, entre otros aspectos, es necesario **medir**.

Los sistemas informáticos enfrentan largas vidas útiles, durante las cuales el entorno de la organización y las condiciones de los requisitos iniciales varían constantemente. Para que tales sistemas respondan a los requisitos funcionales para los que fueron desarrollados y que además se ajusten a las nuevas condiciones, se hace necesario **configurar** y **mantener** el software.

En general los sistemas de software son complejos y su tamaño crece a medida que le son adicionadas nuevas funcionalidades. Las actividades de mantenimiento realizadas para ajustar el software a los nuevos requisitos lo deterioran en su estructura, de manera que en el tiempo el software ya no corresponde a la arquitectura considerada para su desarrollo, el código no corresponde a la documentación y cada vez es más difícil entender el sistema.

Con la idea de que los sistemas de desarrollo de software sean un proceso definido, repetible, en lugar de una actividad artesanal y reinventada para cada nuevo proyecto, este trabajo propone diseñar como actividad de apoyo a la Ingeniería de Software el Proceso de Gestión de Configuración del Software (GCS) para el proyecto Sistema de Gestión de Inventario y Almacenes (SIGIA).

La aplicación de este proceso no solo hará posible disciplinar a los involucrados, permitirá además almacenar y disponer de los datos históricos necesarios para lograr un trabajo más predecible y eficiente.

Como todo proceso, la GCS también puede ser sistematizada y automatizada, lo que se denomina un Sistema de Gestión de Configuración (SGC). Actualmente

existen en el mercado diversas herramientas que permiten apoyar una o más actividades de la Gestión de Configuración como por ejemplo, el CVS, SVN Subversion, Source Safe, Clear Case, Darcs, Plastic SCM, entre otras. La utilización de algunas de estas herramientas resultaría muy costosa en recursos humanos, tiempo y dinero, tal es el caso del Rational ClearCase. (Febles, 2004).

A pesar de todas las facilidades que brinda la Gestión de Configuración del Software, en el proyecto SIGIA no se establecen estrategias para crear disciplina y aplicar las mejores prácticas en el desarrollo de software. Se encuentran problemas relacionados con el control de versiones en los módulos, los desarrolladores efectuaban cambios no supervisados que finalizaban en resultados negativos, por falta de una visión global de la configuración del producto y la repercusión de dicho cambio. Además no existe:

- Procedimiento para llevar a cabo las actividades relacionadas con la configuración en el desarrollo del software.
- Un formato preestablecido para hacer pedidos de cambios.
- Un seguimiento de los cambios de cada versión del proyecto.

Lo que deriva en la mala utilización del recurso humano (los desarrolladores repiten el trabajo antes realizados o en peor caso hacen lo que no tienen que hacer) provocando atrasos en actividades de desarrollo y en la entrega del producto final, lo que sin lugar a dudas afecta la calidad del software.

La necesidad de hacer más eficaz, eficiente y efectiva la producción de software en el proyecto SIGIA, para minimizar las deficiencias apuntadas, y la urgencia de garantizar disciplina en la ejecución de los procesos asociados a la Gestión de Configuración de Software (GCS) lleva a definir el siguiente **Problema de investigación**:

¿Cómo diseñar el Proceso de GCS en el proyecto SIGIA guiado por el Proceso Unificado de Desarrollo (con sus siglas en inglés “RUP”) que garantice la utilización de los estándares internacionales en la GCS?

Después de ser analizados todos los problemas que existen en el proyecto SIGIA se plantea como **Objetivo general:** diseñar el Proceso de GCS en el Proyecto SIGIA guiado por RUP.

El **objeto de estudio**: Proceso de Gestión de Configuración del Software guiado por RUP y el **campo de acción**: Proyecto SIGIA.

Para guiar la investigación se hace necesario apoyarse en las siguientes preguntas:

- ¿Qué elementos componen la versión X del producto?
- ¿Cuál es el impacto de un determinado cambio?
- ¿Cuándo y a quién se distribuyó la release (liberaciones) Y?
- ¿Qué ha ocurrido y cuando ha ocurrido?

Para dar cumplimiento al objetivo se acometieron las siguientes **tareas:**

1. Realizar un estudio del estado del arte sobre el tema y del marco teórico que fundamenta el objeto de la investigación.
2. Evaluar y seleccionar las actividades de GCS planteadas por RUP que formarán parte del proyecto SIGIA.
3. Evaluar y seleccionar las herramientas candidatas para la automatización del proceso de GCS.
4. Proponer las métricas que serán aplicadas para mejorar el proceso de GCS.

Para garantizar que la propuesta del proceso de Gestión de Configuración tenga rigurosidad, se hace necesario estudiar, para luego aplicar los métodos teóricos que se exigen para la determinación de los métodos de investigación, los mismos fueron seleccionados y aplicados sobre la base de los métodos científicos generales.

En la primer etapa del estudio, se procedió a la revisión bibliográfica en Internet, consultas a libros y entrevistas a entendidos en el tema, ordenando el conocimiento ya existente y satisfaciendo las necesidades de búsqueda,

basándose en los **métodos teóricos** a través del **análisis o estudio** de documentos que abordaran de manera directa o indirecta la GCS.

El **método analítico - sintético** permitió la descomposición del tema y analizar su contenido. El método **Inductivo – deductivo** ayudó a entender los análisis y propuestas a la solución del problema, así como conocer las relaciones entre los distintos elementos de la investigación para poder realizar las generalizaciones que parten de los **análisis histórico – lógico** a través de los cuales se profundizó en las tendencias, regularidades y cualidades del objeto de estudio y en los argumentos que antecedieron al problema. En este sentido a través del **método comparativo** se establecieron las analogías y diferencias con relación al objeto de estudio para dar solución al problema en cuestión.

En el caso de los **métodos de nivel empíricos** se partió de la **observación**, esto permitió hacer una valoración de la situación actual del proyecto y de su evolución.

La **significación práctica** consiste de manera general en disponer el diseño del Proceso de GCS para el proyecto SIGIA, el cual incluye procedimientos y la utilización de una herramienta para la GCS, lo que permitirá dar seguimiento al desarrollo de los productos de software en el proyecto, garantizando de esta forma calidad en la entrega del producto final.

Poder aplicar estos procedimientos de forma automatizada hará su aplicación más fácil y cómoda para los desarrolladores garantizando alcanzar más rápidamente una disciplina de trabajo. (Febles, 2000).

La **actualidad y pertinencia social** radica en que el tema propuesto, además de dar respuestas a la problemática actual del Proyecto SIGIA puede ser aplicado en los distintos proyectos productivos de la Facultad-3 y en la universidad en general. Puede ser interpretada como un comunicador al interior de cada proyecto. La propuesta de la GCS para el desarrollo del proyecto SIGIA se ajusta a las nuevas modalidades de la industria del software, por lo que está a tono con las exigencias del mercado del software y con la política organizacional de los proyectos productivos de la Facultad 3. Contribuye además a obtener software de calidad.

La estructura de este trabajo se ha establecido en dos capítulos. El Capitulo 1 *Fundamentación Teórica* es una introducción al entorno de Gestión de Configuración. La intención de esta parte es introducir el tema principal y, lo que es más importante, presentar los conceptos necesarios para los capítulos posteriores. En el Capítulo 2 *Descripción de la solución propuesta* presenta en detalle cada una de las tareas de la Gestión de Configuración que deben ser aplicadas en el proyecto SIGIA y las métricas que serán utilizadas para medir la eficiencia y eficacia del proceso de GCS.

CAPITULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Estado del arte de la Industria del Software.

En nuestros días se vive una revolución informática y los paradigmas con los cuales se veía la economía hace 15, 20, 30 años no son los mismos hoy en día. Actualmente, la empresa más rentable del mundo es una de software al igual que la persona más rica del mundo es un empresario de software, situación impensable hace 40 años, cuando las empresas que integraban esas listas eran petroleras o siderúrgicas.

En ese sentido, el desarrollo de software constituye un sector esencial a escala mundial, se encuentra en el centro de todas las grandes transformaciones; sobre todo si se considera que los grandes temas del momento, como lo son la economía digital, la evolución de las empresas y la administración del conocimiento, se resuelven con software.

La industria del software interviene en todos los procesos que habilitan a la "nueva economía", se le considera una industria blanca que no contamina y que genera fuentes de trabajo bien remuneradas. Entre los casos de éxito que nuestro país debe tomar en cuenta, están el de la India -donde se manufactura software-, Brasil -donde el gobierno incentiva la creación de empresas con el fin de competir eficazmente con Estados Unidos, Irlanda y Canadá.

Sin embargo, generalmente el interés de los gobiernos por producir software no es del todo exitoso. Informes de instituciones dedicadas al análisis de software muestran los siguientes indicadores (Reo, 2002) (Bedini, 1995) (Butler, 2001) (Camou, 2002):

- 25% de los proyectos de software son abortados.
- Se liberan productos a sus clientes con remanentes del 15% de defectos.
- Muchas empresas gastan de 30% a 44% de su tiempo y dinero en retrabajo de software ya liberado.
- Solo en el 53% de los casos se cumplen las planificaciones de tiempo.

A pesar de esos problemas la inserción de los países, principalmente los países de Latinoamérica, es cada vez mayor en el mercado latinoamericano de Tecnología de la Información y las Comunicaciones (TIC). La siguiente tabla muestra datos que demuestran la afirmación anterior.

País	Posición en Mundo	LA	Servicios (unidades)	Gastos Internos (dólares)	Participación (%) Mundo	LA
Brasil	11	1	8.816	3.583	45	1.3
México	19	2	3.316	2.326	21	0.6
Argentina	30	3	1.729	634	10	0.3
Colombia	40	4	694	797	5	0.2
Venezuela	41	5	687	797	5	0.2
Chile	45	6	629	490	4	0.1
Resto			2.252	364	9	0.3
Total LA			18.123	8.991	100	2.9

Tabla 1.1 Mercado Latinoamericano de Tecnología de la información- 2005.
Fuente: www.proargentina.gov.ar

1.1.1 Análisis de la situación actual de la Industria Cubana del Software (ICSW).

Dentro de las proyecciones del gobierno cubano se encuentra el fomento de una Industria Cubana del Software (ICSW), que permita diseñar y proveer equipos electrónicos y sistemas informáticos que beneficien a la sociedad y también con posibilidad de exportarlos para aportar a la base material de todos los programas del país. (SIME, 1997) (Lage, 2000) (González, 2003).

Actualmente Cuba se esfuerza en la formación del capital humano para la producción de software y para la informatización de la sociedad. Muestras claras de este interés, es la creación de la Universidad de las Ciencias Informáticas (UCI), con una matrícula de 10000 estudiantes, futuros profesionales de la Informática y como alternativa para el desarrollo económico del país.

Las empresas cubanas, a pesar de la potencialidad que poseen, tienen índices de exportaciones irrisorios. Se posee un mercado interno relativamente fuerte en comparación con lo exportado que es muy pequeño como se aprecia en la Fig.1.1. (Moreno, 2003).

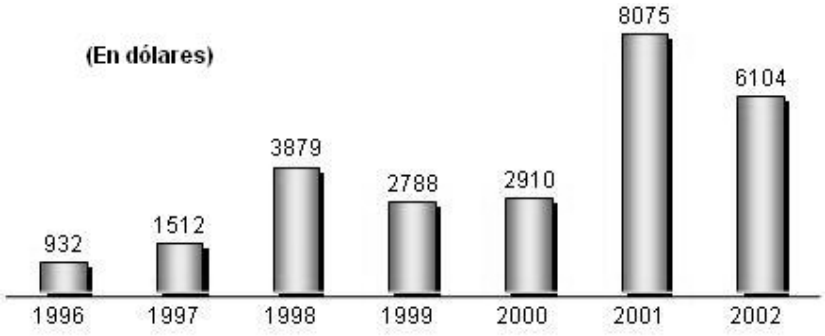


Figura 1.1 Exportación cubana de Software y Servicios informáticos.

De manera similar se destacan como problemas más importantes de la ICSW los siguientes (García1 & Álvarez, 1999) (García, 2000) (Álvarez, 2003) (Febles, 2000) (Febles, 2002) (MIC, 2002) (Moreno, 2003):

- Pobre planificación y organización del trabajo.
- Escaso desarrollo de proyectos de investigación y desarrollo.
- No se aplican estándares de calidad.
- Pobre formación y superación continua de los recursos humanos.
- Poca aplicación de técnicas de comercialización.
- Ausencia de una política salarial y de estimulación coherente para el personal informático.

Con el análisis de estos problemas cabe decir que el reto fundamental es lograr organizaciones de software maduras, caracterizadas por su alta capacidad para administrar los procesos de software con enfoque visionario, que garanticen la motivación, el compromiso, la participación y la destreza del personal, considerando estos elementos parte de la escala de valores del especialista de software. (Febles, 2004).

1.2 Proceso de Desarrollo de Software.

Un Proceso de Desarrollo de Software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

Un PROCESO define: “QUIEN”, “QUE”, “CUANDO”, y “COMO” hay que realizar las cosas para alcanzar un determinado producto de software.

Como la Gestión de Configuración del Software es un <<proceso>> dentro del proceso de desarrollo de software, en su modelación define como sus principales elementos:

Información	Descripción
Administrar la GCS	**¿Quién?** Identifica las responsabilidades y autoridades para dar cumplimiento al Plan de Gestión de Configuración.
Actividades de GCS	**¿Qué?** Identifica todas las actividades de GCS que serán ejecutadas en el proyecto.
Planificar la GCS	**¿Cuándo?** Identifica en qué momento se realizarán las actividades de GCS y su relación con otras actividades del proyecto.
Recursos de GCS	**¿Cómo?** Especifica las herramientas y los recursos humanos que serán necesarios para ejecutar las actividades de GCS.

La Gestión de Procesos está obligada a responder a los procesos que tienen lugar en una organización; donde estos deben estar bien identificados y analizados.

La introducción del enfoque de procesos en el perfeccionamiento de la organización y gestión de las empresas constituye un elemento para elevar su competitividad y gestión al cliente final, o sea, el desarrollo de los procesos debe asegurar simultáneamente, la solución de los problemas de calidad, eficiencia y efectividad.

