Universidad de las Ciencias Informáticas

Facultad 3

Estrategia para la Gestión de Configuración de Software del proyecto Registros y Notarías

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

Autoras: Yadamí Febles Ramos

Eidy Marien Carbó Peña

Tutor: Ing. Maikel Y. Leyva Vázquez

Consultante: DrC. Pedro Y. Piñero Pérez

Junio, 2007

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos a la infraestructura Productiva de la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo su beneficio y como estime conveniente.

Para que así conste firmamos la presente a los ___ días del mes de _____ de 2007.

_____	_____	_____
Eidy Marien Carbó Peña	Yadamí Febles Ramos	Maikel Yelandi Leyva Vázquez
Autor	Autor	Tutor

Dedicatoria

A papito y a la mamita de lulu, que no tuvieron una Doctora pero si una Ingeniera.

A tati ….tú también eres mi vida

A bebé, para que un día llegue hasta aquí.

A moli moli, mi alma gemela.

Marien

A mis padres…….éste es el resultado de todo nuestro esfuerzo junto.

A mis hermanos, Gery , Geandry y Yadi…… para que se sacrifiquen mucho como tata.

A Papito, Cheché, Pity y Tripi……..ya tienen una Ingeniera!!!

Yadamí

Agradecimientos

A mi tutor, Maikel Yelandi Leyva, por estar siempre ahí, para ayudarnos en todos nuestros problemas y por tu amistad, gracias.

A mimi y papi: Gracias a los dos por confiar siempre en mí y por alumbrarme el camino, con sus sabios consejos.

A mis hermanitos Yadita, Geandry y Gery, por ser personitas especiales que alegran cada minuto de mi vida.

A Tripi, por tu amor de siempre y tu apoyo incondicional en todo momento.

A Maye: Flaqui gracias por tu valiosa amistad de casi 15 años, por tu confianza, porque a pesar de la distancia, sigues siendo mi mano derecha.

A Dorita y Rene, por convertirse en mis hermanos, por ser incondicionales, por haberme brindado todo su cariño, porque siempre encontré en ustedes una verdadera familia, los adoro.

A las Chuchas Dayma y Adriana, gracias por su amistad tan bonita, por haberme ayudado en todo momento, por comprenderme y aceptarme, las quiero mucho mucho.

A mi compañera de tesis, gracias por todo.

A Maykel y Karina por habernos ayudado tanto en el desarrollo de la tesis y por convertirse en mis amigos.

A Peter, mi gran amigo, que no se libró de mí, ni estando en Japón. Todos te debemos mucho.

A Darel, Frank y Rene: Al fin logramos utilizar un Patrón de Ramas decentemente.

A Oscar por brindarnos toda la ayuda que necesitamos y por sus excelentes revisiones en las que nos llenó el Word de globitos rojos.

A mis amigas que se convirtieron este año en mis hermanas e hicieron más llevadera la ausencia de moli: Yule, Made y Yainelys, por todo, GRACIAS.

A mi familia por todo el tiempo sin vernos y las pocas horas de cariño.

A mi compañera de tesis, difícil pero llegamos.

A Yasmany: el mejor amigo que pude encontrar en la universidad. Gracias mi vida por tu apoyo estos 5 años y por enseñarme tanto. Te amo.

Resumen

Los procesos registrales y notariales en la República Bolivariana de Venezuela actualmente se realizan de forma independiente y diferente en cada registro, la distribución de los ingresos permite que los registradores y notarios alcancen una retribución excesiva, lo que origina una sustancial diferencia con otros funcionarios de la Administración Pública. Con el objetivo de estandarizar estos procesos se está desarrollando el Proyecto de Modernización de los Registros y Notarías de la República Bolivariana de Venezuela (R&N). Este proyecto ha presentado en el ciclo de desarrollo de su producto una serie de dificultades con las cuales han sido afectadas determinadas disciplinas del proceso, entre ellas la Gestión de Configuración.

En el presente trabajo se propone una estrategia para la Gestión de Configuración de Software (GCS) en el proyecto R&N. En esta propuesta son definidos procedimientos para ser aplicados en R&N y son seleccionadas además herramientas que automatizan las diferentes actividades de la GCS. La estrategia propuesta es inicialmente aplicada en el submódulo Bienes de dicho proyecto.

Palabras Claves: Gestión de Configuración de Software.

“Software configuration management is the unsung hero of software development”

Tom Milligan

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

Introducción

Actualidad del tema

Si bien es cierto que por los años 80, en los albores de la industria del software, existía un mercado poco competitivo y la satisfacción del cliente se reducía a la existencia de homogeneidad en el producto, en la actualidad, el mundo de la informática se torna cada vez más novedoso y competitivo. La gestión de un software eficiente tiene un impacto estratégico y se ha convertido en una de las principales oportunidades de ventaja en el mercado (Dapena, 2005). Se impone, por tanto, la necesidad de obtener productos que integren variadas tecnologías de punta y cuyo tiempo de elaboración sea el mínimo, incrementando de esta forma la productividad de los equipos involucrados en el desarrollo del software.

Es meritorio acentuar que obtener el éxito en la industria del software no es camino fácil. Existen actualmente muchas deficiencias e insatisfacciones por parte de los clientes en este aspecto. En varias ocasiones el software no es entregado en el tiempo establecido, o no satisface completamente las exigencias demandadas por los usuarios.

Al profundizar en las raíces o causas de tales insatisfacciones surgen de manera reiterada la aplicación inadecuada de las disciplinas de Ingeniería de Software, la no utilización de los roles y procesos apropiados para el desarrollo de las tareas de la empresa de software y la no utilización de modelos de calidad en ellas (Estrada, 2003).

Dentro de estas disciplinas de Ingeniería se encuentra la Gestión de Configuración de Software (GCS) como uno de los procesos claves en el desarrollo de un producto informático (Navarro J. A., 2006) y que actualmente su incorrecta aplicación constituye una de las principales causas de fracaso en grandes y millonarios proyectos en el mercado mundial.

Su merecida importancia se debe a que la GCS es la encargada de mantener la integridad de los productos que se obtienen a lo largo del desarrollo de los sistemas de información, garantizando que no se realicen cambios incontrolados y que todos los participantes en el desarrollo del sistema dispongan de la versión adecuada de los productos que manejan.

La GCS está presente en el transcurso de todas las tareas asociadas al desarrollo del sistema y continúa registrando los cambios hasta que este deja de utilizarse. De esta forma facilita el mantenimiento del

sistema, aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un cambio, tanto evolutivo como correctivo.

Llevar a cabo un eficiente proceso de GCS permite controlar el sistema como producto global a lo largo de su desarrollo, obtener informes sobre el estado del mismo, así como reducir el número de errores de adaptación del sistema, lo cual se traduce en un aumento de la calidad del producto, la satisfacción del cliente y en consecuencia, mejoras en la organización.

En la Facultad 3 de la Universidad de la Ciencias Informáticas (UCI), se encuentra en estado de desarrollo el Proyecto de Modernización de los Registros y Notarías de la República Bolivariana de Venezuela (en lo adelante R&N). Este proyecto se asienta en la elaboración de un software capaz de estandarizar los procesos registrales y notariales en la hermana República Bolivariana de Venezuela. Actualmente este proceso se lleva de forma independiente y diferente en cada registro, la distribución de los ingresos permite que los registradores y notarios alcancen una retribución excesiva, lo que origina una sustancial diferencia con otros funcionarios de la Administración Pública. El objetivo de este software es crear un sistema especializado en este proceso, que sea capaz de evitar fraudes de toda índole en los registros y notarías venezolanos.

Esta tarea consta de un tiempo de duración, hasta la fecha, de aproximadamente 22 meses, se lleva a cabo con un grupo de desarrollo de 80 personas, las cuales se encuentran distribuidas en 4 módulos:

- Registros Públicos.
- Registro Mercantil.
- Administración Financiera.
- Servicio Autónomo.

En el proceso de desarrollo de este software se han cometido varios errores, los cuales han producido atraso en la evolución del producto y por ende han afectado los compromisos de entrega establecidos. Uno de los pilares fundamentales en el proceso de desarrollo de un producto, la Gestión de Configuración de Software, se ha visto afectada gravemente en el transcurso de este proyecto, a tal punto que pudiera decirse, ha sido casi nula.

Partiendo del análisis anterior, se han detectado los siguientes problemas vinculados a la GCS, a lo largo del proceso de desarrollo del software:

- No se cuenta con procesos bien definidos y guiados por los principios de la ingeniería y la gestión de software.
- No se encuentra definido el rol de Gestor de Configuración.
- No se cuenta con un procedimiento definido para la Identificación de la Configuración de Software.
- Los cambios en el proceso de desarrollo no se hacen bajo un sistema de gestión de configuración que permitan el control y seguimiento de los mismos.
- En muchos casos no se crea o respeta la línea base del proyecto, permitiéndose la modificación de la misma en cualquier etapa de la vida del proyecto.
- La falta de control de las versiones del software genera descontrol y desorganización en el proceso de desarrollo.
- Cuando se realiza un cambio todas las partes interesadas no se enteran del cambio, ni se realizan los procesos de sincronización ante el mismo, que imposibiliten la pérdida de tiempo o la desorganización en el equipo de desarrollo.
- No se tiene un sistema de informe y análisis estadísticos de los cambios realizados al proyecto, lo que imposibilita una reprogramación y estimación coherente ante los cambios, la toma de decisiones dentro del proyecto por parte del gestor de proyecto comienza a ser por espontaneidad y no acorde a un sistema metodológico.
- No se realizan las inspecciones necesarias a la gestión de configuración, ni revisiones técnicas a los cambios que se hacen.
- No se logra el control de la calidad del cambio solicitado por la no conformidad.
- No se realizan procesos de gestión y mitigación de riesgos, ni tareas que ayuden a la predicción de los mismos.
- Existe descontrol entre los artefactos y sus relaciones.
- No se cuenta con una herramienta estándar para todos los módulos, que automatice las actividades de la GCS.

A partir de los problemas detectados, se plantea el siguiente Diseño Teórico, como marco de referencia de la investigación:

Problema

No realizar un adecuado proceso de Gestión de Configuración en el proyecto R&N influye de forma negativa en el control de cambios sobre los artefactos en desarrollo, así como en la integridad de los mismos.

Objeto de Estudio

El proceso de Gestión de Configuración de Software.

Objetivo General

Desarrollar una estrategia para la Gestión de Configuración de Software en el proyecto R&N.

Objetivos específicos

1. Elaborar el marco teórico de la investigación.
2. Definir procedimientos que regulen y estandaricen el proceso de Gestión de Configuración de software en el proyecto R&N.
3. Seleccionar y justificar las herramientas de software que serán utilizadas para automatizar la GCS en el proyecto R&N.
4. Aplicar en el submódulo Bienes del proyecto R&N, los procedimientos definidos en la estrategia.

Campo de Acción

La Gestión de Configuración de Software en el Proyecto de Modernización de los Registros y Notarías de la República Bolivariana de Venezuela.

Hipótesis

Si se desarrolla una estrategia de Gestión de Configuración de Software adecuada a las particularidades del proyecto R&N entonces se logrará mayor efectividad en el control de los cambios sobre los artefactos en desarrollo y en la integridad de los mismos.

Valor Práctico

Se elabora una estrategia para la GCS que ayuda a llevar un proceso de desarrollo de software mejor controlado en el proyecto R&N. Este control se llevará a cabo de forma organizada, siguiendo una serie de pasos para su ejecución más eficiente. De esta forma se contribuirá en gran medida a la calidad en el

desarrollo del software, la detección y corrección de los errores a tiempo y por ende a la mejoría en el cumplimiento de las fechas de entrega del software.

Para lograr los objetivos trazados y demostrar la hipótesis establecida se acometieron las siguientes Tareas de Investigación:

1. Estudiar el estado actual de la Gestión de Configuración de Software en el mundo.
2. Estudiar las particularidades de la gestión de configuración en el proyecto Registro y Notarías.
3. Definir los procedimientos para identificar los Elementos de Configuración de Software del proyecto.
4. Definir los procedimientos para establecer la línea base del proyecto.
5. Definir las relaciones entre los elementos de configuración de software.
6. Definir los procedimientos para el control de los cambios generados por el equipo de desarrollo y por las no conformidades de los clientes.
7. Definir los procedimientos y herramientas a utilizar para el control de versiones.
8. Definir aspectos evaluativos para el proceso de control interno de la GCS.

Estructura de la Tesis

La tesis quedó estructurada en tres capítulos. El capítulo 1, referido al marco teórico y referencial de la investigación donde se realiza un análisis crítico y valorativo del estado del arte en el tema de Gestión de Configuración de Software. El capítulo 2, se propone una estrategia que sirva de guía al proceso de GCS en el proyecto R&N. En el capítulo 3 tiene lugar la aplicación de dicha estrategia en el submódulo Bienes así como la descripción de algunos resultados observados.

Fundamentación Teórica 1

1.1 La Gestión de Configuración en el Desarrollo de Software

La tendencia actual en la industria del software lleva a la construcción de sistemas cada vez más grandes y complejos. Por tal motivo, la presencia de un proceso bien definido y bien gestionado es la diferencia fundamental entre proyectos altamente productivos y otros que fracasan.

Debido a esta tendencia a nivel mundial, se ha impulsado la necesidad de mejorar las prácticas de ingeniería de software con el objetivo de mantener la integridad en los productos que se obtienen. El atributo integridad en un producto software está dado por las siguientes exigencias (Antonio, 2001):

- Cumplir todos los requisitos del usuario, tanto los que están explícitos como los que se encuentran implícitos.
- Cumplir los requisitos de rendimiento.
- Presentar trazas de su evolución desde que se concibió, y a través de todas las fases de su ciclo de vida.

En (Antonio, 2001), la autora plantea que la integridad de un producto software depende de la acción combinada de tres tipos de disciplinas:

- Desarrollo.
- Gestión.
- Control.

Dentro de las disciplinas de control se encuentra la Gestión de la Configuración del Software, la cual contribuye al mantenimiento de la integridad de los componentes del producto software. Esta área es la encargada de evaluar y controlar los cambios que sobre los ECS se efectúen y además facilita la visibilidad sobre los mismos.

A interrogantes como:

- ¿Le gustaría cumplir siempre las fechas de entrega?
- ¿Necesita mejorar la forma de gestionar los cambios a lo largo del ciclo de vida del desarrollo?

Varios autores (Antonio, 2001) (Estrada, 2003) (Martínez, 2006) responden siguiendo el argumento de que la GCS constituye uno de los pilares fundamentales, para la obtención de un producto con éxito, debido a su rol protagónico dentro de dicho proceso, a la importancia de su aplicación en las empresas productoras de software, así como los caóticos resultados observados por su mala aplicación y en el peor de los casos, por la carencia total de estos procesos en desarrollo de un producto software.

A continuación se muestran muchos de los problemas que actualmente existen en la Industria del Software y que están dados por la ausencia de una adecuada GCS:

- En ocasiones la versión actual del código se sobrescribe por una anterior.
- Se hacen cambios a una versión incorrecta del código.
- Reaparecen errores ya corregidos.
- No se logra determinar qué versiones de ficheros van en una entrega.
- Las construcciones no son reproducibles.
- Muchas posibilidades de error cuando se mantienen múltiples versiones.
- No hay historial de los cambios.
- Los jefes de proyecto no pueden medir el avance.
- Pobre comunicación del equipo.

La raíz de estos problemas está dada en que no se le brinda la debida importancia al proceso de GCS y ello lamentablemente deriva en el descontrol sobre los artefactos en desarrollo y la descoordinación en el equipo que trabaja en la evolución de los mismos.

A medida que pasa el tiempo el software crece y proporcionalmente a esto la cantidad de información asociada, por lo cual resulta más difícil controlarla si no se cuenta con técnicas especializadas para esta tarea. En proyectos grandes, con aplicaciones de larga vida, o que requieren del mantenimiento simultáneo de múltiples versiones, la inexistencia de actividades de GCS hace que el riesgo sea mucho

mayor, debido a que si el equipo no controla bien los cambios que surgen a lo largo del ciclo de vida, es muy probable que los cambios descontrolen al equipo, llevándolos al caos total.

Sin embargo, la presencia de una buena comunicación en el equipo y una gestión de cambios apropiada, constituyen aspectos que influyen de forma positiva en los proyectos a la hora de proporcionar calidad a sus productos. En este sentido, varios autores ofrecen su apreciación sobre cómo el éxito del proceso de desarrollo del software depende de la correcta realización de cuatro tipos de funciones (Antonio, 2001) (Navarro J. A., 2006) (Estrada, 2003):

1. La Gestión del Proyecto.
2. El Desarrollo Técnico.
3. El Sistema de Calidad.
4. El Sistema de Gestión de Configuración.

Una vez más la GCS es señalada entre las principales actividades que contribuyen a la mejora en el desarrollo de software, por autores de gran prestigio (Antonio, 2001) (Babich, 1986) (Estrada, 2003) (Gómez) (Martínez, 2006) (Navarro A.) (Pressman, 2005). Dado el análisis de la GCS y su importancia en el desarrollo de software, resulta meritorio destacar lo que sobre el tema plantea La Primera Ley de la Gestión de Configuración:

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

Otro de los argumentos que defiende la importancia de la GCS en el desarrollo de software es que el proceso de GCS no es una actividad aislada del resto de los procesos partícipes en el desarrollo. Muy por el contrario, puede ser visualizado como un comunicador al interior de cada proyecto (Acevedo, 2004), debido a que:

- Mantiene una estrecha relación de trabajo con todas las entidades del proyecto.
- Informa a cada desarrollador sobre el estado del producto y su evolución.

- Recopila y gestiona la documentación definida y aprobada para el producto, poniéndola a disposición de quien la requiera.
- Cobra gran importancia en la fase de mantenimiento del software.

En este último punto las autoras consideran necesario aclarar la diferencia entre el mantenimiento del software y la Gestión de Configuración de Software. El mantenimiento es un conjunto de actividades de ingeniería de software que se producen después de que el software se haya entregado al cliente y esté en funcionamiento, mientras que la GCS se debe realizar a lo largo de todo el ciclo de vida del producto, tanto en el desarrollo como en el mantenimiento, hasta que el producto sea retirado.

1.2 La GCS en los Modelos y Estándares de Calidad

Durante el epígrafe anterior se brindaron elementos que argumentan la importancia de la GCS en el proceso de desarrollo de Software y es que precisamente su importancia radica en que es una de las disciplinas encargadas de mantener la calidad del producto a lo largo de todo su ciclo de vida. Por tal motivo, la GCS es considerada un elemento importante de garantía de calidad del software. Sobre el tema Pressman (Pressman, 2005) emite su opinión, cuando plantea:

"Es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente".

En lo referente a la Gestión de Calidad del software a nivel mundial se han seguido básicamente dos directrices. La primera se guía por las reglas implantadas por las oficinas internacionales de estandarización para los productos y servicios a través de las normas ISO y la IEEE, mientras que la segunda son las creadas específicamente para el mundo del software como CMM y su evolución CMMI (Navarro J. A., 2006).

