Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de Diploma para Optar por el Título de

*Ingeniero en Ciencias Informáticas*


# Gestión de la Configuración de Software y Control de Cambios para el proyecto productivo IMAC.

**Autor:**

Yunior Armando Hernández Andrade

**Tutor:**

Ivette Barrientos Núñez

**Co-tutor:**

Vladimir Milián Núñez

# DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del presente trabajo y por este medio reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Y para que así conste firmo la presente a los ___ días del mes de _____ de _____

_____

Yunior Armando Hernández Andrade

# AGRADECIMIENTOS

Agradezco la realización de este trabajo principalmente a mi familia, que siempre me ha apoyado, me ha impulsado a estudiar, a superarme y sin los cuales no hubiese llegado a donde estoy. Es para mi mucho más que un honor poderlos llamar “mi familia”.

Agradezco especialmente a mi mamá Alina Andrade porque tuve que crecer mucho para saber el porque no te gustaba el huevo frito en azúcar durante el período especial.

Agradezco especialmente a mis dos papás que pueden tener mil defectos pero para mi son los dos mejores ejemplos de mi vida, ojalá pueda algún día decir que me parezco a Uds.

Agradezco a los profesores Romualdo Filgueiras, Rafael de la Cruz y Oscar Reyes que me mostraron el camino de la ciencia y me iluminaron el túnel del conocimiento para que pudiera andar por él sin tropiezos.

Agradezco a mis amigos de siempre Rayner, Tayser, Sagastume, Mario, Manditín, Delvis, Eduardito, Toyniell, Darién, Albin, Jose y a todos los que no puedo poner aquí por el poco espacio.

Agradezco a todos los que aportaron su granito de arena en la realización de este trabajo de diploma, porque ninguna ayuda fue poca.

Agradezco a mis compañeros de clase de todos los cursos, su ayuda siempre fue bienvenida y sus críticas mucho más aún.

Agradezco a la comunidad de Fruteros, que me llevo a crear tantas buenas amistades, a los Valarien por aguantar mis berros, a los viejos del Emu, a los del PvE, a los tufados de la arena, y hasta a los novatos hablantines. Sin mencionar nombres, gracias.

Agradezco a mis tutores, a mis profesores de todos los años estudiantiles, a mis vecinos, a mis conocidos y a todos los que me aprecian y que ahora mismo no recuerdo, Uds. saben que es por mala memoria, no por olvidarlos.

A todos, muchas gracias.

*A mis dos hermanitas lindas, Omitza y Cyndell. Vivan la vida, estudien, supérense, diviértanse y sean felices. Siempre estaré ahí para Uds.*

*A mis seis abuelos Armando Hernández, Gladis Fuentes, Armando Andrade, Deysi Hernández, Eduardo López y Gladis Gonzales.*

# RESUMEN

El presente trabajo tiene como base la necesidad del establecimiento de un proceso de desarrollo de software con calidad en el proyecto productivo IMAC de la Facultad 2 de la Universidad de las Ciencias Informáticas.

Consta de una investigación cuyo objetivo principal es proponer un Plan de Gestión de la Configuración y Cambios para el proyecto productivo IMAC que tribute, con políticas adecuadas, a la calidad del producto y del proceso en general. Por tanto el tema tratado a lo largo del trabajo se basará en las actividades, herramientas y procedimientos que soporta el rol que atiende la Gestión de la Configuración de Software.

En el capítulo 1 se mostrarán los resultados obtenidos de un estudio valorativo del estado del tema a nivel de universidad y a nivel internacional. También se definirán los términos importantes para el presente trabajo.

En el capítulo 2 se mostrarán las características del proyecto IMAC que son influyentes para el presente trabajo y se desarrollará la propuesta de políticas de gestión de la configuración y cambios que conformarán el plan mencionado para el proyecto.

En el capítulo 3 se muestran los resultados de la ejecución de métodos de validación aplicados para evaluar la propuesta definida.

# ÍNDICE DE CONTENIDO

# INDICE DE IMÁGENES

# INTRODUCCIÓN

El Ministerio de Auditoría y Control de Cuba (MAC) es la entidad gubernamental encargada de llevar el control sobre los procesos, controles y Auditorías que realizan las entidades del país con el objetivo de prevenir, enfrentar y detectar indisciplinas laborales, ilegalidades y corrupción administrativa, promover, fomentar y consolidar un hábito de control y un clima de máxima honradez. Entre sus principales funciones están la realización de las siguientes acciones de control:

Auditoría gubernamental: auditoría externa que realiza el gobierno central a los órganos y organismos de la administración central del estado, a las instituciones financieras bancarias y no bancarias y a las organizaciones económicas estatales; así como, a las entidades del sector cooperativo; a las organizaciones; asociaciones; asociaciones internacionales y demás entidades económicas, cuando reciban, por disposiciones gubernamentales, subsidios, subvenciones, ventajas, exenciones y concesiones, o presten un servicio público o establezcan contratos o compromisos con el gobierno o con entidades estatales; y a cualquier persona natural o jurídica que realice un actividad lucrativa y que esté sujeta a una obligación tributaria generada en el territorio nacional.

Control gubernamental: verificación por comisiones de trabajo que se constituyen a tales efectos con la participación de organismos de la administración central del estado, de la aplicación de las políticas de gobierno y el cumplimiento del plan y el presupuesto, por los organismos de la administración del estado y por las entidades de su sistema, lo que se ejecuta acorde a un plan aprobado anualmente por el consejo de ministros, incluyéndose en este el seguimiento a los realizados en años anteriores.

Fiscalización: acto de verificación, inspección, investigación y comprobación que se realiza a órganos y organismos de la administración central del estado, a organizaciones económicas estatales y a dirigentes y funcionarios designados o aprobados por autoridad u órgano de gobierno facultado, con la finalidad de velar por el cumplimiento de las disposiciones estatales y de gobierno vinculadas con la actividad económico-financiera, preservar la disciplina y la integridad administrativa, así como, prevenir y detectar actos de corrupción administrativa, lo que se ejecuta de conformidad con los planes que se elaboren anualmente sobre la base de intereses estatales y por quejas y denuncias de la ciudadanía vinculadas con la ilegalidad en el control y uso de los recursos del estado y actos de corrupción administrativa.

Atención al Sistema Nacional de Auditoría: sistema conformado por la actividad de Auditoría del Ministerio de Auditoría y Control, por las unidades de Auditoría que integran el Sistema de la Administración Tributaria; las unidades centrales de Auditoría Interna de los Órganos y Organismos del Estado; las unidades de Auditoría Interna de las organizaciones superiores de Dirección Empresarial; los auditores internos de las organizaciones económicas y de las sociedades civiles de servicios y otras formas de organización que practican la Auditoría Independiente.

Debido a sus características el MAC controla un volumen considerable de información que debe ser examinado y verificado en un tiempo limitado. Información cuyo carácter muchas veces es confidencial o inclusive secreto y presenta una importancia vital para el desarrollo apropiado del país.

La mayoría de los procesos y el control de todo el volumen de información en el MAC se hace manualmente con el uso de procesadores de texto y algunas aplicaciones obsoletas que no cubren todas las necesidades del funcionario que las usa. Además en muchos casos el volumen de documentos físicos y el llenado de planillas dificultan la conservación y posterior uso de esta información.

Debido la situación antes planteada y siguiendo la necesidad identificada por el país de informatizar la sociedad cubana para acercarse más al desarrollo sostenible se decidió mediante un convenio de cooperación con la Universidad de las Ciencias Informáticas (UCI) crear un sistema digital capaz de gestionar toda la información de manera segura manteniéndola disponible fácilmente; con la cual agilizar los procesos que realizan los trabajadores del MAC.

Dicho sistema, por la importancia que tiene al ser el primer proyecto de informatización de un ministerio completo desarrollado en el país, cuyo proceso de desarrollo debe servir de guía para futuros proyectos con características similares, debe contar con una calidad excepcional tanto el producto final como todo el proceso de desarrollo.

En la calidad del producto, así como en la calidad del proceso de desarrollo, influyen de manera decisiva las prácticas, métodos, técnicas y herramientas usadas por los integrantes del equipo de trabajo del proyecto durante todas las fases del ciclo de vida del proyecto.

A la disciplina del campo de la informática que se encarga de establecer los principios necesarios para la obtención de un software económico, fiable y que funcione eficientemente se le denomina Ingeniería de Software. Esta disciplina ofrece un

conjunto de metodologías sistemáticas, predecibles y repetibles cuyo fin es mejorar la productividad en el desarrollo y la calidad en el producto de software.

Existen además en el mercado muchas herramientas que brindan apoyo al equipo de trabajo llamadas herramientas CASE (Computer Aided Software Engineering) y muchas más herramientas de desarrollo (IDEs, Frameworks, Middleware) que facilitan el trabajo del programador. Toda esta gama de herramientas son útiles para la construcción y modificación de artefactos durante el ciclo de vida del proyecto pero con ellas se evidencia y acrecienta una variable temida aunque necesaria en el desarrollo de software, el “cambio”.

Cuando los cambios ocurren en fases tempranas del proyecto, reajustar el trabajo, el contenido y las líneas de trabajo de los desarrolladores no es muy difícil, pero cuando ocurren en fases avanzadas puede ser muy problemático, perder parte de la integridad del proyecto o salirse de la línea base propuesta con el evidente desastre que ello conlleva. Esto, sumado a que las herramientas de desarrollo facilitan tanto el cambio que puede llegarse a ver como algo trivial, hace que la necesidad de un estricto control sobre el cambio se imponga. Es absurdo tratar evitar el cambio, pues el sistema siempre cambiará y los deseos de cambiarlo persistirán a lo largo de la vida del proyecto.

Independientemente de la metodología escogida para el desarrollo, esta debe incluir un flujo de trabajo de soporte que contenga un conjunto de procesos dedicados a establecer la validez de todo el producto obtenido durante cualquiera de las etapas de desarrollo y que facilite el mantenimiento del software proporcionando una imagen detallada del sistema en cada etapa a través de un estricto control de los cambios realizados y de la disponibilidad de una versión estable de cada elemento para toda persona involucrada en el proyecto. A este flujo de trabajo se le conoce como Gestión de la Configuración.

El documento que establece las políticas de la Gestión de la Configuración es el Plan de Gestión de la Configuración. En este plan se describen todas las actividades de gestión de la configuración y cambios que serán realizados durante el ciclo de vida del proyecto. Además brinda planificaciones detalladas de las actividades, responsabilidades asignadas y recursos necesarios que incluyen personal, herramientas y equipamiento relacionadas con la gestión de la configuración y cambios del proyecto. La puesta en práctica de las especificaciones descritas en el Plan de Gestión de la Configuración, inclusive cuando las políticas definidas en este

no sean óptimas, aseguran cierto nivel de control sobre los cambios durante el desarrollo lo cual contribuye en gran medida al éxito del proyecto.

Ahora, cada proyecto tiene peculiaridades que lo hacen único y que demandan atención especial durante su desarrollo. Aunque muchos enfoques permitan abordar la producción de software con procesos y actividades similares lo cierto es que cada ciclo de vida obliga a establecer medidas especiales que dependen de las especificidades del problema a resolver con el producto en cuestión, del equipo de trabajo que interviene, de los métodos de comunicación entre las personas vinculadas, de las herramientas seleccionadas y disponibles, etc. Por esto cada Plan de Gestión de la Configuración es también único y específico del proyecto para el cual se ha creado, constituyendo un trabajo novedoso aunque describa unas políticas muy parecidas a otras o use conceptos comunes.

En la UCI, aunque se imparte Ingeniería de Software como asignatura lectiva y se incluye un encargado de la Gestión de Configuración en cada proyecto la realidad es que de forma general los integrantes de equipos de proyecto subvaloran el control de lo cambios y el establecimiento de métodos para controlarlos.

La suma de todo lo planteado da origen al siguiente **problema científico**:

¿Cómo asegurar la liberación de versiones estables e informes de estado de la progresiva evolución del proyecto productivo IMAC de manera eficiente, eficaz y contribuyente a la calidad general del proceso de desarrollo?

Cuyo **objeto de estudio** es el proceso de Gestión de la Configuración y específicamente en el campo de acción de los procesos de Gestión de la Configuración en el proyecto productivo SIGAC.

El **objetivo** de este trabajo es proponer un Plan de Gestión de la Configuración para el proyecto productivo IMAC que sea eficiente, eficaz y que contribuya a la calidad general del proceso de desarrollo.

**Objetivos Específicos:**

- Identificar y definir los Elementos de la Configuración de Software (ECS) que forman parte del sistema que requieren un control de cambios.
- Estabelecimiento y soporte de Bibliotecas de Software.
- Definir la organización y estructura de los objetos de la configuración dentro de las bibliotecas de software.

- Definir los procesos de control de cambios.
- Definir políticas para facilitar la revisión del producto y las actividades de seguimiento de defectos y cambios.

Para darle cumplimiento a los objetivos planteados proponemos la realización de las siguientes **tareas de investigación**:

- Realizar un estudio de la bibliografía especializada actual sobre los procesos de gestión de la configuración y cambios.
- Realizar un estudio de planes de gestión de configuración y cambios de proyectos similares en la UCI.
- Realizar un análisis de los artefactos generados dentro del proyecto productivo SIGAC.
- Definir los ECS que requieren un control de cambios.
- Realizar un estudio comparativo de las herramientas existentes que permiten el control de versiones.
- Realizar un estudio de las características específicas del proyecto para la selección de la herramienta adecuada para el control de versiones.
- Documentar el proceso de instalación y configuración de la herramienta de control de versiones seleccionada.
- Establecer una estructura organizativa para los ECS dentro de la biblioteca instalada.
- Documentar los procesos de control de cambios dentro del proyecto.
- Implementar una estrategia para el seguimiento, custodia y reporteado del estado de la configuración.
- Aplicar las propuestas al proyecto productivo SIGAC para probar resultados.

La idea que defiende éste estudio es: Si se crea un plan de gestión de la configuración adecuado y ajustado a las especificidades del proyecto; y además se aplican adecuadamente y de manera sistemática las políticas y métodos en él definidos se logrará un ciclo de desarrollo óptimo, sostenido y organizado tributando a un mejor producto y a un equipo de desarrollo más preparado.

En la realización de este trabajo se usaron los métodos empíricos, métodos teóricos, métodos estadísticos matemáticos y en especial el método Delphi para la validación por panel de expertos de la investigación.

Los métodos teóricos que se utilizaron fueron: el Análisis y Síntesis, utilizado para extraer la información más importante y significativa en los estudios conceptuales y del estado del arte. Método Histórico Lógico, usado para constatar teóricamente como ha evolucionado la gestión de la configuración dentro de la UCI. Método Inductivo-Deductivo que permitió llegar a un grupo de conocimientos generalizadores, tanto desde el análisis de lo particular a lo general, como desde el análisis de elementos generalizadores a un nivel más específico.

Los métodos empíricos que se usaron fueron vinculados a las técnicas de recolección de datos. Específicamente se usó la encuesta.

Los métodos estadísticos matemáticos se usaron para evaluar los resultados de los datos recopilados en la encuesta que se realizó para el estudio del estado del arte en la Universidad.

El método Delphi se utilizó para validar la propuesta de Plan de Gestión de la Configuración descrita en esta investigación.

Para la organización de la información de este trabajo, el mismo consta de tres capítulos distribuidos de la siguiente manera:

- Capítulo 1: Contiene un estudio de los elementos y conceptos relacionados con el objeto de estudio y un análisis del estado del arte a nivel internacional y dentro de la universidad relacionado con el tema de la investigación.
- Capítulo 2: Contiene un análisis de las características del proyecto productivo IMAC y la descripción de una propuesta de Plan de Gestión de la Configuración para este.
- Capítulo 3: Contiene los resultados de la validación de la propuesta realizada.

# Capítulo 1: Fundamentación Teórica

## INTRODUCCIÓN

En este capítulo se tratan conceptos y elementos teóricos que forman los pilares de esta investigación. Se precisan algunos de los más importantes conceptos y definiciones relacionados con la Gestión de la Configuración de Software (GCS), las características de los procesos que la conforman y su papel e importancia dentro de la Ingeniería de Software. Además se tiene en cuenta el criterio de destacados autores sobre el nivel de importancia que le confieren a este tema.

También se hace un análisis de las tendencias actuales en el mundo y las principales tareas que se realizan. Finalmente se enumeran algunas de las herramientas informáticas más usadas para la Gestión de la Configuración y se realiza un análisis comparativo de estas.

## CONCEPTOS E IMPORTANCIA

### ¿QUÉ ES LA GESTIÓN DE LA CONFIGURACIÓN DE SOFTWARE?

La Gestión de la Configuración de Software ha sido definida de manera ligeramente diferente a criterio de cada autor como:

*"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."* (Babich, 1986)

*"La gestión de configuración del software es una actividad de protección que se aplica a lo largo de todo el proceso del software. La GCS identifica, controla, audita e informa de las modificaciones que invariablemente se dan al desarrollar el software…"* (Pressman R. , 2005)

*"Describe la estructura del producto e identifica los elementos de la configuración que la componen y que son tratados como entidades versionables. La GCS define configuraciones, construcción e identificación, agrupa los artefactos versionados en*

*conjuntos de componentes y brinda seguimiento entre las versiones."* (Rational Unified Proccess, 2003)

*"Gestión de Configuración es la disciplina que abarca todo el ciclo de vida de la producción de software y productos asociados. Específicamente, requiere de la identificación de los componentes a controlar y la estructura del producto, controla todos los cambios sobre los elementos y garantiza mecanismos para auditar todas las acciones."* (IEEE, 1990)

La definición que se considera más completa en esta investigación y por la que ésta se va a regir es:

*"Proceso de soporte cuyo propósito es identificar, definir y almacenar en una línea base los elementos de software, controla los cambios, reporta y registra el estado de los elementos y de las solicitudes de cambio; asegura la completitud, consistencia y corrección de los elementos; controla, almacena, maneja y libera los elementos asociados al producto de software."* (Jacobson, Griss, & Jonson, 2000)

## ¿QUÉ IMPORTANCIA TIENE LA GESTIÓN DE LA CONFIGURACIÓN DE SOFTWARE?

La importancia de la GCS ha sido valorada por diversos autores de cuyos textos podemos apreciar su valor. Por ejemplo Watts Humphrey plantea que en el desarrollo de software, los problemas más frustrantes, son causa de un pobre proceso de Gestión de Configuración. Y resultan frustrantes porque toman tiempo para solucionarlos, ocurren generalmente en momentos críticos y su ocurrencia es totalmente innecesaria. (Humphrey, 1989) Tom Milligan, asesor de IBM Rational Software plantea, según referencia en (Rational, 2004), que: *"La Gestión de Configuración de Software es el héroe no nombrado en el proceso de desarrollo de software"*.

En (Brown, 1998) se valora su importancia como: *"La Gestión de 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"*

## GESTIÓN DE LA CONFIGURACIÓN EN LOS MODELOS DE CALIDAD

Para la gestión de la configuración a nivel internacional se han seguido dos variantes fundamentalmente: Seguir las normas de los estándares de las oficinas internacionales de estandarización IEEE e ISO; o seguir las normas de los estándares de calidad creados específicamente para el software como CMMI. En ambas variantes se puede observar claramente la inclusión de la gestión de la configuración como proceso necesario dentro del modelo.