El eficiente y eficaz funcionamiento del proceso no se realiza automáticamente, sino que requiere de una gestión. Esta gestión es una secuencia de actividades que permiten captar, registrar, trasmitir, procesar y analizar la información y transformar esta información en órdenes a través de decisiones.

La correlación entre los resultados obtenidos en el proceso y los recursos utilizados determina la eficiencia del proceso. La correlación entre los resultados y los efectos logrados constituye la eficacia del proceso. Por último, la correlación entre los recursos utilizados y los efectos logrados determina la efectividad del proceso. El análisis y diseño de los procesos tiene como objetivo incrementar su eficiencia, eficacia y efectividad.

1.3 Gestión de Configuración del Software. Definiciones.

El término gestión se define en el Diccionario Ilustrado Océano de la Lengua Española como la acción y efecto de gestionar, es decir, es hacer diligencias para lograr un negocio o fin. Por otro lado el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) define que una configuración está conformada por requisitos, artefactos de diseño y de implementación que definen una versión particular de un sistema o de un componente de un sistema en un momento dado. (IEEE, 1990) (Buckeley, 1999) (Burrows, 1999) (Ibargüengoitia, 2002).

Para una definición más precisa del término se recurre a algunos de los autores más acreditados que comenzaron en su momento a utilizar el término o bien a las definiciones dadas por organizaciones internacionales de prestigio tales como IEEE o la Organización Internacional para la Estandarización (ISO). Así, se ha seleccionado las siguientes definiciones de Gestión de Configuración:

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. (Antonio, 2001) (Folz, 2000) (Rational Software Corporation, 2003).

El diccionario de términos IEEE en (Piatini, 1996) define Gestión de Configuración como el proceso de identificar y definir los ítems (elementos) de configuración en un sistema, controlando la entrega y el cambio de estos elementos a través del ciclo de vida del sistema, almacenando el estado de los ítems de configuración y de las solicitudes de cambio, y verificando la completitud con respecto a los requerimientos especificados.

La norma ISO 10007 (ISO 10007, 1995) define el objetivo principal del Software Configuration Manager (SCM / GCS) como: Documentar y proveer visibilidad de los productos de software y del estado de progreso en la satisfacción de los requerimientos funcionales y físicos.

Capability Maturity Model (CMM) (Paulk, 1993) es un modelo de capacidad de maduración del software, el cual se refiere a la dimensión administrativa de SCM (GCS). Este modelo establece el propósito SCM como: Establecer y mantener la integridad de los productos de software a lo largo de su ciclo de vida.

Para (Rational Software Corporation, 2003) 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 GC. La GC tiene que ver con la definición de la configuración, así como la construcción, el etiquetado y recolección de versiones de los artefactos.

Para Babich (Babich, 1986) “El arte de coordinar el desarrollo de software para minimizar la confusión se denomina Gestión de Configuración. La Gestión de Configuración es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación. El objetivo es maximizar la productividad minimizando los errores.”

A partir de las definiciones antes presentadas y después de haber establecido analogías y diferencias entre ellas, la autora se adscribe al criterio de Ivar Jacobson, Martín Griss y Patrick Jonson en el libro “Software Reuse: Architecture, Process, and Organizations for Busisness Success” en la que plantean (Jacobson, 2000):

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

Por otra parte algunos autores valoran la GCS como un proceso sumamente importante en el éxito de un producto software. Al respecto la Ley Fundamental de la Gestión de Configuración (Brown, 1998), plantea:

"La Gestión de Configuración es el fundamento de un proyecto software. Sin ella, no importa cuan talentoso sea el equipo, cuan grande sea el presupuesto, cuan robusto sean los procesos de desarrollo y prueba, o cuan 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".

1.4 Gestión de Configuración del Software y Calidad.

Incursionar en otros mercados requiere contar con un aval sólido que demuestre calidad. Es entonces cuando entran en juego los procesos de certificación que aunque para efectos de comercialización a nivel nacional, no son indispensables, llevan a la empresa desarrolladora a fortalecer su estructura interna, de una manera planificada, organizada y con objetivos muy claros.

Una opción importante para incursionar en el mercado internacional es contar con la certificación de calidad en CMM (Capability Maturity Model) con nivel 2 al menos, desarrollado por el Instituto de Ingeniería de Software (SEI) en el año 1991. CMM provee a las organizaciones de una guía para incrementar el rendimiento de sus procesos de negocios, indicando que la productividad y la calidad de los productos y servicios, aumenta conforme crece el nivel de madurez de los procesos que lo crean. El modelo describe cinco niveles de madurez donde cada uno, claramente definido, indica un nivel de prestación o rendimiento particular. . Los niveles de madurez de CMM se pueden representar resumidamente como se muestra en la Fig. 1.2 (Kan, 1995) (Humphrey, 1995).

Para poder escalar a un nivel superior es necesario haber cumplido todos los aspectos que plantea el nivel actual, por lo que es posible incluso, cumplir con aspectos de nivel 2 y de nivel 3 simultáneamente y, aún así, mantenerse en un nivel 1(Delgado, 2006).

Existen algunas estadísticas que aseguran que al cabo de tres años de la aplicación del modelo en empresas que se encontraban en el nivel uno estas alcanzaron el nivel dos. Para alcanzar los dos últimos se necesita un año más en cada caso.

Adicionalmente, su enfoque a procesos, no solo garantiza la calidad de los productos sino de la disciplina productiva, no obliga al uso de una metodología en particular, ni lenguajes de programación, ni tecnologías; como tampoco sugiere estándares específicos de calidad. (Yebra, 2001).

La principal deficiencia de este modelo es lo complejo que resulta su aplicación en las organizaciones, aunque actualmente se han publicado submodelos que facilitan este proceso (Personal Software Process, PSP y Team Software Process, TSP). (Humphrey, 2000). También al ser un modelo tan exigente retarda su aplicación y el paso de un nivel a otro no es tarea de un día lo que implica que muchas organizaciones abandonen su aplicación.

	Características	KPAs	Resultado
5 Optimizado	Bases cualitativas para una inversión capital continua en la automatización y mejoramiento de la retroalimentación de los procesos.	-Prevención de defectos -Administración de la tecnología de cambio -Administración del proceso de cambio	**Productividad y calidad**
4 Administra	(cuantitativo) -Procesos medidos -Control estadístico razonable sobre la calidad del producto	-Medición y análisis de los procesos -Administración de la calidad	
3 Definido	(cuantitativo) -Costo y planificación confiables -Procesos definidos e institucionalizados -Rendimiento de la calidad mejorado pero impredecible	-Definición y mejoramiento de los -procesos organizacionales -Programas de entrenamiento -Administración integrada de software -Coordinación intergrupal -Inspecciones entre compañeros -Ingeniería del producto de software	
2 Repetible	(intuitivo) -Procesos dependientes de los individuos - Costo y calidad altamente variables - Métodos y procedimientos informales y “ad hocs”	-Administración de los requerimientos -Planificación y vigilancia de los proyectos de software -Administración de la subcontratación de software -Aseguramiento de la calidad del software -Administración de la configuración de software	
1 Inicial	-Ad hoc -Caótico -costos, planificación y rendimiento de la calidad impredecibles		*Riesgos*

Figura 1.2. Representación esquemática de CMM.

Fuente: http://www.sei.cmu.edu/cmm/cmms/cmms.integration.html

1.5 Herramientas de apoyo a la Gestión de Configuración del Software.

En la actualidad desarrollar software es una propuesta interesante. Sin embargo, se necesitan facilitadores para poder obtener resultados de alta calidad. Mediante la GCS y herramientas CASE (Computer Aided Sofware Engineering) que se puede traducir como Ingeniería de Software Asistida por Computadora, es posible obtener resultados repetibles y de alta calidad con un mínimo de esfuerzo.

IBM Rational ClearQuest

IBM Rational ClearQuest permite efectuar seguimiento de defectos y cambios en toda la empresa (International Business Machine, 2005).

Entre las actividades que realiza Rational ClearQuest se encuentran:

- Seguimiento de cambios y defectos basado en actividades.
- Soporte para flujos de trabajo, que incluye notificaciones por correo electrónico y opciones de envío.
- Soporte para consultas con generación de múltiples informes y gráficos.
- Interfaz Web para acceder fácilmente desde cualquier navegador Web estándar.
- Integración con Rational ClearCase para conseguir una solución SCM completa.
- Integrado con los Entornos de Desarrollo (IDE) líderes en el sector, como WebSphere Studio, Eclipse y Microsoft® .NET

Sin duda una de las mayores dificultades de los productos Rational es su elevado costo. Así como la alta generalidad con que cuenta en tareas como la generación de informes, que por el grado de flexibilidad y parametrización que posee, tiene una curva de aprendizaje alta, requiriendo de grandes esfuerzos para su asimilación.

Rational ClearCase

"IBM Rational ClearCase" es otra de las herramientas incluidas en la Suite Rational 2003. Ofrece una gestión de los activos de software para equipos de desarrollo de mediano y gran tamaño (International Business Machine, 2005).

Entre las funcionalidades de Rational ClearCase están:

• Gestión del ciclo de vida y control de los activos de desarrollo de software.
• Control integrado de versiones.
• Gestión de Línea Base.
• Integración con los IDEs líderes en el sector, como WebSphere Studio, Eclipse y Microsoft® .NET
• Integración con Rational ClearQuest.

Microsoft Visual SourceSafe. VSS

Microsoft Visual SourceSafe (VSS) es una herramienta para el control de versiones que brinda soporte para puntos de restauración y capacidad de colaboración, que permite a los desarrolladores trabajar simultáneamente sobre una versión de un producto (MSDN, 2005). MS Visual SourceSafe permite:

- Regresos (roll back) a versiones anteriores de un fichero.
- Ramificar, compartir, unir y administrar los entregables.
- Seguimiento de versiones de todo el proyecto.
- Seguimiento a código modular (un fichero que es usado o compartido por múltiples proyectos).
- Cuenta con las funcionalidades básicas de todo gestor de versiones (ultima versión, "check In", "check out")

El VSS, no presenta un costo elevado, en consecuencia con el soporte que brinda a las actividades de GC, pues sólo enmarca su trabajo en el control de versiones, por lo que para el desarrollo individual, podría resultar suficiente, mas al entrar a formar parte del trabajo en equipo, carece de un valor más allá del mero control

de versiones ya que las funcionalidades para la coordinación del trabajo colaborativo no puede brindarla por sí solo.

Otro elemento que no por último resulta menos importante es el carácter propietario de la herramienta, y su procedencia, al ser un producto de Microsoft, corporación estadounidense, no puede ser comercializado con Cuba.

Source Forge

Source Forge es un ambiente de desarrollo colaborativo, herramientas heterogéneas y procesos con un entorno integrado para la administración de proyectos, administración de cambios y capacidades de colaboración (SourceForge, 2005).Entre las funcionalidades de Source Forge se encuentran:

Herramientas de proyectos

- Seguimiento de problemas y cambios.
- Administración de tareas.
- Sistema de liberaciones (release) de ficheros.

Colaboración

- Administración de documentos.
- Forum de discusión y noticias.
- Lista de correos.

Interoperabilidad

- Microsoft Project.
- Microsoft Office.
- Sistema de Control de Versiones (Subversion, CVS, Rational ClearCase).

Plastic SCM

Plastic SCM es el nombre que engloba toda la gama de productos de Gestión de Configuración de Códice Software.

Se compone de un potente control de versiones y de un sistema configurable para la gestión del ciclo de vida del desarrollo. Ofrece:

- Espacios de trabajos configurables.
- Versionado completo de la estructura del proyecto.

- Excelente soporte de desarrollo en paralelo.
- Ciclo de vida del desarrollo configurable.
- Integración con entornos de desarrollo.
- Soporte multiplataforma.
- Completo sistema de seguridad basado en ACLs
- Nuevas fórmulas de visualización.

1.5.6 Subversion y Trac. Herramienta seleccionada.

Subversión es una herramienta para el control de versiones de software. Permite mantener en un repositorio central la última versión de los archivos de un proyecto, coordinando los registros de cambios y la gestión de los usuarios que pueden acceder al proyecto. Subversión es significativamente más completa que otros frameworks de control de versiones como CVS, ya que mejora todas las operaciones de actualización, eliminación, renombramiento y movimiento de archivos. Subversión permite mantener varios proyectos para distintos usuarios, registrando los cambios del código.

Trac es un sistema de Wiki mejorado que permite el manejo y gestión de hitos para desarrollo de proyectos de software. Trac utiliza una aproximación minimalista para el manejo de los proyectos, ya que la misión fundamental es ayudar a los desarrolladores a escribir un buen software manteniendo la gestión de los cambios controlada. Trac provee una interface con Subversion, e integra un Wiki que provee herramientas para realizar gestión de metas y reportes de bugs (defectos).

Al ser un Wiki, trac permite la descripción mediante markups, facilitando el posteo de mensajes, creación de vínculos y referencias independientes entre bugs (defectos), tareas, conjuntos de cambios, archivos y páginas wiki. Una línea de tiempo muestra todos los eventos del proyecto en orden, permitiendo en forma simple la visualización, control y monitoreo del proyecto.

Teniendo en cuenta las potencialidades que ofrecen las herramientas libres como Subversión (para control de versiones) y Trac (para la GCS), se ha seleccionado estas herramientas para la GCS en el proyecto SIGIA, partiendo del criterio que la

integración de estas tecnologías permite un desarrollo altamente escalable para la construcción de productos y proyectos en plataformas heterogéneas, lo cual es completamente extensible a proyectos opensource (código abierto). Además que responde a las expectativas de los proyectos productivos de la UCI de migrar en un futuro a el uso de Software Libre como una alternativa para el desarrollo de software.

Subversion y Trac permite automatizar:

- **Control de versiones**: su staff (desarrolladores) podrán desarrollar software con la confianza de que sus cambios permanecerán seguros y auditables.
- **Control de *releases*** (liberaciones*)*: su staff podrá marcar y obtener *releases* de software, y certificar que sus *releases* correspondan con los entregables.
- **Control de defectos**: mediante *workflow* (flujo de trabajo) integrado de gestión de defectos, se podrá disminuir el tiempo de respuesta para las solicitudes de cambio y defectos que los clientes presenten.