1.2.1 ISO

La ISO, Organización de Estandarización Internacional, ha definido una serie de estándares que son generalmente aplicables a todos los procesos de producción.

La ISO 9000 se ha especializado en todo lo referente a la solución de software en la ISO 9000-3, debido a que esta disciplina tiene características propias diferentes como para distinguirse del proceso de producción en general.

Entre los aspectos que tiene en cuenta la Norma ISO 9000-3 se encuentra la Gestión de Configuración de Software (Martínez, 2006), como uno de los procesos principales de apoyo al desarrollo de productos informáticos.

Por otra parte el estándar ISO/IEC 12207 para Procesos del Ciclo de Vida del Software, establece el proceso de Gestión de Configuración Software como uno de los procesos de soporte del ciclo de vida. Un proceso de soporte "apoya" a otro proceso como una parte integral, con un propósito distinto, y contribuye al éxito y a la calidad del proyecto de software (Hista International S. A., 2006).

1.2.2 IEEE

El Instituto de Ingenieros Eléctricos y Electrónicos, IEEE[1] por sus siglas en inglés, es una Asociación Profesional Técnica de más de 350,000 miembros individuales en 175 países dedicada a la estandarización, entre otras actividades (IEEE, 1998). Dentro de la familia de estándares producidos por la IEEE es importante mencionar el estándar IEEE 730-1998 para el Plan de Aseguramiento de la Calidad de Software (IEEE, 1998), en el cual se encuentran aspectos como la administración, la documentación, el control de código, entre otros. Dentro de la documentación le concede gran importancia al Plan para la Gestión de Configuración de Software mostrado en el estándar IEEE 828-1998 (IEEE, 1998) (Navarro J. A., 2006).

Dicho plan aborda lo referente a la asignación de las responsabilidades, la identificación de las distintas actividades que serán el soporte durante todo el proceso de GCS, la identificación de los elementos de configuración del software, el control de estos elementos que integran la configuración del software, el acceso a las bibliotecas, la aprobación o desaprobación de un cambio y la implementación del cambio, en caso de ser aprobado (Martínez, 2006).

El estándar IEEE Std. 1074-1995 para el desarrollo de procesos del ciclo de vida del software, establece el proceso de Gestión de Configuración de Software como uno de los procesos integrales, los cuales son necesarios para completar exitosamente las actividades del proyecto, y son utilizados para asegurar la

[1] IEEE: Institute of Electrical and Electronics Engineers.

finalización y calidad de las funciones del proyecto. Además consideran necesarias para este proceso, las siguientes actividades (Hista International S. A., 2006):

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

1.2.3 CMMI

El Modelo Integral de Madurez de las Capacidades, CMMI[1] por sus siglas en inglés, es un enfoque para la mejora de procesos que brinda a las organizaciones los elementos básicos sobre procesos efectivos. Su principal función es guiar a los equipos de desarrollo de software en el sentido de la mejora de procesos a lo largo de un proyecto. CMMI ayuda a integrar funciones organizacionales tradicionalmente separadas, otorgar metas y prioridades en la mejora de procesos, brinda además una guía para los procesos específicos de calidad y un punto de referencia para la valoración de los procesos actuales (Martínez, 2006).

Cuenta con cinco niveles, al igual que su predecesor CMM, aunque algunos cambian su nomenclatura. Para obtener Nivel 2 de CMMI (Gestionado) es necesario tener definidos los siguientes procesos (Gracia, 2005):

- Gestión de los requisitos del producto y del proyecto.
- Planificación de los proyectos.
- Seguimiento y control de los proyectos.
- Gestión de acuerdos con los proveedores de productos y servicios.
- Selección y supervisión de los proveedores.
- Medición y análisis.
- Aseguramiento de la calidad del producto y del proceso.
- Gestión de la Configuración de Software.

[1] CMMI: Capability Maturity Model Integration.

La información anterior pone de manifiesto el lugar básico que ocupa la GCS, pues de los cinco niveles que plantea el modelo, desde el número dos, hay que garantizar las actividades de GCS para poder considerar que se cumple con la especificación del mismo.

Partiendo del análisis bibliográfico efectuado, se evidencia que la Gestión de Configuración de Software es valorada como una actividad de gran importancia dentro del proceso de desarrollo por varios autores y consecuentemente es considerada en las principales normas y modelos de calidad de software a nivel mundial. El conocimiento de las funciones incluidas dentro del proceso revela la veracidad de lo anteriormente planteado.

1.3 El Proceso de Gestión de Configuración de Software

Como resultado del proceso de ingeniería de software que se realiza a lo largo del ciclo de desarrollo de un producto, se genera información de diferente naturaleza y que pueden ser clasificadas en (Pressman, 2005):

- Programas de computadora (tanto en forma de código fuente como ejecutable)
- Documentos que describen los programas de computadora (tanto técnicos como de usuario)
- Datos (contenidos en el programa o externos a él).

Los elementos que componen toda la información producida como parte del proceso de ingeniería de software se denominan colectivamente configuración del software. En la Figura 1.1 se visualiza lo anteriormente planteado, para su mejor comprensión:

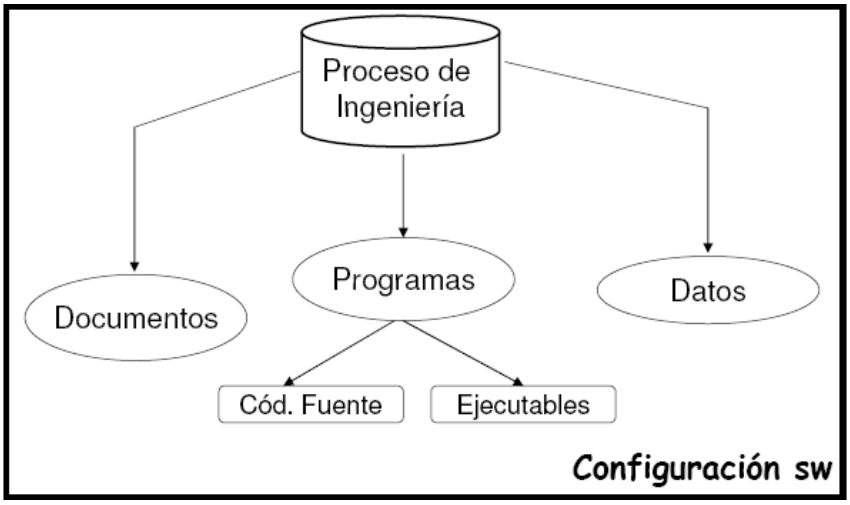


Figura 1. 1 Configuración del Software.

La rama de la Ingeniería de software encargada de controlar la evolución de todos los elementos de configuración existentes en el desarrollo, involucrando un conjunto de técnicas para gestionar con eficiencia las modificaciones que se realicen sobre estos elementos a largo de su ciclo de vida, es la Gestión de Configuración de Software (Rancán, 2003).

Esta disciplina, definida en los años 70 e implantada en los años 90, es conceptualizada en el diccionario de términos IEEE, como el proceso de identificar y definir los elementos de configuración en un sistema, controlando la entrega y el cambio de estos elementos a través del ciclo de vida del producto, almacenando el estado de los elementos de configuración y de las solicitudes de cambio y verificando la completitud con respecto a los requerimientos especificados (Proceso de Control de Cambios Guiado por la Arquitectura del Software).

Según Ivar Jacobson (Ivar Jacobson, 2003) la GCS controla los cambios y mantiene la integridad de los artefactos del proyecto. Esta disciplina se encarga de identificar los elementos de configuración, restringir y auditar los cambios a esos elementos, así como definir y gestionar las configuraciones de estos.

En (Pressman, 2005) se hace referencia a una de las definiciones más completas y utilizadas, planteada por 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.”

Es evidente que la responsabilidad principal de la GCS es el control de los cambios que surgen en todo el ciclo de vida del software. Sin embargo, este proceso es complementado por otras actividades, las cuales tienen como objetivo mantener el control, la organización y el almacenamiento de todo el flujo de información, con la misión de maximizar la producción reduciendo los errores.

Según la investigación realizada, se detectó que diferentes autores proponen varios subprocesos para la GCS, e incluso le atribuyen a algunas distintas nomenclaturas. El estándar de la IEEE y la ISO presentan similitud en los subprocesos que consideran necesarias para la GCS (Antonio, 2001) (Estrada, 2003) (IEEE, 1998):

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

Por su parte en CMMI, se plantean los siguientes procesos o prácticas claves (Estrada, 2003):

- Planificación de las actividades de Gestión de Configuración.
- Identificación de los ECS.
- Control de cambios a los ECS.
- Informar a los grupos e individuos involucrados de los cambios a los ECS.
- Auditoría de la Configuración.

Sin embargo, las autoras consideran que existe una actividad que resulta de vital importancia su aplicación en el proceso de GCS y en los modelos anteriormente referenciados no es tomada en cuenta como una actividad independiente, se trata de la gestión de versiones. Cuando esta actividad es obviada y se mezcla con el control de los cambios en un proceso único, sin delimitar fronteras y actividades específicas, se corre el riego de que puedan diluirse algunas responsabilidades y no se ejecuten muchas tareas que son imprescindibles como la actualización de los repositorios, el control de acceso a los ECS, entre otros. En el capítulo 2 se brindan más detalles sobre cada una de las actividades que fueron seleccionadas para integrar la estrategia propuesta.

1.3.1 Identificación de la Configuración del Software

La identificación de los elementos de la Configuración es considerada una de actividades necesarias y preliminares dentro del proceso de GCS. Esta tarea constituye una tarea de las más importantes en el proceso, ya que el éxito del mismo depende, en cierta medida, de que se haya efectuado una correcta identificación de los elementos existentes.

Una definición bastante clara sobre la Identificación de la Configuración de Software es brindada por Angélica de Antonio (Antonio, 2001), donde enuncia que la Identificación de la Configuración de Software consiste en identificar la estructura del producto de software, sus componentes y el tipo de estos, y en hacerlos únicos y accesibles, de alguna forma.

Como fuera anteriormente expuesto, al conjunto de todos los elementos que se obtienen a lo largo de todo el proceso de desarrollo de Software se les denomina Configuración del Software (Pressman, 2005) y por ende a cada uno de estos elementos, por separado, suele llamársele Elementos de la Configuración del Software (ECS).

Un ECS debe ser un componente que se pueda definir y controlar de forma independiente, es decir, debe ser una unidad en sí mismo. Los ECS constituyen la base de información para las actividades de GCS, debido a que la evolución de los mismos a lo largo del desarrollo del software da paso a la necesidad de controlarlos y almacenarlos (Antonio, 2001). En el Anexo 1 se visualizan varios ejemplos de elementos de configuración de un proyecto de software.

Según varios autores (Martínez, 2006) (Antonio, 2001) para llevar a cabo de forma exitosa el proceso de Identificación de la Configuración de Software, es necesario realizar un conjunto de actividades:

- Establecimiento de una jerarquía preliminar del producto software
- Selección de los elementos de configuración de software.
- Definición de las relaciones en la configuración.
- Definición de un esquema de identificación.
- Definición y establecimiento de líneas base.
- Definición y establecimiento de bibliotecas de software.

Las autoras de este trabajo consideran que a la hora de realizar esta actividad se le debe brindar gran importancia a la selección de los elementos, debido a que tanto la omisión de algún elemento significativo como la selección de un número elevado de los mismos, pueden producir males mayores en tiempos muy cortos. También constituye un paso muy importante, la asignación adecuada de los atributos correspondientes a cada elemento, así como la identificación de las relaciones existentes entre los ECS, lo cual brinda mayor información, una vez que sean consultado y además poder valorar el impacto que tiene la modificación de algún de los ECS.

1.3.1.1 Línea Base

En (Rancán, 2003), el autor plantea que las Líneas Base pueden ser consideradas como hitos en el proceso de desarrollo que se constituyen en función de la aprobación de uno o varios ECS mediante la elaboración de revisiones técnicas formales.

La IEEE (IEEE, 1990) define una línea base como: Una especificación o producto que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios.

En la documentación del Proceso Unificado de Desarrollo de Software (Ivar Jacobson, 2003), se define que Línea Base es una instancia de una versión de cada artefacto dentro del repositorio del proyecto. La misma provee una norma oficial para guiar las siguientes actividades dentro del proceso de desarrollo y sobre las cuales solo se podrán realizar cambios que estén autorizados.

Habitualmente las Líneas Base se ubican a la finalización de determinadas fases del proceso de desarrollo, buscando la obtención de dos objetivos (Rancán, 2003).

1. Identificar los resultados de las tareas realizadas durante la fase.
2. Asegurar que se ha completado la fase, contando con elementos consolidados para iniciar la fase siguiente.

Una línea base puede ser instaurada de dos formas (Antonio, 2001) (Estrada, 2003):

Físicamente: Etiquetando cada Elemento de Configuración del Software y almacenándolos en un Archivo o Biblioteca de Proyecto.

Lógicamente: Publicando un documento de identificación de la configuración, que hace referencia al estado actual del producto en dicho punto del proceso de desarrollo.

En conclusión, la idea de la línea base consiste en permitir cambios rápidos e informales sobre un Elemento de Configuración del Software antes de que se pase a formar parte de dicha línea, pero en el momento en que se establece una línea base se debe aplicar un procedimiento formal para evaluar y verificar cada cambio.

1.3.2 Control de Cambios

La primera Ley de la Ingeniería de Sistemas establece: “Sin importar en qué 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” (Navarro J. A., 2006).

El cambio hay que asumirlo, no evitarlo (Antonio, 2001). Es importante tener en cuenta que la mayoría de los cambios están justificados, ya que a medida que pasa el tiempo se sabe más acerca del problema y de cómo resolverlo. Varios cambios de los que se originan, pueden estar dados por cambios tecnológicos, variaciones en las estrategias o prioridades del negocio, porque al analizar el problema no se hacen las preguntas correctas a las personas correctas o por cambios en el ambiente de negocios.

El cambio incontrolado produce caos. Para evitar que los mismos se apoderen de la marcha del proyecto y lo lleven al fracaso, se debe incluir un mecanismo formal para controlarlos y además este debe ser cumplido estrictamente durante su ejecución. Dicho mecanismo debe ser aceptado por el cliente y conocido por todo el equipo del proyecto (Pressman, 2005).

Varios autores, (Navarro A.) (Pressman, 2005) (Rancán, 2003), plantean que para llevar a cabo un eficiente proceso de gestión de cambios es necesario incluir al menos las siguientes actividades:

- Petición de cambios.
- Evaluación de los cambios.
- Aprobación o desaprobación de los cambios.
- Implementación de los cambios (En caso que haya sido aprobado en el paso anterior).
- Auditorías y revisiones de la configuración.

De estas tareas, las autoras consideran necesario destacar la evaluación del cambio, como una de las actividades de vital importancia dentro de este proceso. En dicha actividad se realiza un análisis detallado del impacto que tendrá un elemento sobre el resto, una vez que haya sido modificado o deje de funcionar. En este análisis se debe modelar toda la estructura como un grafo, donde los elementos de configuración son nodos y sus relaciones de dependencia, son aristas. Esto se traduce en el grafo a encontrar todos los nodos alcanzados en una recorrida a partir del nodo origen del cambio (Machuca). También se debe tener en cuenta la planificación del cambio, es decir, el personal asignado para esta tarea, así como la fecha en la que se va a efectuar la mima.

En ocasiones las personas se sienten incómodas con el nivel de burocracia que implica el proceso de control de cambios. Según Pressman, esta sensación es normal, ya que sin la protección adecuada, el control de cambios puede retrasar el progreso y crear un papeleo innecesario (Pressman, 2005).

Aunque para algunos resulta tedioso, no por ello deja de ser tan importante y más aún en proyectos grandes donde el flujo de información crece rápidamente y la evolución del software resulta más difícil de controlar sin la existencia de técnicas especializadas.

Una vez implementado el cambio, se debe realizar una inspección sobre el sistema con el objetivo de certificar que el problema se ha rectificado o se han satisfecho los nuevos requisitos. Cumplida la inspección se procede a “liberar” o dar de “alta” los ECS modificados.

Los procedimientos de "alta" y "baja" implementan dos elementos importantes del control de cambios: control de acceso y control de sincronización. El control de acceso gobierna los derechos de los ingenieros de software a acceder y modificar objetos de configuración concretos. El control de sincronización asegura que los cambios en paralelo, realizados por personas diferentes, no se sobrescriban mutuamente (Pressman, 2005). En la Figura 2 se ilustra el proceso de control de acceso y sincronización para su mejor comprensión.

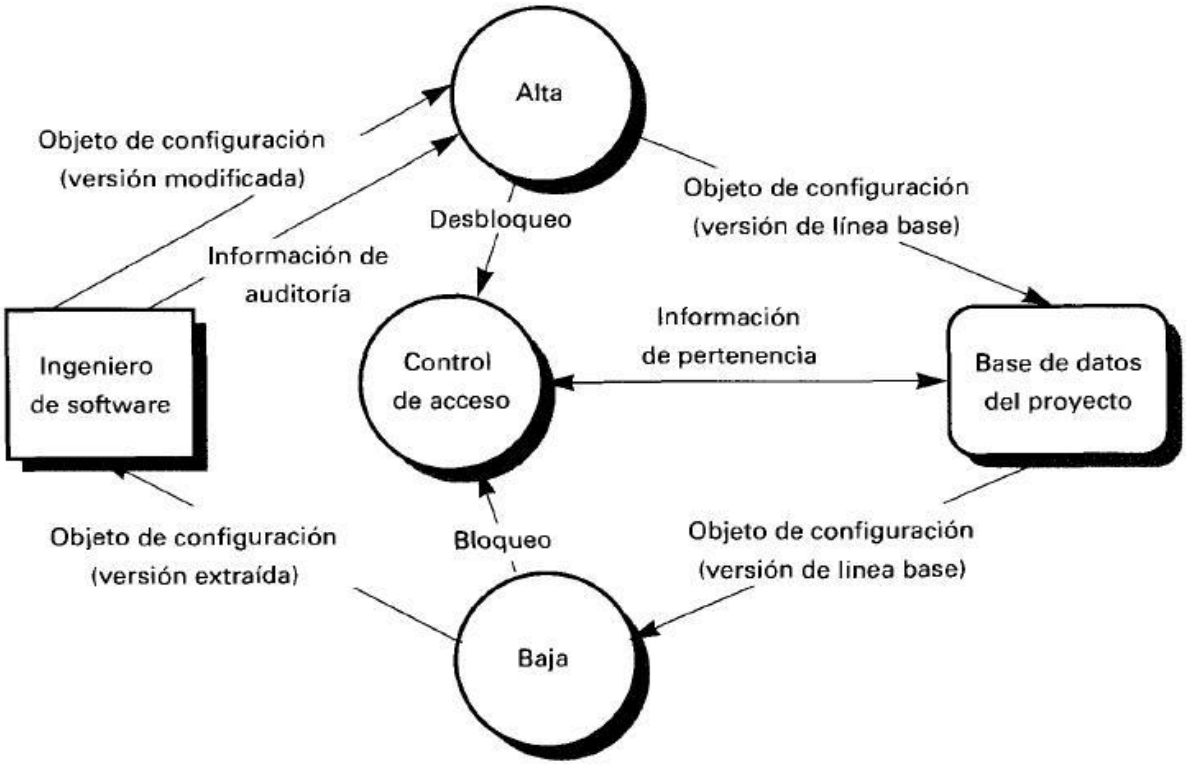


Figura 1. 2 Control de Acceso y Sincronización (Pressman, 2005).