## EN CMMI:

En el Modelo de Madurez de Capacidad del Software, desarrollado en 1991 por el Instituto de Ingeniería del Software con el propósito de guiar a las organizaciones en la selección de estrategias de mejora determinando la madurez del proceso actual e identificando los puntos importantes que se deben estudiar y trabajar para mejorar tanto el proceso como la calidad del software. Este modelo ha ido evolucionando gradualmente y en diciembre del 2001 surge CMMI (Software Engineering Institute). Este modelo cuenta al igual que CMM con cinco niveles, aunque algunos cambian de nombre. Para poder escalar a un nivel superior es necesario haber cumplido todos los aspectos que plantea el nivel actual, por lo que es posible incluso, cumplir con aspectos de nivel 2 y de nivel 3 simultáneamente y, aún así, mantenerse en un nivel 1. Para obtener Nivel 2 de CMMI (Gestionado) es necesario realizar las siguientes tareas (Software Engineering Institute) (Ema, 2002):

- 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 Configuración de Software.

Como podemos observar la GCS ocupa un lugar importante en este modelo pues tiene que ser garantizada desde el segundo nivel de los 5 niveles que tiene CMMI para considerarse que se cumple con este.

## EN ISO

ISO es una red de institutos nacionales de estándares de 156 países que promueve la normalización internacional para facilitar el intercambio de bienes y servicios como de aplicaciones (ISO, 1995). Relacionado con los procesos de software ha sido implementado el modelo de evaluación y mejora del proceso de software ISO 9000, específicamente la guía ISO 9000-3.

Entre los aspectos que tiene en cuenta la Norma ISO 9000-3 se encuentran:

- Sistema de Calidad.
- Especificación de los requisitos a comprobar.
- Planificación del desarrollo.
- Planificación de la calidad.
- Pruebas y validaciones.
- Gestión de Configuración de Software.
- Control de Documentos
- Mediciones

El hecho de que una norma tan reconocida internacionalmente incluya la GCS dentro de su estándar ratifica la importancia que le conceden tantos autores.

## EN IEEE

El estándar IEEE 730-1998 para el Plan de Aseguramiento de la calidad de Software (IEEE 730, 1998), aborda aspectos como la administración, la documentación, el control de código, etc. Dentro de la documentación se le adjudica gran importancia al Plan para la Gestión de Configuración de Software reflejado en el estándar IEEE 828-1998 (IEEE, 1990). Dicho plan trata todo lo referente a la asignación de las responsabilidades, la identificación de las actividades que se realizarán durante todo el proceso, la identificación de la configuración, el reconocimiento de los elementos de configuración, el control de la configuración, el acceso a las bibliotecas, la aprobación o desaprobación de un cambio y la implementación del cambio de ser aprobado (todas estas incluidas entre las tareas de la Gestión de Configuración).

# GESTIÓN DE LA CONFIGURACIÓN EN DETALLES

Si bien todas las normas o estándares vistos anteriormente incluyen la gestión de la configuración brindándole una importancia relativamente alta lo cierto es que en todas

las variantes usadas la lista de actividades que incluye bajo esta categoría es ligeramente diferente.

## ACTIVIDADES POR MODELO

En el modelo **CMM** se incluyen las siguientes actividades: (Paulk, 1993)

- Planificación de las actividades de Gestión de Configuración. *[planif]*
- Identificación de los Elementos de Configuración de Software. *[ident]*
- Control de cambios a los Elementos de Configuración de Software. *[contr]*
- Informar a los grupos e individuos involucrados de los cambios a los Elementos de Configuración de Software. *[report]*
- Auditoria de la Configuración. *[audit]*

La norma **ISO** propone las siguientes actividades: (ISO 10007, 1995) (Burrows, 1999) (García, 2001)

- Identificación de la Configuración. *[ident]*
- Control de Cambios en la Configuración. *[contr]*
- Generación de Informes de Estado. *[report]*
- Auditoria de la Configuración. *[audit]*

El estándar de la **IEEE** define las siguientes actividades: (IEEE, 1990)

- Identificación de la Configuración. *[ident]*
- Control de Cambios en la Configuración. *[contr]*
- Generación de Informes de Estado. *[report]*
- Auditoria de la Configuración. *[audit]*

## ANÁLISIS

Observe que las actividades marcadas como *[ident]*, *[contr]*, *[report]* y *[audit]* referentes a las actividades que con diferentes títulos realizan la “identificación de los elementos a incluir”, “el control de cambios”, “el reporteado o aviso a los implicados” y la “auditoría a los procesos de configuración” respectivamente están presentes en los tres modelos mencionados. Sin embargo la actividad marcada como *[planif]* aparece CMM pero no en ISO ni en IEEE.

Existen otros autores que han incluido una actividad más, la “Gestión de Versiones” *[vers]* (Febles, 2004). Sin embargo ninguno de los estándares analizados la incluye

como un proceso independiente sino como parte de la actividad marcada como *[contr]* (Antonio, 2001) (Gervás, 2002)

La propuesta que se ha elegido para este trabajo coincide con los estándares ISO e IEEE pues considera que la actividad de planificación correspondiente a CMM es la realización de éste propio trabajo. (IEEE, 1990) (Burrows, 1999) (ISO 10007, 1995) (García, 2001)

A continuación pasaremos a explicar detalladamente cada una de las actividades de la gestión de la configuración que propone el modelo usado en este trabajo.

## IDENTIFICACIÓN DE LA CONFIGURACIÓN

Los artefactos resultantes del proceso de Ingeniería de Software pueden agruparse en tres categorías: (Febles, 2004) (Pressman R. S., 2002)

- Programas de Computadora. Código fuente y ejecutables.
- Documentos descriptivos de los Programas de Computadora. Tanto técnicos como dirigidos a usuarios finales.
- Datos. Internos y Externos de los Programas de Computadora.

Todos estos elementos producidos como parte del proceso de ingeniería de software se denominan Elementos de la Configuración de Software *[ECS]* y conforman la unidad básica para la Gestión de la Configuración de Software.

La cantidad de ECS que se generan durante el ciclo de vida del proyecto aumenta a medida que este avanza, así como las relaciones entre los ECS. Esto puede provocar confusión en etapas avanzadas, sobre todo si no se mantiene un control estricto, lo cual puede provocar repetido de información o faltante de ésta por asumir su existencia equivocadamente. De ahí la importancia de la identificación de la configuración en etapas tempranas del proyecto.

Entre las definiciones de Identificación de la Configuración están:

*"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"* (Antonio, 2001)

*"habilidad para identificar qué información va a ser controlada en el uso del proyecto, quién es el dueño de la información (…) Además de tener identificada la última liberación."* (Brown, 1998)

*"proceso de identificar cada Línea Base a ser establecida durante el ciclo de vida del proyecto y describir los ECS y su documentación, que formarán parte de cada una de las Líneas Base. Una vez que han sido seleccionados los ECS y sus componentes, se debe establecer una forma de identificar y acceder a estos"* (NASA, 1995)

En general la actividad consiste en identificar y asignar nombres significativos a los ECS siguiendo una convención previamente determinada, establecer la importancia de cada uno, su procedencia, la versión correcta y la última versión y mantener estos datos accesibles a los implicados.

Ésta actividad incluye varias tareas: (SWEBOK, 2004)

- Establecimiento de una jerarquía preliminar del producto software.
- Selección de los Elementos de Configuración.
- Definición de las relaciones en la configuración.
- Definición de un Esquema de Identificación.
- Definición y Establecimiento de líneas base.
- Definición y Establecimiento de bibliotecas de software

A continuación se describen estas tareas.

### ESTABLECIMIENTO DE UNA JERARQUÍA PRELIMINAR

Se trata de obtener una visión general del sistema producto, de sus elementos componentes y de la estructura de estos organizándolos jerárquicamente acorde a criterios propios variados como importancia, funcionalidad, empaquetamiento, etc. Esta actividad puede realizarse opcionalmente en los primeros pasos del proyecto. En dependencia de la metodología usada en el desarrollo esta tarea puede ser actividad de otro rol dentro del proyecto como suele ser el Arquitecto.

### SELECCIÓN DE LOS ELEMENTOS DE CONFIGURACIÓN

Es indispensable la selección de una cantidad adecuada de ECS y la selección de los ECS adecuados. Seleccionar muy pocos ECS puede convertir el proyecto en poco visible o poco entendible. Lo mismo sucede al seleccionar ECS de baja importancia. Sin embargo seleccionar muchos de ellos puede llevar a un incremento excesivo de especificaciones y documentos que resulten muy difíciles de manejar.

El mantenimiento de un ECS es costoso ya que puede requerir Especificaciones, Manuales, Revisiones y Aprobaciones por parte del usuario, Auditorías Funcionales,

Auditorías Físicas, Contabilidad de Estado y Trazabilidad de Evolución independientes y específicas para cada uno.

Los ECS seleccionados deben incluir funcionalidades distintas en ECS distintos para evitar la reconstrucción y liberación de nuevas versiones muchas veces. La opción de separar funcionalidades diferentes en ECS distintos disminuye el impacto que un cambio en el ECS tiene sobre el proyecto.

Existen criterios documentados que pueden tenerse en cuenta para la selección de ECS: (Antonio, 2001)

- Utilización múltiple: Número de elementos de su mismo nivel o niveles superiores que lo utilizan.
- Criticidad: Gravedad del impacto de un fallo en dicho componente.
- Número de personas implicadas en su mantenimiento.
- Complejidad de su interfaz: Las interfaces de un ECS con otros deberían ser simples. Hay que minimizar el acoplamiento entre ECS.
- Singularidad del componente con respecto al resto.
- Reutilización: Si el componente se va a diseñar especialmente para ser reutilizado.
- Si se trata de elementos reutilizados.
- Tipo de tecnología: Si el componente incorpora nuevas tecnologías no utilizadas en otros componentes.

En los casos en que, al analizar un componente, caiga dentro de uno de los criterios mencionados es muy probable que este se convierta en un nuevo ECS pues requerirá mucha atención por parte del equipo de desarrollo.

## DEFINICIÓN DE LAS RELACIONES DE LA CONFIGURACIÓN

Según se expresa en la literatura consultada los ECS son objetos que pueden estar interconectados mediante relaciones. (Antonio, 2001) (Pressman R. S., 2002)

Esta información es importante porque ayuda a comprender donde se sitúa un elemento con respecto a otro. Normalmente el personal dedicado a GCS no suele llevar toda la idea del sistema que se construye, por esto es importante definir estas relaciones. Además ellas ayudan a notificar cuando se hace necesaria la revisión o modificación de otro elemento por la modificación de otro.

Algunas de las relaciones que puede llevarse entre elementos es: (Antonio, 2001)

- **Equivalencia:** Por ejemplo, cuando un determinado ECS que es un programa está almacenado en tres lugares diferentes (estación de trabajo, un disco duro externo y el diskette del programador), pero todas las copias corresponden al mismo programa.
- **Composición:** Por ejemplo, el ECS "especificación de diseño" estará compuesto de otros ECS, como el "modelo de datos" o el "diseño del módulo N", para cada uno de los módulos que componen el producto software.
- **Derivación:** A partir de qué se ha originado algo. Por ejemplo, el código objeto del código fuente, o una determinada traza de ejecución de un determinado caso de prueba con un determinado programa ejecutable.
- **Sucesión:** En la historia de cambios sobre un elemento desde una revisión a otra. Puede ser muy útil definir un Grafo de Evolución para cada ECS. Este grafo describe la historia de cambios de un objeto y su transición de unas versiones a otras.
- **Interrelación:** Le permiten al Ingeniero de Software determinar que otros objetos de configuración pueden verse afectados.
- **Dependencia:** Cualquier otro tipo de relaciones entre ECS, fundamentalmente en la documentación, y sobre todo para facilitar la trazabilidad de los requisitos.
- **Variante:** Variación sobre un determinado elemento, con la misma funcionalidad, pero que, por ejemplo, funciona más rápido.

Si se identifican y se mantienen estas relaciones se hace mucho más sencillo determinar que otros ECS serían afectados si se llevara a cabo la modificación de otro ECS.

### DEFINICIÓN DE UN ESQUEMA DE IDENTIFICACIÓN

Esta tarea consiste en decidir el método que se utilizará para identificar unívocamente a cada ECS. Debe decidirse un esquema que permita etiquetar cada ECS y que incluya datos de importancia de este que permitan su diferenciación. Algunos de los datos importantes que deben incluirse son: (Antonio, 2001) (Febles, 2004)

- Número o código numérico del ECS.
- Nombre del ECS.
- Descripción del ECS.
- Autor(es) del ECS.
- Fecha de Creación.
- Identificación del proyecto al que pertenece el ECS.

- Identificación de la Línea Base a la que pertenece.
- Identificación de la fase y subfase en que se creó.
- Tipo de ECS (documento, programa, elemento físico, etc.)
- Localización.

Esta información puede ser referenciada como partes separadas o codificada y agrupada en un código único. Esta última opción es la más usada. Se pueden elegir entre dos tipos de sistemas de codificación:

- Significativos.
- No significativos.

El sistema más sencillo es el de codificación no significativa ya que consiste en la asignación de números consecutivos lo cual hace más fácil la selección del código. Sin embargo para su implementación requiere el establecimiento y mantenimiento de registros de asignación de códigos pues el número por sí solo no brinda información alguna.

La codificación significativa puede ser sencilla o compleja según en dependencia del tipo y cantidad de información que se codifica en el ECS. La ventaja es que no es necesario consultar ningún registro para saber de que ECS se trata al hacer referencia a un código. Sin embargo debe usarse codificación significativa solo si ello va a ser útil pues se suele hacer largos códigos de identificación que la convierten en una codificación tan compleja que quienes la usen no van a ser capaces de recordar su significado. En este caso la codificación significativa carece de sentido.

Las herramientas de gestión automática de la configuración suelen ofrecer facilidades para la identificación de los diferentes componentes software del proyecto. Sin embargo la mayoría de ellas no soportan la identificación de tipos específicos de ECS como los documentos y los dispositivos físicos.

## DEFINICIÓN Y ESTABLECIMIENTO DE LÍNEAS BASE

Diferentes conceptos de línea base pueden ser encontrados en la bibliografía consultada, como son:

*"La línea base es un concepto que nos ayuda a controlar los cambios sin impedir seriamente los cambios justificados"* (Pressman R. S., 2002)

*"La línea base es una instantánea (snapshot) 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"* (Rational Unified Proccess, 2003)

*"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."* (IEEE, 1990)

Estos conceptos se complementan entre si.

Se plantea que la idea de la creación de líneas bases consiste en permitir cambios rápidos e informales sobre un ECS antes de que se revise, apruebe y pase a formar parte de una línea base, pero desde entonces se deberá aplicar un proceso formal para evaluar y verificar cada nuevo cambio propuesto sobre este. (Antonio, 2001)

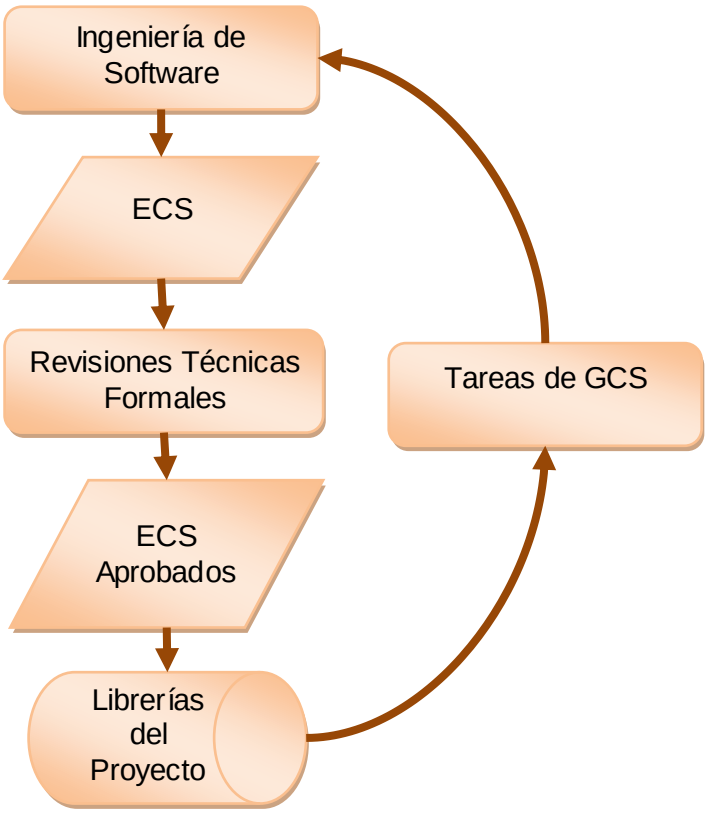
Se puede establecer una línea base de dos formas: (Antonio, 2001) (Febles, MConfig.PM, Modelo de referencia para la Gestión de Configuración en la pequeña y mediana empresa de software, 2004)

- **Físicamente:** Etiquetando cada ECS y almacenándolo en un archivo o librería de proyecto.
- **Lógicamente:** Publicando un documento de identificación de la configuración que referencie el estado actual del producto en dicho punto del desarrollo del proyecto.

Al comienzo del proyecto se debe precisar (Antonio, 2001)

- ciclo de vida a utilizar,
- fases del ciclo de vida,
- productos de cada fase,
- puntos de establecimiento de líneas bases y
- ECS contenidos en cada línea base.

*ILUSTRACIÓN 1* ***LÓGICA DE UNA LÍNEA BASE***




Como resumen podemos decir que las líneas base son puntos de partida para futuros desarrollos del Proceso de Desarrollo de Software. Se establecen al cumplir con un hito del proyecto y una vez establecidas los cambios sobre los ECS que la conforman se realiza bajo un proceso formal de control de cambios.

## DEFINICIÓN Y ESTABLECIMIENTO DE BIBLIOTECAS DE SOFTWARE

Una biblioteca de software se define como “una colección controlada de software y/o documentación relacionada cuyo objetivo es ayudar en el desarrollo, uso o mantenimiento del software.” (Antonio, 2001) (SWEBOK, 2004)

Las bibliotecas facilitan las tareas de GCS, especialmente las de Control de Cambios y Contabilidad de Estado. En ellas se almacenan copias oficiales de los ECS que se encuentran bajo control de cambios. (NASA, 1995)

Existen un grupo básico de bibliotecas a definir: (Antonio, 2001)

- **de Trabajo:** Se establece al inicio del proyecto y comprende el área de trabajo donde los analistas y diseñadores elaboran los documentos del proyecto y donde los programadores desarrollan el software. En esta biblioteca el control de cambios es informal.