Total transparencia Internet: sus procesos podrán ser públicos si así se escoge, mientras que los desarrolladores podrán estar distribuidos en todo el mundo, y trabajar con tranquilidad gracias al cifrado Secure Sockets Layers (Protocolo de Seguridad en transacciones electrónicas).

Después de ser analizadas las principales características y funcionalidades de estas herramientas para la Gestión de Configuración de Software, si bien es cierto que todas estas dan soporte a muchas de las actividades incluidas en la GCS, dejan a un lado cosas muy importante como la **identificación de los elementos de configuración** y la **administración de líneas base** y muchas toman solo como GCS el control de versiones. (Delgado, 2006).

La siguiente tabla muestra un resumen comparativo de algunas de las herramientas de GCS.

RESUMEN COMPARATIVO	**CVS**	**SVN**	**Source Safe**	**Rational Clear-Case**	**Plastic SCM**

Commits atómicos (Si una operación de commit es interrumpida, el sistema no queda en un estado inconsistente)	No	Si	No	Si	Si
Movimiento y renombrado de directorios (¿Soporta el sistema mover directorios Manteniendo la historia de los ficheros y directorios?)	No	Renombra-do, soportado. Movimiento limitado.	No. Se recrea creando un nuevo elemento con el nuevo nombre partiéndo-se la historia en dos.	Si	Si. En Plastic los directorios son elementos versionados de primer nivel, como los ficheros
Permisos del repositorio (¿Es posible dar permisos a diferentes elementos del repositorio?)	No	Limitado. Soporta permisos WebDAV.	Limitado. Solo permisos de lectura, escritura, borrado y borrado del repositorio	Limitado Solo permisos de lectura, escritura borrado y borrado del reposito-rio	SI. Es posible fijar ACLs a todos los elementos del sistema.
Soporte de changesets (Los changesets agrupan modificaciones en un solo paquete)	No	Si. Cada commit lleva un changeset implícito.	No	No	Si. De dos maneras (1)
Conocer todos los contribuidores de un fichero línea a línea, para saber de dónde viene cada cambio (annotate)	Si	Si	No	Si	Si
Control de modificaciones no enviadas al servidor	Si. Permite ver las diferencias entre lo que hay en el espacio de trabajo Y lo que hay en el servidor.	Permite ver las diferencias Entre lo que hay en el espacio de trabajo y lo que hay en el servidor.	Si. Permite ver las diferen-cias entre lo que hay en el espacio de trabajo y lo que hay en el servidor.	Permite ver las diferen-cias entre lo que hay en el espacio de trabajo y lo que hay en el servidor.	Si. Además de permitir ver las diferencias, es capaz de almacenar los cambios intermedios (Checkouts) en el servidor.

Mensajes de commit por fichero Capacidad de asociar un mensaje a cada check-in.	No	No	Si	Si	Si
Facilidad de instalación	No	Dos modos: instalado como un módulo de Apache o con svnserver. Requiere cierto trabajo.	Si. Mediante un paquete de instalación.	No	Si. Un simple instalador, y un wizard de configura-ción inicial.
Portabilidad	No	No	Si	Si	Si
Integración con entornos de desarrollo	Si	Muy buena. Unix, Windows y MacOS	Limitada. Solo Windows.	Mediana. Windows y UNIX.	Muy buena. Windows, Unix/Linux y MacOSX
Soporte técnico.	No	Consultas a los voluntarios En los foros.	Si. Aunque es un producto depre-cated por MS.	Completo	Completo

Tabla 1.2 Resumen comparativo de herramientas de GCS.

(1) Lo soporta de dos maneras: changesets asociados a cada checkin, pudiendo elegir que elementos entran en cada uno, y soporte extensivo de ramas que permite aislar los cambios.

Fuente: www.codicesoftware.com

1.6 Gestión de Configuración del Software.

Existen dos actividades que forman parte del Sistema de Control del Proyecto: el Sistema de Calidad y el Sistema de Gestión de Configuración, considerada también como una actividad de garantía de calidad. (Antonio, 2001).

Como ya se ha dicho anteriormente la GCS tiene como objetivo principal establecer y mantener la integridad de los productos generados durante un proyecto de desarrollo de software y a lo largo de todo el ciclo de vida del producto.

La integridad de un producto software significa que el producto cumple las siguientes condiciones (Antonio, 2001).

- Satisface las necesidades del usuario (cumple todos los requisitos del usuario, tanto los explícitos como los implícitos).
- Cumple los requisitos de rendimiento.

- Se puede trazar su evolución desde que se concibió, y a través de todas las fases de su ciclo de vida.

Es importante diferenciar entre mantenimiento del software y la GCS. La primera es un conjunto de actividades de ingeniería del software que se producen después de que el software haya sido entregado al cliente y esté en funcionamiento. En cambio, la GCS es un conjunto de actividades de seguimiento y control que comienza cuando se inicia el proyecto de ingeniería del software y termina solo cuando el software está fuera de circulación. (Pressman, 2002).

Las actividades correspondientes al área de GCS según (Gervás, 2002 (Antonio, 2001) (IEEE, 1987) (IEEE, 1990)] son las siguientes:

1. **Identificación de la Configuración**: Consiste en identificar la estructura del producto, sus componentes y el tipo de estos, y en hacerlos únicos y accesibles de alguna forma.
2. **Control de Cambios en la Configuración**: Consiste en controlar las versiones y entregas de un producto y los cambios que se producen en él a lo largo del ciclo de vida.
3. **Generación de Informes de Estado**: Consiste en informar acerca del estado de los componentes de un producto y de las solicitudes de cambio, recogiendo estadísticas acerca de la evolución del producto.
4. **Auditoria de la Configuración**: Consiste en validar la completitud de un producto y la consistencia entre sus componentes, asegurando que el producto es lo que el usuario quiere.

Con igual propósito en **CMM**, para la Gestión de Configuración, se proponen los procesos o prácticas claves siguientes (Paulk, 1993) (Paulk, 1999):

1. Planificación de las actividades de Gestión de Configuración

2. Identificación de los ECS
3. Control de cambios a los ECS
4. Informar a los grupos e individuos involucrados de los cambios a los ECS
5. Auditoria de la Configuración

Por otra parte **ISO** sugiere para esta área los siguientes procesos (ISO 10007, 1995) (Burrows, 1999):

1. Identificación de la Configuración.
2. Control de Cambios a la Configuración.
3. Informe del Estado de la Configuración.
4. Auditoria de la Configuración.

Para Rational Unified Process **(RUP)** las actividades correspondientes al Gestor de Configuración del software son (Rational Software Corporation, 2003):

1. **Escribir el Plan de Gestión de Configuración (PGC)**: consiste en describir todas las actividades de GC que serán ejecutadas durante el ciclo de vida del proyecto.
2. **Establecer las políticas de Gestión de Configuración (GC):** consiste en monitorear y defender los bienes del proyecto.
3. **Establecer el ambiente de la GC**: consiste en establecer el ambiente tanto de hardware como de desarrollo donde los productos serán desarrollados y construidos.
4. **Reporte de estado de la configuración**: Consiste en informar acerca del estado de los componentes de un producto y de las solicitudes de cambio, recogiendo estadísticas acerca de la evolución del producto.
5. **Realizar auditoría de la configuración**: consiste en evaluar los procedimientos para llevar a cabo los cambios.
6. **Crear la unidad de despliegue**: consiste en crear la unidad de despliegue lo suficientemente completa para ser descargada, instalada y ejecutada.

Puede apreciarse que existe consenso entre los autores en cuanto a los procesos a incluir dentro de la GCS.

1.6.1 Configuración del Software y Elementos de Configuración.

La configuración del software es el resultado de las actividades de Ingeniería de Software. Contempla todos los productos generados o utilizados en un proyecto.

Se entiende por Elemento de Configuración de Software (ECS) cada uno de los componentes básicos de un producto software sobre los que se realizará un control, tiene un nombre y puede evolucionar. Es la unidad de trabajo para la GCS (Antonio, 2001).

Para controlar y gestionar los ECS, se debe identificar cada uno de forma única (con nombre, descripción, atributos y relaciones) y luego organizarlos mediante un enfoque orientado a objetos. Según (Pressman, 2002) se pueden identificar dos tipos de objetos:

- Objetos básicos.
- Objetos compuestos**.**

Un objeto básico es una “unidad de texto” creado por un ingeniero de software durante el análisis, diseño, codificación o pruebas. Un objeto compuesto es una colección de objetos básicos y de otros objetos compuestos.

1.6.2 Relaciones en la Configuración

Los objetos de configuración estarán “conectados” entre ellos mediante relaciones, lo cual permite a los desarrolladores determinar que otros elementos se verán afectados si se produce un cambio en un determinado elemento.

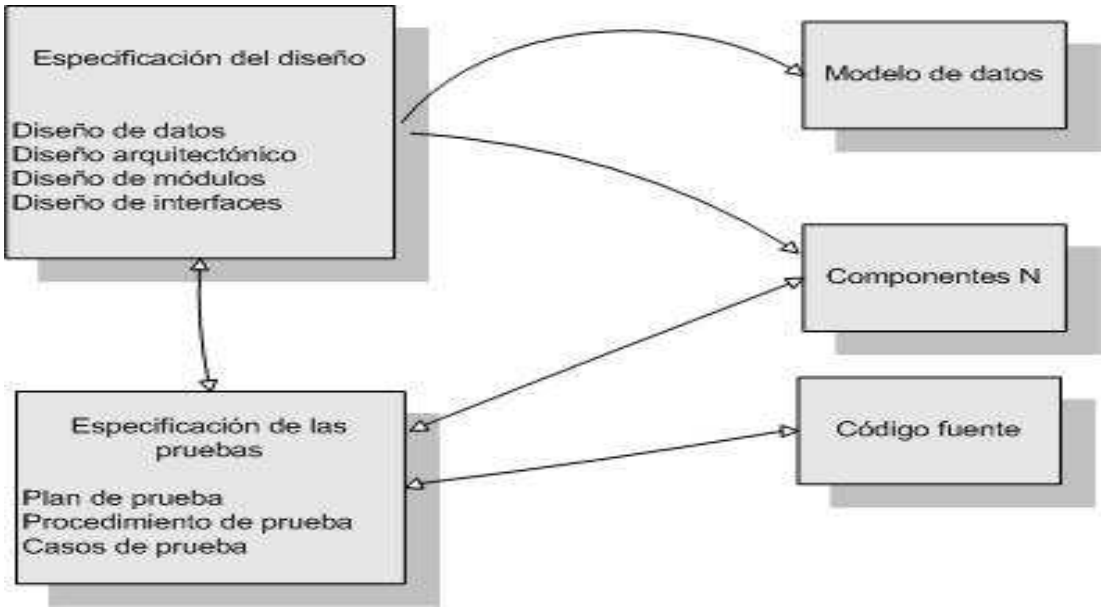


Figura 1.3 Objetos de configuración. (Pressman, 2002).

De acuerdo a la figura anterior los objetos de configuración**, Especificación de diseño, modelo de datos, Componentes N, código fuente y Especificación de prueba** están definidos por separado. Sin embargo, cada objeto esta relacionado con otros como muestran las flechas. Una flecha curvada representa una relación de composición, es decir **Modelo de datos** y **Componentes N** son parte del objeto Especificación de diseño. Una flecha recta con dos puntas representa una interrelación. Si se lleva a cabo un cambio sobre el objeto **código fuente**, se afecta el objeto **Especificación de pruebas**.

Las interrelaciones entre los objetos de configuración se pueden representar con un Lenguaje de Interconexión de Módulos (LIM) (Pressman, 2002). Un LIM describe las interdependencias entre los objetos de configuración y permite construir automáticamente cualquier versión de un sistema.

En (Antonio, 2001) se establecen entre otras, las siguientes relaciones:

- **Equivalencia**: cuando determinado ECS está almacenado en varios lugares diferentes pero todas las copias corresponden al mismo ECS.
- **Composición**: ECS que esté compuesto por otros artefactos de la metodología o de otros módulos del producto.

- **Dependencia**: Relaciones entre ECS, fundamentalmente para facilitar el seguimiento de los requisitos.

1.6.3 Líneas Base

Una línea base es una configuración de referencia en el proceso de desarrollo del software a partir de la cual las revisiones se han de realizar de manera formal. (RTF). (Pressman, 2002).

También se puede definir como un elemento que ha sido revisado y aceptado, que va a servir como base para desarrollos posteriores y que solamente puede cambiarse a través de un proceso formal de control de cambios. (Antonio, 2001).

Tiene como objetivo controlar los cambios en el software, sin impedir llevar a cabo aquellos que sean justificados.

Se definen al comienzo del proyecto, coincidiendo con los hitos marcados y generalmente se corresponden con los resultados de las fases.

Cuando un ECS se convierte en una línea base se introduce en una Base de Datos del Proyecto.

Es importante destacar que los cambios sobre un ECS de línea base produce una nueva versión del elemento.

1.6.4 Bibliotecas de Software

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. Las bibliotecas de software facilitan las tareas de GCS, especialmente en cuanto al Control de Cambios y la Contabilidad de Estado. (Antonio 2001).

El establecimiento de las bibliotecas de software es una de las actividades de la Identificación de la Configuración.

En (Antonio, 2001) se definen un grupo básico de bibliotecas, a ser establecidas:

- **Biblioteca de trabajo**: Se establece al inicio del proyecto, y comprende el área de trabajo donde los analistas y diseñadores elaboran los documentos del proyecto y donde los programadores desarrollan el software, es decir, donde se realiza la codificación y pruebas unitarias. Una vez 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**: 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.
- **Biblioteca de Producción**: Está compuesta por la Biblioteca de trabajo, la de integración y la Biblioteca de Soporte al Proyecto
- **Biblioteca maestra**: Se usa para almacenar ECS liberados para su entrega al cliente o distribución en el mercado. Los elementos en la biblioteca maestra se encuentran sujetos a un control de cambios formal y estricto. Y normalmente esta biblioteca tiene fuertes restricciones de acceso para escritura, aunque normalmente no los tiene para lectura. En esta biblioteca se almacenan las liberaciones (releases) del sistema.
- **Repositorio de Softwar**e: Es la 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 (cinta). 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).