Posteriormente se realizan actividades de garantía de calidad y prueba. Se compila una nueva Línea Base para la ejecución de las etapas de prueba y se ingresa en la base de datos del proyecto. La finalización de procesos de cambio y/o modificaciones normalmente ocasiona un cambio de versión del producto software.

1.3.3 Control de Versiones

Difícilmente un archivo de código o un documento de texto están terminados con la primera escritura; casi siempre necesitan cambios o reescrituras para corregir errores, modificar su contenido. A medida que el documento cambia existen dos opciones: mantener un historial de cambios o dejar que evolucione sin memoria. El control de versiones es un método estándar para mantener esta memoria haciendo además que sea útil para el desarrollo futuro.

En documentos sencillos como un pequeño programa la memoria no es algo esencial, pero en la escritura de programas con millares de líneas de código y una docena de manos, esta técnica resulta necesaria. La frase clave es: Mantener un control de las versiones de todos los archivos de un proyecto provee una manera completamente estandarizada de trabajar.

Pressman (Pressman, 2005) cita a Clemm cuando describe el control de versiones en el contexto de la Gestión de Configuración: "La gestión de configuración permite a un usuario especificar configuraciones alternativas del sistema de software mediante la selección de las versiones adecuadas. Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar y construir una configuración describiendo el conjunto de atributos deseados".

Por tal motivo una versión puede ser considerada como una instancia de un elemento de configuración, en un momento dado del proceso de desarrollo, que es almacenada y que puede ser recuperada en cualquier momento para su uso o modificación. A su vez, la versión del sistema viene identificada por las versiones de los elementos de configuración de software.

Resulta habitual centralizar el almacenamiento de los componentes de un mismo sistema, incluyendo las distintas versiones de cada componente. Este almacén común se denomina *"repositorio".* El repositorio permite ahorrar espacio de almacenamiento, evitando guardar por duplicado elementos comunes a varias versiones o configuraciones, además facilita el almacenar información de la evolución del sistema.

La Gestión de Configuración permite además, especificar y gestionar distintas variantes de los elementos de configuración. Las variantes son versiones de un ECS que coexisten en un determinado momento y que se diferencian entre sí en ciertas características. Las variantes representan la necesidad de que un objeto satisfaga distintos requisitos al mismo tiempo (Antonio, 2001). A diferencia con las revisiones, que son estrictamente secuenciales y sólo existe una como revisión actual, las variantes se desarrollan en paralelo.

Una variante no reemplaza a otra, como ocurre con las versiones, sino que abre un nuevo camino de desarrollo. Las variantes se reconocen fácilmente en los grafos de evolución, puesto que aparecen como una ramificación de éste. A su vez, una variante puede pasar por una sucesión de revisiones, que deberán ser identificadas convenientemente. Las variantes suelen darse cuando por ejemplo, se tiene una misma versión para diferentes plataformas.

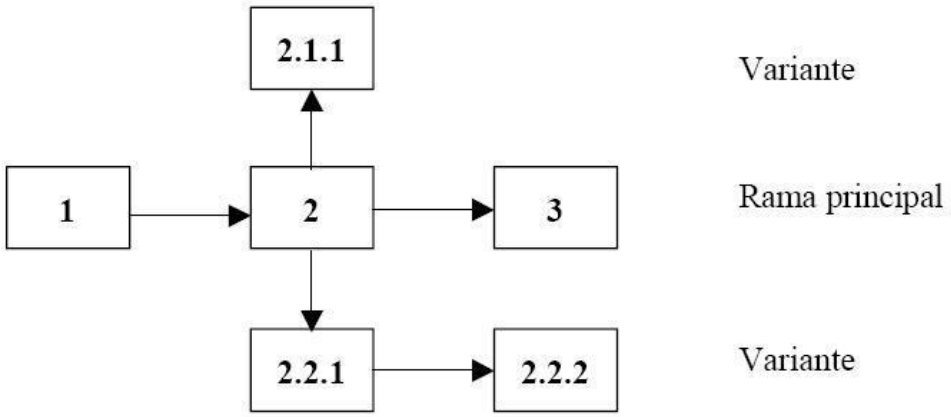


Figura 1. 3 Ejemplo de la existencia de diferentes variantes.

Las variantes pueden ser temporales o permanentes. Una variante temporal es toda aquella que se acabará mezclando con otra variante en algún momento del desarrollo (Antonio, 2001). En ciertas ocasiones es necesario que varias personas trabajen simultáneamente sobre la misma versión de un objeto, y para que no ocurran conflictos entre ellas, se crea una variante distinta para cada persona, para que puedan trabajar en paralelo. Una vez que todas ellas han acabado las modificaciones, es necesario mezclar todas las variantes para que la versión resultante contenga todos los cambios realizados.

Las autoras de este trabajo consideran necesario resaltar que las variantes temporales deben evitarse y en todo caso mezclarse lo antes posible, ya que a medida que pasa el tiempo divergirán más de la versión original y entre sí y resultará más difícil su fusión.

1.3.4 Auditoría de la Configuración

La Auditoría de la Configuración de Software (ACS) es una de las actividades que se realizan con el objetivo de comprobar que se efectuó correctamente el Proceso de Gestión del Cambio. No es suficiente solo con evaluar y autorizar el cambio, sino que hay que asegurarse de que se ha realizado y de forma correcta, esta es la premisa de dicha actividad.

Esta actividad tiene la función de complementar las Revisiones Técnicas Formales (RTF) que se hacen sobre los elementos de configuración (Pressman, 2005), debido a que en dichas revisiones, los encargados se concentran solamente en la corrección técnica del elemento que ha sido modificado y no

tienen en cuenta en sus revisiones otros aspectos de vital importancia que si son protagonistas el proceso de la Auditoría de la Configuración.

Suelen distinguirse dos tipos de Auditorías de Configuración del Software (Rancán, 2003):

Auditoría Funcional: Su objetivo es comprobar que se han completado todas las pruebas necesarias para el elemento de configuración auditado, y que, teniendo en cuenta los resultados obtenidos en las pruebas realizadas, se puede afirmar que el elemento de configuración de software satisface los requisitos que se impusieron sobre él.

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

Se puede concluir planteando que la ACS es una de las actividades que tributa en gran medida a que el proceso de Gestión de Configuración de Software se efectúe con calidad y su gran importancia radica en que mediante el estricto cumplimiento de sus actividades se logra verificar de que todos los requisitos han sido cumplidos y que el software y su documentación están completos y listos para entregar.

Las autoras consideran necesario adicionar que, debido a su importancia, la ACS es considerada la más costosa de las actividades de GCS, requiere de personal experimentado, y con un gran conocimiento del proceso de desarrollo. Por tal motivo, debe ser realizada por personal ajeno al equipo de desarrollo técnico para mantener la objetividad de la auditoría (Pressman, 2005). Si el proceso de control de cambios se realiza mediante un mecanismo formal, se recomienda que el proceso de ACS se lleve a cabo por personal del equipo de calidad del proyecto.

1.3.5 Elaboración de Informes de Estado

La generación de informes de estado de la configuración, también denominada contabilidad de estado, desempeña un papel vital en el éxito del proyecto de desarrollo de software (Pressman, 2005). Cuando aparece involucrada mucha gente es muy fácil que no exista una buena comunicación. Pueden darse errores entre las personas desarrolladoras del software. La generación de Informes de Estado ayuda a eliminar esos problemas, mejorando la comunicación entre todas las personas involucradas. Los Informes de Estado de la Configuración informan sobre lo que aconteció, cuándo y quien lo hizo, así como las demás personas afectadas.

Varios autores recomiendan la generación de un Informe de Estado de la Configuración con las siguientes frecuencias (Navarro A.) (Pressman, 2005).

- Cuando se establece una nueva identificación a un ECS.
- Cuando se envía una orden de cambio.
- Cuando se realiza una ACS.
- Habitualmente, para mantener informados a los desarrolladores de los cambios significativos que ocurren en el desarrollo.

1.4 La GCS y Metodologías de Desarrollo de software

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que se obtiene es insatisfacción por parte de los clientes con el resultado y desarrolladores aún más insatisfechos.

1.4.1 Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo, RUP[1] por sus siglas en inglés, es una metodología de Ingeniería del Software que proporciona una visión disciplinada para la asignación de tareas y responsabilidades en las organizaciones de desarrollo de software. RUP enmarca la disciplina Gestión de Configuración en todo el ciclo de vida del proyecto, desde que este se inicia hasta que se cierra (desde la fase de concepción hasta la de transición) y su propósito es el de controlar los diferentes artefactos que son producidos o elaborados por muchas personas que participan en un mismo proyecto (Ivar Jacobson, 2003).

[1] RUP: Rational Unified Process

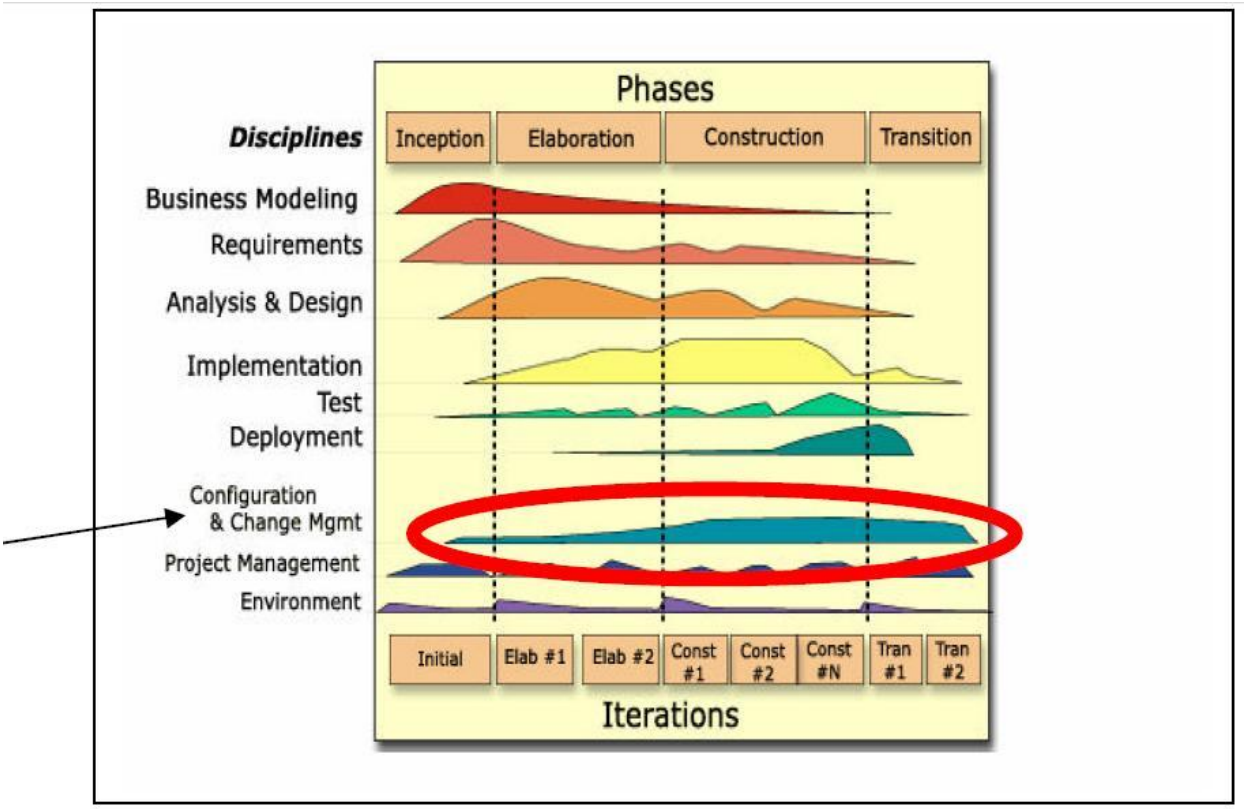


Figura 1. 4 Fases y Flujos de Trabajo de la Metodología RUP.

El Proceso Unificado de Desarrollo plantea que los cambios a los artefactos del proyecto se proponen mediante solicitudes de cambio. Luego estos son utilizados para documentar y controlar defectos, solicitudes de mejoras o cualquier otra solicitud de cambios al proyecto. El beneficio de utilizarlos es que proporcionan un registro de las decisiones y debido a su proceso evaluativo, se asegura que los impactos de los cambios sean entendidos por todos los miembros del equipo del proyecto (Bañeres, 2005).

Entre los objetivos que plantea RUP en lo referente a la GCS se encuentran:

1. Las actividades de administración de configuración de software deben ser planificadas.
2. Los productos de software seleccionados deben estar identificados, controlados y disponibles.
3. Los cambios a productos identificados de software son controlados.
4. Los grupos e individuos afectados son informados del estado y contenido del lineamiento base del software.

1.4.2 Métrica V3

Métrica Versión 3 es una metodología de desarrollo elaborada por el Consejo Superior de Informática del Ministerio de Administraciones Públicas del Reino de España, que ofrece a las organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software. (Ministerio de Administraciones Públicas).

Esta nueva versión de MÉTRICA contempla el desarrollo de Sistemas de Información para las distintas tecnologías que actualmente están conviviendo y los aspectos de gestión que aseguran que un proyecto cumple sus objetivos en términos de calidad, coste y plazos.

La interfaz de gestión de configuración de MÉTRICA Versión 3 permite definir las necesidades de gestión de configuración para cada sistema de información, recogiéndolas en un plan de gestión de configuración, en el que se especifican las actividades de identificación y registro de productos en el sistema de gestión de configuración durante el desarrollo y posterior mantenimiento del sistema de información (Ministerio de Administraciones Públicas).

Si en la organización ya existe un sistema de gestión de configuración estándar, para el sistema de información en concreto deberán analizarse las necesidades de configuración específicas respecto a dicho sistema estándar y determinar las diferencias, si las hubiera, así como aquellas necesidades concretas que no se encuentren recogidas, estableciendo así el plan de gestión de configuración del sistema de información.

1.5 Patrones de rama de la GCS

A partir del estudio realizado en (Ruiz Arroyo & García Peñalvo, 2007) (Appleton, Berczuk, Cabrera, & Orenstein) se proporcionan algunos patrones de ramas dada su importancia para el desarrollo en paralelo y las ventajas que el uso de los mismos proporciona. La selección de los patrones a utilizar dependerá en gran parte de las compensaciones deseadas entre seguridad y productividad. El uso de un número elevado de ramas para el mayor aislamiento de desarrolladores o tareas, reduce riesgos de seguridad, pero a expensas de un mayor esfuerzo en la combinación y la integración de las diferentes ramas. El esfuerzo, en estas dos últimas actividades mencionadas, es directamente proporcional a la capacidad de

las herramientas de soportar los diferentes tipos de patrones. A continuación se muestran características de estos patrones, así como su funcionamiento básico:

1.5.1 Rama por Proyecto

Se basa en la creación de una única rama, sobre la que todos los usuarios trabajan simultáneamente actualizando sus cambios.

Ventajas

1. No es necesario, con este patrón, utilizar controles de versiones con un fuerte soporte para ramas, puesto que hay solo una.
2. Este patrón al ser el más sencillo es soportado por todas los herramientas que brindan soporte a la gestión de la configuración.

Desventajas

1. El principal problema de este modelo es que no hay puntos intermedios en los que almacenar los cambios antes de pasar a la línea principal, por lo cual esta última es fácil de corromper.
2. Cada desarrollador es responsable de la estabilidad del código del producto.
3. El código pasa mucho tiempo fuera del controlador de versiones a tendiendo que los envíos al repositorio son menos frecuentes para evitar corromper la línea base con funcionalidades no probadas.
4. Siempre que se introduce un cambio en la línea principal de trabajo, los desarrolladores trabajaran sobre una línea base inestable hasta que el cambio finalice la fase de prueba.
5. Su utilización no posibilita separar la corrección de errores, en versiones anteriores de un producto, de las nuevas funcionalidades que se implementen para el mismo.
6. Un desarrollo con una única rama no es soportable a lo largo del tiempo para productos con múltiples versiones comercializadas o con equipos de desarrollo muy grande.

1.5.2 Rama por Release

Con este patrón, al finalizar cada versión del producto, se crea una nueva rama sobre la cual se continuará el desarrollo y la detección de defectos tal y como se hacía en el patrón de ramas por proyecto.

Ventajas

1. Facilita el mantenimiento de las versiones que estén siendo usadas por los usuarios ya que posibilita tener separado las nuevas funcionalidades de las errores de versiones anteriores.
2. Permite que los errores corregidos en versiones actuales sean actualizados también en las ramas que contienen las versiones anteriores.

Desventajas

1. Los defectos corregidos en versiones anteriores tienen que ser propagados hacia las siguientes versiones y generalmente los cambios son tan profundos o chocan entre sí que es necesario corregirlos nuevamente en las demás versiones.
2. Como su modo de trabajo es muy similar al patrón Rama por Proyecto incluye las desventajas 1, 2, 3, 4 enumeradas con la pasada sección.
3. En la corrección de un error desde la versión donde se corrigió hasta la actual, el entorno de trabajo varía constantemente, ya que debe cambiarse de una línea de desarrollo a otra para propagar la corrección.

1.5.3 Rama de Mantenimiento

Este patrón es una variante del anterior, por cada versión del producto se crea una nueva rama, pero en este caso la nueva rama se usa para el mantenimiento y detección de errores de esa versión.

Ventajas

1. Permite que un error corregido en una versión pueda ser incorporado fácilmente en la línea principal de trabajo, sin que haya que ir propagándolo hacia abajo pasando por todas y cada una de las versiones.
2. Este patrón suele combinarse con los patrones de rama por desarrollador y rama por tarea entre otros.
3. Permite separar los errores corregidos en versiones anteriores de las nuevas funcionalidades que se implementan.

Desventajas

1. En este patrón se mantienen las desventajas 1, 2, 3, 4 a las cuales se hizo referencia en el patrón de Rama por Proyecto

1.5.4 Rama por Desarrollador

En este patrón el desarrollo mantiene activas múltiples ramas: la rama principal del proyecto y además una rama adicional para cada desarrollador, donde cada uno trabajará de forma aislada a los demás y a la línea principal de desarrollo. Al finalizar el trabajo cada desarrollador lo integrará en la línea principal, aunque antes debe actualizar su rama con el contenido de la rama principal.

Ventajas

1. Permite, con el aislamiento, que los desarrolladores puedan tener almacenada en el repositorio las versiones intermedias e inestables de su código sin comprometer la integridad de la rama principal.
2. Durante todo el desarrollo del producto la línea principal se mantiene con una versión estable del código.
3. Propicia el desarrollo en paralelo pudiendo coexistir en cualquier momento diversas versiones para un mismo elemento (una en cada rama).