- **de Integración:** Es de esta biblioteca de donde se toman los ECS para su integración en ECS de nivel superior del sistema, que generalmente incluyen varios ECS independientes, los cuales al ser integrados aumentan en valor para el sistema.
- **de Soporte al Proyecto:** En esta biblioteca se almacenan los ECS aprobados y transferidos desde la Biblioteca de Trabajo o desde la Biblioteca de Integración. Cuando un elemento pasa a esta biblioteca, se encuentra sujeto a un control de cambios interno y semi-formal.
- **de Producción:** Esta clasificación agrupa la Biblioteca de trabajo, la de integración y la Biblioteca de Soporte al Proyecto
- **Maestra:** Se usa para almacenar ECS liberados para su entrega al cliente o distribución en el mercado. Los elementos en la biblioteca maestra se encuentran sujetos a un control de cambios formal y estricto. Normalmente esta biblioteca tiene fuertes restricciones de acceso para escritura, aunque no así para lectura. En esta biblioteca se almacenan las liberaciones (Release) del sistema.
- **Repositorio de Software:** Es la entidad en la que se archivan los ECS de un proyecto tras su cierre. Se transfieren a él desde la Biblioteca Maestra. Opcionalmente se puede identificar un segmento especial en el que se almacenarán los elementos reutilizables. Todo lo que se almacena en el repositorio debe estar adecuadamente identificado y catalogado, para facilitar su recuperación en caso de necesidad. Se supone que es un almacenamiento a largo plazo, por lo que puede ser de recuperación lenta. Es central y común a todos los proyectos, mientras que la biblioteca de Producción y la Maestra son individuales para cada proyecto.
- **de Seguridad:** También debe estar adecuadamente identificada, aunque su contenido no está sujeto a Gestión de Configuración (las copias contenidas en ella no están catalogadas en los registros de Gestión de Configuración).

Los ECS pasan de una biblioteca a otra con procesos de revisión en caso de pase a una de control más riguroso o con procesos de cambio (defecto, mejora, etc.) a una de control menos riguroso. En este último caso el ECS generaría una nueva versión. La tarea de bibliotecario suele estar encomendada a uno de los miembros del proyecto.

Además de elegir los tipos de biblioteca que se van a usar y mantener en el proyecto, se deben definir procesos para:

- Establecimiento de una biblioteca.
- Introducción de elementos en una biblioteca.
- El acceso a la biblioteca.

## CONTROL DE CAMBIOS DE LA CONFIGURACIÓN

Según Pressman los cambios son un hecho inevitable durante el desarrollo de un software (Pressman R. S., 2002) resumen que además se verifica claramente en la Primera Ley de la Ingeniería de Bersoff, la cual enuncia: *"No importa en que momento en el ciclo de vida se encuentra el sistema, este va a cambiar y el deseo de cambio persistirá a lo largo de todo el ciclo de vida del mismo."* (Bersoff, 1980) Esto es un hecho que ocurre principalmente porque, con el tiempo, varían las necesidades o requisitos del software y aumentan los conocimientos del equipo de desarrollo. (Febles, MConfig.PM, Modelo de referencia para la Gestión de Configuración en la pequeña y mediana empresa de software, 2004) (Febles, 2002)

Han de considerarse fundamentalmente dos tipos de cambios: (Antonio, 2001)

- **Corrección de un Defecto:** Corresponde a reparación de una funcionalidad no correcta en el sistema.
- **Mejora del Sistema:** Corresponde al agregado de nuevas funcionalidades no existentes o mejoras a funcionalidades que ya existen y funcionan.

Para determinar de qué tipo es un cambio es importante tener en cuenta la trazabilidad de los requisitos.

Suelen establecerse varios niveles de control de cambios, entre ellos:

- **Informal:** Este ocurre antes de que el ECS pase a formar parte de una línea base, momento en el cual aquel que haya desarrollado el ECS puede realizar cualquier cambio justificado sobre él.
- **Semi-formal:** También llamado "a nivel de proyecto". Una vez que el ECS pasa la revisión técnica formal y se convierte en parte de una línea base, para que el encargado de su desarrollo realice cambios sobre él debe recibir la aprobación de: (Pressman R. S., 2002) (Antonio, 2001)
- El director o líder del proyecto, si es un cambio local.
- El Comité de Control de Cambios, si el cambio tiene impacto sobre otros ECS.

- **Formal:** Se suele adoptar una vez que el producto se ha comenzado a comercializar, cuando se transfieren los ECS a la Biblioteca Maestra. Todo cambio debe ser aprobado por el Comité de Control de Cambios.

En los procesos con niveles formal o semi-formal aparece una nueva figura, el Comité de Control de Cambios. Este comité es una persona o grupo de personas encargados de tomar las decisiones finales acerca del estado y prioridad de las peticiones de cambio. Su obligación es tener una visión general del producto para poder evaluar el impacto de cada cambio en un determinado Elemento de Configuración sobre otros Elementos de Configuración, así como el impacto sobre la calidad del producto, su rendimiento, su fiabilidad, la visión que el cliente tiene del producto, etc. Esta labor se suele delegar en el Comité de Revisión del Diseño.

## EL PROCESO DE CONTROL DE CAMBIOS

Se plantea que no existe ningún estándar para el control de cambios de tipo informal o interno, aunque existen algunas recomendaciones en “IEEE STD 1042 Guide to Software Configuration Management”. (Antonio, 2001)

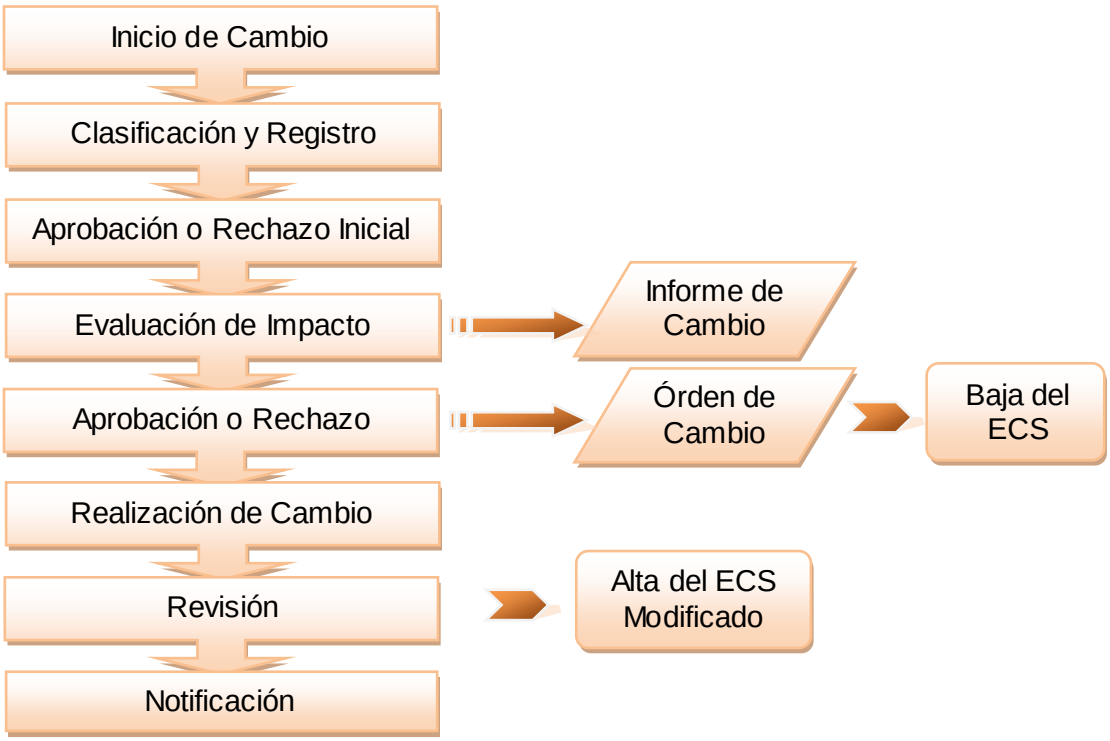
Un proceso de control de cambios poco estricto creará problemas y en un proyecto grande creará caos rápidamente, sin embargo un proceso de control de cambios demasiado estricto, con mucho control también crea problemas pues las mismas fuerzas que hacen necesario el control de cambios lo hacen molesto. Sin dudas nos enfrentamos a una situación de equilibrar. (Bach, 1998) (Pressman R. S., 2002)

En cuanto al control de cambios de tipo formal, existen diversas formas en las que se puede estructurar aunque si existen etapas típicas para el proceso de hacer cambios sobre una línea base. (Antonio, 2001)

1. Iniciación del Cambio, se presenta una solicitud de cambio, que puede venir provocada por un problema que se ha detectado o por un cambio en los requisitos.
2. Clasificación y registro de la solicitud de cambio.
3. Aprobación o rechazo inicial de la solicitud de cambio. De ello suele ser responsable el Comité de Control de Cambios.
4. Evaluación de la solicitud de cambio, si ha sido aprobada, para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y el coste estimado del cambio. Como resultado se obtiene un Informe de Cambio.

5. Se presenta el Informe de Cambio al Comité de Control de Cambios. Si se considera que el cambio es beneficioso se genera una Orden de Cambio (también llamada Orden de Cambio de Ingeniería), que describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y de auditoria. Esta Orden de Cambio es asignada a alguno de los ingenieros de software para que se encargue de llevarlo a cabo. En este momento, el objeto a cambiar se da de baja en la Biblioteca de Soporte al Proyecto.
6. Se realiza el cambio, entrando en un proceso de seguimiento y control.
7. Una vez finalizado el cambio, se certifica, mediante una revisión, que se ha efectuado correctamente el cambio y con ello se ha corregido el problema detectado o bien se han satisfecho los requisitos modificados. El objeto se devuelve a la Biblioteca de Soporte al Proyecto.
8. Se notifica el resultado al que originó el cambio.

*ILUSTRACIÓN 2* ***ESTRUCTURA TÍPICA DEL PROCESO DE CONTROL FORMAL DE CAMBIOS*** (ANTONIO, 2001)

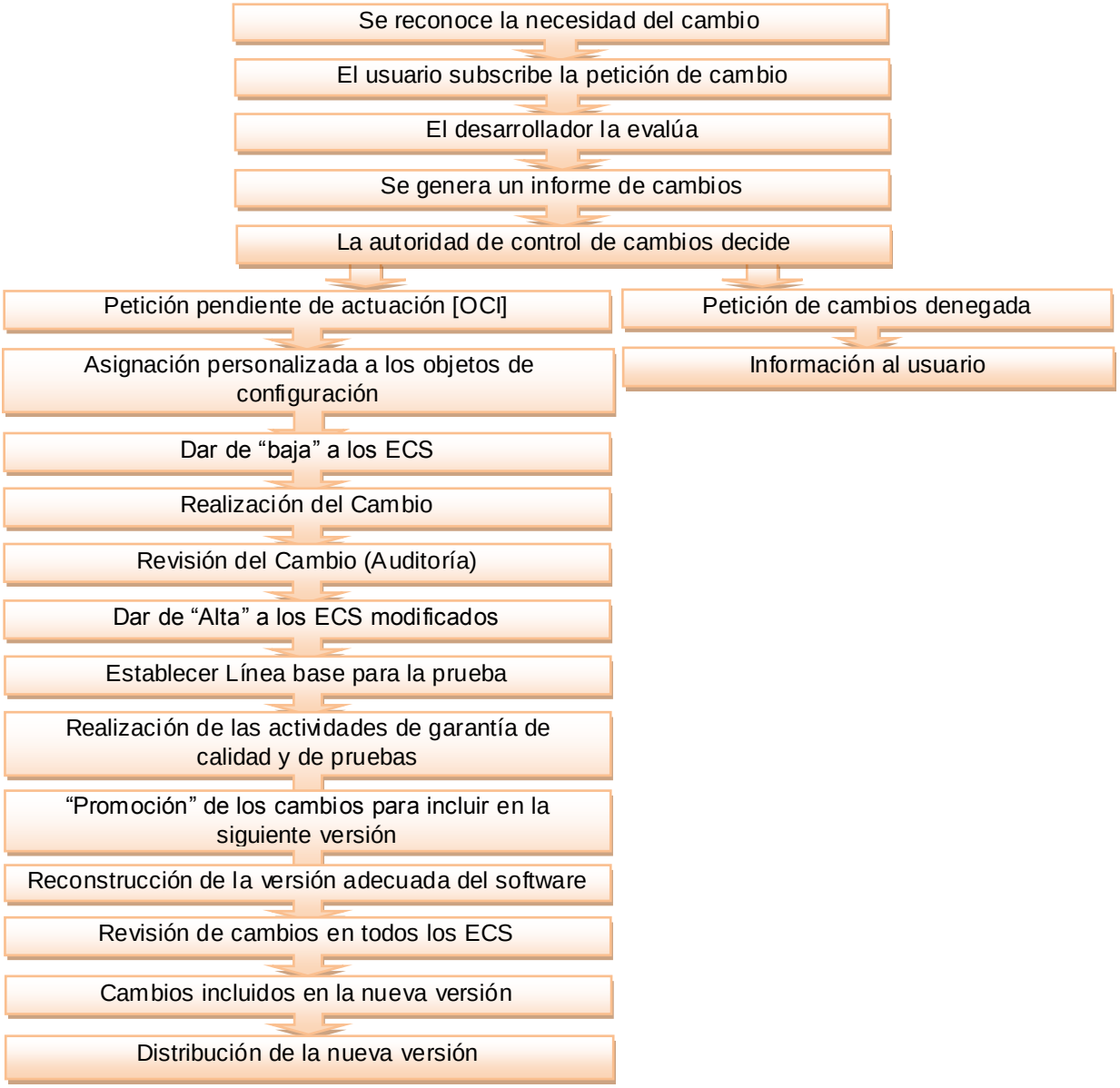



Otras fuentes (Pressman R. S., 2002) proponen un proceso de cambios formal con pasos más específicos pero mantienen la idea del control de cambios informal antes de que el ECS pase a formar parte de una línea base. En este momento el desarrollador puede realizar sobre el ECS cualquier cambio justificado por los requisitos técnicos y las necesidades del proyecto siempre que este no influya en otros requisitos del sistema que no formen parte del ámbito de trabajo de dicho

desarrollador. El proceso de control de cambios formal de (Pressman R. S., 2002) se describe a continuación.

Se hace una petición de cambio y se evalúa para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y sobre otros objetos de la configuración. Los resultados de la evaluación se presentan como un informe de cambios a la autoridad de control de cambio (ACC), persona o grupo que toma la decisión del estado y la prioridad del cambio. Para cada cambio aprobado se genera una orden de cambio de ingeniería (OCI). La OCI describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y auditorías. El objeto a cambiar es “dado de baja” de la base de datos del proyecto; se realiza el cambio y se aplican las adecuadas actividades de Gestión de Calidad de Software (SQA). Luego el objeto es “dado de alta” en la base de datos del proyecto y se usan los mecanismos de control de versiones apropiados para crear la siguiente versión del software.

*ILUSTRACIÓN 3* ***PROCESO DE CONTROL DE CAMBIOS*** (PRESSMAN R. S., 2002)



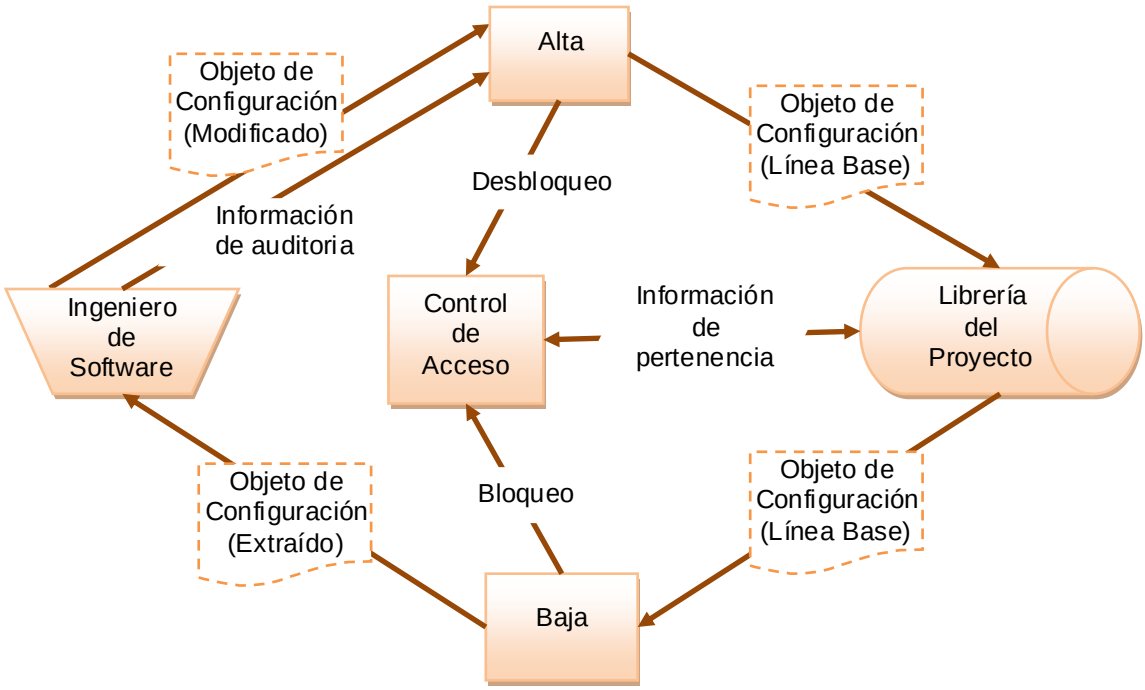

En ambos procesos formales de control de cambios podemos observar las tareas de "alta" y "baja" de los ECS antes y después de su modificación. Las tareas de "alta" y "baja" implementan dos elementos importantes del Control de Cambios: (Pressman R. S., 2002) (Antonio, 2001)

- **Control de Acceso:** Gobierna los derechos de los desarrolladores a acceder y modificar objetos de configuración concretos.
- **Control de Sincronización:** Asegura que los cambios en paralelo, realizados por personas diferentes, no se sobrescriban.

El flujo de control de acceso y de sincronización se explica como proceso según Pressman: "….De acuerdo con una petición de cambio aprobada y una orden de cambio de ingeniería, un ingeniero de software da de baja a un objeto de

configuración. Una función de control de acceso comprueba que el ingeniero tiene autoridad para dar de baja al objeto, y el control de sincronización bloquea el objeto en la base de datos del proyecto, de forma que no se puedan hacer más actualizaciones hasta que se haya remplazado con la nueva versión. Fíjese que se puede dar de baja otras copias, pero no se podrá hacer otras actualizaciones. El ingeniero de software modifica una copia del objeto de línea base, denominada versión extraída. Tras la SQA y la prueba apropiada, se da de alta la versión modificada del objeto y se bloquea el nuevo objeto de línea base." (Pressman R. S., 2002)

*ILUSTRACIÓN 4* ***CONTROL DE ACCESO Y DE SINCRONIZACIÓN*** (PRESSMAN R. S., 2002)




## MECANISMO DE SOLICITUD DE CAMBIOS

Este es el primer mecanismo que es necesario definir. Constituye el mecanismo necesario para solicitar cambios sobre los ECS.

El primer paso es definir una plantilla o formulario para solicitar los cambios. Este formulario debe contener la información suficiente para determinar: (Antonio, 2001)

- Para que se necesita el cambio,
- qué debe ser cambiado,
- quién solicita el cambio,
- descripción detallada del problema,
- otros ECS que puedan afectarse,

- otros cambios con los que se relacione,
- aprobación del cambio,
- como se solucionó el problema.

Este formulario debe ser simple y aceptado por las personas que lo tendrán que manejar.

Muchas veces se usa el formulario de Solicitud de Cambios para todas las operaciones relacionadas con pedidos de cambios sin embargo es común que se definan formularios diferentes para tipo de cambio.