1.6.5 Control de acceso y de sincronización

Los procesos de “alta” y “baja” de los elementos de configuración en el repositorio del proyecto, implementan dos elementos importantes del control de cambios: control de acceso y control de sincronización (Pressman, 2002). El control de acceso gobierna los derechos de los ingenieros de software acceder y modificar objetos de configuración concretos. El control de sincronización asegura que los cambios en paralelo, realizados por personas diferentes, no se sobrescriben mutuamente. (Harter, 1989).

Según (Pressman, 2002) una función de control de acceso comprueba que el ingeniero tiene autoridad para dar de baja a un elemento y el control de sincronización bloquea al elemento en el Repositorio del proyecto, de manera que no se pueden hacer más actualizaciones hasta que se haya remplazado con la nueva versión del ECS.

1.6.6 El cambio en el proceso de desarrollo de software

Hoy en día la mayoría de las empresas que se dedican a la producción de software realizan una muy baja GCS, esto se debe a que la GCS es una actividad bastante compleja. La causa principal de su complejidad consiste en que surge el cambio, sin importar en que momento de desarrollo se encuentre el software (Pressman, 2002). Por otra parte existen una gran cantidad de elementos a controlar porque a medida que avanza el proyecto, crece también el número de elementos que lo conforman.

El cambio se puede producir en cualquier momento y por cualquier razón. La Primera Ley de la Ingeniería de Sistemas (Bersoff, 1980) establece:”Sin importar en que momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida”.

En (Pressman, 2002) se definen cuatro causas fundamentales por las que se puede originar un cambio:

- Nuevos negocios o condiciones comerciales que dictan los cambios en los requisitos del producto o en las normas comerciales.
- Nuevas necesidades del cliente que demandan la modificación de los datos producidos por sistemas de información, funcionalidades entregadas por productos o servicios entregados por un sistema basado en computadora.
- Reorganización o crecimiento/reducción del negocio que provoca cambios en las prioridades del proyecto o en la estructura del equipo de Ingeniería de Software.
- Restricciones presupuestarias o de planificación que provocan una definición del sistema o producto.

Para mantener el control de los cambios producidos es necesario definir un Proceso de Control de Cambios, pero antes hay que tener en cuenta los tres niveles fundamentales de control de cambios (Antonio, 2001):

- **Control de cambios informal**: Antes de que el ECS pase a formar parte de una línea base, aquel que haya desarrollado dicho elemento podrá realizar cualquier cambio justificado sobre él.
- **Control de cambios al nivel del proyecto o semi-formal**: Una vez que el ECS pasa la revisión técnica formal(RTF) y se convierte en una línea base, para que el encargado del desarrollo pueda realizar un cambio debe recibir la aprobación de:
    - El director del proyecto, si es un cambio local
    - El Comité de Control de Cambios, si el cambio tiene algún impacto sobre otros ECS.
- **Control de cambios formal**: Se suele adoptar una vez que se empieza a comercializar el producto, cuando se transfieren los ECS a la Biblioteca Maestra. Todo cambio deberá ser aprobado por el Comité de Control de Cambios.

En un proceso formal o semi-formal, el que toma las decisiones finales acerca de la aprobación o no de un cambio, evalúa el impacto del cambio sobre otros ECS, así como el impacto sobre la calidad del producto, es el Comité de Control de Cambios (CCC).

1.6.7 Versiones, Revisiones, Variantes y Releases (liberaciones)

Versión: Instancia de un ECS 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).

Desde la perspectiva del sistema en su conjunto, una versión es una instancia de un sistema software que difiere en algo de otras instancias (nuevas funcionalidades, opera en hardware diferente, etc.)

Las versiones pueden clasificarse en:

- **Versiones mayores**: que representan importantes despliegues de software y hardware y que introducen modificaciones importantes en la funcionalidad, características técnicas, etc.
- **Versiones menores**: que suelen implicar la corrección de varios errores conocidos puntuales y que a menudo son modificaciones que vienen a implementar de una manera correctamente documentada soluciones de emergencia.
- **Versiones de emergencia:** modificaciones que reparan de forma rápida un error conocido.

Como pueden llegar a existir múltiples versiones es conveniente definir una referencia o código que los identifique unívocamente. El sistema universalmente aceptado es:

- **Versiones mayores**: 1.0, 2.0, etc.
- **Versiones menores**: 1.1, 1.2, 1.3, etc.
- **Versiones de emergencia**: 1.1.1, 1.1.2, etc

En su ciclo de vida una versión puede encontrase en diversos estados: desarrollo, pruebas, producción y archivado.

La siguiente figura ilustra gráficamente la evolución temporal de una versión:

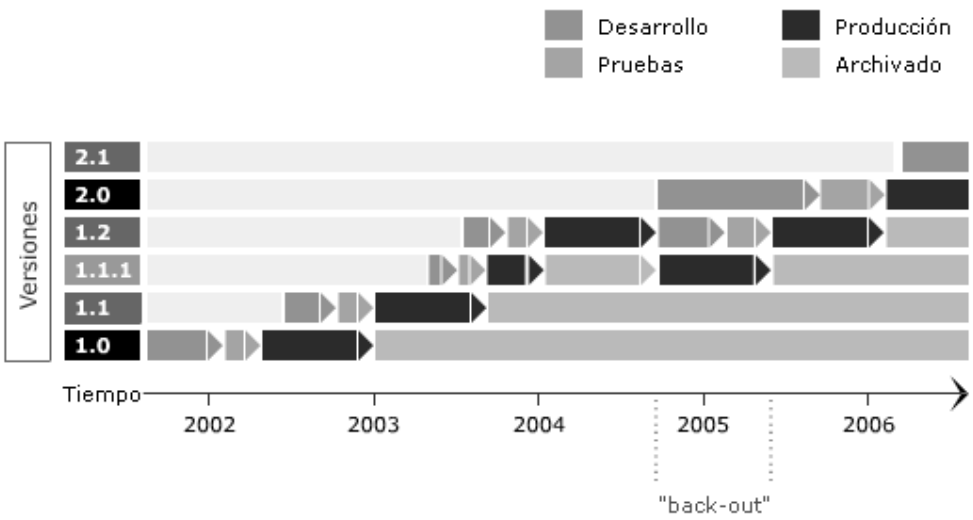


Figura 1.4. Evolución temporal de una versión.

Fuente:
http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_versiones/vision_general_gestion_de_versiones/vision_general_gestion_de_versiones.php

El despliegue de nuevas versiones puede realizarse de diferentes maneras y es responsabilidad de la **Gestión de Cambios** el determinar la forma más conveniente de hacerlo. Entre las opciones más habituales cabe destacar:

- **Versión delta**: sólo se testean (prueban) e instalan los elementos modificados. Esta opción tiene como ventaja su mayor simplicidad pero conlleva el peligro de que puedan aparecer problemas e incompatibilidades en el entorno de producción.
- **Versión completa**: Se distribuyen todos los elementos afectados ya hayan sido modificados o no. Aunque esta opción es obviamente más trabajosa es más improbable que se generen incidentes tras la instalación si se han realizado las pruebas pertinentes.
- **Paquete de Versiones**: La Gestión de Cambios puede optar por distribuir de forma sincronizada diferentes paquetes de versiones, de esta forma se ofrece una mayor estabilidad al entorno. En algunos casos esta opción es obligada por incompatibilidades entre una nueva versión con software o hardware previamente instalado. Por ejemplo, en la migración a un nuevo sistema operativo que requiere hardware más avanzado y/o nuevos versiones de los programas ofimáticos.

Revisión: Modificaciones secundarias, es decir, son las distintas versiones de un ECS según se va avanzando en el desarrollo de dicho elemento.

Variante: Las variantes son versiones de un elemento de configuración que coexisten en un determinado momento y que se diferencian entre sí en ciertas características (p.ej.: software para múltiples sistemas operativos o para diferentes idiomas). Las variantes representan la necesidad de que un objeto satisfaga distintos requisitos al mismo tiempo.

A diferencia de las revisiones, que son estrictamente secuenciales, y sólo existe una como revisión actual, las variantes se desarrollan en paralelo, y puede haber varias sobre las que se esté trabajando simultáneamente (Antonio, 2001).

Release: Versión que se distribuye a los clientes.

1.7 Métricas en el proceso.

El (IEEE, 1993) define métrica como "una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado".
Un ingeniero del software recopila medidas y desarrolla métricas para obtener indicadores. Un indicador es una métrica o una combinación de métricas que proporciona una visión profunda del proceso, del proyecto de software o del producto en sí (Pressman. 2002).

Se debería recopilar métricas para que los indicadores del proceso y del producto puedan ser ciertos. Los indicadores del proceso permiten a una organización de ingeniería del software tener una visión profunda de la eficacia de un proceso ya existente. También permiten que los gestores evalúen lo que funciona y lo que no. Las métricas del proceso se recopilan durante un largo periodo de tiempo. Su intento es proporcionar indicadores que lleven a mejoras de los procesos de software a largo plazo. (Pressman, 2002).

CAPITULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

En el presente capítulo se diseñan cada una de las actividades del Proceso de Gestión de Configuración del Software para el Proyecto SIGIA, tomando como referencia el flujo de trabajo de RUP.

Para (Rational Software Corporation, 2003) el Gestor de Configuración del Software es el responsable de proporcionar toda la infraestructura de la GC y el ambiente del equipo de desarrollo del producto. Además tiene que garantizar que el ambiente de la GC (se refiere a las bibliotecas) facilita productos revisados y actividades de rastreo de defectos y de cambios.

2.1 Mapa del Proceso de Gestión de Configuración del Software.

Descripción del proceso.	
Código	P01
Proceso	Proceso de Gestión de Configuración del Software.
Responsable	Gestor de Configuración.
Participantes	Gestor de Configuración. Gestor de cambios.
Clientes	Clientes Internos • Equipo de desarrollo.

I – Misión
Identificar, controlar y organizar la evolución del software.

II	**Funciones**
1	Apoyar las actividades de desarrollo del producto a fin de que los desarrolladores tengan su propio espacio de trabajo.
2	Garantizar la disponibilidad de todos los artefactos para la liberación de una determinada versión.

3	Realizar actividades de rastreo de defectos y de cambios.
4	Escribir el PGC.
5	Informar el progreso de las solicitudes de cambio.

IV- Esquema del proceso

Figura 2.1 Esquema del Proceso de Gestión de Configuración del Software.

V- Relación de actividades del proceso				
Código	**Actividad**	**Responsable**	**Participantes**	**Resultados a obtener**
P0101	Plan de Gestión Configuración.	Gestor de Configuración.	Gestor de Configuración. Gestor de Control de Cambios	PGC.
P0102	Creación del entorno de la GC.	Gestor de Configuración	Gestor de Configuración.	Repositorio del Proyecto.
P0103	Reporte de estado de la configuración.	Gestor de Configuración.	Gestor de Configuración.	1.Reportes 2. Informes
P0104	Administrar líneas base y liberaciones.	Gestor de Configuración.	Gestor de Configuración	Unidad de despliegue.

2.2 Actividad: Plan de GCS.

El Plan de Gestión de Configuración (PGC) es un documento que se debe producir al inicio del proyecto. Define las políticas, estándares y procedimientos que se van a utilizar para gestionar la configuración en el transcurso del proyecto. Incluye el propósito, alcance, definiciones, acrónimos, abreviaturas y referencias del plan.

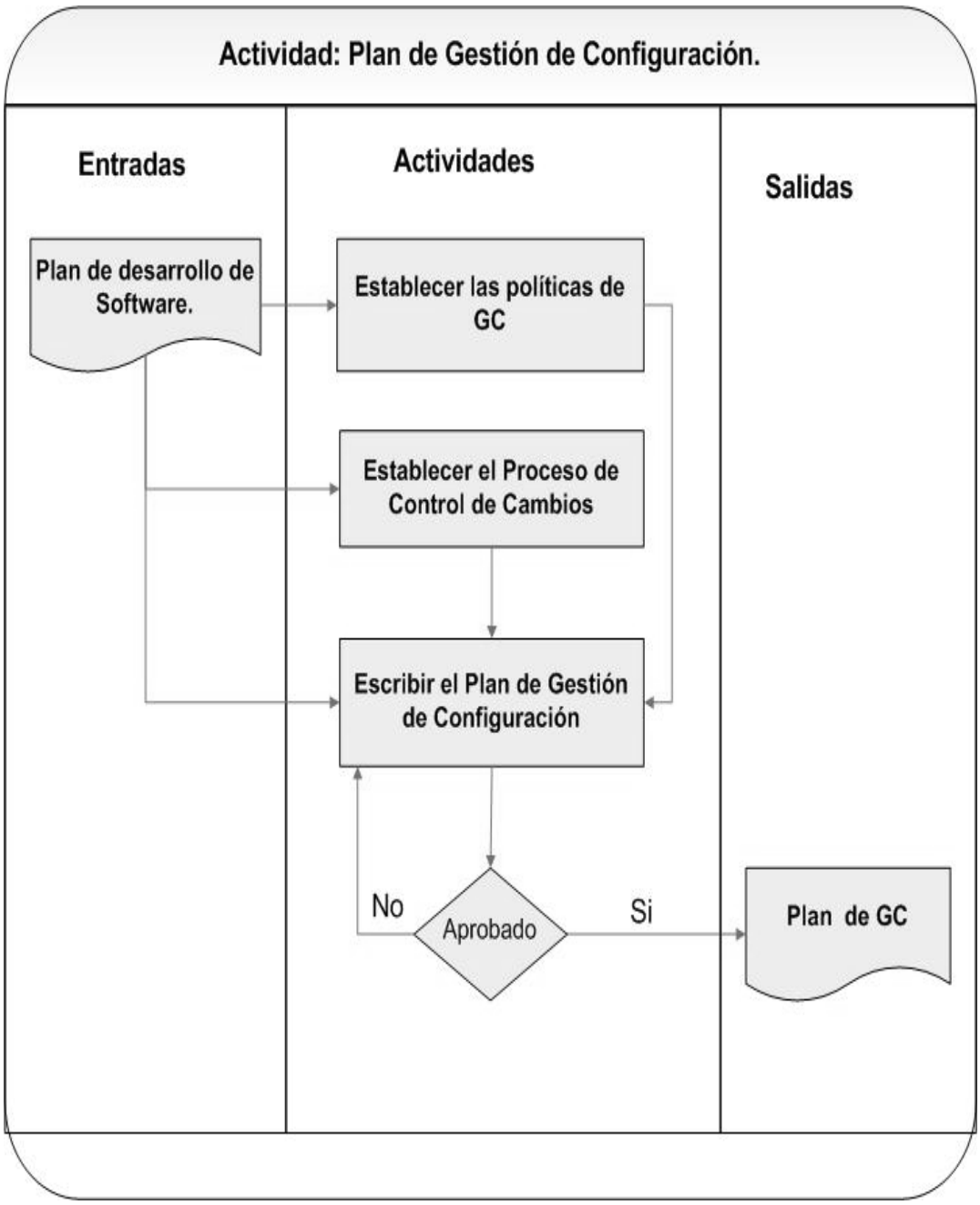


Figura 2.2 Esquema del procedimiento.