Desventajas

1. A diferencia de los patrones anteriores, es necesario emplear una herramienta de control de versiones que tenga buen soporte para ramas.

2. Los desarrolladores necesitan tener un conocimiento mayor sobre el funcionamiento del controlador de versiones.

1.5.5 Rama por Tarea

Utilizando este patrón se crea una nueva rama por cada tarea asignada a cada desarrollador y no suele ser este último el encargado de introducir los cambios en la línea principal del producto, sino que una persona cumple el rol de integrador y es quien se encarga de dicha labor de integración.

Ventajas

1. El aislamiento garantiza la protección de los datos en todo momento en el repositorio (dentro de la rama de la tarea).

2. Garantizan mayor estabilidad e integridad en el proyecto ya que posibilitan que se realicen almacenamientos controlados que no sean sobre la línea principal garantizando así la integridad de la misma.
3. Al estar cada cambio (cada tarea) en una rama diferente, se consigue una trazabilidad completa del mismo.
4. Permiten elegir de acuerdo a un proceso concreto cuándo un cambio se incorpora en el producto.

Desventajas

1. Se generan un elevado número de ramas, por lo que la herramienta a utilizar debe brindar un buen soporte el uso de las mismas.

De los patrones analizados anteriormente el más utilizado es el patrón de Rama por Proyecto, esto debido principalmente al desconocimiento de los demás patrones y también a la carencia de una herramienta que brinde un buen soporte a los mismos. Las autoras consideran que lo ideal sería generalizar la utilización de los patrones de Rama por Desarrollador o Rama por Proyecto en combinación con el patrón de Rama de Mantenimiento por las posibilidades que estos brindan, pero siempre hay que tener presente lo que se planteó al inicio de este epígrafe y es que a un mayor número de ramas es necesario un mayor esfuerzo en la combinación y la integración de las mismas.

1.6. Herramientas de Apoyo a la GCS

Con el desarrollo de las tecnologías de la informática han aparecido un conjunto de herramientas que proporcionan en mayor o menor medida soporte a las diferentes actividades de la gestión de configuración, las propuestas más robustas las ofrecen soluciones propietarias, con un alto precio en licencias. Es por esto, que como alternativa, los equipos de desarrollo deben seleccionar un conjunto de herramientas libres que por separado brinden las funcionalidades de las herramientas privativas.

Como ejemplo de un conjunto de herramientas representativas del mundo del software propietario se pueden citar: IBM Rational ClearCase, Visual Studio TeamSystem, Plastic SCM[1], Telelogic, Surround SCM. Mientras que CVS[1], Subversion (SVN), Darcs y Bit son ejemplo de herramientas libres.

[1] SCM: Software Configuration Manager.

Las herramientas citadas son frecuentemente denominadas por muchos autores controladores de versiones o sistemas de configuración de software (SCM por sus siglas en inglés) indistintamente de la funcionalidad que ofrezcan. Pero las autoras consideran importante aclarar que en el presente trabajo se utilizará el término Controlador de Versiones, si ésta es la única funcionalidad que cumple la herramienta, pero si además de dicha función brinda soporte a otras de las actividades de la gestión de configuración entonces se le denominará SCM.

Es importante destacar que a la hora de seleccionar cualquiera de estas herramientas se debe tener en cuenta las características del equipo de desarrollo y cuál es la mejor propuesta en dependencia de las necesidades propias del proyecto ya que si un equipo de trabajo tiene sus desarrolladores distribuidos geográficamente deberá utilizar una herramienta que soporte este tipo desarrollo y sería una solución diferente a la que necesitaría un equipo que no posea dichas características. También se debe tener en cuenta la forma de trabajo a seguir con la herramienta pues existen dos tipos de estas: los centralizados y los distribuidos, los cuales implican formas distintas de trabajar.

Distribuidos

En los distribuidos no existe un punto central de desarrollo sino que los repositorios están distribuidos y descentralizados en diversas máquinas que pueden o no ser independientes entre sí, y técnicamente no hay ninguno más importante que otro. El modo de trabajo suele ser que cada desarrollador tenga su repositorio propio sobre el cual trabaje de forma independiente, y periódicamente se pongan en común los trabajos de todos en algún repositorio convenido a tal efecto. Este tipo de herramientas son muy utilizados para el desarrollo por la comunidad de software libre, como ejemplo se pueden citar: Darcs, Arch, Bazaar, GIT, Monotone, Codeville y Bitkeeper (Lucarella & Bertogli, 2006).

Centralizados

Por otro lado, los centralizados se basan en un repositorio único central al que todos los desarrolladores se conectan para guardar sus cambios. Este repositorio es quien se encarga de manejar las distintas variantes de implementación de un mismo programa. Este tipo de herramienta implementa dos tipos de modelo: Bloquear–Modificar-Desbloquear o Copiar-Modificar-Mezclar.

[1] CVS: Concurrent Versions System.

Modelo: Bloqueo-Modificación-Desbloqueo

Utilizando este modelo se sigue como método de trabajo un desarrollo en serie ya que una vez bloqueado un archivo por parte de un desarrollador si otro necesita modificarlo no le será posible hasta que el primero en bloquearlo termine de hacer sus modificaciones y entonces desbloquee el archivo. O sea, la segunda persona debe esperar por el archivo o cambiar de tarea, lo cual introduce como desventajas pérdida de productividad y desestabilización del producto, ya que se estará presionando al programador que tiene el archivo bloqueado para que termine lo antes posible. Esa presión deriva en muchas ocasiones a que se introduzcan nuevos errores. El Visual Source Safe (VSS) es una herramienta representativa de este tipo de modelo.

Modelo: Copiar-Modificar-Mezclar

Es la alternativa al bloqueo de ficheros del modelo visto anteriormente y al desarrollo en serie, aquí cada desarrollador al descargar un determinado elemento del repositorio hace una copia de él en su espacio de trabajo, pudiendo ambos desarrollar en paralelo sobre un mismo fichero e integrando los cambios privados al finalizar la tarea. Se puede citar a Subversion, Plastic como herramientas que soportan este modelo. Aunque se considera necesario aclarar que las versiones recientes de Subversion permiten, si se desea, seguir el modelo Bloqueo-Modificación-Desbloqueo, todo depende de las opciones que se seleccionen a la hora de su utilización.

1.6.1 Caracterización de herramientas

Durante el desarrollo del capítulo se ha venido presentando un conjunto de herramientas que apoyan el proceso de gestión de configuración, estas han sido puestas como ejemplos en diferentes escenarios de acuerdo a sus características y potencialidades. Como cierre de este epígrafe se muestra a continuación aspectos relacionados con tres herramientas que son mundialmente utilizadas por las novedades y las prestaciones que ofrecen dando. Esta descripción permite a las autoras seleccionar en el capítulo 2 la herramienta más factible al proceso de GCS del proyecto R&N.

IBM Rational ClearCase

Es una herramienta que en su integración con IBM Rational ClearQuest logra una solución completa de gestión de configuración de software con un control integrado de versiones, una gestión del espacio de trabajo automatizado, una gestión de línea base, gestión del proceso de compilación y liberaciones y un seguimiento fiable y flexible de defectos y solicitudes de cambio.

Características

1. Permite el desarrollo en paralelo, pues brinda un excelente soporte al manejo de múltiples ramas.
2. Está disponible tanto para sistemas operativos propietarios como open-source
3. Brinda integración con el resto de las herramientas de la Suite de Rational, con Microsoft Visual Studio .NET y con Eclipse**.**
4. Automatiza el proceso de integración continua, reduciendo además los ciclos de compilación y obtención de productos entregables.
5. Posee un precio elevado, alrededor de $4,021.88 por licencia.
6. Es ideal para equipos de desarrollo entre medianos y grandes pero su configuración suele ser compleja.
7. Con Rational ClearCase MultiSite brinda soporte a equipos geográficamente distribuidos.
8. Es muy útil para productos con múltiples versiones comercializables

Plastic SCM

Plastic SCM es una herramienta española relativamente nueva que salió al mercado en noviembre de 2006. Puede ser empleada únicamente como controlador de versiones aunque sus desarrolladores están trabajando en un sistema de flujo de trabajo que podrá ser utilizado para la gestión completa del ciclo de vida del desarrollo integrando la gestión de tareas, entregas, pruebas, gestión de los resultados, así como la trazabilidad entre todos ellos.

Características

1. Como rasgo distintivo incorpora una visualización en 3D de un árbol de versiones que muestra la vida de un elemento concreto dentro del repositorio, ya sea un fichero o un directorio.
2. Una de sus prioridades es permitir el desarrollo en paralelo y esto lo consigue gracias al robusto modelo de ramas que implementa.
3. Posee soporte para Windows, Linux y Solaris y se integra con Eclipse, para entornos Java/J2EE y Visual Studio.Net.
4. Permite versionado completo de directorios por lo que los maneja tan bien como a los ficheros.

5. A pesar de ser también una herramienta propietaria su precio por licencia es mucho menor que el Rational Clear Case siendo de 595 euros la licencia por desarrollador.
6. Algo muy meritorio de esta herramienta y de su grupo de desarrollo es el soporte y la atención que brindan ante cualquier duda o inquietud relacionada con el funcionamiento de la misma.
7. Su utilización en desarrollos geográficamente distribuido es necesario realizarlo auxiliándose de una red privada virtual, VPN[1] por sus siglas en inglés.
8. Puede ser utilizada en equipos de trabajo de cualquier tamaño.
9. Puede ser de mucha utilidad en productos con múltiples versiones comercializables.

Subversion

Es un controlador de versiones centralizado y multiplataforma que fue diseñado para reemplazar al antiguo Concurrent Versions System (CVS). Esta es una herramienta libre (open-source) y permite como otros controladores de versiones saber quién y en que revisión se escribió cada línea de código.

Características

1. Usa el modelo Copiar-Modificar-Mezclar aunque versiones más actuales permiten seguir si el usuario lo desea el modelo copiar-bloquear-modificar.
2. Brinda la opción de trabajo con múltiples ramas aunque no de manera tan eficiente como las herramientas descritas anteriormente.
3. Logra muy buena integración con diferentes IDE de desarrollo, por ejemplo: con eclipse a través de su cliente Subclipse y también con Visual Studio.Net con su cliente Ank, además tiene al TortoiseSVN que es su cliente para su integración con Windows.
4. El soporte a esta herramienta depende de la comunidad desarrolladora.
5. Puede ser utilizadas para equipos de cualquier tamaño
6. Su utilización en desarrollos geográficamente distribuido es necesario realizarlo auxiliándose de una VPN.

[1] VPN: Virtual Private Networks.

1.7 Análisis Crítico del proceso de GCS de R&N

Durante el desarrollo de los módulos iniciales de R&N no se llevó a cabo un proceso continuo y eficiente de gestión de configuración que estuviera acorde al tamaño y complejidad de este proyecto.

En un primer instante se identificaron los elementos de configuración de software (ECS) y fueron descritos en un documento que no fue nunca actualizado para incluir nuevos elementos o sustituir los carentes de funcionalidad u obsoletos. Ante las numerosas solicitudes de cambios y no conformidades por parte de los clientes, el Equipo de Calidad Interna[1] o Calisoft[2], no fueron revisados dichos ECS para analizar la profundidad del cambio atendiendo a las relaciones de dependencia entre los ECS, presentándose situaciones en las que un cambio afectaba a varios módulos y no todos eran informados del cambio inmediatamente, pudiendo existir un lapso de tiempo de hasta una semana hasta que se percataran que el cambio afectaba también a otros módulos. En el proyecto no estaba definida una línea base ni los procedimientos para introducir o tomar un determinado ECS de la misma.

Los mayores esfuerzos estuvieron dedicados al control de versiones del código y de la documentación. El equipo de desarrollo divido en módulos poseía cada un repositorio habilitado en dos herramientas de control de versiones: los módulos de Inmobiliario y Mercantil desarrollados en Visual Estudio 2003 utilizaron como herramienta el Visual Source Safe, producto a que ambos son de Microsoft. El módulo de Administración Contable empleaba Subversion con su cliente ANK para Visual Estudio y para la documentación y los script de la base datos utilizaban el cliente de Windows Torstoise, Servicio Autónomo empleaba en un primer inicio CVS pero luego migraron a Subversion utilizando como cliente Subclipse producto que éste módulo es desarrollado utilizando Eclipse como entorno de desarrollo. La utilización de varias herramientas de control de versiones ocasionó que ante el uso de componentes comunes entre diferentes módulos la transferencia de uno a otro no se hiciera a través del controlador de código sino que tuvieran que buscar alternativas manuales, pudiendo ser desde la ubicación del componente en una máquina común hasta el envío por correo.

[1] Equipo de Calidad Interna, es un concepto utilizado en los proyectos de software de la Universidad de Ciencias Informáticas, que consiste en un grupo de personas que forman parte del propio proyecto y su actividad principal es garantizar la calidad del producto en desarrollo.

[2] Calisof es un equipo establecido en la UCI, que se encarga de supervisar la calidad en los productos de software que pretenden ser liberados.

Este proyecto tiene la particularidad que se desarrolla tanto en Cuba como en Venezuela, la forma de compartir la documentación y el código es mediante FTP[1], las no conformidades, los entregables del producto y la documentación eran colocados en un FTP y descargados por ambas partes, este proceso debido a problemas de conectividad es bastante incómodo y puede demorar por lo que hay pérdida de tiempo.

Ante la llegada de no conformidades por parte de calidad interna, del grupo Calisoft o de los venezolanos estas eran asignadas a los jefes de módulo quienes se encargaban con su equipo de darle respuesta, posteriormente una vez bien avanzado el proyecto se comenzó a analizar estas no conformidades con el jefe de proyecto y algunos integrantes de los módulos, este pequeño equipo, pudiera pensarse en un símil con el comité de gestión de cambios aunque en verdad este no estaba establecido en el proyecto.

Como resultado final se puede evaluar que la gestión de configuración en R&N no fue un proceso continuo sino que más bien se fue estructurando sobre la marcha a medida de las exigencias del desarrollo. Haciendo valer su importancia en momentos críticos del proyecto en los que se hizo inminente la puesta en práctica de algunas de las actividades que comprenden esta disciplina. Partiendo de la experiencia acumulada y del estudio realizado se considera de vital importancia la propuesta de una estrategia que organice el proceso de GCS para futuros desarrollos de este proyecto.

1.8 Conclusiones del Capítulo

En el presente Capítulo se realizó un estudio del estado del arte de la Gestión de Configuración de Software a nivel internacional, también se realizó un estudio profundo en el proyecto R&N de la Universidad de la Ciencias Informáticas, por ser el marco donde se desarrolla la Investigación, dándole cumplimiento, de esta forma al objetivo número uno de este trabajo, trazados al inicio de la investigación. El estudio realizado arrojó las siguientes conclusiones:

1. La Gestión de Configuración del Software es valorada de gran importancia dentro del proceso de desarrollo por varios autores y consecuentemente es considerada en las principales normas y modelos de calidad de software a nivel mundial como son ISO, IEE, CMMI.

[1] FTP: Protocolo de Transferencia de Ficheros.

2. La Gestión de Configuración se encarga de mantener la integridad de los productos que se obtienen a lo largo del desarrollo de los sistemas de información, garantizando que no se realizan cambios incontrolados y que todos los participantes en el desarrollo del sistema disponen de la versión adecuada de los productos que manejan.
3. El conjunto de actividades presentes en la GCS permiten controlar el sistema como producto global, gestionar las modificaciones que sobre él se generen, obtener informes sobre el estado del desarrollo del software, así como auditar la ejecución de los procesos, lo que se traduce en un aumento de calidad del producto, de la satisfacción del cliente y, en consecuencia, de mejora de la organización.
4. Las herramientas que brindan un mayor soporte a las actividades de GCS son herramientas propietarias pero estas tienen un elevando costo.
5. Subversion es una herramienta de control de versiones de la comunidad de software libre que brinda potencialidades a la altura de las herramientas propietarias.

Estrategia de referencia para la GCS en R&N 2

2.1 Introducción

En el presente capítulo se muestra una Estrategia de referencia de Gestión de Configuración (ErGCS), para ser aplicada en el Proyecto de Modernización de los Registros y Notarías de la República Bolivariana de Venezuela (R&N). La misma es obtenida como resultado de tres elementos fundamentales:

1. Las particularidades presentes en el equipo de desarrollo de R&N.
2. Las deficiencias detectadas en el proceso de GCS en R&N.
3. La investigación realizada en el Capítulo 1, teniendo en cuenta los aportes que sobre el tema han emitido varios autores y organizaciones (Estrada, 2003) (IEEE, 1998) (Antonio, 2001) (Martínez, 2006) (Pressman, 2005) (Rancán, 2003) .

En la estrategia se siguen algunos de los principios de RUP, por ser la metodología que actualmente guía el proceso de desarrollo del proyecto R&N, además se tiene en cuenta lo que sobre el tema plantea el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE). En la ErGCS se proponen actividades que, luego del estudio del estado del arte del tema, las autoras consideran deben estar presentes en el proceso de GCS del proyecto. Además se muestran cada uno de los procedimientos a seguir en cada una de dichas actividades, así como la asignación de un rol responsable de ejecutarla. También se lleva a cabo la selección de herramientas que soporten un conjunto de actividades y procedimientos presentes en dicha propuesta.

2.2 Estructura del equipo de GCS

La descentralización excesiva en el proceso de GCS puede generar descoordinación en el mismo, llevándolo al caos, es por ello que una de las proposiciones iniciales de la ErGCS es la creación de un equipo de GCS bien estructurado. Las autoras de este trabajo tuvieron en cuenta este primer paso, debido a que permite centralizar el proceso, asignar responsabilidades bien definidas, organizar el trabajo de los involucrados, así como facilitar un mejor control sobre los mismos. Todo lo anterior influye de forma positiva a la hora de aumentar el nivel de productividad y por ende calidad en el proceso de GCS.

Dicho equipo tendrá la siguiente estructura:

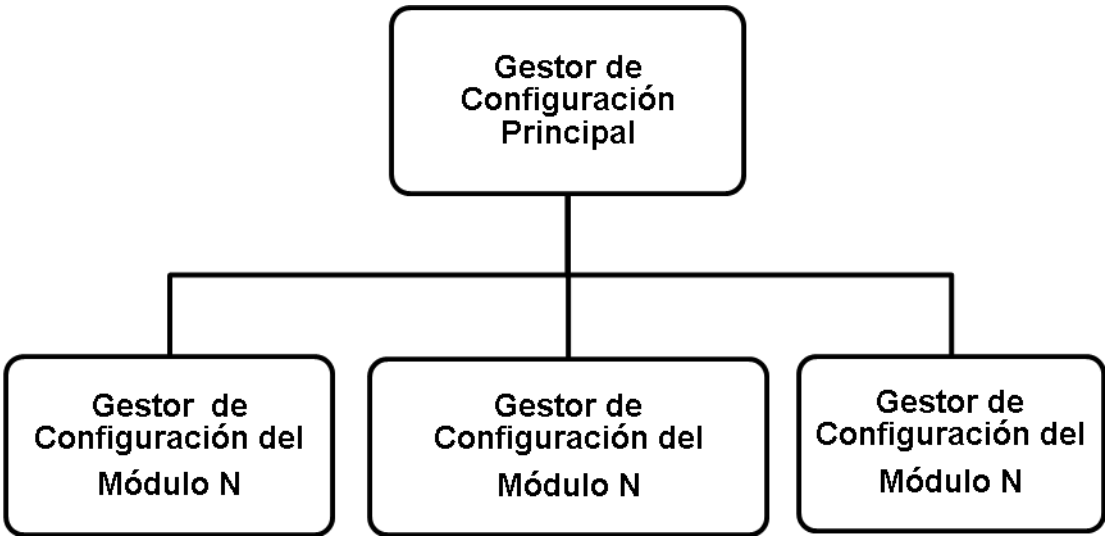


Figura 2. 1 Estructura del equipo de GCS.