Al formulario usado para informar problemas y que derivarán, si son aprobados, en un cambio de tipo “Corrección de Defecto” se les denomina “Informes de Incidencias” o “Informes de Problemas”. Estos recogen información adicional sobre el incidente que develó la existencia del problema. (Antonio, 2001)

- Fecha y hora en que ocurrió,
- descripción del incidente,
- efectos producidos,
- de que forma se puede repetir (duplicar) el incidente, (si se ha podido duplicar)
- volcado de datos,
- referencia al tipo de prueba que se estaba realizando.

Los informes de incidencias son analizados por los desarrolladores y estos pueden recomendar una de las siguientes acciones: (Antonio, 2001)

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

En la última acción se menciona un nuevo formulario, Notificación de Cambio, en la que se puede dar respuesta a varios informes de incidencias con una sola de ellas, cuando todos ellos corresponden al mismo problema.

También suele utilizarse un formulario especial para solicitar cambios en la documentación que suele llamarse Formulario de Seguimiento de la Documentación a los que también pueden dar lugar los informes de incidencia cuando la deficiencia reportada en el informe o la mejora sugerida tiene impacto sobre la documentación. Sin embargo también pueden usarse los formularios anteriores para reemplazar al Formulario de Seguimiento de la Documentación.

## MECANISMO DE DECISIÓN SOBRE LAS SOLICITUDES DE CAMBIO

Aunque no son únicos se proponen algunos criterios a tener en cuenta para tomar la decisión de aprobar o rechazar una solicitud de cambio: (Antonio, 2001)

- Valor del cambio para el proyecto/organización.
- Retorno de la inversión.
- Tamaño.
- Complejidad.
- Impacto sobre el rendimiento del producto (uso de memoria y CPU)
- Recursos disponibles para efectuar el cambio (humanos y materiales)
- Relación con otros cambios ya aprobados y en progreso.
- Tiempo estimado para completar el cambio.
- Relación con las políticas de la empresa (satisfacción del cliente, competitividad, etc.)
- Existencia de alternativas.

Otras fuentes proponen (Álvarez Chang & Cortina Blanco, 2006)

- Alcance del cambio.
- Costos y recursos necesarios.
- Beneficios vs costos.
- Negociación con el cliente.

## SEGUIMIENTO DE LOS CAMBIOS. GESTIÓN DE PROBLEMAS

Cuando un cambio es aprobado debido a un problema se debe hacer un seguimiento del mismo. A este proceso se le denomina Gestión de Problemas.

La Gestión de Problemas consiste en gestionar la evolución de los problemas detectados sobre el software desarrollado, ya sea los que se detectan en etapa de pruebas como los que detecta el usuario. Se considera una actividad complementaria al control de cambios.

Las tareas que conlleva son: (Antonio, 2001)

- Admisión y registro de notificaciones de problemas (vía llamadas telefónicas, correo electrónico, herramienta específica)
- Asignación del problema a una persona responsable.
- Asociación de información al problema y mantenimiento de la misma en una Base de Datos de Problemas.
- Monitorización del estado del problema.
- Registro de actividades efectuadas para resolver un problema.
- Generación de informes acerca de los problemas.
- Resolución de interrogaciones sobre la Base de Datos de Problemas.
- Análisis estadísticos acerca de los problemas, como por ejemplo el estudio de correlaciones entre problemas y componentes.

De estos resulta interesante almacenar los siguientes datos:

- Descripción.
- Severidad.
- Urgencia o prioridad.
- Causa del problema (omisiones en el análisis, error entrada, falta de experiencia,...)
- Solución al problema.
- Módulos afectados.
- Persona que lo notificó.
- Persona responsable.
- Fechas de notificación, resolución, etc.
- Fase/etapa en la que se originó el problema.
- Fase/etapa en la que se detectó el problema.

### GENERACIÓN DE INFORMES DE ESTADO DE LA CONFIGURACIÓN

Esta tarea también es denominada contabilidad de estado y es la tarea que responde a las siguientes preguntas (Pressman R. S., 2002) (Antonio, 2001)

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

Generar Informes de Estado de la Configuración (IEC) reviste una importancia vital para el desarrollo exitoso del proyecto ya que ayuda a mejorar posibles problemas de comunicación entre los desarrolladores del proyecto. Para su realización es necesario ir registrando toda la información necesaria de lo que va ocurriendo en el proyecto con cada uno de los ECS y generando los informes necesarios sobre el estado de estos para ser informados a quienes fuese necesario (preferentemente a todos los implicados en el proyecto).

Por lo tanto esta tarea implica la realización de estas actividades básicas (Antonio, 2001)

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

Los productos de esta tarea son entonces (Antonio, 2001)

- Registros.
- Informes.

Los IEC se suelen almacenar en la Base de Datos del Proyecto de forma que los encargados del desarrollo o del mantenimiento del software puedan acceder a la información de Cambio cada vez que lo necesiten. Lo que hace tan importante esta tarea es que permite: (Antonio, 2001)

- Mantener la continuidad del proyecto. Se trata de permitir que el proyecto siga adelante cuando, por ejemplo, el jefe de proyecto deja la empresa.
- Evitar la duplicidad del trabajo. Si no se guarda información acerca de lo que ya se ha hecho, se puede estar repitiendo el trabajo ya hecho.
- Evitar caer en los mismos errores una y otra vez.
- Ser capaces de repetir aquello que salió bien.
- Ayudar a encontrar las causas de un fallo.

Básicamente la información necesaria para la realización de esta tarea proviene de las otras 3 tareas de la GCS.

Resulta difícil a veces seleccionar cual información capturar y cual no. Ya que la cantidad y tipo de la información a capturar depende de las características específicas del proyecto, de su tamaño y su complejidad. Sin embargo al comienzo del proyecto no se sabe que información puede llegar a ser útil por lo que es preferible guardar la mayor cantidad posible de información. Aun así el Plan de GCS deberá indicar la información mínima a capturar.

Además es necesario decidir donde almacenar esta información capturada. La mejor solución parece ser almacenarla en la base de datos del proyecto y gestionarla con una herramienta automatizada.

Esta información contenida en la base de datos para conseguir informes de estado se puede utilizar además para facilitar otras tareas no contenidas dentro de la GCS como son el post-mortem del proyecto y la realización de estimaciones para futuros proyectos.

## REGISTROS

Algunos ejemplos de registros que pueden mantenerse durante el proceso de desarrollo son: (Antonio, 2001)

- **Registro de elementos de configuración:** Conteniendo toda la información relativa a los diferentes elementos de configuración.
- **Registro de Líneas base:** Conteniendo toda la información relativa a cada línea base: Nombre, Fecha de establecimiento, Elementos de configuración que la componen.
- **Registro de Solicitudes de cambios:** El tipo de información que se suele mantener acerca de cada solicitud de cambio es la recogida a través del formulario de Solicitud de Cambio, incluyendo:
    - Código de solicitud.
    - Información acerca del solicitante.
    - Descripción del cambio.
    - La documentación que apoya la petición de cambio, por ejemplo, una referencia a un Informe de Incidencia.
    - La resolución o disposición acerca del cambio (aprobado, aplazado, denegado,…)
- **Registro de Cambios:** El tipo de información que se suele mantener acerca de cada cambio es la recogida a través del Informe de Cambio, la Orden de

Cambio, el proceso de Gestión de Problemas, etc. Puede contener información acerca de:

- Solicitud del cambio a la que corresponde.
- Evaluación del cambio:
  - Coste.
  - Esfuerzo.
  - Tiempo.
- Soluciones alternativas.
- Plan de implementación del cambio.
- Restricciones y criterios de revisión.
- Impacto sobre la configuración:
  - Líneas base afectadas.
  - Elementos de configuración afectados.
  - Versiones afectadas.
- Historia del cambio: Puesto que un cambio es algo que evoluciona, es necesario mantener una historia de cada cambio. En cuanto a los datos que se deben mantener en la historia de un cambio, se pueden considerar:
  - Fecha de la solicitud de cambio.
  - Fecha de aprobación del cambio.
  - Fecha de rechazo del cambio.
  - Fecha de cancelación del cambio.
  - Fecha de implementación del cambio.
  - Fecha de cierre del cambio.

- **Registro de Incidencias:** El tipo de información que se suele mantener acerca de cada incidencia es la recogida a través del Informe de Incidencia, del tipo:
  - Información del incidente.
  - Resultado de la Incidencia.
    - Disposición del CCC.
    - Número de la solicitud de cambio a la que dio lugar (si es aplicable).
    - Número de Formulario de Seguimiento de Documentación (si es aplicable).
    - Número de Notificación de Cambio asignada (si es aplicable).
  - Historia:
    - Fecha de la incidencia.

        - Fecha de resolución acerca de la incidencia.
        - Fecha de cierre de la incidencia.
- **Registro de Modificaciones del Código:** Puede contener información del tipo:
    - Número de identificación de la modificación.
    - Descripción de la modificación.
    - Número de notificación de cambio a la que corresponde (si es aplicable).
    - Número de solicitud de cambio a la que corresponde (si es aplicable).
    - Nombre de los módulos afectados.
    - Versiones modificadas.
    - Persona responsable de la modificación.
    - Fecha de inicio.
    - Fecha de terminación.
- **Registro de Modificaciones sobre bases de datos:**
    - Número de identificación de la modificación.
    - Descripción de la modificación.
    - Número de notificación de cambio a la que corresponde (si es aplicable).
    - Número de solicitud de cambio a la que corresponde (si es aplicable).
    - Base de Datos modificada.
    - Número de versión modificada.
    - Registros modificados.
    - Persona responsable de la modificación.
    - Fecha de inicio.
    - Fecha de terminación.
- **Registro de Modificaciones sobre documentación:**
    - Número de identificación de la modificación.
    - Descripción de la modificación.
    - Número de formulario de seguimiento de documentación al que corresponde.
    - Documento modificado.
    - Número de versión modificada.
    - Persona responsable de la modificación.
    - Fecha de inicio.
    - Fecha de terminación.

- **Registro de Liberaciones y Variantes:** Su objetivo es describir la composición y estado de una versión liberada del producto:
    - Código de liberación o variante.
    - Fecha de liberación.
    - Elementos de configuración que la componen.
    - Versión de los elementos de configuración que la componen.
    - Medio en el que se encuentran.
    - Diferencias con la liberación anterior.
    - Cambios incorporados.
    - Cambios pendientes.
- **Actas de las reuniones del Comité de Control de Cambios:**
    - Fecha.
    - Lista de miembros asistentes.
    - Propósito de la reunión.
    - Acciones del CCC:
        - ECS etiquetados.
        - Líneas base revisadas/cambiadas.
        - Disposición de Solicitudes de Cambio, Informes de Incidencias, Notificaciones de Cambio, Formularios de Seguimiento de Documentación, Informes de Cambios, etc.
        - Líneas base aprobadas.
    - Resultados de las auditorías:
        - Deficiencias detectadas.
        - Plan de resolución de las deficiencias encontradas.
        - Recomendaciones.

## INFORMES

Los informes se pueden categorizar como: (Antonio, 2001)

- Planificados.
- Bajo demanda.

Algunos ejemplos de los tipos de informes que se pueden generar son: (Antonio, 2001)

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

- **Inventario de elementos de configuración**: Es para ofrecer visibilidad sobre el contenido de las bibliotecas de proyecto.
- **Informe de incidencias:** Es un resumen del estado en que se encuentran todas las incidencias originadas durante un determinado período de tiempo y las acciones a las que han dado lugar.
- **Informe de modificaciones:** Es un resumen de las modificaciones que se han efectuado en el producto software durante un determinado período de tiempo.
- **Informe de diferencias entre versiones:** Resumen de las diferencias entre las sucesivas versiones de un elemento de configuración.

En cualquier caso, al comienzo de cada proyecto será necesario decidir qué tipo de registros se van a mantener y qué tipo de informes se van a generar y para quién.

## AUDITORÍAS DE LA CONFIGURACIÓN

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

Las ACS son el complemento de las tareas anteriores de GCS con el objetivo de asegurar que un cambio ha sido implementado correctamente (Pressman R. S., 2002)

Se realizan dos tipos de actividades en este sentido:

- Revisiones técnicas formales.
- Auditorías de Configuración del Software.

La Revisión Técnica Formal debe realizarse para cualquier cambio que no sea trivial y se centra en la corrección técnica del ECS que ha sido modificado para determinar su consistencia con otros ECS, las omisiones o posibles efectos secundarios, etc.

La Auditoría de Configuración del Software complementa la Revisión Técnica Formal comprobando características que generalmente no se tienen en cuenta en la revisión. Específicamente se refiere a responder las preguntas: (Antonio, 2001) (Pressman R. S., 2002)

1. ¿Se ha hecho el cambio especificado en la orden de cambio? ¿Se han incorporado modificaciones adicionales?
2. ¿Se ha llevado una Revisión Técnica Formal para evaluar la corrección técnica?
3. ¿Se ha seguido el proceso del software y se han implementado adecuadamente los estándares de ingeniería del software?
4. ¿Se han "resaltado" los cambios en los ECS? ¿Se han especificado la fecha del cambio y el autor? ¿Reflejan los cambios los atributos del objeto de configuración?
5. ¿Se han seguido procedimientos de GCS para señalar el cambio, regístralo y divulgarlo?
6. ¿Se han actualizado adecuadamente todos los ECS relacionados?

Esta es la actividad de la GCS más costosa pues requiere personal experimentado, con gran conocimiento del proceso de desarrollo y que sin embargo sea ajeno al equipo técnico para poder mantener la objetividad de la auditoría.

Dentro de ella se pueden diferenciar tres tipos de actividades (Antonio, 2001)

- **Revisiones de fase:** Se realizan al finalizar cada fase del desarrollo y su objetivo es examinar los productos de dicha fase. Las revisiones propias de la Gestión de Configuración son aquellas en las que se establecerán las líneas base. El objetivo de esta revisión es descubrir problemas, no comprobar que todo está bien. Hay que ser capaz de desenmascarar los problemas ocultos y sutiles, no sólo los que son obvios.
- **Revisiones de cambios:** Se realizan para comprobar que los cambios aprobados sobre una línea base se han realizado correctamente.
- **Auditorías:** Se realizan al final del proceso de desarrollo de software y su objetivo es examinar el producto en su conjunto.

Las revisiones deben realizarse continuamente durante el proceso de desarrollo, no al final de este.

La tarea de revisión implica tres tipos de funciones (Antonio, 2001)

- Verificar que la configuración actual del software se corresponde con lo que era en fases anteriores. Debe haber correspondencia y trazabilidad entre los elementos de configuración que aparecen en una línea base y los que

aparecen en las líneas base que la preceden y que la siguen. La verificación se realiza con respecto a la línea base precedente.

- Validar que la configuración actual del software satisface la función que se esperaba del producto en cada hito del proceso de desarrollo. La validación se realiza con respecto a los requisitos del sistema.
- Valorar si una determinada línea base, teniendo en cuenta los resultados de la verificación y validación, y otro tipo de comprobaciones, se debe considerar aceptable o no.

El papel que juegan las revisiones en el proceso de Gestión de Configuración es el siguiente (Antonio, 2001)

- Los productos generados durante el proceso de desarrollo se agrupan, al llegar determinados hitos, en una “línea base pendiente de aprobación”.
- Tiene lugar la revisión de fase.
- Si en la revisión se encuentran deficiencias en los ECS que componen la línea base, se generan los correspondientes Informes de Problemas, o Informes de Discrepancias, y se entregan al CCC, el cual recomienda ciertos Cambios sobre la Configuración.
- Una vez implementados los cambios propuestos, se pasa a una nueva revisión de cambios en la que se comprueba si los cambios se han implementado correctamente.
- Si en la revisión no se encuentra ninguna deficiencia, se declara “aceptable” la “línea base pendiente de aprobación”. Si el CCC está de acuerdo con la resolución de los revisores, la línea base se aprueba.

De las auditorías podemos mencionar los siguientes clasificaciones: (Antonio, 2001)

- **Auditoría Funcional:** Cuyo 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, se puede afirmar que el Elemento de Configuración satisface los requisitos que se impusieron sobre él.
- **Auditoría Física:** Cuyo objetivo es verificar la adecuación, completitud y precisión de la documentación que constituye las líneas base de diseño y del producto. Se trata de asegurar que representa el software que se ha codificado y probado. Tras la auditoría física se establece la línea base de Producto.

Tiene lugar inmediatamente después de haberse superado la auditoría Funcional.

- **Revisión Formal de Certificación:** Cuyo objetivo es certificar que el Elemento de Configuración del Software se comporta correctamente una vez que éste se encuentra en su entorno operativo.

## APLICACIÓN DE LAS POLÍTICAS DE GCS

Cada proyecto tiene sus características especiales que lo definen y que influyen de manera definitiva en las políticas de GCS que se establecen en su Plan de Gestión de la Configuración. Sin embargo puede ser de ayuda exponer ejemplos de políticas de gestión de la configuración y cambios establecidos en proyectos a nivel mundial conjuntamente con las características del proyecto ya que las políticas establecidas para características específicas se pudieran reutilizar para proyectos con las mismas características. (Burrows, 1999)

Sin embargo prácticamente ningún proyecto expone la documentación de su expediente de proyecto al público lo cual hace muy difícil saber con exactitud las políticas aplicadas en el Plan de Gestión de la Configuración o el documento homólogo que utilicen para el caso.

Los 3 ejemplos que a continuación se exponen constituyen proyectos importantes dentro del campo de acción para los que se han diseñado.

### EJEMPLO DE MANGOS

Massive Network Game Object Server (MaNGOS) o “Servidor de Objetos de Juego Masivo en Red” es un proyecto para la creación de una plataforma servidora de juegos masivos en línea desarrollada en C++ por una comunidad de poco mas de 5000 miembros de todos los países. El proyecto lleva casi 3 años de desarrollo en los que ha producido aproximadamente 15 versiones estables del producto. (MaNGOS Community)

El proyecto esta enfocado a obtener las soluciones más óptimas posibles para el código fuente que desarrollan, escribiéndolas de manera comprensible, detallada y elegante para tributar a la facilidad de entendimiento de la solución y a un estilo estándar.

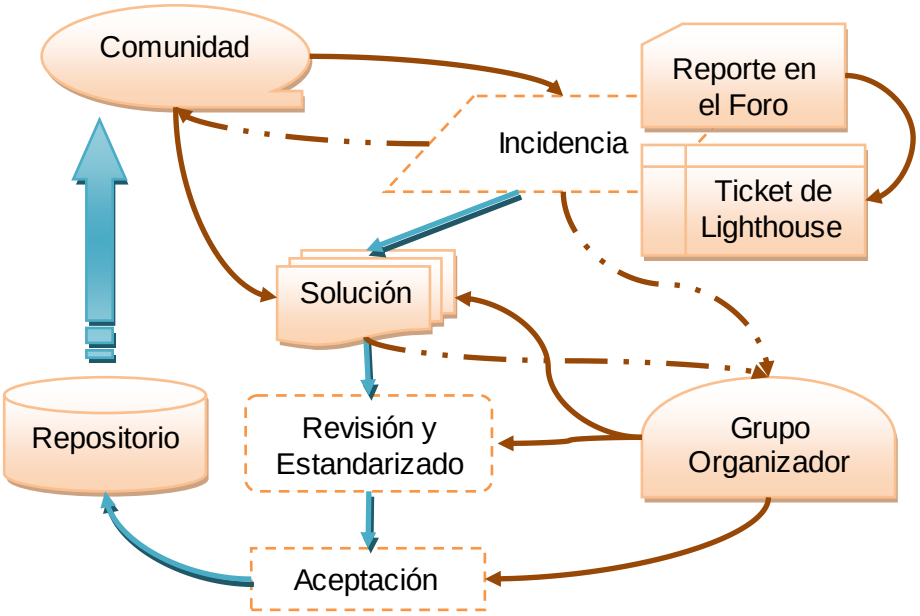
El equipo está conformado por un líder de proyecto quien es a la vez arquitecto, planificador y administrador de la configuración, un equipo de organización que se encarga de gestionar la información, realizar las decisiones importantes sobre el proyecto y apoyar al líder en sus tareas, y el resto de la comunidad. La comunidad también está dividida entre desarrolladores activos, moderadores del foro, desarrolladores ocasionales y usuarios.