Relación de las actividades.

Código	Actividad	Responsable	Participantes	Resultados a obtener
010101	Establecer las políticas de GC.	Gestor de Configuración.	Gestor de Configuración.	1. Identificación de la configuración. 2. Líneas bases establecidas.
010102	Establecer el Proceso de Control de Cambios.	Gestor de Control de Cambios.	Gestor de Control de Cambios.	1. Procedimiento para la Solicitud de Cambio (SC). 2. Formulario de Solicitud de Cambio. 2. Comité de Control de Cambios (CCC).
010103	Escribir el Plan de GC.	Gestor de Configuración.	Gestor de Configuración.	Documento: Plan de GC.

Descripción del procedimiento.

010101: Establecer las políticas de GC

La realización de esta actividad incluye los siguientes pasos:

1. El Gestor de Configuración tiene que identificar cada uno de los Elementos de Configuración (EC) haciéndolos únicos de manera accesible. La identificación de los EC servirá para almacenar estos elementos en el repositorio. La estructura para identificar los EC es:

"Proy.-Tipo-Código-versión" ejemplo **(SIGIA-BD-01-1.0)** donde:

Proy: significa el nombre del proyecto.

Tipo: se refiere al tipo de ECS, por ejemplo BD-Base de datos, Doc-Documentos, Z-Ficheros compactados, Cod- Código fuentes, Case-Ficheros generados por case, E- Ejecutables.

Código: El código del elemento de configuración.

Versión: versión en la que se encuentra el ECS.

2. El Gestor de Configuración establece las líneas base con el objetivo de no impedir los cambios justificados. Las líneas bases serán establecidas al final de cada fase ya que marcan un hito o son creadas cuando se llegan a cumplir los objetivos planificados en algún momento específico dentro del proyecto, que una vez creadas las mismas, los cambios sobre los ECS que la conforman se realizan bajo un proceso formal de cambio.

010102: Definir los requerimientos para el reporte de estado de la configuración.

1. El Gestor de Configuración junto al Gestor del proyecto definen que cambios serán reportados, a quien y con qué frecuencia.
2. El Gestor de Configuración realizará los reportes de estado en el momento en que sean requeridos , pero generalmente se realizarán al final de cada iteración teniendo en cuenta :
   - Cantidad de cambios solicitados.
   - Cantidad de cambios asignados.
   - Cantidad de cambios resueltos.
   - Prioridad de los cambios.

010103: Establecer el Proceso de Control de Cambios.

Esta actividad comprende los siguientes pasos:

1. El Gestor de Control de Cambios define un procedimiento para las SC para asegurar que los cambios producidos sean evaluados e implementados de manera controlada (Anexo 1). El procedimiento

incluye los siguientes pasos:

1. Un cliente o miembro del proyecto decide realizar una SC (Anexo 2) la cual puede estar motivada por la detección de una falla o error en el Sistema o por variaciones en los requisitos.
2. Registro y clasificación de la SC: El CCC recibe la solicitud, asignándole un código de clasificación para su rápida ubicación y rastreo. Tras su registro se deben asignar a las SC una prioridad dependiendo de la urgencia y el impacto de la misma. La prioridad es el dato relevante para establecer el calendario de cambios a realizar. Se incluyen los siguientes niveles de prioridad:

- **Baja:** puede ser conveniente realizar este cambio junto a otros cuando, por ejemplo, se decidan actualizar ciertos paquetes de software o se compre nuevo hardware, etc.
- **Normal:** Es conveniente realizar el cambio pero siempre que ello no entorpezca algún otro cambio de más alta prioridad.
- **Alta:** un cambio que debe realizarse sin demora pues está asociado a errores conocidos que deterioran apreciablemente la calidad del servicio. El **CCC** debe evaluar este cambio en su próxima reunión y adoptar las medidas pertinentes que permitan una pronta solución.
- **Urgente:** es necesario resolver un problema que esta provocando una interrupción o deterioro grave del sistema.

3. Evaluación del cambio: El CCC evalúa el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y sobre otros ECS. Para la evaluación del cambio se pueden realizar las siguientes preguntas:
    - ¿Cuáles son los beneficios esperados del cambio propuesto?
    - ¿Justifican esos beneficios los costes asociados al

proceso de cambio?

- ¿Cuáles son los riesgos asociados?
- ¿Disponemos de los recursos necesarios para llevar a cabo el cambio con garantías de éxito?
- ¿Puede demorarse el cambio?

4. Si el cambio no es aprobado el responsable del CCC se lo informa a la persona que solicito el cambio.
5. Si el cambio es aprobado se inmovilizan los ECS afectados, es decir se "dan baja" para evitar una superposición de modificaciones.
6. Generar OCI: el responsable del CCC genera una *Orden de Cambio de Ingeniería* (OCI) (Anexo 3).La OCI describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y de auditoría.
7. La OCI es asignada a uno de los desarrolladores para que implemente el cambio.
8. Revisión del cambio: EL equipo encargado de realizar las pruebas realiza una auditoría a los cambios para ver si se han implementado correctamente. Si el cambio se implementó correctamente el equipo de calidad certifica el cambio.
9. El CCC libera los EC que han sido modificados y compila una nueva línea base del elemento y se genera una nueva versión del producto con los cambios implementados.

2. Una vez definido el procedimiento para las SC, se establece el Comité de Control de Cambios (CCC). El objetivo del CCC es evaluar las SC y decidir si se aprueban o se rechazan los cambios. Asegura que los cambios propuestos son analizados, revisados y documentados a fin de poder ser rastreados y auditados.

El CCC estará integrado por los siguientes miembros del proyecto:

- Jefe del proyecto (Presidente del CCC).
- Arquitecto.
- Jefe de GCS.
- Planificador.

3. El Gestor de Control de Cambios tiene que notificar el cambio realizado a todos los miembros del proyecto.

010103: Escribir el Plan de Gestión de Configuración.(PGC)

1. El Gestor de Configuración escribe el PGC para describir todas las actividades de gestión de configuración que serán realizadas durante el ciclo de vida del proyecto. Describe el cronograma de actividades, las responsabilidades asignadas y los recursos requeridos.

2. Una vez que el Gestor de Configuración haya escrito el PGC, todos los miembros del proyecto deben revisar el PGC para que el Gestor de Configuración esté seguro de que el plan es entendido y va a ser adoptado por el equipo de desarrollo. Seguidamente el Gestor de Configuración le entrega el PGC al Jefe de proyecto para que el plan sea aprobado.

2.1 Si Jefe del proyecto no aprueba el plan el Gestor de Configuración tiene que volver a escribir el plan.

3. Finalmente el Gestor de Configuración tiene que dar mantenimiento al PGC para garantizar que el PGC esta actualizado y aplicable. El PGC será revisado al comienzo de cada fase.

2.3 Actividad: Crear el entorno de GCS.

El propósito de esta actividad es establecer un ambiente donde el producto pueda ser desarrollado, construido, y este disponible para los interesados.

Figura 2.3 Esquema del procedimiento.

Descripción del procedimiento.
011201: Crear el entorno de Gestión de Configuración. 1. El Gestor de Configuración establece el ambiente de hardware para la GC. El Gestor de Configuración junto al Jefe de proyecto tiene que asignar los recursos requeridos para instalar y configurar las herramientas de GC. En orden de prioridad, la computadora donde estará el repositorio del proyecto (PC servidora) debe cumplir los siguientes requisitos: • Requerimiento de memoria ▪ PC Servidora: 256 MB mínimo. ▪ PC Cliente: 128 MB mínimo. • Ancho de banda: 100 Mbps • Espacio de disco en el repositorio: 10 GB. 2. El Gestor de Configuración define las bibliotecas que serán utilizadas en el proyecto: • Biblioteca de trabajo. • Biblioteca de soporte al proyecto. • Biblioteca maestra • Biblioteca de Backup.

3. El Gestor de Configuración estructura la biblioteca de trabajo del proyecto, la organiza lógicamente para asegurarse de que los artefactos resultantes de cada flujo de trabajo o de cada fase están ubicados en el lugar correcto. La estructura de esta biblioteca teniendo en cuenta los subsistemas y los elementos de cada subsistema será la siguiente:

- Requerimientos del Sistema
    - Modelo
        - Modelo de Casos de Uso.
    - Documentos
        - Visión
        - Glosario
        - Lista de requisitos.
        - Especificaciones de Casos de Uso.
- Análisis y Diseño.
    - Modelo de análisis.
        - Realización de Casos de Uso.
    - Modelo de diseño.
        - Subsistema de diseño.
        - Interface.
    - Documentos
        - Documento de arquitectura.
- Implementación.
    - Código.
    - componentes
- Prueba
    - Plan de prueba
    - Casos de prueba
    - Datos de prueba
    - Resultados de prueba
- Despliegue

        - Plan de despliegue
        - Nota de liberación
        - Manuales de usuario.
- Gestión de Configuración y Cambios.
    - Registros.
    - Informes.
- Gestión de Proyecto.
    - Planes
        - Plan de desarrollo de software
        - Plan de iteración
        - Lista de riesgos
        - Plan de Gestión de Configuración.
        - Plan de Calidad.
- Herramientas
        - Rational Unified Process (Rational Rose)
        - Eclipse
        - SVN 1.1.1
        - Edwin.

4. El Gestor de Configuración define los permisos en las bibliotecas del proyecto para asegurar la integridad y el control de cada uno de los elementos de configuración del software. Los permisos se pueden establecer de manera general restringiendo las acciones en las bibliotecas o de manera más específica restringiendo las acciones de los miembros del equipo en cada uno de los EC dentro de las bibliotecas.

Tomando como partida los grupos de trabajo definidos en el proyecto, el Gestor de Configuración dará permisos de lectura (r) y/o escritura (w) a los miembros del proyecto según el módulo del proyecto en el que estén trabajando.

Los desarrolladores pueden realizar operaciones de “check in” y “check out”.

La operación de “check out” crea copia de los elementos de las bibliotecas en

el espacio de trabajo de los desarrolladores. Cuando se modifica el elemento y se desea subir para la biblioteca se utiliza “check in” y se crea una nueva versión del elemento. La Figura 2.4 representa el proceso de check out/ in.

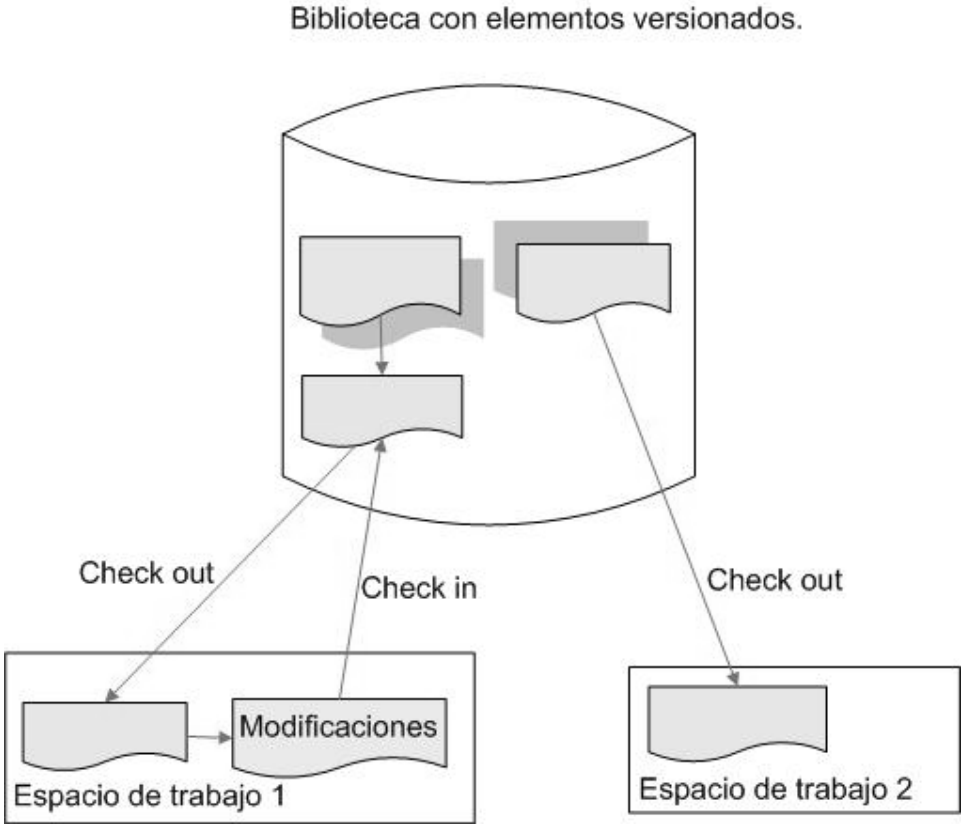


Figura 2.4 Espacios de trabajo y check out/in.

2.4 Actividad: Reporte de estado de la Configuración.

Esta actividad tiene como propósito mantener actualizados de los cambios y su dinámica a gestores y desarrolladores.