A continuación se muestran las funciones asignadas a cada rol dentro del equipo de GCS:

Gestor de Configuración Principal:

- Controla y dirige el proceso de GCS a nivel de proyecto.
- Define políticas de acceso al repositorio del proyecto.
- Integra todos los elementos de las líneas base a nivel de módulo en la línea base del proyecto.

- Ejecuta controles internos al proceso de GCS, con el objetivo de analizar las deficiencias y erradicarlas.
- Coordina y controla las actividades de GCS no solo en Cuba, sino también las que se realizan por parte del equipo de desarrollo en Venezuela.
- Convoca y dirige las reuniones del Comité de Control de Cambio.
- Es el encargado junto al Líder del Proyecto, de tomar decisiones futuras respecto a la metodología de trabajo usada en la GCS.

Gestor de Configuración del Módulo N:

- Controla que el proceso de GCS se lleve a cabo correctamente por todos los miembros del módulo.
- Gestiona la evolución de la línea base a nivel de módulo y verifica que esta sea actualizada en el período de tiempo establecido.
- Verifica que los desarrolladores del módulo estén utilizando los componentes según la versión actual del producto.
- Encargado de solicitar al Gestor de Configuración Principal, en caso que sea necesario, alguna versión anterior del software para analizar en su módulo.

2.3 Herramientas propuestas

Uno de los elementos clave en la propuesta es la utilización de herramientas que apoyen y automaticen diferentes actividades de la GCS. Específicamente se propone para la actividad de Control de Versiones (sección 2.4.2), la instalación de Subversion sobre un servidor Apache, para de esta forma poder disfrutar de la robustez, autenticación HTTP[1], compresión de datos, cifrado, etc. que ofrece Apache.

Se considera que Subversion es la herramienta que debería regir el control de versiones en el proyecto R&N, atendiendo que:

[1] HTTP: Protocolo de Transferencia de Hipertexto.

- Es una herramienta de la comunidad de software libre, que brinda prestaciones equiparables a las propietarias, a pesar del elevado costo de estas últimas en el mercado.
- La política de la facultad y la universidad favorecen la migración a software libre, por lo que éste es un buen paso siguiendo esta nueva línea.
- Logra muy buena integración con los dos entornos de desarrollo actualmente utilizados en la construcción del producto: Visual Studio.NET 2003 y Eclipse.
- Su soporte multiplataforma permite continuar su uso en una futura migración de plataforma o ambiente de desarrollo.
- Facilita la tarea de realizar Backups[1] al repositorio del proyecto hacia los servidores Gforge de la universidad ya que estos están basados en Subversion (Vázquez Acosta).

La propuesta incluye además el uso combinado del Subversion con el TRAC pues este último permite mediante su sistema de tickets la gestión de las tareas en el proyecto, ya sean errores a arreglar, mejoras pendientes, entre otras. Además sus capacidades de Wiki permiten crear dichos reportes de tickets totalmente personalizados, cambiar la apariencia del sitio o agregar páginas informativas del proyecto.

Partiendo de la particularidad del proyecto R&N, que su desarrollo se lleva a cabo en dos lugares distantes geográficamente, se propone además, la gestión y uso de una Red Privada Virtual (VPN) para la utilización de un repositorio único mediante una conexión segura entre los dos países.

2.3.1 Estructura del repositorio

Una vez instalado el servidor de Subversion será creado un repositorio sobre el cual se estructurará el expediente del proyecto. Formando parte de esta estrategia se realiza una propuesta de cómo estructurar dicho expediente en aras de tener una mejor organización. Para ello se parte de la propuesta del departamento de calidad en la universidad pero adaptándolo a las características y necesidades del proyecto.

El repositorio puede estar estructurado mediante una carpeta con el identificador del proyecto y dentro un subsistema de carpetas principales:

[1] El término Backups puede ser traducido como la creación de copias de seguridad en una ubicación distinta a la original para, en caso de pérdida o deterioro de archivos, tener la opción de recuperarlos.

- Desarrollo Técnico.
- Sistema de Calidad.
- Seguridad Informática.
- Gestión de Proyecto.
- Gestión de Configuración.
- Línea Base Proyecto

La carpeta de desarrollo técnico estará interiormente divida por módulos y contendrá además dos directorios: uno para los elementos comunes, y otro para almacenar las liberaciones y diferentes versiones del producto. Cada módulo estará dividido a su vez por flujos de trabajo. En el directorio correspondiente al flujo de trabajo de implementación es específicamente donde se almacenará el código organizado en tres directorios principales: Trunk, Branch, Tags (sección 2.3.3.1).

Esta organización posibilita que ante la necesidad de trabajar en un nuevo módulo su inclusión en el repositorio sea tan sencilla como adicionar una carpeta dentro del directorio de desarrollo técnico con el identificador del nuevo módulo en cuestión.

2.3.1.1 Estructura del expediente de GCS

El expediente de GCS contendrá los procedimientos, plantillas y artefactos que se generen a lo largo de todo el desarrollo del ciclo de vida del software. Para una mejor organización se propone que su estructura quede definida mediante el siguiente sistema de carpetas:

- **ECS:** Almacena, divididos por módulos y por categorías, todos los ECS que se seleccionen en el proyecto.
- **Solicitudes de Cambio:** Contiene todas las solicitudes de cambio generadas, ya sean internas o externas al equipo de desarrollo. Estas se encuentran divididos por módulos y separadas por las diferentes etapas o fechas en las que son producidas.
- **Informes de Estado:** Es contenedora de los informes generados durante el desarrollo del proyecto. Estos están también organizados por módulos.

- **Control Interno de la Configuración:** Contiene los informes o reportes generados a partir de las revisiones que el gestor principal realiza al proceso de GCS, midiendo así el desempeño de los gestores de cada módulo.
- **Plantillas de Desarrollo Técnico:** Contiene todas las plantillas a utilizar en los diferentes flujos de trabajo.
- **Procedimientos:** Contiene los procedimientos por los que se regirá el proceso de GCS en el proyecto.
- **Biblioteca de ECS:** Este es un directorio privado que contiene los ECS que son retirados de la línea base. A estos directorios solo tendrán acceso un número limitado de personas, por ejemplo el Líder del Proyecto y el Arquitecto Principal.

2.3.1.2 Política de Acceso al Repositorio.

Una vez estructurado el repositorio es primordial el establecimiento de políticas de acceso a seguir por integrantes del proyecto. En este sentido se propone que el acceso a los diferentes directorios se corresponda según las responsabilidades del rol que desempeña cada desarrollador en un momento determinado. El Gestor Principal de la Configuración junto al Líder de Proyecto es el encargado de establecer la política de seguridad y acceso al repositorio. El proceso se puede describir mediante los siguientes pasos:

- Crear dos ficheros, uno que contendrá la clave correspondiente a cada usuario y otro para definir los permisos a los directorios del repositorio.
- Crear usuarios que interactuarán con el Subversion con sus respectivas claves de acceso.
- Definir grupos de usuarios en el Subversion, por ejemplo: administradores, programador de presentación, programador de datos, arquitectos, etc.
- Asociar a un usuario el grupo al que pertenece.
- Asignar los permisos correspondientes a cada grupo de usuarios.

2.3.2 Metodología de trabajo con la Herramienta

2.3.2.1 Propuesta de Patrón

Al comenzar a utilizar un controlador de versiones es importante que se defina la metodología a seguir en el uso de la herramienta. En esta estrategia se propone utilizar el patrón de rama por tareas, delimitando el término tareas, a la creación de los distintos subsistemas del proyecto y a la integración de cada uno de ellos. Este patrón será utilizado del siguiente modo:

- Sobre el directorio Trunk se almacenará la línea principal de desarrollo. A partir de él se creará una rama para cada subsistema a desarrollar, garantizando de esta forma la integridad de la línea principal ya que el trabajo es realizado sobre cada una de las ramas y no sobre el directorio principal.
- El Directorio Branch contendrá las ramificaciones creadas para cada subsistema y contendrá además una rama de integración, donde, como su nombre lo indica, se integrarán los diferentes subsistemas creados durante una iteración. Una vez terminada dicha iteración y que el equipo de calidad interna certifique que el contenido de la rama es estable y libre de defectos se procederá a integrarlo con la línea principal de desarrollo.
- En el directorio Tags se almacenarán etiquetadas las versiones estables que se obtengan a partir de la estabilización de la rama de integración y su certificación por calidad.

Siguiendo el uso de este patrón de la forma descrita anteriormente se garantiza una total integridad de la línea principal de desarrollo durante todo el ciclo de vida de creación del producto. Además posibilita el desarrollo de nuevos subsistemas a partir de un punto estable y permite tener en todo momento una versión estable del producto para presentar ante un cliente o evento que se presente.

2.3.2.2 Estandarizando el trabajo con las herramientas propuestas

Para el trabajo con la herramienta se proponen algunas reglas que pueden ser de gran utilidad ya que facilitan y estandarizan el uso de la misma:

1. Debe ser una premisa no corromper la línea principal de desarrollo pues de esta forma se ve afectado automáticamente todo el equipo de desarrollo.

2. Asociar obligatoriamente cada envío al repositorio con un comentario, ser preciso y exhaustivo en los mismos, esto ayuda a documentar el proyecto y permite a los compañeros de equipo claridad a la hora de reconocer los cambios realizados.
3. Actualizar la copia de trabajo frecuentemente para no invertir posteriormente demasiado tiempo en la resolución de los cambios realizados por cada desarrollador que entren en conflictos.
4. Cerrar los tickets al finalizar las tareas. Para evitar olvidar la realización de esta acción se propone crear un gancho (hooks) para Subversion que cierre automáticamente un ticket abierto que sea especificado en el comentario del commit[1]. Por ejemplo si al realizar el envío al repositorio, es puesto como comentario *"Cerrado #342"*, el ticket #342 pasará a cerrarse automáticamente en el Trac.
5. Realizar Backups del repositorio periódicamente.

La ErGCS, propuesta por las autoras, brinda un mecanismo para controlar el cumplimiento de las reglas enunciadas anteriormente a través del Control Interno de la Configuración (sección 2.4.4).

2.4 Actividades presentes en ErGCS

Varios autores (Antonio, 2001) (IEEE, 1998) (Pressman, 2005) (Rancán, 2003) plantean diferentes actividades presentes en la GCS, e incluso les atribuyen a algunas, diferentes nomenclaturas. Según la investigación realizada, las autoras proponen como actividades presentes en ErGCS las siguientes:

- Identificación de la Configuración del Software.
- Gestión de Cambio de la Configuración.
- Control de Versiones de la Configuración del Software.
- Generación de Informes de Estado.
- Control Interno al proceso de GCS.

Estas actividades fueron seleccionadas, primeramente, por la importancia que tiene su cumplimiento en el proceso de GCS. Por otro lado, fue necesario tener en cuenta que el proyecto en el cual se van a aplicar

[1] Commit: comando de la herramienta SVN que se utiliza para subir los cambios al repositorio.

tiene la característica de contar con un equipo grande de desarrollo y además se encuentra distribuido geográficamente, por lo que se le atribuye gran importancia a la organización y calidad que se obtendrá en el proceso de desarrollo, como resultado de la correcta aplicación de estas actividades.

2.4.1 Procedimiento para la Identificación de la Configuración (IC)

El proceso de identificación de la configuración incluye entre sus actividades la gestión de los ECS, el establecimiento de las líneas base y las relaciones entre los ECS. A continuación se describirán los procedimientos a seguir en cada una de ellas.

2.4.1.1 Gestión de los ECS

Mediante esta actividad se identifican, almacenan y notifican los elementos de los cuales se desea tener control de su evolución en el proceso de desarrollo del software. En el caso del proyecto R&N, esta tarea es de vital importancia dentro de la GCS, debido al gran flujo de información existente y al crecimiento elevado de la misma en el tiempo, lo cual se traduce en un aumento continuo del número de ECS. Para que el proceso identificación de los ECS fluya exitosamente se plantean las siguientes acciones:

1. Identificar los ECS.

Las autoras consideran que la identificación de los ECS debe ser llevada a cabo por personas que posean gran conocimiento sobre la arquitectura del sistema, debido a que tanto la omisión de algún elemento significativo como la selección de un número elevado de los mismos, pueden producir males mayores en tiempos muy cortos. Por lo anteriormente expuesto, se propone que en el proceso de selección participen fundamentalmente los siguientes roles:

- Arquitecto Principal.
- Jefe de módulo.
- Gestor de Configuración Principal.

2. Clasificar los ECS.

Se realiza con el objetivo de organizar los ECS y facilitar de esta manera su localización en el repositorio del proyecto. Se propone que todos los ECS que pertenezcan a una categoría sean almacenados en un mismo documento.

Para ello se establecieron las siguientes categorías:

- Base de Datos.
- Componentes.
- Documentación
- Ejecutables.
- Aplicaciones.

3. Almacenar los ECS.

Una vez que hayan sido correctamente identificados, se procede a almacenar toda la información referente a estos elementos en el repositorio del proyecto, con el objetivo de controlar su evolución a lo largo de su ciclo de vida en el desarrollo del software. Inicialmente en el proyecto R&N se propuso una planilla que contiene la siguiente información del ECS para ser almacenada:

- Número o código del ECS
- Versión
- Nombre del ECS
- Descripción del ECS
- Autor/es del ECS
- Datos de los autores
- Fecha de creación
- Listado de elementos dependientes del ECS
- Listado de los módulos que los usan
- Tipo de ECS
- Localizaciones

En el caso del número o código del ECS se plantea el siguiente formato para su formación: IdentificadorProyecto-TipoEcsMódulo-NúmeroIncrementado.

A continuación se muestran posibles datos que pueden adquirir estas variables:

- Identificador del Proyecto→ R&N
- Tipo de ECS→ Base de Datos (BD), Código (Cod), Documentación(Doc), Ejecutables (Eje),
- Módulo→ Inmobiliario (Imb), Mercantil (Mer), Servicio Autónomo (SA).
- Número Incrementado→ 01, 02, etc.

Sin embargo las autoras consideran necesario adicionar un campo más a dicha planilla y es la clasificación que posee el elemento según la importancia que posee el mismo en la arquitectura del software en Alta, Media o Baja. Esta decisión está sustentada bajo la importancia que tiene este atributo en el elemento a la hora de evaluar una solicitud de cambio sobre el mismo.

4. Notificar los ECS identificados

El proceso iniciado con la identificación de los ECS finaliza una vez que se halla informado a todos los desarrolladores del proyecto de los elementos seleccionados y la localización de los documentos que contienen los datos de cada uno de ellos. Esta documentación quedará expuesta a futuras actualizaciones, ya sea para adicionar nuevos elementos, modificar el estado de los mismos o eliminar los que no se utilicen o carezcan de validez. El encargado de actualizar este documento será el Gestor de Configuración Principal.

2.4.1.2 Establecimiento de la Línea Base

Como se planteara en el Capítulo 1, la línea base puede ser establecida en diferentes hitos del desarrollo de software y pueden ser creadas ya sea para un sistema completo o para subsistemas del mismo. En el caso específico del proyecto R&N atendiendo a que su desarrollo está dividido por módulos y a la dimensión de cada uno de ellos se propone establecer líneas base a:

- Nivel de módulos.
- Nivel de proyecto.

De esta forma se brinda cierto grado de autonomía a la hora de gestionar los cambios sobre la línea base, específicamente en las que se construyen a nivel de módulo, debido a que las solicitudes de cambios sobre ECS que solamente tienen impacto dentro del ámbito del mismo pueden ser analizadas por los integrantes y el jefe de módulo sin necesidad de elevarlo a una comisión de gestión de cambios a nivel de proyecto. Aunque es importante aclarar que si este ECS ya se encuentra integrado en la línea base del

proyecto, entonces su análisis debe efectuarse mediante un proceso formal de cambios (sección 2.4.3), es por ello que se plantea al inicio, que esta estructura provee cierto grado de autonomía en el proceso.

2.4.1.2.1 Línea base a nivel de proyecto

Esta actividad es coordinada por el Líder del Proyecto, el Arquitecto Principal y llevada a la práctica por el Gestor Principal de la Configuración. Para su establecimiento se considera necesario realizar los siguientes pasos:

1. Establecer la línea base a nivel de proyecto, al alcanzar los objetivos de cada una de las fases definidas en la metodología RUP.
2. Determinar qué elementos de configuración del software conformarán la línea base en cada hito:

   Para este paso las autoras proponen los posibles artefactos que pueden integrar las distintas líneas base:

   - Los documentos de planificación del proyecto, los artefactos generados en RUP para el flujo de trabajo de modelación del negocio y en el de requisitos, son ECS que pueden integrar la línea base al finalizar la fase de Inicio.
   - Al finalizar la fase de Elaboración la línea base contará con la actualización de los ECS identificados en la fase de Inicio e incorporará los artefactos que se generen en esta fase tales como: modelos de análisis, modelos del diseño, modelos de datos y de despliegue de cada módulo así como la documentación de la arquitectura.
   - En cada iteración que se finalice en la fase de construcción, se propone incluir en la línea base la versión liberada de cada módulo todas los artefactos de la disciplina de pruebas que se obtengan, así como los informes de estado y los artefactos de soporte que hayan sido generados hasta ese momento.
   - En la fase de transición se parte de una línea base que esté lo suficiente madura como para ser desplegada en el ambiente final del usuario. En esta fase se incluirá el producto final junto a los materiales de soporte al usuario (manual de usuario y materiales de entrenamiento).

2.4.1.2.2 Línea base a nivel de módulo

Cada módulo a partir de sus planificaciones semanales debe establecer hitos menores en los cuales crear su propia línea base a partir de los diferentes ECS que vayan liberando. Estos hitos se establecerán con

mayor frecuencia que los que se establezcan para la línea base a nivel de proyecto, y serán definidos por el Jefe de Módulo, y el arquitecto de cada módulo.

2.4.1.2.3 Integración de las líneas base de cada módulo en la línea base principal

En cada uno de los hitos anteriormente mencionados el Gestor de Configuración Principal conformará la línea base del proyecto realizando una copia referencial de la línea base de cada uno de los módulos, mientras que estos no estén integrados, o la versiones liberada del producto una vez que se logre la integración.