El proyecto se divide en 5 componentes principales: el framework propio que es la base para la aplicación y contiene clases generales, el sistema de juego que son las clases que definen las reglas del juego, la base de datos, la aplicación que autentica y la aplicación que mantiene el juego.

La documentación del proyecto se encuentra alojada en una wiki pública (MaNGOS Community, 2009) y el conocimiento y métodos en hilos explicativos del foro de la comunidad. Usan un repositorio de Git con una copia modelo para cada uno de los integrantes del grupo organizativo del proyecto y una copia base central. Para el seguimiento y control de incidencias realizan una combinación de reporteado eficiente en el foro y gestión de incidencias usando una herramienta propia llamada Lighthouse que está basada en Trac.

El flujo de trabajo de control de cambios lo tienen definido de la siguiente manera:

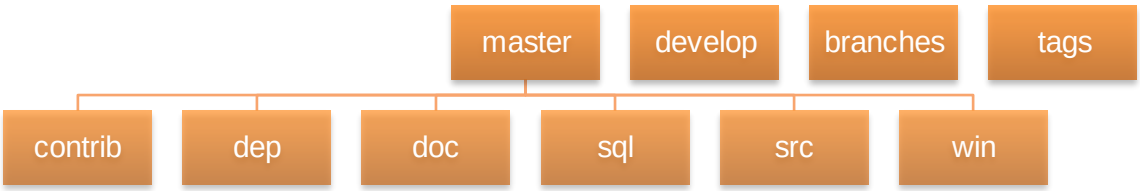
*ILUSTRACIÓN 5* ***FLUJO DE CONTROL DE CAMBIOS DEL PROYECTO MANGOS***




Para el nombrado de archivos usan notación acamellada mientras que para el nombrado de directorios como identificativos palabras completamente en minúscula.

La distribución de archivos dentro de su repositorio es como sigue:

*ILUSTRACIÓN 6* ***DISTRIBUCIÓN DEL REPOSITORIO DE MANGOS***




A pesar de que usan una herramienta de control de versiones de modelo distribuido la usan en “modo centralizado”. Esto significa que poseen un repositorio al que llaman maestro (master) hacia el cual sólo el equipo de organizadores puede subir y que usan como modelo estable. Poseen además una copia de desarrollo (develop) para cada organizador y el resto lo conforman las ramas que tiene la comunidad. Además de ello tienen un espacio especial para los liberados (tags).

Tanto el maestro como las copias de desarrollo y las ramas poseen la misma estructura. Un directorio para las herramientas de terceros que les son necesarias (contrib), un directorio para las herramientas de las que depende el código desarrollado (dep), un directorio para la documentación mínima para el usuario (doc), un directorio para los script de base de datos necesarios (sql), un directorio para el código fuente (src) el cual tiene dentro un directorio para cada módulo y un directorio para los archivos de proyecto (win).

Para informar del estado usan un subforo especial en el foro de la comunidad al cual solo pueden agregar nuevos hilos los organizadores del proyecto, aunque una vez creados los informes en nuevos hilos de discusión se les permite a la comunidad comentarlos. (MaNGOS Community)

El establecimiento de líneas base lo realizan planificando con antelación una serie de funcionalidades que se deben cumplir, cuando estas estén completamente funcionales se genera una nueva versión, se le asigna un identificador a esta versión, se marca como “cumplido” el hito relacionado con esas funcionalidades y se crea una nueva línea base con todo el código fuente de dicha versión que se usará como base para desarrollos futuros. También generan nuevas líneas base cuando fusionan con otras ramas con mejores funcionalidades o fusionan otras herramientas de terceros que aportan mejoras. En este caso se completa un hito no planificado pero no se genera una nueva versión.

## EJEMPLO DE MERINDE

MeRinde es el proyecto de Metodología de la Red Nacional de Integración y Desarrollo de Software Libre del Centro Nacional de Tecnologías de Información (CNTI) del gobierno de Venezuela. Este proyecto propone un estándar abierto para el proceso de desarrollo de software orientado a planes específicamente diseñado para los proyectos libres del CNTI. (CNTI)

Con el proceso de desarrollo y con las plantillas que proponen se busca a su vez estimular la transferencia del conocimiento entre las comunidades desarrolladoras de software libre, con lo cual no solo se pretende que sea compartido los códigos fuente de los sistemas sino que también se compartan la documentación como guía de referencia para mejoras por terceros al sistema o para que sirva como modelo a otras comunidades para el desarrollo de sus propios sistemas.

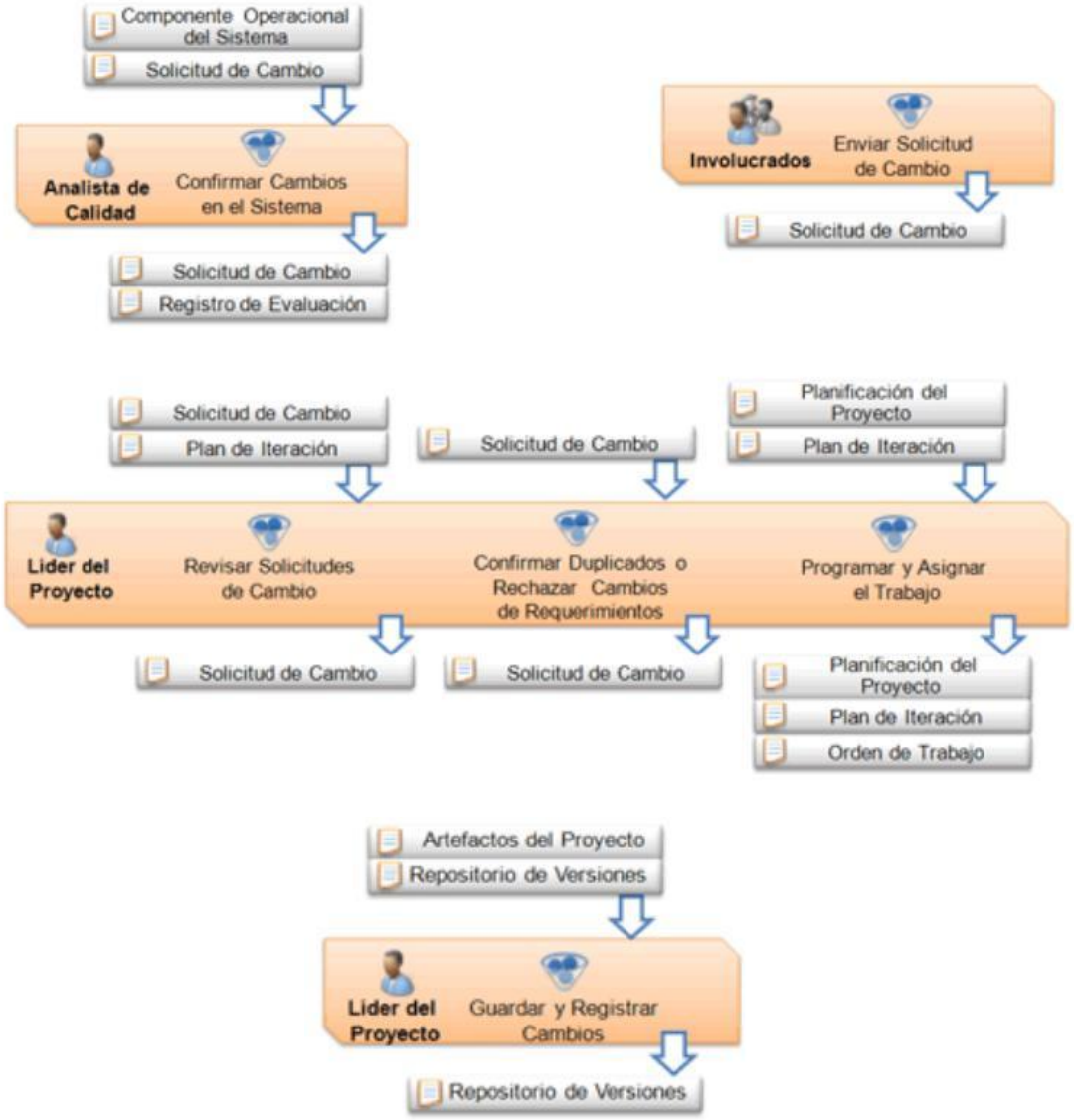
Dicho en otras palabras, MeRinde propone todo una metodología para los proyectos libres de la CNTI dentro de la cual existe por supuesto un área para la GCS.

La metodología que propone esta fuertemente influida por RUP, de ahí que mantengan muchos nombres tomados de ella y los roles sean prácticamente los mismos.

A pesar de que no proponen específicamente una herramienta de control de versiones ni una herramienta de gestión de proyecto en su metodología, si es importante destacar que recomiendan el uso de una herramienta llamada FindeForge que está alojada en un dominio venezolano y que es propia de Rinde. También recomiendan el uso de un gestor central de recursos de las tecnologías de información y documentación de consulta llamado ARTIC, también propio. (Equipo MeRinde)

El modelo que proponen para las políticas de control de cambios es como sigue:

*ILUSTRACIÓN 7* ***FLUJO DE CONTROL DE CAMBIOS DE MERINDE*** (EQUIPO MERINDE)




MeRinde tampoco propone una distribución específica dentro de la biblioteca de software escogida por el proyecto ni un sistema de establecimiento de líneas bases ni un sistema de reportes específico. Aunque si proponen la creación de todos los puntos mencionados pero dejan a cada proyecto decidir que hacer al respecto.

## EJEMPLO DE RATIONAL 2003

Conocido popularmente como la “ayuda del rational” es una aplicación web que explica el contenido de la metodología del Proceso Unificado de Rational (RUP). Esta metodología, robusta y muy completa, es una de las más populares y conocidas a nivel internacional. No es una metodología Ágil, por lo que muchos la han despreciado pero también cuenta con muchísimos seguidores y en general es ascendente de muchas otras metodologías que se han basado en ella.

Como metodología al fin, RUP propone una serie de políticas y procesos generales que pueden ser adaptados a una u otra situación y que pueden ser usados o desechados según se necesite.

El flujo de cambios que propone esta dividido en dos partes seguidas una de la otra: Control de Pedidos de Cambio y Control de Cambios:

*ILUSTRACIÓN 8* ***FLUJO DE CONTROL DE PEDIDOS DE CAMBIO EN RUP***

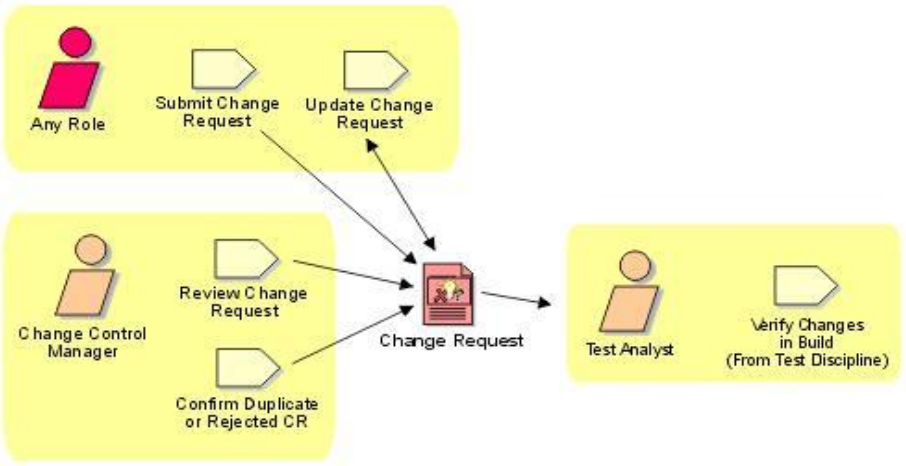
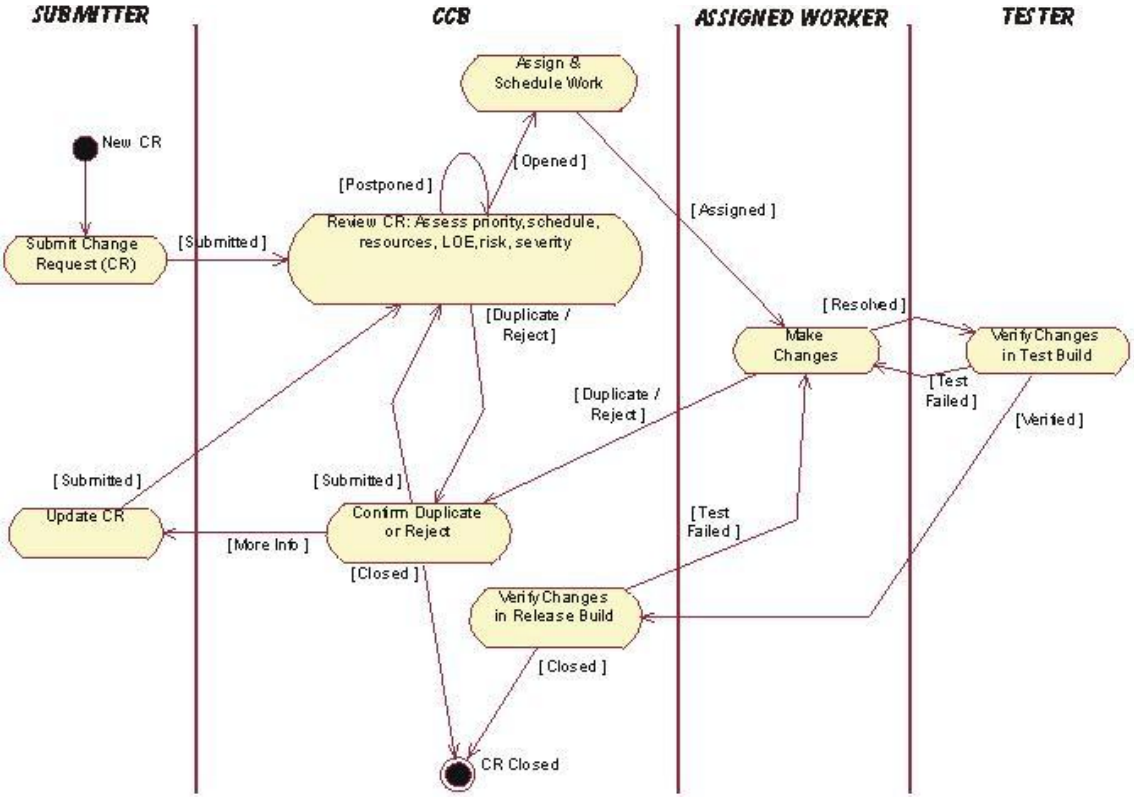



*ILUSTRACIÓN 9* ***FLUJO DE CONTROL DE CAMBIOS EN RUP***




RUP tampoco propone al usuario ninguna herramienta de control de versiones específica, se limita a señalar la necesidad de una.

## HERRAMIENTAS PARA LA GESTIÓN DE LA CONFIGURACIÓN

Existen en el mercado muchísimas herramientas diseñadas para facilitar el trabajo de la gestión de la configuración. Algunas están enfocadas a un aspecto específico de la gestión de la configuración o inclusive enfocadas a la integración de las tareas de gestión de la configuración con otras tareas de la gestión de proyecto, denominadas aplicaciones de gestión de ciclo de vida. Estas aplicaciones incrementan la productividad pues eliminan en cierta medida las barreras entre la colaboración fácil y el flujo normal de la información. Existen también herramientas denominadas específicamente como herramientas de gestión de proyecto que cubren todas, o al menos muchas, de las necesidades de la gestión de la configuración.

Una de las herramientas más completas es Telelogic® Synergy™ de IBM® Telelogic, anteriormente denominada SYNERGY/CM. Ésta brinda muchas facilidades ya que posibilita la gestión automática de la configuración de software. Sus características son definidas como: *"Telelogic Synergy, una solución de gestión de la configuración basada en tareas, permite a los equipos de desarrollo de software trabajar más rápida y fácilmente aumentando la comunicación y la colaboración. Telelogic Synergy acelera la gestión de releases y los procesos de gestión de builds, maximiza la eficacia de los recursos de desarrollo limitados y une los equipos de desarrollo distribuidos. Telelogic Synergy proporciona un eficaz repositorio distribuido y un enfoque probado de flujo de trabajo orientado a equipos para el desarrollo de software"*

*"Telelogic Synergy ofrece capacidades avanzadas para ayudar a las organizaciones a mejorar la calidad de sus procesos de desarrollo, flujo de trabajo y comunicación"* (IBM)

Telelogic IBM brinda además un grupo de herramientas perfectamente integrables con Synergy que abarcan casi toda la gama de tareas del desarrollo de software, control de cambios con Telelogic Change, eficiencia personal con Telelogic Dashboard, administración de pruebas con Telelogic Tester, entre otras.

Sin embargo a pesar de las bondades que ofrece, este software es privativo y tiene un precio elevado. Su uso gratuito es por un periodo de tiempo limitado de 30 días lo que lo convierte en una herramienta con pocas probabilidades de ser usada por pequeños desarrolladores o grupos con pocos recursos.

## HERRAMIENTAS DE GESTION DE PROYECTOS

Las herramientas denominadas de gestión de proyecto se pueden categorizar atendiendo a la filosofía usada en: “privativo” o “de código abierto” y atendiendo a la arquitectura usada en: “basado en web” o “de escritorio”.

Para evaluar las herramientas de gestión existentes se hace necesario definir un conjunto de parámetros que sirvan de guía puntual. Sin embargo no existen guías para la selección de parámetros de evaluación de este tipo.

Dichos parámetros deben estar sobre todo dirigidos a las bondades que los hacen necesarios, las funcionalidades que proveen y a las políticas de su uso que, por lo general, son las que deciden con más fuerza sobre el uso de una u otra herramienta.