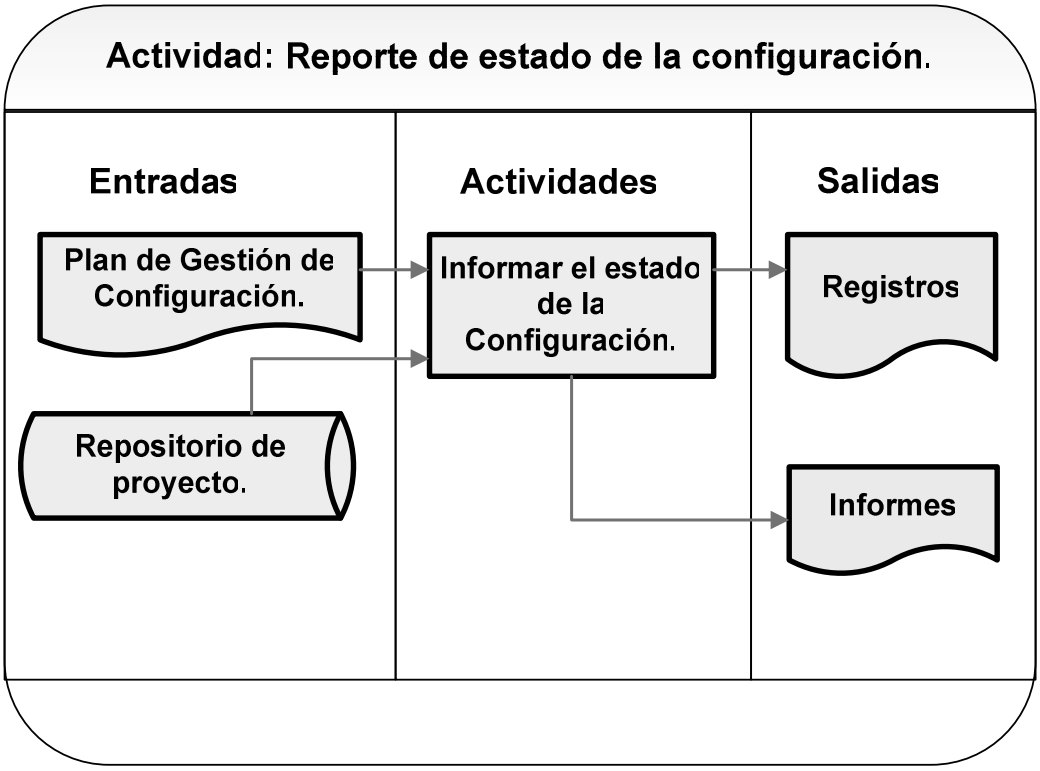


Figura 2.5 Esquema del procedimiento.

Descripción del procedimiento.
010301: Reporte de estado de la configuración. 1. Cuando se origina un cambio en el proyecto el CCC tiene que registrar este cambio para gestionarlo eficientemente. Los registros que serán usados en el proyecto son: ▪ Registros de Solicitudes de Cambio. ▪ Registros de Orden de Cambio. ▪ Registros de Incidencias.

- Actas de reuniones del CCC.

La información referente a cada registro estará almacenada en el repositorio del proyecto.

2. Una vez que los cambios son registrados el CCC tiene que informar a los miembros del proyecto el estado de los cambios y el estado de la configuración. Para esto es necesario la generación de dos informes: los informes de cambio y de estado de la configuración.

- Informe de Cambios: contiene un resumen del estado en que se encuentran las Solicitudes de Cambio registradas durante un determinado período (Anexo 4).
- Informe de estado de la configuración: Provee un resumen del estado actual de las actividades de GCS (Anexo 5).

2.5 Actividad: Administrar Líneas base y liberaciones.

El propósito de esta actividad es asegurar que el conjunto de artefactos consistentes puedan ser identificados como parte de una línea base para varios propósitos, tales como la identificación de liberaciones candidatas, versiones de productos, artefactos con un alto nivel de madurez o en su totalidad.

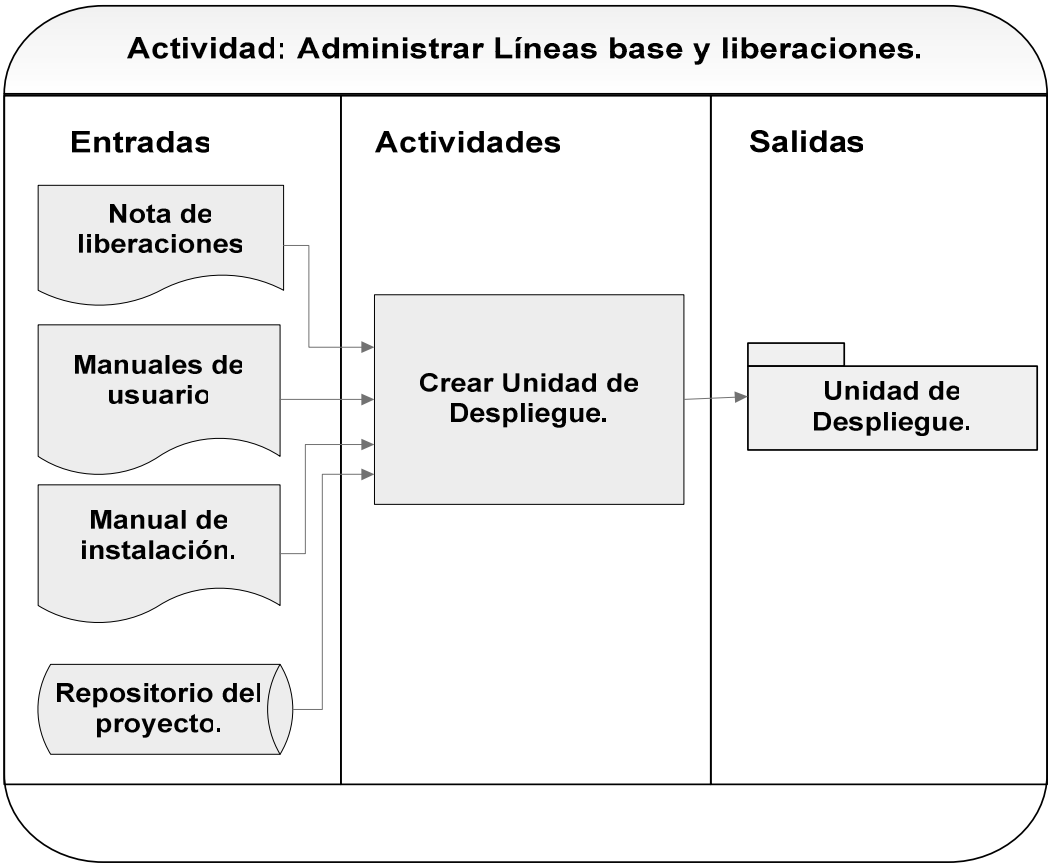


Figura 2.6 Esquema del procedimiento.

Descripción del procedimiento.
010401: Crear Unidad de Despliegue. 1. El Gestor de Configuración toma los componentes ejecutables, los manuales de usuario, los manuales de instalación y los artefactos necesarios almacenados en el repositorio del proyecto y crea la unidad de despliegue lo suficientemente completa como para ser descargada, instalada y ejecutada. 2. El Gestor de Configuración crea una copia de los elementos entregables que se encuentran en el repositorio del proyecto o en la Biblioteca Maestra. 3. El Gestor de Configuración entrega al Jefe del Proyecto la copia de los elementos entregables para su posterior entrega a los clientes.

2.6 Actividad: Auditoria de 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 de 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).

En (Pressman, 2002) se definen dos formas de asegurarse que los cambios se han implementado correctamente:

- Revisiones técnicas formales (RTF).
- Auditorías de configuración del software.

Las RFT se centran en la corrección técnica del elemento de configuración que ha sido modificado. Se lleva a cabo una RTF para cualquier cambio que no sea trivial.

Una auditoría de configuración del software complementa la RTF al comprobar características que generalmente no tiene en cuenta la revisión. En una auditoria se pueden plantear entre otras, las siguientes preguntas:

- ¿Se ha hecho el cambio especificado por la OCI? ¿Se han incorporado modificaciones adicionales?
- ¿Se ha llevado a cabo una RTF para evaluar una corrección técnica?
- ¿Se han resaltados los cambios en el ECS? ¿Se han especificado la fecha de cambio y el autor? ¿Reflejan los cambios los atributos del objeto de configuración?
- ¿Se han seguido procedimientos de GCS para señalar el cambio, registrarlo y divulgarlo?
- ¿Se han actualizado adecuadamente todos los ECS relacionados?

Después de haber revisado la bibliografía de Pressman [Pressman, 2002] se ha considerado que la auditoría de la GCS se llevará a cabo en el proyecto independientemente por el grupo de garantía de calidad.
Durante la auditoria se utilizan diferentes formularios para solicitar una mejora y para informar de un problema o deficiencia detectada, una prueba o el uso del sistema (y que posiblemente requerirá un cambio).

El formulario Informes de Incidencias o Informes de Problemas se usan para informar de problemas. Recogen información adicional sobre el incidente que ha desvelado la existencia del problema (Anexo 6).

Este Informe de Incidencia es analizado por los desarrolladores, y estos pueden recomendar alguna de las siguientes acciones (Antonio, 2001):

- No requiere acción: Cuando lo que se describe en el informe de incidencia no es realmente una deficiencia. Esta situación suele ser debida a malentendidos acerca de la forma de funcionamiento del sistema. También se puede dar esta situación cuando ya se ha informado previamente de una incidencia similar, y se están tomando las acciones correctivas necesarias.
- Solicitud de Cambio: La implementación se corresponde con el diseño del sistema, pero una mejora en el diseño del sistema solucionaría el problema. Se genera entonces una Solicitud de Cambio, que se tratará por los cauces normales.
- Notificación de Cambio: Cuando la deficiencia que se describe en el informe de incidencia se debe a una mala implementación que debe ser corregida. Se informa entonces al CCC y se pasa a corregir la deficiencia.

2.7 Establecimiento de métricas en el proceso.

Es importante y razonable medir para luego utilizar las métricas resultantes para ayudar a mejorar los procesos de software local y la calidad del producto final.

En este sentido se ha utilizado un conjunto de medidas fácilmente recolectables basadas en objetivos dentro del proceso de Gestión de Configuración del Software que requieran mejoras.(Pressman,2002).

2.7.1 Reducir el tiempo para evaluar e implementar las solicitudes de cambio.

Medidas:

- Tiempo (horas o días) que transcurren desde el momento que es realizada una petición de cambio hasta que se complete su evaluación,

  $t_{cola.}$

- Esfuerzo (horas-persona) para desarrollar la evaluación, $W_{eval.}$
- Tiempo (horas o días) transcurrido desde la terminación de la evaluación a la asignación de una orden de cambio al personal, $t_{eval.}$
- Esfuerzo (horas-persona) requeridas para realizar el cambio. $W_{cambio.}$
- Tiempo requerido (horas o días) para realizar el cambio, t_{cambio}.

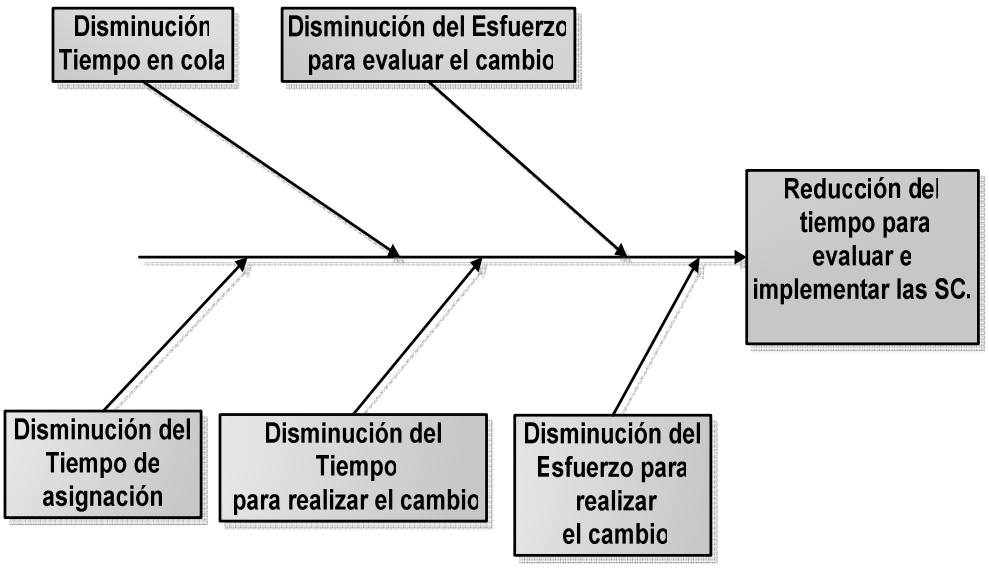


Figura 2.7 Diagrama de espina.

Una vez que estas medidas han sido recogidas para un cierto número de solicitudes de cambio, es posible calcular el tiempo total transcurrido desde la petición de cambio hasta la implementación de dicho cambio y el porcentaje de tiempo consumido por las solicitudes en colas, la asignación de cambios y la implementación del cambio propiamente dicho.

De manera similar, el porcentaje de esfuerzo requerido para la evaluación y la implementación pueden también ser determinado.

Los porcentajes proporcionan un análisis interno para buscar el lugar donde los procesos de petición de cambio se ralentizan y pueden conducir a unos pasos de mejoras de proceso para reducir t_{cola}, t_{eval}, W_{eval}, W_{cambio}.

2.7.2 Lograr un mejor soporte de nuestro producto.

Medidas:

- ¿Contienen las preguntas de solicitud de cambio la información que se requiere para evaluar adecuadamente el cambio y de esa forma realizarlo en el menor tiempo?
- ¿Se sigue convenientemente el Proceso de Control de Cambios?
- ¿Se llevan a cabo los cambios de alta prioridad de manera oportuna y sincronizada?

2.7.3 Lograr estabilidad en el producto (basado en los cambios que ocurren con cada versión del producto).

El estándar (IEEE, 1990) sugiere un índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto software basado en lo cambios que ocurren con cada versión del producto. Se determina la siguiente información:

Mt= número de módulos en la versión actual.

Fc= número de módulos en la versión actual se han cambiado.

Fa= número de módulos en la versión actual que se han añadido.

Fd= número de módulos de la versión anterior que se han borrado en la versión actual.

IMS= [Mt-(Fa+ Fc+ Fd)]/Mt

A medida que el IMS se aproxima a 1.0 el producto se empieza a estabilizar. El IMS puede emplearse también como métrica para la planificación de las actividades de mantenimiento del software.