Las líneas bases, tanto la del proyecto como la de los módulos, serán almacenadas en carpetas asignadas dentro del repositorio y serán además etiquetadas. El modo de etiquetar puede variar aunque pudiera seguirse la nomenclatura: **LBI#F**

- LB→ Significa Línea Base.
- I → Identificador (P: para el proyecto y en el caso de los módulos se toma el identificador que se haya decidido en el esquema de identificación de los ECS).
- # → Número de la línea base (siguiendo el orden de creación).
- F → Fase en que es establecida.

2.4.1.2.4 Modificaciones sobre la línea base

Siempre que sea necesario realizar una modificación sobre un elemento que integre la línea base se debe seguir un proceso formal de cambios. En consecuencia a esta evolución de la línea base, el Gestor de Configuración Principal retirará el ECS sujeto a cambio e integrará el mismo actualizado, luego que este haya sido revisado y aprobado. El elemento que se retira de la línea base no es eliminado del repositorio del proyecto sino que será almacenado en el directorio privado: "*Biblioteca de ECS* "(sección 2.3.1.1). En el caso que sea necesario consultar dicho ECS, se debe:

1. Solicitar al Arquitecto y/o al Gestor de Configuración Principal la versión del ECS que necesita y explicar la necesidad de su uso.
2. Si la petición es aprobada entonces se obtendrá una copia del ECS del directorio privado y se le hará entrega al desarrollador que la solicitó.
3. El desarrollador, al finalizar, debe eliminar de su espacio de trabajo el ECS.

2.4.1.3 Relaciones entre los ECS

De las relaciones definidas en (Antonio, 2001), las que mayormente se establecen entre los ECS del proyecto de R&N son: Dependencia y Composición. Para facilitar el manejo de las mismas las autoras proponen la utilización de una base de datos que se desarrolló en Access. La misma muestra una lista de ECS que al ser seleccionados muestran en otras dos estructuras los elementos que se relacionan con dicho elemento ya sea por dependencia o composición. La necesidad de la implementación de esta herramienta surgió debido a que:

- Las herramientas de apoyo utilizadas no ofrecen ninguna funcionalidad que automaticen esta actividad.
- Resulta engorroso, ante un número elevado de ECS, plasmar todas las relaciones en un documento Word.
- La base de datos consta de tres tablas:

Elementos: Almacena los elementos de configuración.

Dependencia: Contiene las dependencias entre dos elementos.

Composición: Guarda las relaciones de composición entre dos elementos.

Figura 2. 2 Interfaz de aplicación que almacena las relaciones entre los ECS.

2.4.2 Control de versiones

El control de versiones no solo incluye el uso de herramientas, también combina junto a ella procedimientos que regulan como llevar a cabo esta actividad. A continuación se definen una serie de pasos para el control de versiones en el proyecto R&N, basado en el procedimiento que se utiliza en la facultad y en el descrito en (Estrada, 2003):

Responsables:

- Líder del Proyecto.
- Equipo de Gestión de Configuración
- Posibles estados de las versiones de los componentes:

Desarrollo: Versión disponible para los desarrolladores**.**

Estable: Versión que forma parte de la línea base.

Liberación: Versión entregada a un cliente.

Prueba: Versión entregada al equipo de calidad.

Actividades en el procedimiento

1. Se recibe notificación del problema. Si es el problema incide sobre componentes en estado estable o liberación se pasa al Proceso de Gestión de Cambios (Ver sección 2.4.3).
2. Se bloquea en el repositorio la última versión del ECS y se le entrega al desarrollador correspondiente para su modificación.
3. El desarrollador modifica el ECS.
4. El ECS es probado por el equipo de calidad interna. En caso de que elemento no cumpla los requisitos establecidos ir al paso 3.
5. Se actualiza la nueva versión del ECS, en el repositorio, al finalizar la modificación y se desbloquea.
6. Se informa a todo el equipo de desarrollo de la nueva versión del ECS
7. La versión antigua se deposita en la biblioteca de software del proyecto.

2.4.2.1 Reglas de versionado del producto

Los números de versión están compuestos por tres números X.Y.Z Cada número tiene el siguiente significado:

X: Cambios Mayores en la aplicación, lo cual puede estar dado por un gran número de nuevas funcionalidades adicionadas, la incorporación de un módulo importante, por un cambio notable de arquitectura o de tecnología (lenguaje de desarrollo, motor de base de datos).

Y: Cambios Menores en la aplicación, lo cual puede estar dado por una nueva funcionalidad adicionada en la aplicación. Cuando se estrena una versión mayor se deja la versión menor a cero pero aún así se incluye, de modo que la siguiente versión mayor sería la 2.0.

Z: Indica modificaciones que se reparan de forma rápida. Cuando se estrena una versión menor, es decir, cuando la segunda versión menor es igual a cero; suele omitirse. Cuando se está en desarrollo puede seguirse la misma nomenclatura pero adicionándole al lado la revisión de Subversion en la que se encuentra, quedaría como en el siguiente ejemplo: versión 1.0-svn 537.

2.4.3 Gestión de Cambio de la Configuración (GCC)

Inicialmente se plantearon como objetivos principales para el proceso GCC, ofrecer instrumentos que permitan:

- Solicitar cambios sobre los ECS del producto
- Analizar y valorar el impacto que motivará la implementación del cambio sobre el producto y sobre la organización de desarrollo.
- Aprobar o rechazar la solicitud de cambio.
- Priorizar las solicitudes de cambio.
- Controlar la ejecución del cambio solicitado.
- Certificar que el cambio realizado ha sido correctamente implementado.

Partiendo de las premisas planteadas, se proponen un conjunto de pasos o actividades organizadas con el objetivo de establecer un procedimiento capaz de llevar a cabo de forma controlada el proceso de gestión de cambios en el proyecto R&N. Se tiene en cuenta para dicha propuesta lo que sobre el tema plantean varios autores como: (Antonio, 2001) (Pressman, 2005) (Rancán, 2003).

Procedimiento para la Gestión de Cambios de la Configuración

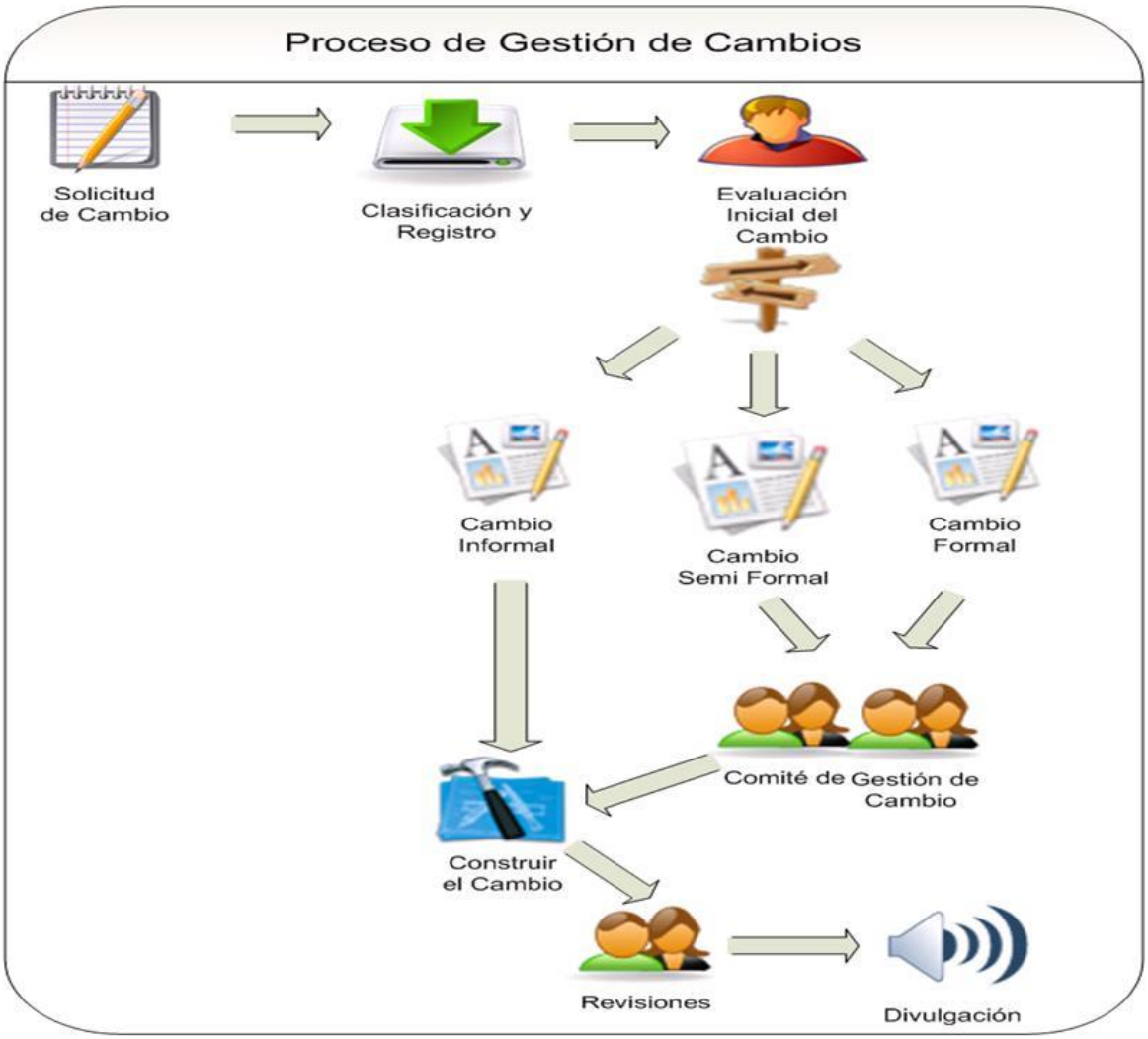


Figura 2. 3 Proceso de Gestión de Cambios.

El proceso de Gestión de Cambio comienza cuando se reciben no conformidades, ya sea por parte de Calidad Interna, Calisoft o el cliente. Una vez recibidas estos documentos de no conformidades se proponen realizar los siguientes pasos:

1. Verificar la claridad y el formato de la No conformidad.
2. Decidir si procede o no la no conformidad.
3. En caso que la no conformidad proceda, la misma pasa a ser una solicitud de cambios.

Con tal propósito, se decidió utilizar la Planilla de Solicitud de Cambio (PSC), definida por el equipo de Calisoft, con el objetivo de propiciar un formato estándar para la petición de cambios y además para tener un control estricto de los mismos por parte de la dirección del proyecto, contribuyendo así a la toma de decisiones en la organización.

Se propone clasificar las solicitudes de cambio acuerdo a su origen en: Solicitudes Externas o Solicitudes Internas. Las externas son las que se llevan a cabo por parte de los clientes y también por el departamento de calidad central (Calisoft). Mientras que las internas son las propuestas por el equipo de desarrollo del proyecto y como organización de calidad dentro del mismo, el equipo de Calidad Interna. De esta forma se puede llevar un control estadístico sobre las solicitudes emitidas por el cliente y las que surgen por parte de equipos involucrados con la calidad del proyecto como Calisoft y Calidad Interna.

A continuación se muestran algunos de los campos presentes en esta planilla:

- Fecha de la solicitud
- Nombre del solicitante
- Origen de la Solicitud del cambio (Externa o Interna).
- Causa de la solicitud del cambio (Error, Defecto, Mejora,)
- Resumen del cambio

En el Anexo 2 se encuentran más detalles de la Planilla de Solicitud de Cambios.

En cuanto al impacto del cambio, las solicitudes pueden ser clasificadas en:

- **Solicitudes de cambio informal:** Se aplica antes de que un ECS se convierta en una línea base.
- **Solicitudes de cambio semiformal:** Se efectúa una vez que un ECS se convierte en una línea base. Este cambio puede tener un impacto local o global, según este impacto, será la formalidad con la que se aplique el cambio.
- **Solicitud de cambio formal:** Se adopta cuando se distribuye el producto software a los clientes (release).

Se decidió darle estas clasificaciones a las solicitudes de cambio, debido a que se tendrán en cuenta a la hora de analizar dicho pedido. Por ejemplo, cuando la solicitud del cambio es informal, o sea, la misma es sobre ECS que no afecta la arquitectura del software y además no forma parte de la línea base del

proyecto, en aras de reducir el nivel de burocracia en el proceso, las autoras proponen que estos cambios sean analizados y resueltos internamente por el Jefe de Módulo y su equipo de desarrollo.

Por otra parte, si el cambio es semiformal, o sea, que afecta la línea base del proyecto, se propone seguir un procedimiento para este proceso, que tendrá lugar en dependencia del impacto del cambio, el cual puede ser local o global. En caso de ser local, se procede a su análisis dentro del entorno donde este se produce, si presenta un impacto global, este debe ser colegiado por el comité de gestión de cambios, el cual realizara un análisis profundo del mismo. Mientras que si las solicitudes son emitidas por los clientes, una vez entregado el producto, siempre van a ser llevadas a cabo mediante un proceso formal de cambios, con el objetivo de tener un mayor control del producto en esta etapa de su desarrollo. Vale aclarar que efectuar esta clasificación a las solicitudes de cambio es responsabilidad del Líder de Proyecto, el cual presenta una amplia visión del producto.

Una vez efectuada la solicitud del cambio, la misma es clasificada nuevamente, ahora en: Mejora o Defecto, teniendo en cuenta su misión. Esta sub-clasificación ayuda en gran medida al establecimiento de prioridades a las solicitudes e implica un análisis de las mismas por parte la Comisión de Gestión de Cambios. Posteriormente la solicitud se almacena en el repositorio del proyecto, específicamente en la carpeta nombrada *Historia de Cambios,* asignándosele un código único para su mejor localización. El código presenta el siguiente formato:

<Origen_Solicitud > <día> <mes> <año>

Luego se procede a la valoración del cambio. Esta actividad es realizada por la Comisión de Gestión de Cambio. Las autoras proponen que la CGC esté conformada por los siguientes roles:

- Arquitecto del proyecto.
- Gestor de configuración del Proyecto.
- Jefes de cada uno de los módulos.
- Diseñador de cada uno de los módulos.
- Planificador del proyecto.

Esta selección está sustentada bajo la importancia que tiene la presencia de cada uno de estos roles a lo largo del desarrollo del software, por lo que presentan mayor capacidad para analizar profundamente el cambio, determinar si aceptado o rechazado y en caso de la primera opción, planificar la ejecución del

mismo teniendo en cuenta los posibles riesgos en la marcha del proyecto. En la siguiente tabla se muestran características presentes en estos roles que justifican su selección.

Tabla 2.1 Características de los Roles que integran la Comisión de Gestión de Cambios

Rol	Características
Arquitecto del Proyecto.	Tiene gran dominio sobre la arquitectura funcional del software, debido a esto, posee gran visibilidad sobre los posibles elementos que pueden ser afectados por el cambio. Tiene conocimiento si la afectación es sobre elementos arquitectónicamente significativos o no. Por ello puede brindar detalles sobre el impacto del mismo sobre otros elementos
Gestor de Configuración Principal.	Tiene conocimientos sobre la configuración del software, de los elementos que integran la misma, así como la frecuencia de afectación que estos han presentado.
Jefes de Módulo.	Presenta gran dominio sobre el impacto del cambio en su módulo. Puede apoyar tanto al arquitecto y al líder de proyecto a la hora de analizar el cambio. Tiene conocimientos sobre la carga de trabajo que presentan cada uno de los desarrolladores que se encuentran en su módulo, esto ayuda a la hora de asignar personal para la implementación del cambio.

Rol	Características
Planificador del Proyecto	Muestra gran dominio sobre la planificación en tiempo real del proyecto, así como su estado actual. Sus conocimientos pueden ser de gran utilidad a la hora de decidir si es factible realizar el cambio o no, en caso que se decida efectuar el cambio, puede proponer el momento en que se debe realizar el mismo, según la planificación establecida, así como el personal asignado para el cumplimiento de dicha tarea.
Analista de Módulo.	Tiene gran dominio sobre el análisis y diseño del módulo por lo que su aporte puede ser muy importante a la hora de la toma de decisiones.

En la evaluación realizada por la CGC se tienen en cuenta los siguientes aspectos:

- Esfuerzo técnico que requiere el cambio.
- Posibles efectos secundarios a raíz del cambio.
- Impacto global sobre otras funciones del sistema y sobre otros objetos de la configuración.
- Influencia que tiene la implementación del mismo sobre la planificación, en tiempo real, del proyecto.
- Análisis del personal capacitado y disponible para la ejecución del cambio, en caso de que este sea aprobado.

Una vez analizados estos elementos, el resultado y las observaciones de la evaluación se recogen en un Informe de Cambio (IC). En el Anexo 3 se muestran los detalles de la planilla para el informe de cambio.

El resultado puede ser: Aceptado o Rechazado. En cualquiera de los dos casos anteriores, se procede a anexar el IC a la solicitud que le dio origen. En caso que sea Rechazado, se le comunica la novedad al emisor del cambio, así como los elementos que fundamentan la decisión tomada. En caso de ser Aprobado, la CGC le asigna una prioridad al cambio para ser ejecutado, según el análisis realizado sobre el mismo. Seguidamente se genera una Orden de Cambio de Ingeniería (OCI). Este documento tiene como función principal planificar la ejecución del cambio, teniendo en cuenta la disponibilidad del personal para asignar las diferentes tareas, además tiene la función de describir las restricciones que deben ser respetadas así como los criterios de calidad para evaluar la ejecución del cambio, asignando un probador encargado de garantizar la calidad en este proceso.

A partir de este momento se siguen una serie de pasos en función de la ejecución del cambio:

1. Se bloquea el/los ECS sobre el que se realizará el cambio y los ECS relacionados, en el repositorio de subversión, con el objetivo de evitar superposición de cambios.
2. Se informa a todos los implicados que el cambio se va a efectuar y se les muestra los elementos afectados mediante un *Informe de Ejecución de Cambio* (IEC).
3. Se realiza el cambio.
4. Se aplican las adecuadas actividades de aseguramiento de la calidad de la ejecución del cambio.
5. Se desbloquea el/los ECS sobre el que se realizo el cambio y los ECS relacionados.
6. Se informa a los implicados que ya el cambio ha finalizado y que los elementos afectados ya están disponibles.
7. Se actualiza la línea base del proyecto.
8. Se distribuye la nueva versión del producto.

2.4.4 Control Interno de la Configuración

En el Capítulo 1 se presentó la opinión que, sobre las auditorías de la configuración, han ofrecido diferentes autores y organizaciones de reconocimiento internacional. Las autoras de la investigación apoyan la importancia de estos planteamientos, pero a su vez consideran que la forma de implementar las auditorías en el proyecto R&N debe estar enfocada a controlar el proceso de GCS en cada módulo, verificando el cumplimiento de los procesos y mecanismos definidos en la estrategia y cediéndole la tarea de auditar el proceso general, al equipo de Calidad Interna del proyecto, el cual está compuesto por

personal ajeno al proceso de desarrollo técnico, lo cual facilita la efectividad de la auditoría. No se proponen momentos exactos en los cuales realizar la revisión, dejándolo a decisión del Gestor de Configuración Principal, quien es el encargado de realizarla, en cambio sí se definen un conjunto de aspectos a tener en cuenta a la hora de ejecutarla. Estos indicadores han sido agrupados según la actividad de GCS que controlarán:

Identificación de la Configuración.