Los parámetros elegidos en esta investigación son:

- **Gestión de Proyecto (GP):** Que permita la identificación completa y correcta del proyecto que alberga, sus integrantes, objetivo, etc.
- **Software Colaborativo (SC):** Que esté enfocado a la colaboración entre proyectos o al menos que permita su uso como multiproyecto o con subproyectos.
- **Sistema de Control de Incidencias (SCI):** O control de errores, que provea un sistema para controlar las incidencias del ciclo de vida: mejoras, errores, etc.
- **Portafolio del Proyecto (PF):** O gestión del Expediente de proyecto específica para este.
- **Gestión de Recursos (GR):** Que provea gestionado de recursos humanos, materiales y temporales o al menos dos de esos tipos de recursos.
- **Gestión Documental (GD):** Que provea una solución para la gestión de documentos independiente del SCM.
- **Gratuito (F):** Que sea de uso funcional completamente gratuito, o al menos que tenga una versión con todas las funcionas que queremos evaluar que sea gratuita. Este parámetro será el de mayor fuerza.
- **Publicación Personal (PP):** Que provea un espacio personal por desarrollador. Este parámetro será opcional y de poco valor.
- **Sistema de Comunicación (CM):** Que provea facilidades de comunicación para el equipo de desarrollo, ya sea con funciones integradas o con integraciones con otras aplicaciones. Este parámetro será opcional y de poco valor.

Algunas de las herramientas de Gestión de Proyecto se listan en la siguiente tabla con sus funcionalidades:

| **Código abierto de Escritorio** | **GP** | **SC** | **SCI** | **PF** | **GR** | **GD** | **F** |
|---|---|---|---|---|---|---|---|
| Open Workbench | X | | | | X | | ** |
| GanttProject | X | | | | X | | X |
| **Código abierto basado en Web** | **GP** | **SC** | **SCI** | **PF** | **GR** | **GD** | **F** |
| dotProject | X | | X | | | X | * |
| Project.net | X | X | X | X | X | X | |
| Trac | X | X | X | | | | X |
| SharpForge | X | X | X | | | X | |
| Redmine | X | X | X | | | | X |
| **Privativo de Escritorio** | **GP** | **SC** | **SCI** | **PF** | **GR** | **GD** | **F** |
| RationalPlan | X | X | | X | X | X | |
| Planisware 5 | X | X | X | X | X | X | |
| **Privativo basado en Web** | **GP** | **SC** | **SCI** | **PF** | **GR** | **GD** | **F** |
| Trac+ | X | | X | | X | | |
| Teamwork | X | X | X | X | X | X | |

Observemos en la tabla anterior que el promedio de cantidad de funcionalidades por categoría de las herramientas es superior en las herramientas privativas, constituyendo estas sin lugar a dudas las más funcionales. Podemos observar también que las herramientas basadas en tecnología web son, como promedio, más funcionales que las de escritorio. Podemos y debemos tener en cuenta además que las herramientas basadas en web son independientes del sistema operativo.

La opción más clara debiera ser sin dudas usar herramientas basadas en web y privativas sin embargo estas son en su totalidad herramientas no gratuitas por lo que su selección y uso está más limitado a los grupos de desarrollo con recursos. Además el uso de herramientas privativas conlleva también al estudio de posibilidades que brinde la licencia de esta, la que usualmente incluye comisiones mayores si el software que desarrolla el proyecto tiene mercado.

En cuanto a las herramientas de código abierto debemos señalar que algunas de ellas también son herramientas no gratuitas que solo incluyen períodos cortos de prueba (de 30 días generalmente) gratuitos después de los cuales hace falta pagar una licencia para continuar usándolos o en algunos casos la versión gratuita no contiene todas las funcionalidades necesarias para el trabajo profesional con que cuenta la versión pagada.

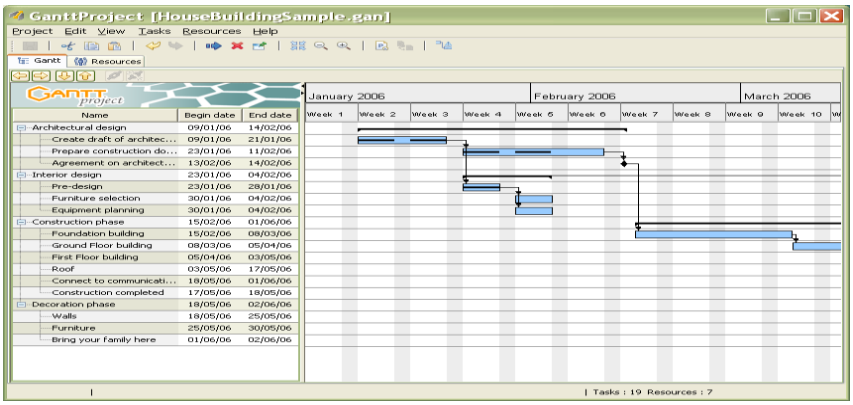
Fuera de la privatización y/o las licencias pagadas solo quedan 4 herramientas de la lista cuyo uso es completamente gratuito. GanttProject, dotProject, Trac y Redmine.

## GANTTPROJECT

Según sus propios desarrolladores GanttProject es una herramienta de gestión de proyecto de escritorio escrita en Java bajo licencia GPL y probada para Windows, Linux y MaxOS cuyas principales funcionalidades incluyen: (The GanttProject Team)

- Jerarquía y dependencias de tareas.
- Diagrama de Gantt.
- Gráfico de carga de recursos.
- Generado de diagramas PERT.
- Reportes en PDF y en HTML.
- Importado y exportado desde Microsoft Project.
- Intercambio de datos con aplicaciones de hojas de cálculo.
- Basado en grupos de trabajo WebDAV.

*ILUSTRACIÓN 10* ***CAPTURA DE PANTALLA DE GANTTPROJECT***




Posee una interfaz gráfica muy amigable y su uso es bastante intuitivo. La instalación es también relativamente sencilla.

Como se puede observar GanttProject es una herramienta enfocada a la planificación de proyecto con funcionalidades bastante completas en este sentido pero inexistentes en cuanto a otras funciones necesarias en la gestión de proyectos.

## DOTPROJECT

La descripción puesta para dotProject por sus desarrolladores en el sitio de desarrollo colaborativo Sourceforge.net es: “una aplicación y un framework para la gestión de proyecto basada en web que incluye módulos para empresas, proyectos, tareas (con

diagrama de Gantt), foros, archivos, calendarios, contactos, incidencias, buró de ayuda, soporte multiusuario y multilenguaje, permisos por usuario y por módulo y temas." (SourceForge, Inc.)

dotProject es una herramienta web relativamente completa diseñada en PHP5 y usando MySQL como base de datos. Surgió en 2001 bajo licencia BSD primeramente y luego GPL a partir de la versión 2.0 en 2005, sus funciones de gestión y elementos básicos incluyen:

- Gestión empresarial (compañías, departamentos y contactos)
- Proyectos y tareas.
- Recursos (equipamiento, locales, etc)
- Lista de pendientes.

También incluye elementos adicionales como:

- Foros.
- Control de acceso vía ACL.
- Diagramas de Gantt.
- Reporteado.
- Historial de todas las actividades.
- Calendario.

Su interfaz es relativamente sencilla una vez que se entiende cuidadosamente pero puede abrumar al usuario que entra por primera vez por la cantidad de información por página que puede llegar contener la cual a veces desorienta.

*ILUSTRACIÓN 11* ***CAPTURA DE PANTALLA DE DOTPROJECT***

En noviembre de 2007 el equipo de desarrollo comenzó las planificaciones para una versión 3 usando el framework de Zend que debía convertirse en el mejor producto de su tipo a nivel mundial. En julio de 2008 salió a luz la versión 2.1.2 estable con la cual para agosto de 2008 llegaron a más de 11000 usuarios en el foro de dotProject y entre 500 y 700 descargas diarias. (SourceForge, Inc.) Sin embargo, a la espera de la versión 3 las estadísticas del proyecto cayeron casi a 0 en diciembre de 2008. Desde julio de 2008 a mayo de 2009 no ha habido cambios de más de 40 archivos en el repositorio del proyecto. En abril de 2009 hubo un pequeño pico de trabajo de unos 400 archivos pero volvió a caer en "tiempo muerto". Estas estadísticas sugieren que dotProject esta prácticamente descontinuado.

### TRAC

Según sus desarrolladores, Trac es "un sistema de seguimiento de incidencias y wiki mejorada para proyectos de desarrollo de software… Utiliza un acercamiento minimalista a la gestión de proyectos basada en web cuya misión es ayudar a los desarrolladores a escribir un buen software a la vez que se mantiene fuera del camino imponiéndose lo menos posible en las políticas y procesos de desarrollo escogidos por el equipo de trabajo" (Edgewall Software)

Es una herramienta web escrita en Python y por tanto independiente del sistema operativo que integra gestión de proyecto y gestión de incidencias pero que a la vez sirve como interface web a sistemas de control de versiones (SCV) soportando los SCV más populares como Subversion, Git, Mercurial, Bazaar y Darcs. Trac puede configurarse para funcionar sobre un servidor web usando tecnologías CGI, FastCGI ó mod_Python y alternativamente posee un servidor con protocolo propio. El sistema de Trac usa una base de datos cuyo gestor puede ser escogido libremente entre MySQL, PostgreSQL, Sqlite. La instalación de este sistema es relativamente sencilla pues funciona como un módulo de Python. Hasta mediados de 2005 estaba disponible bajo licencia GPL pero con la versión 0.9 la licencia cambió a BSD modificada, se distribuye libre y gratuito.

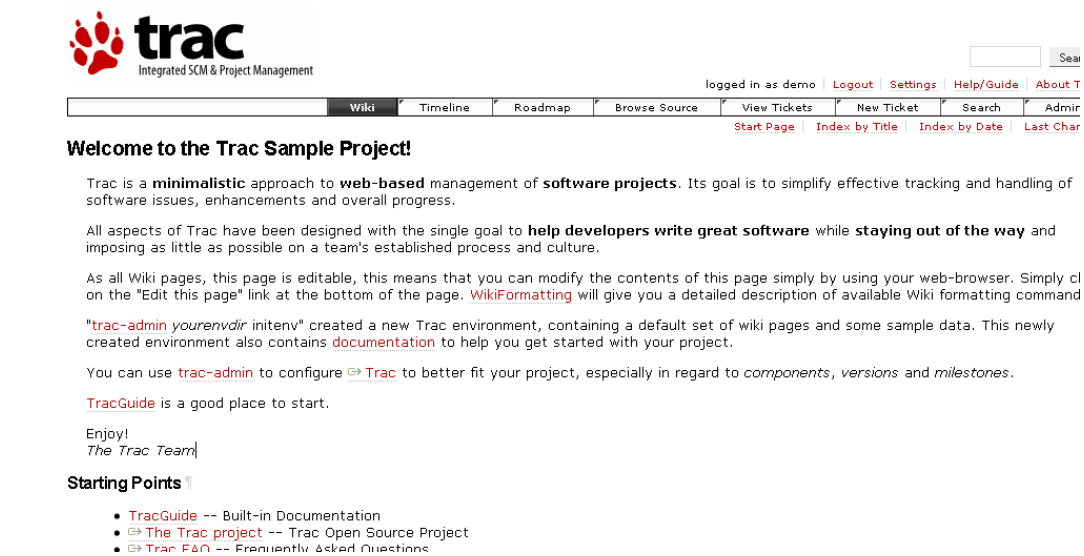
Entre sus principales funcionalidades están:

- Gestión de proyecto con:
  - Hoja de Ruta (roadmap)
  - Hitos.
- Sistema de tickets que incluye:
  - Control de incidencias.

    - Seguimiento de tareas.
- Fino control de permisos.
- Cronología de todas las actividades recientes.
- Wiki propia.
- Sistema de reportes personalizados.
- Interfaz web de SCV.
- Fuentes RSS.
- Notificaciones por correo.
- Soporte para múltiples proyectos.
- Ambiente extensible usando plug-in escritos en Python.
- Exportado a iCal.

La interfaz que proporciona es bastante intuitiva aunque puede ser modificada a gusto pues usa un motor propio de salida para la web llamado Genshi y permite personalizaciones de la interfaz.

*ILUSTRACIÓN 12* ***CAPTURA DE PANTALLA DE TRAC***




Se han reportado oficialmente más de 450 instituciones en todo el mundo que usan esta herramienta, entre las que se encuentran los laboratorios de Propulsión a Chorro de la NASA y el proyecto Integrated Runtime de Adobe (Edgewall Software)

Trac permite hipervínculos de información entre la base de datos de incidencias, el control de versiones y el contenido de la wiki lo cual lo convierte en una poderosa herramienta para referenciar información dentro del proyecto.

Una de sus más interesantes características es su extensibilidad, puede agregársele prácticamente de todo sin modificar el núcleo o base del sistema siempre que se programe en Pyhton. Otro punto de interés es el sistema de reportes realmente flexible.

Además del equipo de desarrollo principal, cuenta con una comunidad dedicada a escribir extensiones funcionales para el sistema que contiene muchísimas extensiones, modificaciones, scripts y temas. Algunas de estas extensiones son realmente interesantes como AccountManager que provee características para la gestión de las cuentas de usuario, LDAP-Plugin que provee autenticación contra servidores LDAP, AgileMethodsPlugin que provee funcionalidades para el trabajo con metodologías ágiles, TimeAndEstimationPlugin que provee características para el registro de tiempo y la planificación, etc. (The TracHacks Community)

Aunque Trac clasifica como una herramienta para múltiples proyectos la realidad es que el soporte para este tipo de instalación es pobre pues no posee por defecto un método de consulta o administración a nivel superior de todos los proyectos a la vez por crear instancias separadas para cada proyecto. Sin embargo esta funcionalidad puede extenderse usando un conjunto de plug-ins propuestos por la comunidad bajo la categoría de *"multi-project plug-ins"*.

## REDMINE

Según la descripción de sus propios desarrolladores Redmine es "…una flexible aplicación web para la gestión de proyectos. Escrita en Ruby usando el framework Ruby on Rails, es multiplataforma y puede usar varios tipos de base de datos." (Redmine)

Redmine es una herramienta de gestión de proyecto y seguimiento de errores escrita en Ruby usando el motor "Ruby on Rails", cuyo diseño está significativamente influenciado por Trac, inclusive tiene un sistema de empaquetado similar. Ella incluye ayudas visuales a la representación del proyecto y que soporta multiproyectos. Sus principales características funcionales son:

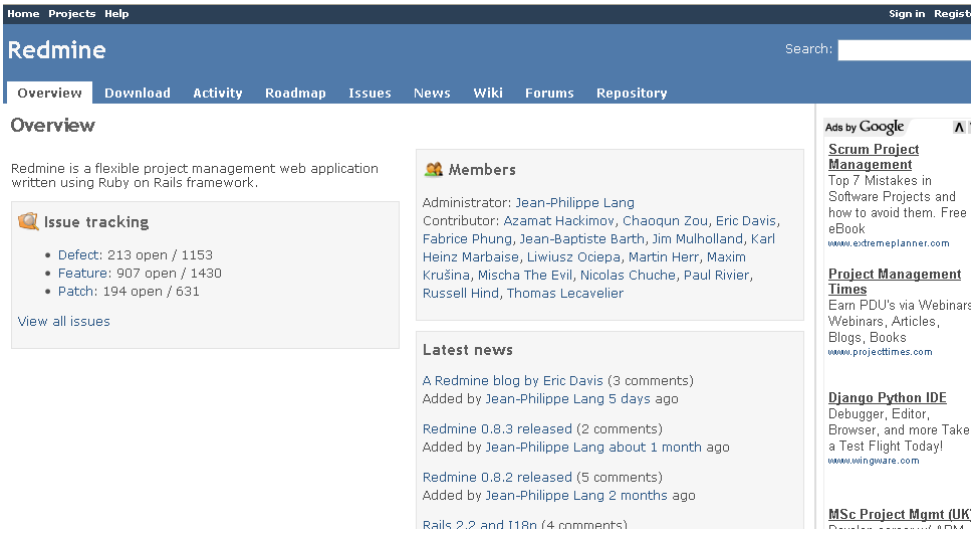
- Soporte para múltiples proyectos.
- Control de acceso flexible basado en roles.
- Sistema de control de incidencias.
- Gráficos de Gantt y calendarios.
- Notificaciones por correo.

- Wiki y Foro por proyecto.
- Seguimiento de tiempo.
- Integración con SCV (SVN, CVS, Git, Mercurial, Bazaar y Darcs)
- Soporte para autenticación por LDAP múltiple.

Se han reportado cerca de 30 instalaciones de esta herramienta alrededor del mundo entre las que se encuentran la forja de código de PhpBB y el “verano de código” de Gentoo (Redmine)

Redmine posee una interfaz muy amigable y atractiva, fuertemente influenciada por la arquitectura de la información de TRAC pero más estilizada cuyo tema predeterminado es muy agradable a la vista.

*ILUSTRACIÓN 13* ***CAPTURA DE PANTALLA DE REDMINE***




Quizás la característica más interesante que posee, además de la cantidad de funcionalidades que la instalación predeterminada contiene, es su habilidad para manejar múltiples proyectos de manera amena y sencilla.

La comunidad de Redmine provee también aproximadamente 20 plug-ins completados para extender las funcionalidades básicas y otros tantos en proceso de desarrollo. (Redmine)

Para la instalación de esta herramienta, sus desarrolladores proveen, además del código fuente, un ejecutable multiplataforma que instala Redmine por defecto con todas sus dependencias y herramientas de terceros necesarias como el motor On Rails de Ruby, el servidor web Apache2, etc. Sin embargo este ejecutable hace una

instalación expresa sin verificar la existencia anterior de estas otras herramientas necesarias.

Redmine es muy a menudo detraído precisamente por estar escrito en Ruby y depender de un framework poco popular.

## FORJAS DE PROYECTO

Las forjas de proyecto son un concepto consensual, o sea que no ha sido oficialmente definido sino que es una nominación que se ha popularizado y adoptado con el tiempo.

El término surgió cómo referencia a los prefijos o sufijos utilizados por los varios sistemas de alojamiento y gestión de proyectos en internet que fueron representativos en el concepto.

Cuando se habla de forja de proyecto o forja de software se refiere de manera general a una plataforma de colaboración que permite el trabajo de desarrollo colaborativo de proyectos sobre la red. Estas plataformas normalmente agregan un conjunto de aplicaciones y herramientas de gestión de proyecto, de desarrollo de software, de comunicación, etc. integradas mediante una interfaz web y que generalmente alojan varios proyectos independientes. Los desarrolladores, que se registran a los proyectos alojados, pueden hacer uso de las herramientas de la plataforma para el trabajo en el proyecto.

Algunas de las funcionalidades más comunes que proveen estas forjas de proyecto son:

- Gestión de código fuente (usando herramientas de control de versiones)
- Listas de correo.
- Foros de discusión.
- Wikis.
- Servicios de alojamiento y descarga de archivos.
- Sistemas de seguimiento y control de incidencias.

Algunas van un poco más adelante proveyendo de herramientas de planificación y otras similares.

Existen muchas forjas de proyecto en internet, la mayoría con alojamiento gratuito y con características funcionales muy interesantes para la gestión de proyecto en general. Sin embargo el uso de una de estas forjas implica el alojamiento del código

fuente del proyecto fuera de los dominios del equipo de desarrollo, característica muchas veces temida por las implicaciones subyacentes.

Algunos ejemplos de forjas de proyecto en internet:

- **SourceForge:** Archiconocida forja que en agosto de 2008 alojaba aproximadamente 180000 proyectos y cerca de 1.9 millones de usuarios registrados.
- **GoogleCode:** Una de las más recientes forjas cuyo acceso esta bloqueado para Cuba.
- **OpenFoundry:** Forja muy popular sobre todo en la zona de Asia. Su interfaz está completamente en caracteres orientales.
- **GNU Savannah:** Forja de software libre de GNU, su principal característica es que solo aloja proyectos bajo licencias libres.