2.7.4 Costo relativo

El costo relativo es el costo de los cambios relativos sobre el costo del proyecto.
El costo relativo (CR), se calcula usando la siguiente ecuación:

CR= (*Costo de los cambios / Costo total del proyecto) * 100*

Donde el costo de los cambios y el costo total del proyecto se determinan usando mediciones definidas en el *seguimiento del proyecto* y *en el plan de medición.* El costo de las mediciones se calcula usando los datos registrados en el registro de cambios (ej. Tamaño, horas).

A medida que transcurre el tiempo, el valor del costo de los cambios, debería ir disminuyendo, dado que se supone que el producto software comienza a nivelarse y hacen falta cada vez menos modificaciones. (Pascuttinni, 2002).

El costo relativo nos provee un valor, indicando el esfuerzo realizado para llevar adelante los cambios.

2.7.5 Tiempo promedio de cambio.

El Tiempo promedio de cambio (TPC), es el tiempo promedio que transcurre para la realización de un cambio, tomado desde que se realizó el pedido hasta la finalización del mismo.
El TPC se calcula usando la siguiente ecuación:
TPC = SUM (TCi - TRi) / A
Donde A es la cantidad total de cambios, TRi es el hora en que se requirió el cambio i, y TCi es la hora en que se finalizó con el cambio i.

Es necesario destacar que los datos utilizados de muestreo, posean la misma complejidad. Este es un detalle muy importante, ya que sino se tiene en cuenta en el cálculo del TPC nos daría un resultado incorrecto. Por lo tanto, en un caso real, primero habría que agrupar los cambios por su nivel de complejidad y luego realizar el cálculo del tiempo promedio de cambio.

Un valor alto en TPC es indicador de un pobre mantenimiento, en cambio un valor bajo es indicador de un buen mantenimiento.

2.8 De lo abstracto a lo concreto.

2.8.1 Las bibliotecas de Software.

En (Antonio, 2001), se establece que se deben definir, además de las bibliotecas, los procedimientos para:

• El establecimiento de una Biblioteca.
• La introducción de elementos en la biblioteca.
• El acceso a la biblioteca.

El repositorio del proyecto estará estructurado en subdirectorios llamados **trunk** para la "línea principal de desarrollo ", **branches** (variantes) para alojar las diferentes ramas con nombre de la línea principal de desarrollo y **tags**, que es el directorio donde se almacenarán los ECS que se encuentran estables y no necesitan ser modificados constantemente. Los permisos de escritura en este directorio son restringidos, solo el Jefe de proyecto y el Gestor de Configuración, aunque todos los miembros del proyecto pueden leer.

El proceso de movimiento de una versión de un ECS de una biblioteca a otra, depende completamente del nivel de estabilidad, de madurez o de acabado que presente la versión del ECS.

En la biblioteca de trabajo, como se describe en el Capítulo 1, se almacenarán los ECS que se están desarrollando por los trabajadores del proyecto. En esta biblioteca los ECS no están aún estado estable. Una vez que se termine de implementar la versión del ECS, debe ser probada su implementación, para que así se esté seguro de que los cambios se realizaron de manera correcta. Esto conlleva a que se mueva la versión del ECS a la biblioteca de integración, donde será probada por un trabajador del proyecto. Si la prueba resultó exitosa, y se concluyó la implementación de la versión del ECS, este puede moverse a la biblioteca de Soporte del proyecto, donde empezará a estar bajo un proceso de control de cambios interno e informal. Esto quiere decir que la versión del ECS presentará un mayor nivel de estabilidad. Una vez que se esté seguro que la versión de un ECS, tenga calidad para ser entregado a un cliente o para que se pueda distribuir en el mercado, dicho versión de ECS se mueve para la biblioteca maestra, donde se encuentra entonces bajo un proceso de control de cambio formal y estricto.

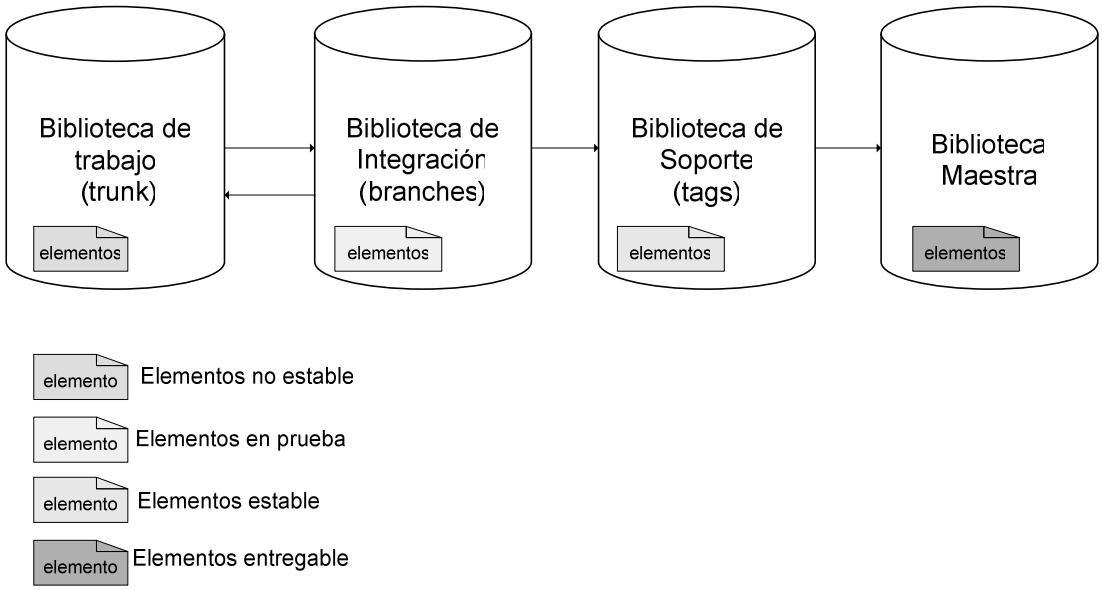


Figura 2.8 Almacenamiento de ECS en las bibliotecas del proyecto.

En el trabajo diario con el Subversion los desarrolladores generan múltiples versiones de ECS, no todas con igual importancia y en ocasiones difieren muy poco unas de otras. Usualmente es necesario distinguir versiones importantes de otras sin relevancia, y a los efectos del Subversion, las versiones n, n+1 y n+2 de un ECS, simplemente son variaciones de sus respectivas versiones anteriores, y no puede distinguir si una de ellas es una versión que corresponde con un hito del proyecto, es decir, si se ha convertido en línea base. Es necesario el factor humano como parte del sistema; es por ello que el administrador de la configuración debe mantener los elementos que forman parte de las líneas base, separados físicamente del resto de los ECS que se encuentran en la Biblioteca de trabajo. Por esta razón en la Biblioteca de Soporte se almacenarán aquellos ECS que se han convertido en línea base, con el objetivo de garantizar su integridad, y prevenir de malas manipulaciones que puedan introducir inconsistencia. En esta biblioteca solo tienen permisos de escritura el administrador de la configuración y el jefe de proyecto, los demás usuarios tienen solamente acceso de lectura. Esta propuesta es una garantía para la seguridad del proyecto ya que se tendría almacenado de forma física las distintas líneas bases y por consiguiente, las versiones estables de los ECS que conforman el proyecto durante el ciclo de vida.

2.8.2 Controlando los Cambios.

Cuando surge un cambio es muy probable que el ECS que se desea cambiar esté vinculado a otros ECS. El administrador de la configuración puede confirmar en efecto cuáles ECS se encuentran relacionados directamente con el cambio, ya que tiene una visión más exacta y abarcadora de la configuración, gracias a las relaciones establecidas entre los ECS (Figura 2.9).

El grafo de relaciones entre ECS es una estructura de datos que almacena ECS y relaciones entre ellos (flechas curvadas y rectas) en un formato de grafo dirigido y etiquetado. Una flecha curvada representa una relación de composición

<<contiene >> y una flecha recta representa una relación de dependencia <<utiliza>>. Las relaciones permiten determinar al CCC que otros ECS pueden verse afectados.

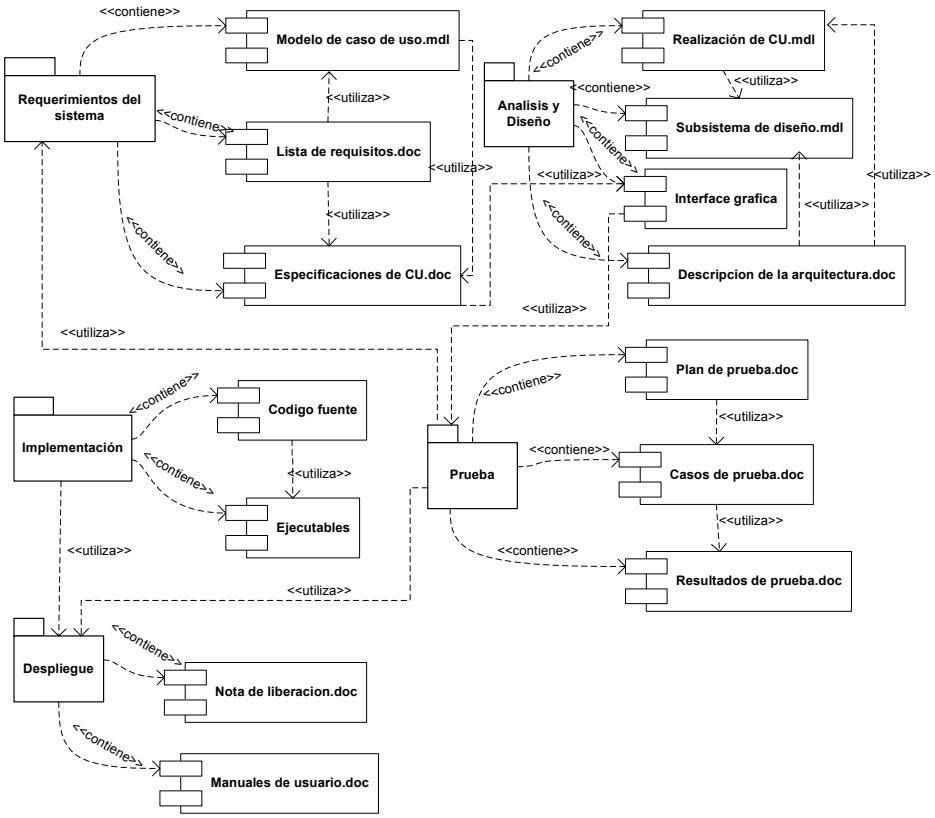


Figura 2.9 Relaciones entre los ECS de los flujos de trabajo.

Se propone publicar en el Trac la figura anterior lo cual permite que todos los miembros del proyecto tengan visibilidad en cuanto al impacto que podría tener un cambio en un determinado ECS sobre otros elementos.

Una vez que se ha valorado el posible impacto que tiene modificar un ECS sobre otros ECS y se ha decidido aprobar el cambio, el planificador del proyecto procede a planificar su ejecución, disponiendo de los recursos humanos necesarios.

El Trac permite la asignación de tareas por lo que el planificador puede asignar tareas a los desarrolladores y ver el cumplimiento de las mismas. En la descripción de la tarea el planificador pudiera adicionar los ECS que se ven afectados con el cambio; de este modo el responsable del cambio también estaría informado de los elementos que se encontrarán relacionados con el cambio a realizar.

Como el Trac esta integrado con Subversion soporta la trazabilidad en los ECS. (Anexo 7) y además permite visualizar en una línea de tiempo los cambios realizados en los hitos, en el repositorio y en la wiki. (Anexo 8).

2.8.3 Control de versiones con Subversion.

Subversion es un sistema centralizado para compartir información. La parte principal de Subversion es el repositorio, el cual es un almacén central de datos, maneja ficheros y directorios a través del tiempo, lo que permite recuperar versiones antiguas de los datos, examinar el historial de cambios, explorar el repositorio, modificar versiones, control de acceso, adicionar nuevos ficheros entre otras. El repositorio guarda información en forma de *árbol de archivos,* una típica jerarquía de archivos y directorios. Cualquier número de *clientes* puede conectarse al repositorio y luego leer y/o escribir en esos archivos. La misión principal de un sistema de control de versiones es permitir la edición colaborativa y el uso compartido de los datos.

Sin embargo existe un problema muy común a la hora de compartir los datos:
los usuarios sobrescriben accidentalmente los cambios de los demás en el repositorio. Para resolver este problema Subversion utiliza el modelo de versionado ***copiar-modificar- mezclar*** como alternativa a la solución *bloquear – modificar-desbloquear.* En el primer modelo, el cliente de cada usuario se conecta al repositorio del proyecto y crea una copia de trabajo personal, una réplica local de los archivos y directorios del repositorio. Los usuarios pueden entonces trabajar en paralelo, modificando sus copias privadas. Finalmente, todas las

copias privadas se combinan (o mezclan) en una nueva versión final. El Subversion a menudo ayuda con la mezcla, pero en última instancia es un ser humano el responsable de hacer que esto suceda correctamente. (Collins-Sussman, Fitzpatrick, & Pilato, 2004)

2.8.3.1 Manejando la información de una copia de trabajo con Subversion.

Tomando como base que Subversion registra para cada fichero de una copia de trabajo la revisión en la que está basado el fichero de la copia de trabajo (esto se llama la *revisión de trabajo* del fichero), y una marca de tiempo con la fecha de la última actualización del fichero desde el repositorio, se proponen un conjunto de aspectos a tener en cuenta a la hora de trabajar en las copias de trabajo. A continuación se muestra el estado en el que se pueden encontrar los ficheros de las copias de trabajo y los comandos de SVN a ejecutar. (Collins-Sussman, Fitzpatrick, & Pilato, 2004)

- Sin cambios y actualizado: el fichero no ha sido modificado en la copia de trabajo ni se ha enviado ningún cambio sobre ese fichero al repositorio desde su revisión de trabajo. Un **svn commit** de ese fichero no hará nada, y un **svn update** del fichero tampoco hará nada.
- Modificado localmente y actualizado: el fichero ha sido modificado en la copia de trabajo pero no se ha enviado ningún cambio sobre ese fichero al repositorio desde su revisión base. Hay cambios locales que no han sido enviados al repositorio, por lo que un **svn commit** del fichero publicará con éxito sus cambios, y un **svn update** del fichero no hará nada.
- Sin cambios y desactualizado: el fichero no ha sido modificado en la copia de trabajo, pero sí en el repositorio. El fichero debería ser actualizado para sincronizarlo con la revisión pública. Un **svn commit** del fichero no hará nada, y un **svn update** del fichero introducirá los últimos cambios en su copia de trabajo.