1. Verificar que todos los trabajadores que intervienen en el proceso cumplan las actividades establecidas según el rol que desempeñan.
2. Verificar que todos los ECS existentes sean identificados, clasificados según las categorías establecidas, almacenados correctamente en el repositorio y notificados a todos los que por su rol puedan interesarle.
3. Verificar que la nomenclatura de cada ECS se encuentre asociadas a la estructura predefinida.
4. Verificar el establecimiento de las líneas base a nivel de módulo, así como la actualización de las mismas según el período de tiempo inicialmente establecido para esta actividad.
5. Verificar que exista correspondencia entre la línea base y el documento que recoge la fecha de establecimiento, las versiones y los ECS que conforman la misma.
6. Verificar que las versiones etiquetadas de las líneas base de los módulos estén en su carpeta Tags correspondiente.
7. Verificar que se haya llevado a cabo una correcta Integración de las líneas base de cada módulo en la línea base principal.
8. Verificar la existencia de los informes que avalan que antes de formar parte de una línea base del proyecto, los ECS pasaron por un control de calidad, o sea que estén debidamente probados.

Gestión de Cambios de la Configuración

1. Verificar que se realizan todos los pasos necesarios para el proceso formal de cambios.
2. Verificar que las solicitudes de cambio aceptadas tengan correspondencia con la actualización del ECS en el repositorio.

3. Comprobar que los cambios especificados en las peticiones de cambio aceptadas han sido realizadas.
4. Verificar si las modificaciones adicionales a una petición de cambio han sido registradas.
5. Verificar que el Proceso de Gestión del Cambio se hayan seguido procedimientos de señalar el cambio, registrarlo y divulgarlo.
6. Cuando se produce un cambio, verificar que todos los ECS que estén relacionados son debidamente actualizados.

Informes de Estado

1. Verificar que existe un historial de los Informes de Estado.
2. Verificar que los informes estén públicos en el controlador de versiones.
3. Verificar que se tiene control de las versiones de los productos entregados al cliente o al departamento de calidad.
4. Verificar que se tiene control sobre las diferentes versiones de los ECS correspondientes a cada módulo.
5. Validar la estabilidad y la estructura del repositorio.
6. Verificar que cada rol tenga los permisos de acceso al repositorio correspondientes a su responsabilidad.
7. Verificar que no se esté desarrollando sobre versiones de ECS luego de que estos hayan sido retirados de la línea principal de desarrollo hacia el repositorio privado.

2.4.5 Generación de Informes de Estado

La comunicación entre los miembros de un proyecto es uno de los aspectos más importantes para alcanzar el éxito. En el proyecto R&N este aspecto constituye un elemento clave debido al gran número de personas que su desarrollo involucra.

La actividad de generación de informes de estado ayuda a guardar datos históricos del desarrollo de un producto y además posibilita que todos los integrantes del proyecto conozcan el estado del mismo, de ahí la importancia que se le brinda a esta actividad en la ErGCS. El encargado de elaborar los Informes de Estado es el responsable de la GCS o el Líder del Proyecto. Se proponen como informes de estado:

1. Documento que registre el contenido de cada línea base cuando estas son creadas o actualizadas, el mismo puede contener el identificador de la línea base (etiqueta), la fecha en que se establece, los ECS que la componen y sus versiones.
2. Documento que registre que versiones del producto son entregados a los clientes o al departamento de calidad.
3. Documento que resuma el estado de las solicitudes de cambios registradas en un periodo de tiempo.
4. Documento que registre el estado de cada uno de los ECS existente (Desarrollo, Prueba, Estable, Liberación).

A pesar de que pueden ser definidos en un inicio del proyecto, los diferentes tipos de informes que se deseen generar pueden ir determinándose a lo largo del desarrollo en dependencia de las situaciones que se vayan presentando y en la experiencia que se vaya adquiriendo.

2.5 Conclusiones del Capítulo

En el transcurso del actual Capítulo, se han desarrollado cada uno de los procedimientos que tienen lugar en la Estrategia de Referencia para la Gestión de Configuración de Software (ErGCS), la cual cumple con los objetivos específicos propuestos inicialmente, partiendo de la necesidad de centralizar, organizar y controlar el proceso de GCS en el proyecto R&N. La realización de este capítulo ha arrojado las siguientes conclusiones:

1. En la estrategia se siguen algunos de los principios de RUP, por ser la metodología que actualmente guía el proceso de desarrollo del software R&N. Además se tiene en cuenta lo que sobre el tema platea la IEEE.
2. Se puede afirmar que la ErGCS brinda una serie de técnicas y procedimientos sólidos encaminados a proporcionar mejorías en el proceso de GCS.
3. La utilización de dos herramientas, Subversion y Trac, para automatizar algunas actividades de la GCS forma parte de la estrategia, desempeñando un papel primordial y de vital importancia en el momento de su aplicación.
4. La finalidad de la ErGCS será su puesta en práctica en el submódulo Bienes, el cual se encuentra en proceso de desarrollo actualmente.

Aplicación de la ErGCS 3

3.1 Introducción

En el presente Capítulo se describen los pasos que se siguieron para la aplicación de la ErGCS en el submódulo Bienes del proyecto R&N. Se realiza además, un análisis inicial valorativo, teniendo en cuenta el impacto de la aplicación de esta estrategia en el proceso de Gestión de Configuración de Software.

Antecedentes de la GCS en Bienes

La ErGCS ha sido concebida para el proyecto de software R&N en su totalidad. Condiciones presentes en este proyecto como el estado de algunos módulos en etapas muy avanzadas de su desarrollo y teniendo en cuenta que los mismos están siendo producidos en la República Bolivariana de Venezuela, dieron lugar a la aplicación de esta estrategia en el submódulo Bienes como etapa de prueba piloto de la misma.

El submódulo Bienes ha sido enmarcado en dos etapas, la primera, definida por la ausencia de estrategias concretas de Gestión de Configuración por parte de la dirección del proyecto y la segunda, después de la puesta en práctica del período de experimentación de la ErGCS propuesta por los autores de este trabajo.

Características de la primera etapa:

- Utilización de dos controladores de versiones: Visual SourceSafe para los artefactos de construcción del producto y Subversion para los artefactos de documentación asociados al proceso de desarrollo.
- Insuficiencia en el proceso de identificación de los ECS.
- Ausencia de definición de las relaciones de dependencia o composición entre los diferentes ECS.
- Ausencia de definición de un procedimiento formal de la gestión de cambio.
- Ausencia de un equipo oficial de control de cambios.

- Ausencia de generación de Informes de Estado, que implicaron una desinformación parcial del equipo de proyecto sobre las condiciones reales del mismo.
- Ausencia de un repositorio soportado por el sistema de control de versiones Subversion para el almacenamiento de los artefactos de construcción del producto.

Por otra parte, los aspectos positivos de esta primera etapa están relacionados con la utilización de no conformidades generadas por el equipo de Calidad Interna y la especialista funcional, que fueron transformadas en solicitudes de cambio y analizadas por el Arquitecto, el diseñador principal de Base Datos y los jefes de equipo del submódulo.

3.2 Aplicación de la ErGCS

Constituyó uno de los pasos iniciales en la aplicación de la ErGCS, la designación del personal a cada uno de los roles establecidos en la misma. El líder de proyecto en Cuba fue designado para asumir el rol de Gestor de Configuración Principal, encargado de supervisar el trabajo del gestor de configuración del submódulo, este rol fue asumido por el jefe del submódulo. Este último es el responsable de administrar el repositorio, elaborar la capacitación de la herramienta y poner en práctica las actividades de gestión de configuración. También fueron seleccionados los integrantes del comité de control de cambios, teniendo en cuenta los roles propuesto en la estrategia para integrar este grupo, el cual se encarga de analizar debidamente las peticiones de cambios que presentan un impacto a nivel de proyecto.

Otro de los pasos iniciales fue adaptar el trabajo que se venía realizando en el proceso de desarrollo del submódulo Bienes a lo establecido en la estrategia y teniendo como premisa evitar la pérdida del historial de cambio. Para ello fue necesario suprimir la anterior metodología de trabajo, consistente en la utilización de dos controladores de versiones diferentes y en su lugar establecer la herramienta Subversion como un único controlador de versiones tanto para los artefactos de documentación como para los de código.

Para dar cumplimiento a dicha tarea se realizó la búsqueda de una herramienta que permitiera efectuar la migración sin pérdida de información entre repositorios Microsoft Visual SourceSafe (VSS) y Subversion (SVN).

Como resultado de la búsqueda efectuada, se decidió utilizar la herramienta vss2svn, la cual es desarrollada por la comunidad libre de Subversion y se encarga de convertir en un fichero entendible por el SVN las bases de datos del VSS. Para ello se siguieron los siguientes pasos:

1. Instaurar un nuevo servidor de SVN.
2. Crear un repositorio en el servidor de SVN, el cual contendrá los datos de la migración.
3. Obtener una copia de la base de datos del VSS.
4. Ejecutar la migración con el programa vss2svn. Durante este proceso tienen lugar los siguientes pasos:
    - Se ejecuta la migración desde la consola de Windows a través del comando: vss2svn.exe –vssdir <dir>, donde:

        <dir> Representa el camino donde se encuentra la base datos del repositorio de VSS (aislada con anterioridad).
    - Al finalizar la ejecución del programa se genera un fichero que luego es subido al repositorio de Subversion. Para esto se utiliza el comando load: svnadmin load <repositorio> <ficherodump>, donde:

        <repositorio> es el camino donde está ubicado el repositorio que almacenará los elementos migrados.

        <ficherodump> es el fichero que se obtiene al correr el programa vss2svn.exe.

De esta forma se obtuvo en el nuevo repositorio de SVN los datos y las estructuras que se encontraban en el VSS.

A cerca de la instalación de la herramienta SVN:

- Fue instalada la versión 1.4.2- r22196 del servidor de Subversion.
- Se instaló la versión 2.0.55 de apache.
- Como cliente de Subversion para el Visual Studio.NET los programadores utilizaron el Ankh 1.0.0.2668-RC4.
- Fue instalada la versión 1.4.1.7992. de Torstoise como cliente de SVN para Windows.

- Seguidamente fueron reestructurados el Directorio de Desarrollo Técnico y el Expediente de Gestión de la Configuración. En las siguientes figuras se muestra la información antes mencionada:

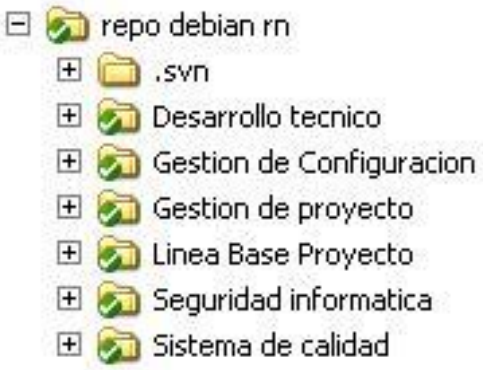
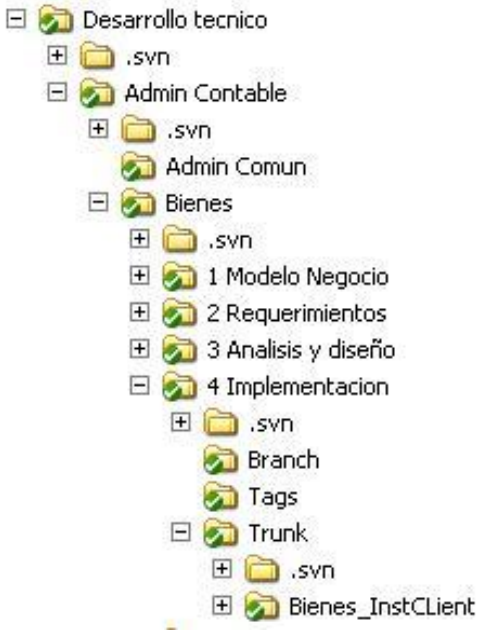
Repositorio	Directorio de Desarrollo Técnico
	

Figura 3. 1 Estructura del Repositorio y del directorio de Desarrollo Técnico de R&N.

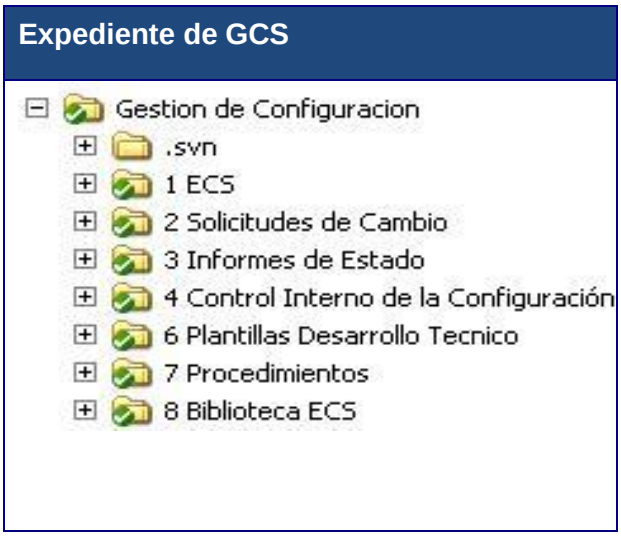


Figura 3. 2 Estructura del expediente de GCS.

De esta forma se resolvieron algunos de los problemas detectados en la primera etapa del submódulo Bienes, debido a que se logró estandarizar el uso de la herramienta dedicada al control de versiones y se obtuvo un expediente de proyecto mejor organizado. Ello influyó en los siguientes aspectos:

- Aumento de la productividad, pues al estar todos los módulos sobre un mismo repositorio, los componentes comunes para varios módulos podían ser accedidos fácilmente haciéndose uso de la propia herramienta de control de versiones, manteniéndose así el historial de cambios sobre dicho elemento y evitando tener que tomarlos de manera manual.
- Se unificaron todos los ECS en una única herramienta lo que agilizó el proceso de backup.
- Mayor seguridad sobre los elementos de configuración, debido a que cada integrante del proyecto, seguía determinadas políticas de acceso, en dependencia del rol que desempeñara, .para acceder al repositorio SVN
- Mayor organización de la información almacenada en el repositorio del SVN, mediante la estructura organizada del expediente de proyecto.
- Rápida localización de la información necesaria para la elaboración de alguna tarea determinada.
- Mayor control sobre los artefactos en desarrollo.

3.2.1 Identificación de la Configuración

3.2.1.1. Identificación de los ECS

Mediante la actividad de identificación de la configuración fueron detectados un total de 24 elementos de configuración de software, los cuales fueron clasificados según las categorías propuestas en el Capítulo 2. Siguiendo las bases definidas en la ErGCS se establecieron varios documentos para la identificación de los ECS. Como resultado de esta tarea se obtuvo un documento por cada categoría de las seleccionadas, el cual almacenaba la información referente a todos los ECS clasificados según esa categoría.

El siguiente gráfico muestra la distribución de los elementos identificados por las categorías establecidas.

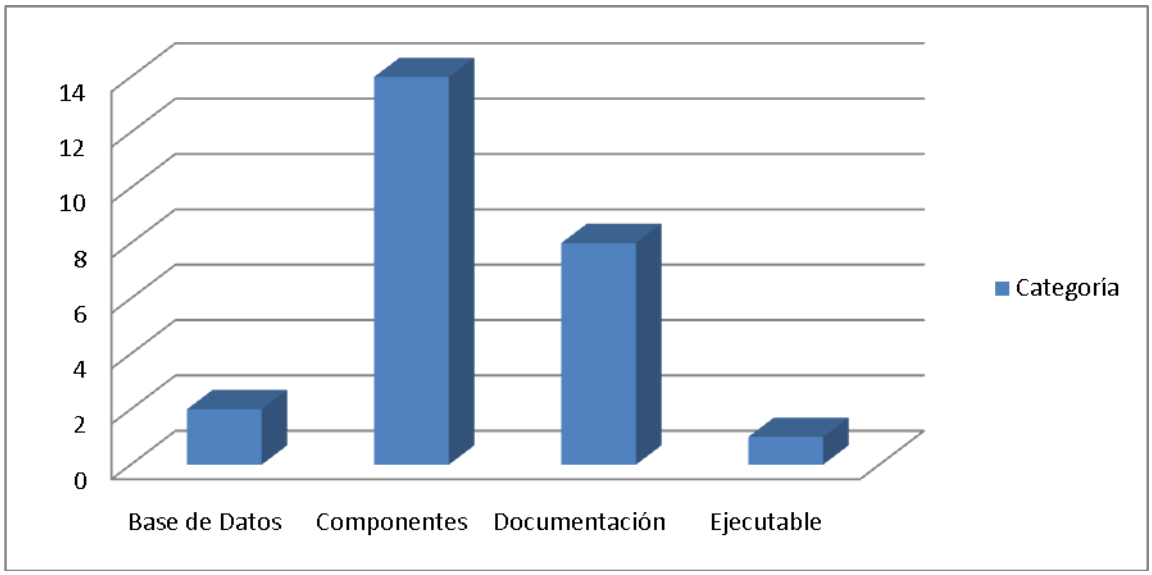


Figura 3. 3 Elementos de configuración de software por categorías.

La documentación generada permitió conocer de cada ECS: los autores, su estado, la importancia de cada uno según la arquitectura del sistema, así como los módulos que lo utilizaban. Además posibilitó tener la localización exacta de cada uno de ellos y la versión con que se trabajaba en ese momento.

3.2.1.2. Relaciones entre los ECS

Una vez identificados los ECS, se analizaron y establecieron las posibles relaciones de dependencia y/o composición, tanto entre los nuevos ECS identificados, como entre éstos y los ya existentes en el repositorio SVN. Toda esta información fue automatizada mediante la aplicación en Access, referenciada en la ErGCS. Una vez insertados los ECS en la herramienta, se adicionaron sus relaciones de

dependencias con los demás ECS, de esta forma se actualizó la base de datos del Access, la cual es consultada en el momento que se solicita cambiar un ECS, con el objetivo de medir el impacto que este tendrá sobre otros ECS, una vez modificado.

3.2.1.3. Línea Base

Con respecto a la definición y evolución de la línea base, se puede plantear que en el submódulo Bienes, la misma fue concebida por el arquitecto del submódulo en las fases iniciales del mismo y luego construida de forma incremental durante el proceso de desarrollo.

Esta línea base quedó almacenada en la carpeta Tags del submódulo Bines en el repositorio SVN, la misma fue etiquetada de la siguiente manera: ***LBBienes1C*** Su composición estuvo dada por los artefactos de código y documentación generados hasta ese momento.