Existe también software disponible para configurar forjas de proyecto propias con las cuales se pueden usar las ventajas de estas aplicaciones sin la desventaja de tenerlas fuera del dominio controlado por el equipo de desarrollo. Entre ellos se encuentran:

- SourceForge Enterprise Edition.
- GForge.
- Trac.
- FusionForge.

**SourceForge Enterprise Edition** (SFEE) es el software sobre el que funciona la forja de SourceForge.net. Originalmente concebido como software libre SFEE fue comercializado en su versión 2.5 y eventualmente relanzado con una licencia propietaria. El código base original fue utilizado por otros desarrolladores de forjas como base para sus propios software como Savane de GNU y GForge de uno de los desarrolladores del SourceForge original.

**GForge** surgió como una bifurcación de proyecto (del inglés “project fork”) de la última versión libre de SourceForge, la 2.6 y se fusionó con Debian-SF que era otra bifurcación de proyecto del mismo SourceForge mantenida libremente. Distribuida bajo licencia GPL. Sin embargo en junio de 2006 los desarrolladores de GForge sacaron a luz GForge Advanced Server (GForgeAS) que esta vez no era de código abierto y su uso gratuito estaba, aunque ilimitado en tiempo, limitado a una cantidad específica de proyectos. El sitio entero y el equipo de desarrollo dejaron a un lado el soporte del antiguo GForge para enfocarse únicamente en GForgeAS. La rama GPL de GForge

fue nuevamente bifurcada en FusionForge en febrero de 2009 en un intento renovado por revivir la rechazada base de código libre y fusionar algunas otras bifurcaciones libres de GForge en un solo proyecto.

**Trac** aunque comenzó y aun se considera solo una herramienta para el control y seguimiento de incidencias; la realidad es que con la configuración adecuada y los plug-ins de la comunidad específicos Trac puede convertirse fácilmente en una aplicación web con las características de una forja de proyectos. Las recomendaciones para ello pueden encontrarse en la lista de correo y en los foros de discusión de Trac en http://www.trac.edgewall.org. Existe también un proyecto comunitario libre que implementa una solución empaquetada para usar Trac como forja de proyectos integrada con algunas otras herramientas. Dicho proyecto se denomina OForge y aunque su última actualización es de diciembre de 2008, el equipo de desarrollo no ha cumplido con los plazos de 4 versiones estables al año y el proyecto apunta hacia la descontinuación; sin embargo es que es una herramienta muy interesante. (OForge, 2009)

**FusionForge** es un descendiente del SourceForge original y del GForge GPL relativamente nueva. Surge como alternativa libre al SFEE y al GForgeAS. FusionForge puede encontrarse en los repositorios de sistemas operativos libres como Debian, Ubuntu, RedHat y Fedora en un paquete llamado GForge, nombre que debe cambiar pronto a la nueva denominación pero que de momento es usable al cambiar el GForge original a GForgeAS.

## HERRAMIENTAS DE CONTROL DE VERSIONES

El número de herramientas para el control de versiones en el mercado es tan amplio como la existencia de herramientas de gestión y quizás más. Se les conoce comúnmente como “Sistemas de Control de Versiones” (SCV) o control de revisiones, nombre adoptado y adaptado del inglés “Source Code Management” (SCM). A pesar de ser control de versiones conceptualmente diferente a control de revisiones la jerga popular los utiliza indistintamente para referirse a lo mismo.

El control de versiones es popularmente conocido por herramientas independientes diseñadas para esa tarea. Sin embargo el control de versiones se encuentra incrustado en muchas aplicaciones de uso común como los procesadores de texto (Microsoft Word, OpenOffice.org Writer, etc), de hojas de cálculo (Microsoft Excel,

OpenOffice.org Calc) y en varios sistemas de control de contenido. También es una funcionalidad clave para aplicaciones de wiki como MediaWiki, TWiki, etc.

Las herramientas de control de versiones son cada vez más reconocidas como necesarias para proyectos de múltiples desarrolladores (EETimes, 2007)

Existen dos modelos de sistemas de control de versiones, uno **centralizado** y otro **distribuido**. Existen controversias sobre que modelo es mejor pero la realidad es que cada modelo resuelve un problema específico para el que prácticamente no puede ser reemplazado.

## EL MODELO CENTRALIZADO

Tradicionalmente los sistemas de control de versiones usan un modelo centralizado (SCVC) en el que todas las funciones de control de versiones se ejecutan en un servidor compartido. De esta manera el servidor compartido o servidor central es la copia de fe o copia buena de los cambios del producto al que se le hace control de versiones y todos los desarrolladores chequean esta copia para obtener la mejor versión a la vez que registran sus cambios en esa copia central.

Si dos desarrolladores acceden al mismo archivo sin un método de gestión de acceso pueden terminar ambos sobrescribiendo el trabajo del otro. Para prevenir esto los sistemas de control de versiones usan dos enfoques:

- Bloqueo de Archivos.
- Fusión de Versiones.

El **bloqueo de archivos** es el método más simple: Consiste en “bloquear” el acceso de escritura al archivo en cuestión mientras haya un desarrollador trabajando en él. De esta manera se asegura que en cada uno de todos los momentos dados solo haya una persona escribiendo en cada archivo por vez. Aun así todos tienen acceso de lectura al archivo por el tiempo que dura el bloqueo. Este enfoque tiene ventajas y desventajas. Puede proveer cierta protección contra conflictos de fusión difíciles cuando un usuario esta haciendo cambios radicales a muchas porciones de archivos o grupos de estos. Sin embargo si algún usuario deja archivos bloqueados exclusivamente por mucho tiempo otros desarrolladores pueden intentar sobrepasar el bloqueo para poder hacer modificaciones y conducir a problemas más graves.

La **fusión de versiones** es la más usada actualmente pues la mayoría de los sistemas de control de versiones permiten múltiples usuarios trabajando sobre el mismo archivo

a la vez. El primer desarrollador que “termine” sus modificaciones siempre tiene éxito en registrar sus cambios en el servidor compartido. El sistema provee entonces facilidades para “fusionar” los cambios en el servidor central de manera que los próximos usuarios que vayan a registrar sus cambios preserven los cambios hechos por el usuario que registró antes. Los próximos usuarios que registren cambios deberán tener en cuenta la fusión para asegurarse de que sus propias modificaciones son compatibles con los cambios hechos anteriormente y la fusión no introduzca errores lógicos en la funcionalidad propia del software desarrollado.

## EL MODELO DISTRIBUIDO

El control de versiones distribuido o descentralizado (SCVD) es una innovación relativamente reciente en el software de control de versiones. El SCVD tiene características que lo diferencian del SCVC más tradicional, a pesar de esto la línea divisoria entre SCVC y SCVD es algo borrosa ya que los SCVD pueden ser usados en “modo centralizado”.

Los SCVD usan un enfoque de “igual a igual” (peer-to-peer) opuesto al enfoque de “cliente-servidor” de los SCVC. Este enfoque de “igual a igual” significa que cada copia de la base de código en un “igual” es una copia de fidelidad, copia de fe o copia buena del repositorio. La sincronización es conducida intercambiando parches de igual a igual. Esto crea las diferencias fundamentales con los SCVC que pueden resumirse en:

- No existencia de una copia canónica o de referencia por defecto, solo copias de trabajo.
- La comunicación solo es necesaria cuando “empujas” o “tiras” cambios de otros iguales, por tanto las operaciones básicas son realmente rápidas.
- Cada copia de trabajo es efectivamente una copia de respaldo de la base de código y del historial de cambios proveyendo seguridad natural contra la pérdida de datos.

Existen en este enfoque usuarios especiales llamados “Tenientes” que son los encargados de decidir que ramas deben fusionarse y cuales no.

Este enfoque tiene algunas ventajas y desventajas con respecto al enfoque centralizado. Veamos algunas ventajas:

- Permite a los usuarios trabajar productivamente aunque no estén conectados a la red.
- Hace las operaciones más rápido pues no se involucra la red.
- Permite la participación en proyectos sin requerir permiso de las autoridades de proyecto y por tanto fomenta una cultura de méritos en vez de requerir un estado.
- Permite el trabajo privado de manera que se pueda usar control de versiones incluso de contenido que aún no se quiere publicar.
- Evita confiar en una única máquina física o en políticas externas de resguardo de datos, un accidente con el disco duro del servidor no es un evento preocupante con el enfoque distribuido.

Algunas desventajas pueden ser:

- El concepto de SCVD es más difícil de aprovechar por los desarrolladores pues requiere más conocimientos de infraestructura.
- No es posible cambiar o eliminar el historial por un solo grupo de personas lo cual puede ser un problema para equipos que quieren tener control total del historial.
- Los servidores centralizados son más favorecidos en ambientes corporativos pues se logra la gestión de acceso y visibilidad de código más fácilmente, por tanto en este sentido se refuerza la seguridad de los datos. No así con servidores distribuidos en los que es mucho más difícil lograr esto.
- No existe una versión dedicada en cada momento, solo variantes.

## COMPARACIÓN DE HERRAMIENTAS

Para comparar algunas herramientas de control de versiones utilizaremos parámetros de interés específico definidos por esta investigación clasificados en tres grupos:

- Generales.
- Por funcionalidades.
- Por Interfaces.

### GENERALES

- **Modelo de Repositorio (MR):** Especifica si usa modelo centralizado (C) o distribuido (D).

- **Modelo de Concurrencia (MC):** Especifica si usa bloqueo (B), fusión (F) o permite el uso de ambos (A).
- **Licencia (LS):** Tipo de licencia que se usa en su distribución, Propietaria (Prop), GNU Public License (GPL), Berkley Software Distribution o una de sus derivadas (BSD).
- **Gratuito (G):** Especifica si es de distribución gratuita o pagada.
- **Identificador de Revisión (IR):** Especifica el tipo de identificador usado internamente para referirse a una versión específica, Nombre (NMS), Identificador pseudo-aleatorio (PSA), Hash en SHA-1 (SHA), Numérico con hash en SHA-1 (NSH).
- **Protocolos de Red (PR):** Especifica los protocolos de red que puede usar para la comunicación.

| Herramienta | MR | MC | LS | G | IR | PR |
|---|---|---|---|---|---|---|
| **AccuRev** | C | A | Prop | No | NMS | propio |
| **Bazaar** | D | F | GPL | Si | PSA | http, sftp, ftp, propio, sobre ssh, paquetes de correo, WebDAV |
| **ClearCase** | C | A | Prop | No | NMS | http, propio (ccfs), driver mvfs |
| **CVS** | C | F | GPL | Si | NMS | pserver, ssh |
| **Git** | D | F | GPL | Si | SHA | propio (git), sobre ssh, rsync, paquetes de correo |
| **Mercurial** | D | F | GPL | Si | NSH | http, propio sobre ssh, paquete de correo |
| **Subversion** | C | A | BSD[1] | Si | NMS | propio(svn), sobre ssh, http+ssl sobre WebDAV |
| **Team Fundation Server (TFS)** | C | A | Prop | No | NMS | soap sobre http o https |
| **Visual Source Safe** | C | A | Prop | No | NMS | Ninguno[2] |

[1] Subversion es distribuido bajo una licencia libre al estilo de Apache2/BSD pero no es específicamente ninguna de ellas.
[2] Visual Source Safe no usa protocolos de red específicos, accede a archivos remotos mediante compartido físico.

### POR FUNCIONALIDADES

- **Envíos atómicos (EA):** Si permite envíos en modo transacción, o se hacen todos los cambios o no se hace ninguno.
- **Versionado de Directorios (VD):** Si permite versionado de directorios.
- **Renombrado (RN):** Si permite renombrado de archivos y directorios.
- **Ganchos pre y post eventos (GE):** Indica la capacidad para desencadenar comandos antes y después de una operación.
- **Firmado digital de revisiones (FD):** Si posee firmado digital de las de revisiones integrado.

- **Registro interactivo de cambios (RI):** Si permite al usuario escoger que partes de sus cambios, inclusive dentro de archivos, quiere aplicar.
- **Actualización parcial (AP):** Si permite la habilidad para obtener solo un subdirectorio específico del repositorio.
- **Referencias Externas (RE):** Embebido de repositorios remotos dentro del árbol de directorios del repositorio actual.
- (Dribin, 2007) (Dribin, 2007)

| Herramienta | EA | VD | RN | GE | FD | RI | AP | RE |
|---|---|---|---|---|---|---|---|---|
| **AccuRev** | Si | Si | Si | Si | Si | Des[3] | Si | Si |
| **Bazaar** | Si | No | Si | Si | Par[2] | Des[3] | No | No |
| **ClearCase** | No | Des[3] | Si | Si | Si | Des[3] | Des[3] | Des[3] |
| **CVS** | No | No | No | Lim[1] | No | No | Si | Des[3] |
| **Git** | Si | Si | Si | Si | Si | Si | No | Si |
| **Mercurial** | Si | Si | Si | Si | Si | Si | Si | Si |
| **Subversion** | Si | Si | Si | Si | No | No | Si | Si |
| **TFS** | Si | Si | Si | Si | Des[3] | Des[3] | Des[3] | Des[3] |
| **Visual Source Safe** | No | No | Si | Si | No | Des[3] | Des[3] | Des[3] |

[1] CVS solo admite ganchos para algunos pocos eventos.
[2] El firmado digital de Bazaar es parcial.
[3] Estas funcionalidades no pudieron probarse ni encontrarse documentadas, por eso están marcadas como Desconocidas (Des).

## POR INTERFACES

- **GUI (GUI):** Describe la tenencia de aplicaciones propias o de terceros que provean interfaz gráfica para usuario y/o para administración. Pueden ser aplicaciones para Windows (W), Windows con integración de Explorador (W+), Windows con Tortoise, (W*), para en general Linux (L), para Unix (U), para MacOS, o plataformas especificadas (GTK), (Qt), (KDE), (Solaris), (JavaVM), etc.
- **Web (Web):** Describe la tenencia de interfaces web. Puede ser con interfaz incluida propia (incluida), o mediante otras aplicaciones o plug-ins de integración web especiales.
- **Plug-ins e** Integración **(PI):** Describe si las funcionalidades están disponibles desde algún IDE usando algún plug-in o aplicación de integración ya sea por defecto o usando aplicaciones de terceros.

| Herramienta | GUI | Web | PI |
|---|---|---|---|
| **AccuRev** | W+, L, U, Mac, BeOS | Incluida | Idea, Eclipse, VisualStudio |
| **Bazaar** | GTK, Qt, W* | webserve, Trac | Eclipse, VisualStudio, |

| | | | |
|---|---|---|---|
| | | | TeamMate |
| **ClearCase** | W, U | Incluida | Eclipse, VisualStudio, KDevelop, Idea |
| **CVS** | W*, Mac, GTK, Qt | cvsweb, ViewVC | Eclipse, KDevelop, Idea, EMacs |
| **Git** | git-gui, gitk, qgit, W* | gitweb, github, trac | Eclipse, Emacs, TextMate, VIM, Idea |
| **Mercurial** | W*, L | Incluida, Trac | Eclipse, NetBeans, VisualStudio9, Emacs, Vim |
| **Subversion** | Qt, W*, KDE, Java, Mac, Nautilus | Apache2, WebSVN, ViewSVN, ViewVC, Trac, SharpForge, sventon | Eclipse, VisualStudio, NetBeans, Idea, KDevelp, TextMate, Emacs, MonoDevelop |
| **TFS** | W, Mac, Unix | Incluido | VisualStudio, Java for Eclipse |
| **Visual Source Safe** | W, L, Mac, Solaris, JavaVM | SSWI, VSS Remoting | VisualStudio, Idea |

## ANÁLISIS

En cuanto a los parámetros generales consideramos de mayor importancia el modelo del repositorio (MR), el tipo de licencia bajo la que se distribuye la aplicación (LS) y la gratuidad (G) de esta que es el parámetro de mayor valor. En este sentido las mejores aplicaciones son Git, Bazaar, CVS, Mercurial y Subversion.

De los parámetros de comparación por funcionalidades consideramos más importantes los envíos atómicos (EA), el versionado de directorios (VD), la habilidad de usar ganchos pre o post eventos (GE) y la posibilidad de realizar referencias externas (RE). Según estos parámetros las mejores aplicaciones son AccuRev, Git, Mercurial, Subversion y Team Fundation Server (TFS) de Microsoft.

Por último, de los parámetros de comparación por interfaces consideramos que la importancia esta dada por la cantidad de sistemas populares para los que existe interfaz de gráfica usuario, la cantidad de IDEs populares para los que tiene integración y las facilidades de acceso web. En este sentido casi todas están al mismo nivel, excepto Subversion que se destaca por encima de las demás en cuanto a interfaces e integración.

Cruzando los resultados de las comparaciones por las tres categorías de parámetros seleccionados podemos concluir que las aplicaciones para el control de versiones más completas son: Git, Mercurial y Subversion. Destacándose ésta última por la cantidad de interfaces que posee.

## GESTIÓN DE LA CONFIGURACIÓN EN LA UCI

Es importante saber como se comporta la GCS en los proyectos de la Universidad de las Ciencias Informáticas. Conocer las tendencias actuales, las políticas más comúnmente aplicadas, las herramientas más usadas y las justificaciones de todo ello pues un proceso de soporte tan importante en el desarrollo de software debe estar ya hace mucho tiempo documentado para las especificidades de la UCI, debe estarse aplicándose centralizadamente y controlado con rigor.

Los primeros trabajos de diploma que se pueden encontrar en la biblioteca UCI y en el sitio tesis.uci.cu datan del año 2006. Sin embargo los medios digitales de la UCI para guardar tesis son escasos y no excluimos que existan tesis anteriores sobre el tema. De cualquier modo parece probable que el tema se ha investigado solo desde hace muy poco en la universidad.

Aproximadamente en el año 2004 en los proyectos de la UCI se realizaba control manual de versiones o se utilizaba Visual Source Safe, sobre todo en aquellos proyectos que podían permitirse usar el naciente Visual Studio como IDE para desarrollar. Se utilizaba sobre todo porque venía junto con el paquete de Visual Studio y estaba muy cómodamente integrado con este.

Algunos años más tarde comenzó a usarse Subversion, casi de manera puntual, por las increíbles capacidades que proveía. La buena experiencia rápidamente se distribuyó por los proyectos de la universidad.

El estado actual de la gestión de la configuración en los proyectos de la universidad, las políticas que se siguen más frecuentemente, las herramientas que usan y sus justificaciones, etc. es un tema para el cual se decidió la realización de una encuesta.

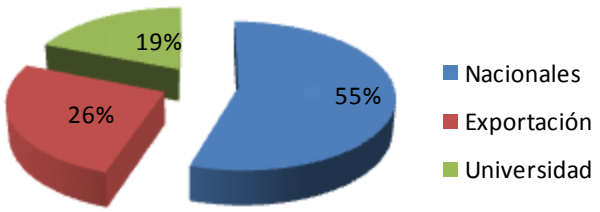
## ENCUESTA PARA DETERMINAR EL ESTADO ACTUAL GCS EN LA UCI

El objetivo de la encuesta era determinar el estado de la GCS en los proyectos de la Universidad de las Ciencias Informáticas, identificar las políticas más comúnmente establecidas y usadas, determinar las herramientas más usadas y sus justificaciones, determinar la preparación de los integrantes de proyecto en el tema de GCS y saber si las políticas definidas se ajustaban a las necesidades para las cuales se habían descrito.