- Modificado localmente y desactualizado: el fichero ha sido modificado tanto en la copia de trabajo como en el repositorio. Un **svn commit** del fichero fallará dando un error de "desactualizado". El fichero debe ser actualizado primero; un **svn update** intentará mezclar los cambios públicos con los cambios locales. Si Subversion no puede combinar los cambios de manera convincente automáticamente, dejará que sea el usuario el que resuelva el conflicto.

2.8.3.2 Las ramificaciones en el control de versiones.

Crear ramas, o línea principal de desarrollo, es una parte fundamental del control de versiones y como se utilizará Subversion para gestionar los datos, entonces ésta es una característica de la cual se acabará dependiendo.

Subversion tiene comandos para ayudar a mantener ramas paralelas de ficheros y directorios. Permite crear ramas copiando los datos, y recordando que las copias están relacionadas unas a otras. También ayuda a duplicar cambios de una rama a otra. Finalmente, se puede reflejar el contenido de diferentes ramas en partes de la copia de trabajo local, para poder "mezclar y probar" diferentes líneas de desarrollo en el trabajo diario.

Como buena práctica se recomienda que los desarrolladores tengan copias locales del tronco. Cuando alguien necesita hacer un cambio a largo plazo que puede molestar en el tronco, un procedimiento estándar es crear una rama privada y realizar ahí los cambios hasta que todo el trabajo esté completo. De este modo un desarrollador no interfiere en el trabajo de otros. De lo contrario puede que cuando un desarrollador haya terminado con su rama, sea casi imposible fusionar los cambios con el tronco sin un alto número de conflictos.

Para fusionar los cambios de la rama con el tronco, primero se debe tener una copia local del tronco, porque el comando que utiliza Subversion para fusionar (**svn merge**) compara dos árboles y aplica las diferencias en una copia local.

Luego se compara el estado inicial de la rama con su estado final. Usando **--stop-on-copy** con **svn log** en la rama, es posible ver el número de la revisión en la que fue creada y el estado final se obtiene con la revisión HEAD. Esto significa que se quiere comparar las revisiones de creación y HEAD del directorio de la rama, y aplicar esas diferencias a una copia local del tronco.

Es importante que cuando se "suban" los cambios al repositorio el mensaje de cambios mencione de forma explícita el rango de cambios que fue fusionado en el tronco, porque si después se continúa trabajando en la rama y se desea fusionar solamente los nuevos cambios y no los cambios que ya fueron fusionados, basta con ejecutar el comando **svn log** y buscar un mensaje de cambios sobre la última vez que se realizó una fusión desde la rama, de este modo es posible fusionar los nuevos cambios realizados en la rama desde la última vez que fueron fusionados con el tronco.

2.9 Valoraciones preliminares.

En términos globales, se espera que la implantación de la Gestión de Configuración en el proyecto SIGIA permita:

- El mantenimiento de la integridad de los elementos, en una atmósfera de cambio continuo.
- La evaluación y ejecución de los cambios, en un ambiente controlado.
- La reducción del tiempo de desarrollo, ayudando a mantener el "orden" en el proyecto.
- Que el estado de la configuración proporcione evidencia objetiva y concreta de la creación y evolución del producto.

Sin embargo, los resultados mencionados anteriormente no serán palpables en el marco de esta investigación, debido a que el proyecto en cuestión se interrumpió y se considera que no es posible implantar la propuesta de Gestión de Configuración a dicho proyecto.

De igual modo para demostrar la eficacia de la propuesta, se decidió aplicar el método de validación de expertos Delphy. Este método plantea que la característica principal que deben tener los expertos es la competencia, que consiste en su nivel de calificación en una determinada esfera del conocimiento.

Después de determinar la competencia de los siete posibles expertos seleccionados, se consideró que ninguno cumplía con los requisitos establecidos y por consiguiente no era factible tomar en cuenta sus criterios, porque sus valoraciones podrían resultar erróneas.

A pesar de los inconvenientes antes mencionados, se entrevistó a la MSc en Sistemas Digitales, Eugenia Muñiz Lodos, Investigadora Auxiliar del ICID y con más de 35 años de experiencia en la producción de software. La misma considera que la propuesta de Gestión de Configuración debe ser aplicada a un proyecto desde la fase de inicio, para de esta forma estar a tono con los planes de proyecto y no en etapas avanzadas, debido a que es muy difícil de comprender y además lejos de avanzar en el proyecto, provoca atrasos en las actividades de desarrollo.

Considera también que no es necesario aplicar la GCS en un proyecto para obtener productos de calidad, pero si reconoce la necesidad de esta práctica en la reducción de esfuerzos y tiempo de desarrollo. Resaltó el alto valor que tiene la propuesta de este proceso, teniendo en cuenta que define un conjunto de procedimientos, métricas y propone una potente herramienta para el control de versiones, que sin lugar a dudas mejorará el desarrollo del proyecto.

CONCLUSIONES

Es importante tener un control preciso, predecible y repetido sobre los procesos de producción y los productos de software. La Gestión de Configuración del Software es un proceso que ayuda a mantener el control de los elementos en el proyecto.

Después de la realización de este trabajo se arribaron a las siguientes conclusiones:

- La Gestión de Configuración del Software se encarga de mantener la calidad del software y la capacidad de duplicar el producto en el tiempo.

- Se propusieron procedimientos y herramientas automatizadas que permitirán de forma eficiente implantar el proceso.

- Se seleccionaron cuatro actividades fundamentales del Proceso de GCS para el proyecto SIGIA: Plan de Gestión de Configuración, creación del ambiente de configuración, reporte de estado de configuración y Administración de Líneas base y liberaciones.

- Se definieron para cada actividad las entradas, salidas, resultados y responsables, facilitando su comprensión para su futura implantación.

- Se propuso Subversion como herramienta libre para el control de versiones integrada con Trac para alcanzar un desarrollo altamente escalable.

- Se propusieron métricas para mejorar el proceso, la planificación, seguimiento y control de los cambios.

RECOMENDACIONES

1. Ampliar la búsqueda de expertos que valoren la propuesta.
2. Aplicar la propuesta del Proceso de Gestión de Configuración en nuevos proyectos de la facultad desde la fase de inicio para de esta manera demostrar su real utilidad práctica.

REFERENCIAS BIBLIOGRÁFICAS

- Álvarez, S. (2003). Un modelo de calidad para una empresa de software, Evento Calidad. C.Habana. Cuba.
- Antonio, A. d. (2001). Gestión de Configuración del software. Chile.
- Babich, W. (1986). *Software Configuration Management.* Adison- Wesley.
- Bedini, A. G. (1995). *Calidad de software.* Novática.
- Bersoff, E. (1980). *Software Configuration Management.* Prentice-Hall.
- Brown, N. (1998). *Little Book of Configuration Management.* AIRLIE.
- Buckeley, F. (1999). *Implementing Configuration Management: hardware, software and firmware* (2 ed.).
- Burrows, C. (1999). Configuration Management. *Crosstalk* .
- Butler, T. (2001). Software Configuration Management: A discipline with added value. *Crosstalk , 7*.
- Camou, E. (2002). *Software Development: The Venezuelan competitive project.* Cavecom.
- Collins-Sussman, B., Fitzpatrick, B. W., & Pilato, C. M. (2004). *Control de versiones con Subversion: Revision 2555.*
- Delgado, R. (2006). ConfigCASE 3.0 Herramienta de apoyo a la Gestión de Configuración . Propuesta arquitectónica. C.Habana.
- Febles, A. (2000). Case Corporativo para el proceso de control de cambios. Tesis de maestría. C.Habana.
- Febles, A. (2002). Un sitio de soporte de software, acercando la empresa al cliente. C. Habana.
- Febles, A. (2004). MConfig.PM. Modelo de referencia para la Gestión de Configuración en la pequeña y mediana empresa de software. C.Habana.
- Fernández, H., & Febles, A. (2000). CASE Corporativo para la creación de la línea base de una empresa de Software. *Ingeniería Industrial* .
- Folz, A. G. (2000). Configuration Management Tips, Embedded Systems.

- García, L. (2000). Metodología para la evaluación de la calidad del análisis y diseño orientado a objeto usando ADOOSI. Tesis de doctorado. C. Habana, CEIS, ISPJAE.
- García, L., & Álvarez, S. (1999). Una visión general sobre la certificación de calidad y la evaluación del proceso de desarrollo de software. *Ingeniería Industrial* (4).
- Gervás, P. (2002). Estándares y Gestión de Configuración.
- González, P. I. (2003). *Discurso pronunciado en la inauguración de la Convención Informática.* Cuba.
- Harter, R. (1989). Configuration Management. HP professional. *3* (6).
- Humphrey, W. S. (1995). *A discipline for software engineering.* Addison-Wesley.
- Humphrey, W. S. (2000). *Introduction to the Team Software Process.* Addison-Wesley.
- Ibargüengoitia, G. (2002). Administración de la configuración del software, Primer Taller Internacional de Calidad en desarrollo de software. México.
- IEEE. (1987). Guide to Software Configuration Management. American National Standards Institute.
- IEEE. (1990). Standard for Software Configuration Management Plans. American National Standards Institute.
- IEEE. (1993). Software Engineering Standards. American National Standards Institute.
- International Business Machine. (2005). *Rational ClearCase*. Retrieved from http://www.ibm.com/software/awdtools/clearcase/
- International Business Machine. (2005). Rational ClearCase.
- ISO 10007. (1995). Quality Management − Guidelines for Configuration Management.
- Jacobson, I. (2000). *Software reuse: Architecture, Process for Busisness Success.* Addison Wesley Longman Inc.
- Kan, S. H. (1995). *Metrics and Models in Software Quality Engineering.* Addison Wesley Longman, Inc.

- Lage, C. (2000). Discurso pronunciado en la Inauguración de la Convención Informática. C.Habana.Cuba.
- MIC. (2002). *Comisión de Productos y Servicios-Fuerza de Tareas de la Industria de Software.* C.Habana.
- Moreno, B. (2003). Conferencia magistral, Evento de Calidad. Convención Informática. C.Habana. Cuba.
- MSDN. (2005). Microsoft Visual SourceSafe Roadmap.
- Narayanaswany, K., & Scacchi, W. (1987). *Maintatning Configurations of Evolving Software Systems. Software Engineering* (Vols. SE-13).
- Norm, B., & Evans, M. (1998). *Litle Book of Configuration Manager . Computer & Concepts Associates.*
- Pascuttinni, P. (2002). Control de Configuración, su impacto en la Gestión del proyecto.Tesis de Maestría.
- Paulk, M. (1993). Capability Maturity Model for Software.
- Paulk, M. C. (1999). The Capability Maturity Model, Guidelines for Improving the Software Process.
- Piatini, M. G. (1996). Análisis y Diseño detallado de aplicaciones informáticas de gestión. *Rama* .
- Pressman, R. S. (2002). *Ingeniería de Software, un enfoque práctico* (5 ed., Vol. 1). Mc Graw-Hill.
- Ragland, B. (1995). Measure, Metric or Indicator: What's the Difference? *Crosstalk , 8* (3).
- Rational Software Corporation. (2003). Rational Unified Process Help. *Versión 2003.06.00.65, Copyright 1987-2003* .
- Reo, D. A. (2002, septiembre 4). La gallina, el cerdo y el modelo CMM, América XXI.
- SIME. (1997). Lineamientos estratégicos para la informatización de la sociedad cubana. C.Habana. Cuba.
- SourceForge. (2005).VA Software Corporation.
- Yebra, I. (2001). CMM un enfoque práctico (Presentación Power Point).

Sitios Web visitados.

- http://www.itba.edu.ar/capis/rtis/articulosdeloscuadernosetapaprevia/pascuttini8.pdf 26-01-07
- http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/rodolfo.pdf 1-02-2007
- http://www.amautacorp.com/index.php 1-02-2007
- http://www.inf.utfsm.cl/ visconti/papers/paperscm1999.pdf 1-02-2007
- http://es.wikipedia.org/wiki/Transport_Layer_Security 3-02-2007
- www.codicesoftware.com 3-02- 2007.
- http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_versiones/vision_general_gestion_de_versiones/vision_general_gestion_de_versiones.php 4-02-2007.
- http://www.grupopf.com.mx/general/view.aspx_22 -02-2007
- http://trac.edgewall.org/ 3-04-2007
- http://consol.com.mx/2007/general/proposals/550 3-04-2007
- http://subversion.tigris.org 3-04-2007
- http://www.gestion.uco.es/gestion/aplicaciones/docs/sgc_manual_usuario.pdf 5-03-2007
- http://www.gestiopolis.com/dirgp/adm/cambio.htm 5-03-2007
- http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_versiones/introduccion_objetivos_gestion_de_versiones/introduccion_objetivos_gestion_de_versiones.php
- http://lml.ls.fi.upm.es/ep-4/0304/doc/Versiones.ppt
- http://www.fdi.ucm.es/profesor/anavarro/10._Gestion_de_la_configuracion_software.pdf 5-03-2007
- http://av.rds.yahoo.com/_ylt=A0Je5XSe4exFHK4AYYJrCqMX;_ylu=X3oDMTBvdmM3bGlxBHBndANhdl93ZWJfcmVzdWx0BHNlYwNzcg--/SIG=135sd2i0e/EXP=1173238558/**http%3a//www.itba.edu.ar/capis/epg-tesis-y-tf/rancan-trabajofinaldeespecialidad.pdf 5-03-2007
- http://www.inf.udec.cl/revista/ediciones/edicion9/febles.pdf 1-05-2007

- http://athenea.ort.edu.uy/publicaciones/ingsoft/ortsf/areas/IntroduccionSCM.pdf 1-05-2007
- http://www.emb.cl/gerencia/seminarios/2007/itilc1/all.htm 1-05-2007