El establecimiento de la línea base proporcionó tres ventajas fundamentales:

- Reproducción posterior de una liberación del producto.
- Establecimiento de las relaciones de predecesor-sucesor entre los artefactos del proyecto.
- Realización de reportes basados en la comparación del contenido de una línea base con otras.

3.2.2 Gestión de Cambios de la Configuración

En el desarrollo de esta actividad se presentaron un total de 8 no conformidades, detectadas por el equipo de calidad interna conjuntamente con el especialista funcional Luego se llevó a cabo una revisión de las mismas con el objetivo de comprender lo que se solicitaba en cada una de ellas, en el caso de que su descripción no estuviera lo suficientemente clara, se determinaba devolver la no conformidad a su lugar de origen, solicitando mejor claridad en la explicación de sobre la misma. Respecto a este aspecto, no fue necesario devolver ninguna no conformidad.

Posteriormente todas las no conformidades que procedieron, fueron convertidas en solicitudes de cambio, con el objetivo de que permanecieran organizadas en un formato apropiado y almacenadas en el repositorio SVN. De todas las solicitudes de cambio reveladas, 2 fueron clasificadas en defectos y 6 en mejora. Luego tuvo lugar el análisis de las mismas por parte del Líder de Proyecto, el cual concluyó que no era necesario efectuarles un proceso formal de cambios, debido a que la ejecución de las mismas no tenía un impacto a nivel de proyecto, por lo que se decidió llevar su análisis al marco del submódulo, ejecutarlas y finalmente actualizar el historial de solicitudes de cambios.

De esta manera el proceso de gestión de cambios en el submódulo Bienes, se convirtió en un proceso debidamente organizado y controlado, permitiendo obtener los siguientes resultados:

- Mayor seguimiento y control del estado de cada una de las solicitudes de cambios recibidas.
- Profundidad en el análisis de las solicitudes de cambios, a través de la ejecución de procesos formales de cambios.
- Disminución del tiempo de respuesta a cada solicitud de cambio emitida.

3.2.3 Informes de Estado

En la ejecución de esta actividad se obtuvieron diferentes informes que no solo almacenaron información de determinadas etapas del proyecto sino que permitieron medir y llevar la contabilidad de los elementos y el estado de los mismos. Al finalizar dichos documentos, la información obtenida fue graficada para una rápida compresión de los datos por parte de las personas interesadas en revisar estos informes, los informes generados fueron los siguientes:

1. Se generó un documento que contenía los ECS registrados en la línea base. Esto permitió tener bien definido y organizado a donde pertenecía cada ECS y sus versiones, así como la cantidad de los mismos. Esto da una medida, además, de los elementos que aún faltan por integrar la línea base.
2. Se generó un informe de estado que contenía toda la información recogida en el Control Interno de la Configuración realizado (sección 3.2.4).
3. Se obtuvo un documento que resume el estado de aceptación (rechazada, aprobada) de las no conformidades generadas antes de aplicar la estrategia. También se recoge en dicho informe el estado de las mismas (En cola, en espera de datos, resuelta). A continuación se muestra la representación gráfica de los datos obtenidos:

Figura 3. 4 Estado de las No Conformidades antes de aplicar ErGCS.

3.2.4 Control Interno de la Configuración

Con el objetivo de mantener un control estricto sobre proceso de Gestión de Configuración de Software en el submódulo Bienes, el líder del proyecto, haciendo uso de la estrategia propuesta, efectuó un control interno a los trabajadores involucrados. En dicha actividad, se tuvieron en cuenta los siguientes parámetros evaluativos:

- Verificar que todos los ECS existentes fueron identificados, clasificados según las categorías establecidas, almacenados correctamente en el repositorio y notificados a todos los que por su rol puedan interesarle.
- Verificar que la nomenclatura de cada ECS se encuentre asociadas a la estructura predefinida.
- Verificar que exista correspondencia entre la línea base y el documento que recoge la fecha de establecimiento, las versiones y los ECS que conforman la misma.

- Verificar que las versiones etiquetadas de las líneas base de los módulos estén en su carpeta Tags correspondiente en el repositorio.
- Verificar la existencia de los informes que avalan que antes de formar parte de una línea base del proyecto, los ECS pasaron por un control de calidad.
- Verificar que existe un historial de los informes de estado.
- Verificar que los informes estén públicos en el repositorio.
- Validar la estabilidad y la estructura del repositorio.
- Verificar que cada rol tenga los permisos de acceso al repositorio correspondientes a su responsabilidad.
- Verificar que no se esté desarrollando sobre versiones de ECS luego de que estos hayan sido retirados de la línea principal de desarrollo hacia el repositorio privado.

Para la evaluación de los mismos, se decidió establecer clasificaciones que definen el nivel de cumplimiento del parámetro que se evalúa. Para ello se clasificaron en:

- Bien: Se clasificaron de esta forma a los procesos que se encontraban entre el 70% -100% de su cumplimiento.
- Regular: Se clasificaron de esta forma a los procesos que en ese momento se encontraban entre el 50%-70% de su complimiento
- Mal: Se clasificaron de esta forma a los procesos que en ese momento se encontraban por debajo del 50% de su cumplimiento.

Los resultados arrojados en el Control Interno a la Configuración son representados en el siguiente gráfico:

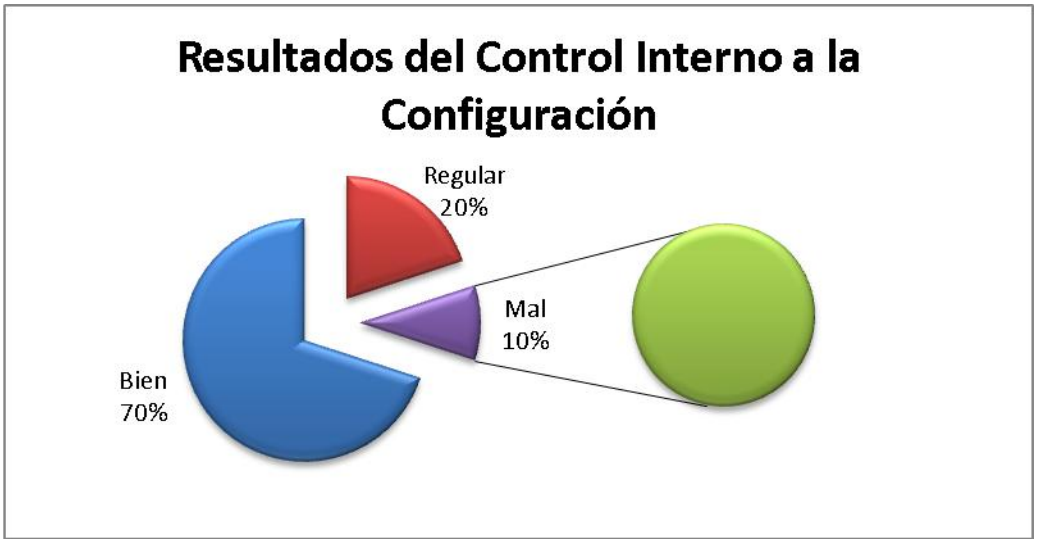


Figura 3. 5 Resultados del Control Interno a la Configuración.

Posteriormente se elaboró en un Informe de Estado, el cual contenía la descripción detallada del proceso de Control Interno a la Configuración, así como los resultados observados en el mismo.

La realización del control interno de la configuración constituyó un medidor importante para el líder del proyecto, a la hora de evaluar la calidad con la que los trabajadores estaban efectuando los procesos. De esta forma se logró tener un control sobre el proceso en general, así como detectar las vulnerabilidades existentes en el mismo.

3.3 Conclusiones del Capítulo

A modo de conclusiones se puede afirmar que a lo largo del capítulo se han desarrollado los objetivos planteados al inicio del mismo, o sea, se han seguido una serie de pasos para describir todo el proceso de aplicación de la estrategia, también tuvo lugar el análisis de los resultados obtenidos con posterioridad a la aplicación de la misma. Referente a los aspectos abordados en el capitulo, se pueden concluir los siguientes elementos:

1. Fue necesaria la migración de datos de VSS a Subversion, con el objetivo de establecer la Herramienta Subversion como único controlador de versiones en el proyecto.

2. Se realizó una capacitación para el trabajo con la herramienta de Subversion lo cual generó en los trabajadores involucrados, habilidades con el uso de la misma.
3. Mediante el establecimiento de roles para la GCS se logró una mayor especialización por parte de los trabajadores involucrados y a su vez facilitó el seguimiento y control sobre las tareas asignadas.
4. Mediante el proceso de Identificación de la Configuración fueron identificados, clasificados, almacenados y notificados un total de 24 ECS del submódulo Bienes.
5. Mediante la generación de Informes de Estados aumentó el nivel informativo de cada miembro del proyecto a cerca de estado del producto en desarrollo y también influyó en gran medida en la toma de decisiones por parte de la directiva del proyecto.
6. La ejecución de controles internos a la configuración proporcionó un mayor control y seguimiento de las actividades asignadas, así como la detección de insuficiencias en el proceso y posibles mejoras para mitigarlos.

Conclusiones Generales

A manera de conclusiones de este trabajo se puede plantear que durante el desarrollo del mismo fueron cumplidos tanto los objetivos generales, como específicos planteados en un inicio. Fue aplicada en el submódulo Bienes la ErGCS definida en el Capitulo 2. La misma tuvo como objetivo principal, definir correctamente los procesos de Gestión de Configuración de Software, lo cual se traduce en los siguientes elementos:

1. Se realizó un estudio del estado del arte de los procesos Gestión de Configuración de Software que permitió:
    - Adquirir profundidad en los conocimientos de Gestión de Configuración de Software.
    - Caracterizar un conjunto de herramientas que brindan soporte y automatizan los procedimientos propuestos en la ErGCS.
    - Analizar diferentes patrones de rama de la GCS.
2. Se definieron procedimientos capaces de regular y estandarizar el proceso de Gestión de Configuración de Software en el proyecto R&N.
    - Se obtuvo un esquema de definición de roles para una mejor aplicación de la GCS, lo cual permitió contar con un equipo de GCS bien estructurado y tareas bien identificadas.
    - Se establecieron mecanismos para la identificación de la Configuración, permitiendo alcanzar gran madurez y organización en este proceso.
    - Se establecieron mecanismos formales de cambios, capaces de controlar el proceso de Gestión de Cambios.
    - Se establecieron procedimientos para el control de versiones del producto lo cual permitió controlar los ECS que componían cada una de las versiones liberadas.
    - Se estableció una metodología de trabajo haciendo uso del patrón de Rama por Proyecto, garantizando en todo momento la integridad de la línea principal de desarrollo
    - Se establecieron mecanismos para el Control Interno de la Configuración, siendo esta una forma de evaluar el trabajo de los gestores de cada módulo.

- Se establecieron mecanismos para la contabilidad del estado del producto siendo este un elemento informativo al alcance de todos los miembros del equipo de desarrollo.

3. Se seleccionó y justificó adecuadamente las herramientas de software para ser utilizadas en la automatización de la GCS en el proyecto R&N proponiendo.
    - Subversion como única herramienta para el control de versiones, proporcionando homogeneidad en el uso de herramientas controladoras de versiones.
    - Una base de datos en Acces para la identificación de las relaciones entre los ECS, con el objetivo de valorar el impacto que tendrá un ECS, una vez que haya sido modificado.
4. La propuesta para la Gestión de Configuración se aplicó en el submódulo Bienes del proyecto R&N:
    - Se ejecutaron diferentes informes de estados, los cuales aumentaron el nivel informativo de los miembros del equipo de desarrollo.
    - Se efectuaron controles internos a la configuración, obteniendo un mayor control sobre el proceso.
    - Se seleccionaron los ECS existentes y se establecieron sus relaciones.
    - Se estableció la line base del proyecto lo cual proporcionó tres ventajas fundamentales:
        - Reproducción posterior de una liberación del producto.
        - Establecimiento de las relaciones de predecesor-sucesor entre los artefactos del proyecto.
        - Realización de reportes basados en la comparación del contenido de una línea base con otras.

Recomendaciones

Finalmente y como parte de las recomendaciones de este trabajo se propone:

1. Aplicar la estrategia de GCS definida en este trabajo para todo el proyecto de R&N y proyecto con características similares.
2. Realizar un estudio profundo de las herramientas que posibilitan el desarrollo distribuido geográficamente.
3. Implementar en la Universidad de Ciencias Informáticas (UCI) una herramienta de apoyo a la gestión de configuración que abarque todas sus actividades. Dicha herramienta debe soportar un buen manejo de múltiples ramas de forma tal que puedan ser implementado todos los patrones de ramas vistos en el capítulo 1.
4. Evaluar la ErGCS que se describe en este trabajo, mediante las valoraciones de expertos en el tema de GCS.
5. Extender este trabajo a la elaboración de una estrategia genérica para los procesos de GCS, la cual pueda ser aplicada en todos los proyectos productivos de la facultad, favoreciendo de esta forma, la homogeneidad en de los procesos de GCS.
6. Insertar el contenido de este trabajo en el plan de cursos optativos en para impartir en el proyecto R&N.

Referencias Bibliográficas:

Acevedo, R. V. (Abril de 2004). Mejoramiento del Proceso de Gestión. Chile.

Antonio, A. d. (2001). *La Gestión de la Configuración del Software.*

Appleton, B., Berczuk, S. P., Cabrera, R., & Orenstein, R. (s.f.). *www.cmcrossroads.com*. Recuperado el 17 de febreo de 2007, de cmcrossroads: http://www.cmcrossroads.com/bradapp/acme/branching/

Babich, W. (1986). *Software Configuration Management.* Addison-Wesley.

Bañeres, J. P. (2005). *Visión ejecutiva de procesos y prácticas para desarrollo de software.*

Dapena, M. D. (2005). UNA PROPUESTA DE INTRODUCCIÓN A LAS REVISIONES EN EL PROCESO DE DESARROLLO DEL SOFTWARE.

Estrada, A. F. (2003). Un modelo de Referencia para la Gestión de Configuración en la PYME de Software. Ciudad de la Habana, Cuba.

Gestion de la Configuracion del software. Práctica de GCSW. (s.f.). Recuperado el 6 de Febrero de 2007, de http://www.cvc.uab.es/shared/teach/a21290/projecteRUP/Plan%20de%20Desarrollo%20Software.doc

Gómez, J. A. *Gestión de la Configuración.*

Gracia, J. (26 de Noviembre de 2005). *Ingeniero Software.* Obtenido de http://www.ingenierosoftware.com/calidad/cmm-cmmi-nivel-2.php

Hista International S. A. (2006). Recuperado el 25 de Abril de 2007, de http://www.histaintl.com/soluciones/configuracion/configuracion.php

IEEE. (1990). Recuperado el 12 de Abril de 2007, de http://www.ieee.org.ni/aboutieee.htm

IEEE. (1998).

Ivar Jacobson, G. B. (2003). *Ayuda del Rational Unified Process version 2003.06.00.65*.

Lucarella, L., & Bertogli, A. (22 de septiembre de 2006). *LugFi*. Recuperado el 5 de marzo de 2007, de http://lug.fi.uba.ar/documentos/scms/index.php

Machuca, S. R. *Análisis de impacto en la gestión de cambios de Servicios de Telecomunicaciones.*

Martínez, R. D. (2006). *ConfigCASE 3.0 HERRAMIENTA DE APOYO A LA GESTIÓN DE CONFIGURACIÓN. PROPUESTA ARQUITECTÓNICA.* Ciudad de la Habana , Cuba.

Ministerio de Administraciones Públicas. (s.f.). *Map.es.* Recuperado el 5 de abril de 2007, de http://www.csi.map.es/csi/metrica3/index.html

Navarro, A. *Ingeniería del Software.*

Navarro, J. A. (Junio de 2006). Entorno Unificado para la Gestión de Configuración de Software. Ciudad de la Habana, Cuba.

Pressman, R. S. (2005). *Ingeniería del Software. Un Enfoque Práctico.*

Proceso de Control de Cambios Guiado por la Arquitectura del Software. (s.f.). Recuperado el 2 de Febrero de 2007, de html.rincondelvago.com/control-de-cambios-del-software.html

Rancán, C. J. (Julio de 2003). Gestion de Configuración de Productos Software en Etapa de Desarrollo.

Rodriguez, K. L. *Gestión de Configuraciones Plan de gestión – Comparación de estándares.*

Ruiz Arroyo, B., & García Peñalvo, F. J. (2007). Sistemas SCM para la Gestión de la Configuración del Software. *Solo Programadores* (147), 46-51.

Vázquez Acosta, M. (s.f.). *sitio de Producción e Investigación de la Facultad 3.* Recuperado el 13 de abril de 2007, de http://facultad3.uci.cu/ProdF3v2

Anexos

Anexo # 1

A. Ejemplos de Elementos de Configuración

1. La especificación del sistema.

2. El plan del proyecto software.

3. La especificación de requisitos software.

4. Un prototipo, ejecutable o en papel.

5. El diseño preliminar.

6. El diseño detallado.

7. El código fuente.

8. Programas ejecutables.

9. El manual de usuario.

10. El manual de operación e instalación.

11. El plan de pruebas.

12. Los casos de prueba ejecutados y los resultados registrados.

13. Los estándares y procedimientos de ingeniería de software utilizados.

14, Los informes de problemas.

15. Las peticiones de mantenimiento.

16. Los productos hardware y software utilizados durante el desarrollo.

17. La documentación y manuales de los productos hardware y software utilizados durante el desarrollo.

18. Diseños de bases de datos

19. Contenidos de bases de datos.

B. Datos de un elemento de configuración de software

1. Número o código del Elemento de Configuración del Software.

2. Nombre del ECS

3. Descripción del ECS

4. Autor/es del ECS

5. Fecha de creación

6. Identificación del proyecto al que pertenece el Elemento de Configuración.

7. Identificación de la línea base a la que pertenece.

8. Identificación de la fase y subfase en la que se creó.

9. Tipo de Elemento de Configuración (documento, programa, elemento físico, etc.)

10. Localización

Anexo # 2

Formulario de Solicitud de Cambio

Nombre del proyecto: _____

Producto: _____ Versión: _____

Fecha de solicitud:_____

Solicitado por:

Nombres y Apellidos	Rol desempeñado

Título del cambio:_____

Descripción:_____

Elementos de Configuración Afectados:

_____ _____

Beneficios o razones para el cambio_____

Impacto:_____

Estado: ____En Cola ____Aprobado ____ En desarrollo

____En prueba _____Rechazada ____ En espera de información

Prioridad: □Máxima □Media □Mínima

Aprobado por:_____

Rechazado por: _____

Anexo # 3

Orden de Trabajo

Nombre del proyecto: _____

Producto: _____ Fecha de Creación:_____

Título: _____

Descripción:_____

Tipo de trabajo: _____

Código de PC o Problema asociado:_____

Asignado a:

Rol desempeñado	Nombres y Apellidos
Desarrollo	
Probador	

Prioridad: □Máxima □Media □Mínima

Fase actual:

_____Determinación de Requerimientos _____Estudio Preliminar

_____Análisis _____Diseño _____Codificación _____Otra