## POBLACIÓN Y MUESTRA

Definimos como población para la encuesta los 127 proyectos existentes en la UCI, los cuales pueden categorizarse como Nacionales, de Exportación o propios de la universidad. La idea óptima era realizar la encuesta al menos a dos integrantes de cada proyecto, intentando que estos fueran en la medida de lo posible, el líder del proyecto y el administrador de la configuración.

*ILUSTRACIÓN 14* ***DISTRIBUCION DE LOS PROYECTOS UCI***




Sin embargo si se lograba una mayor participación que solo dos estudiantes por proyecto el resultado de esta sería mucho menos aproximado. Por ello se distribuyó la encuesta en formato digital utilizando un enfoque spam y tentando a responder con comentarios y facilidades como: “solo le tomará 5 minutos”, “los datos que puedan relacionar su respuesta con Ud. o con su proyecto no se brindarán a terceros”, “sea franco y escriba sin tapujos, que buscamos la verdad no la oficialidad”, “es una encuesta para un trabajo de tesis”, etc.

Aún así previmos que no iban a responder todas las personas que queríamos y más aún, que probablemente no respondieran desde todos los proyectos como sucede en muchas encuestas, ya sea por apatía, desinterés, posponer continuamente, no recibir la encuesta por problemas técnicos u otros factores, etc. Por ello y apoyándonos en (Álvarez, 1997) decidimos que con un 15% que los 127 proyectos que respondieran estaba bien lo cual nos deja en aproximadamente 19 proyectos.

## ¿POR QUÉ AL LÍDER Y AL ADMINISTRADOR DE LA CONFIGURACIÓN?

El Administrador de la Configuración, o su equivalente, es el encargado de definir y planificar de antemano todas las políticas relacionadas con la GCS al comenzar el proyecto y además es el principal encargado de hacer valer estas políticas. El líder por su parte, a pesar de que no es su principal preocupación si debe estar muy empapado con el proceso de desarrollo seleccionado por el proyecto y por ende con el trabajo del Administrador de la Configuración. Por estas razones consideramos que son los dos

integrantes de proyecto que mejor pueden responder las preguntas que necesitamos encuestar.

## DISEÑO DE LA ENCUESTA

El diseño de la encuesta lleva una atención especial ya que es este el aspecto preponderante para la obtención de la información que se requiere. La encuesta debe ser precisa, directa, encaminada a cubrir los aspectos que se necesitan de manera concreta y usando un lenguaje claro. Asegurando así minimizar la posibilidad de respuestas erróneas o en blanco.

La encuesta usada tiene 54 preguntas, de ellas 33 son preguntas cuya respuesta es solo “Si”, “No” o “No se”; 4 son preguntas cuya respuesta es un rango numérico; 9 son preguntas abiertas, que son menos eficaces pero necesarias para obtener cierta información necesaria, las demás preguntas son de selección dentro de un conjunto.

En todos los casos en que la pregunta no era abierta se configuró la encuesta de manera que solo se pudiese seleccionar fácilmente una de las respuestas brindadas, inclusive en las preguntas de rango numérico.

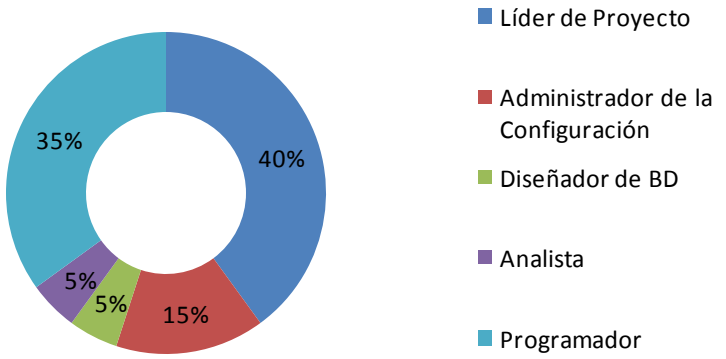
| Objetivo | Preguntas |
|---|---|
| Identificar el rol de quien respondía para determinar la cantidad de conocimiento que debía poseer | 1, 2 y 3 |
| Introducir al encuestado en el tema de la encuesta | 4 y 5 |
| Determinar carga de trabajo del Administrador de la Configuración y posibles relaciones inherentes con otros roles | 6, 7 y 8 |
| Determinar la existencia, conocimiento y uso de políticas de GCS en el proyecto | 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 y 20 |
| Determinar la eficiencia de la organización de la biblioteca de software del proyecto | 21 |
| Determinar la existencia, composición, trabajo activo, seguimiento y control del Comité de Control de Cambios en el proyecto | 22, 23, 24 y 25 |
| Determinar la existencia, funcionamiento correcto, periodicidad y eficiencia de los reportes de estado de la configuración en el proyecto | 26, 27, 28 y 29 |
| Determinar la cantidad de conocimiento del encuestado sobre control automático de versiones en la práctica | 30, 31 y 32 |
| Determinar el uso y justificación de herramientas de control de versiones en el proyecto | 33, 34 y 35 |
| Determinar el estado práctico de la cultura de GCS en el encuestado y el proyecto | 36, 37, 38, 39 y 40 |
| Determinar la necesidad, uso y modo de uso de la gestión documental en el proyecto | 41, 42, 43 y 44 |
| Determinar uso activo y justificación de herramientas de gestión de proyecto | 45, 46, 47 y 48 |

| | |
|---|---|
| Determinar la cantidad de conocimiento del encuestado sobre gestión automática de proyecto en la práctica | 49 y 50 |
| Determinar cobertura de la herramienta de gestión de proyecto seleccionada | 51, 52 y 53 |
| Conocer la opinión del encuestado respecto a la GCS de su proyecto | 54 |

## ANÁLISIS DE LOS DATOS DE LA ENCUESTA

Se respondieron 20 encuestas, una más que las el número que consideramos aceptable. Las respuestas vinieron de roles diferentes como se ve en la gráfica:

*ILUSTRACIÓN 15* ***COMPORTAMIENTO DE ROLES ENCUESTADOS***




A pesar de que la encuesta estaba pensada principalmente para líderes de proyecto y administradores de la configuración la mayoría entre los roles encuestados está en líderes de proyecto y programadores. Esto, lejos de empañar el resultado de la encuesta da una perspectiva más desde el individuo que usa las políticas de GCS en vez de quien las define lo cual nos permitirá saber además si estas políticas llegan realmente y ayudan en el desarrollo de proyectos en la UCI.

Los resultados más transcendentes de la encuesta son como sigue:

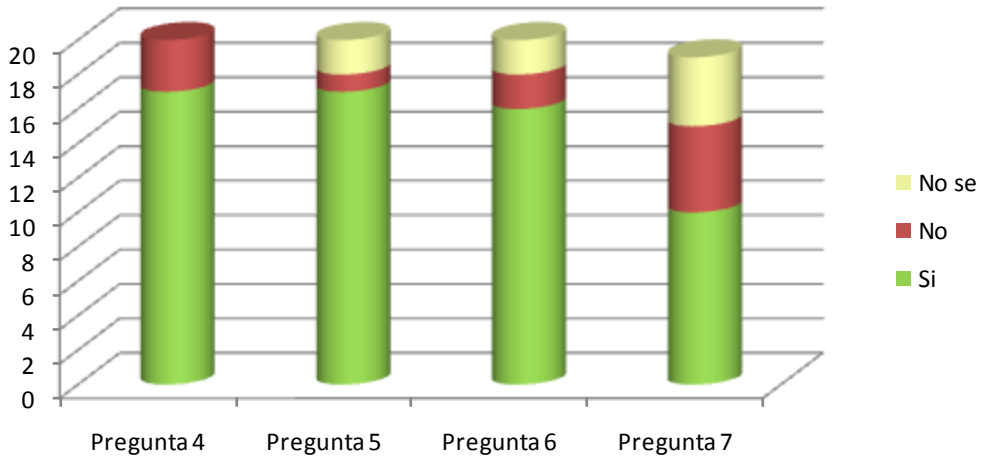



*ILUSTRACIÓN 16* ***ROLES ADICIONALES DEL ADMINISTRADOR DE LA CONFIGURACIÓN***

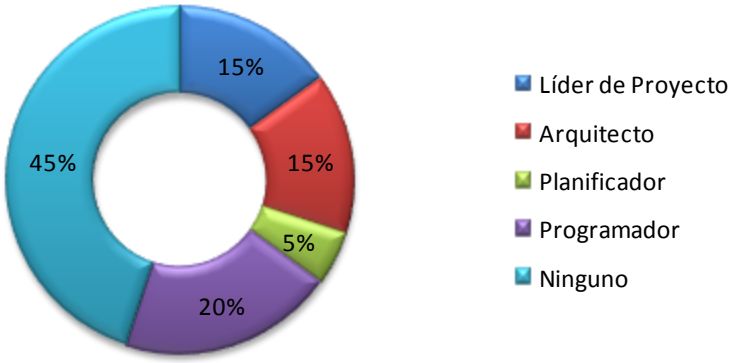
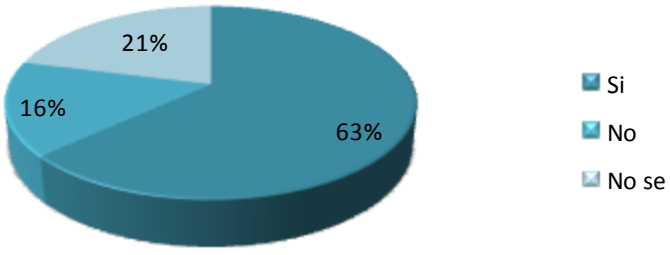



*ILUSTRACIÓN 17* ***EXISTENCIA DE PLAN DE GCS DEFINIDO***



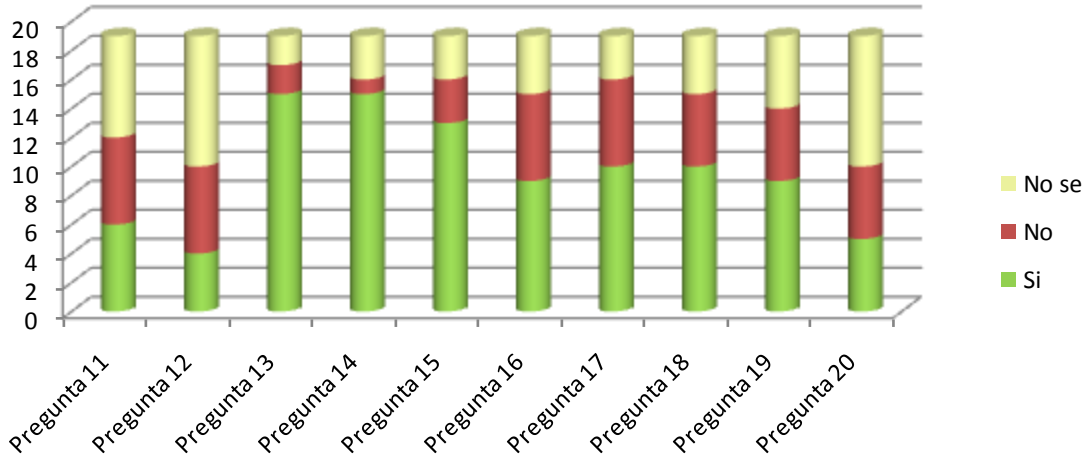



*ILUSTRACIÓN 18* ***ESTADO DE LA ORGANIZACION DEL REPOSITORIO***

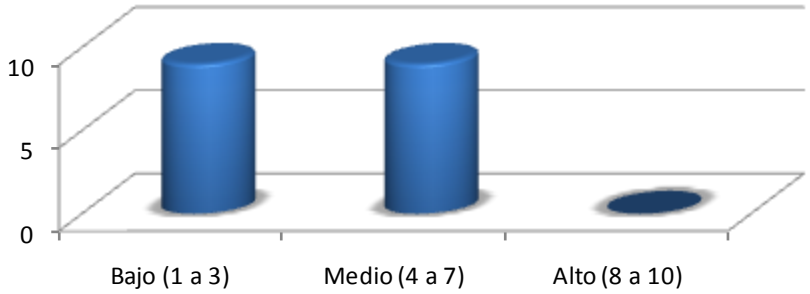



*ILUSTRACIÓN 19* ***EXISTENCIA DE COMITÉ DE CONTROL DE CAMBIOS***

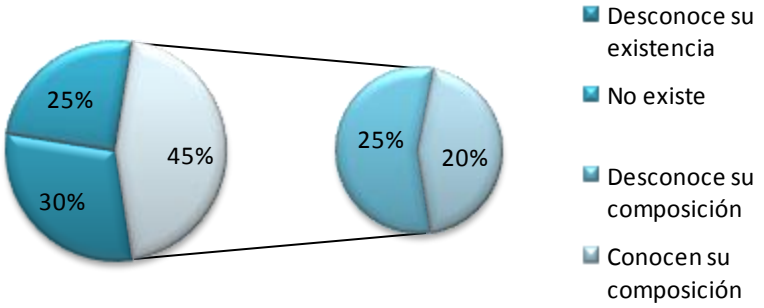



*ILUSTRACIÓN 20* ***COMPOSICIÓN DEL COMITÉ DE CONTROL DE CAMBIOS***

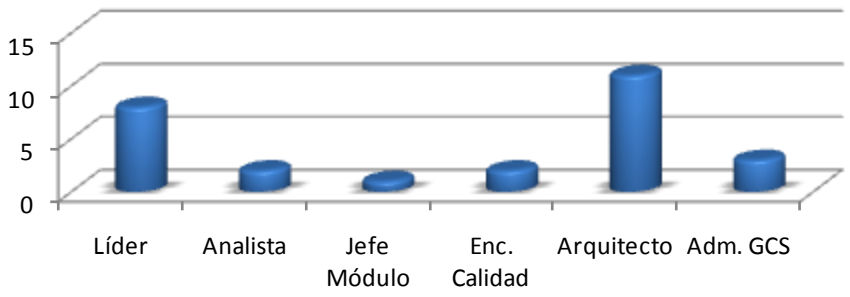


*ILUSTRACIÓN 21* ***FUNCIONAMIENTO DEL COMITÉ DE CONTROL DE CAMBIOS***

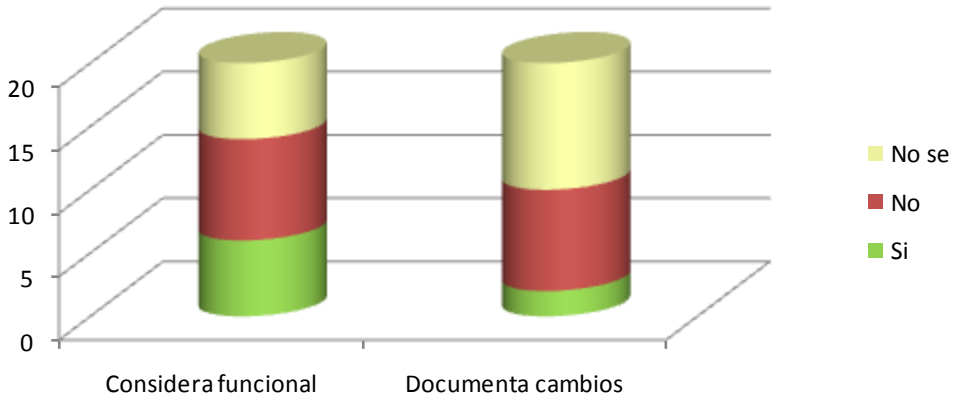



*ILUSTRACIÓN 22* ***PROTAGONISMO DEL CCC***

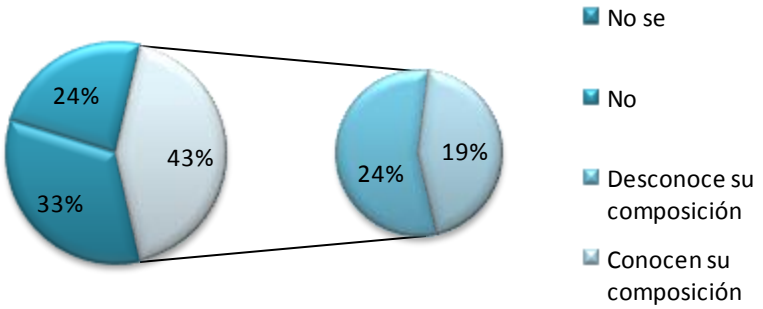



*ILUSTRACIÓN 23* ***REALIZACIÓN DE INFORMES DE ESTADO***

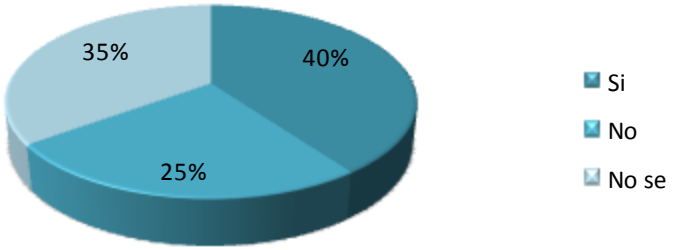



*ILUSTRACIÓN 24* ***PERIODICIDAD DE LOS INFORMES***

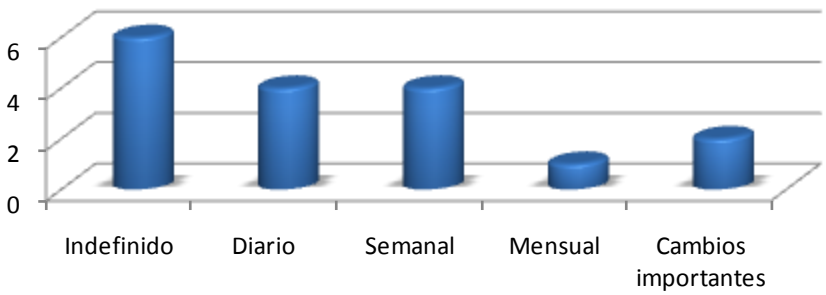


*ILUSTRACIÓN 25* ***DESTINATARIOS DE LOS INFORMES***

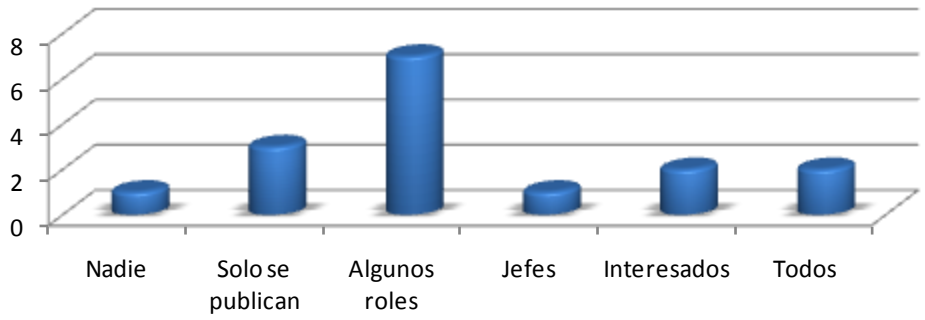



*ILUSTRACIÓN 26* ***CONTENIDO DE LOS INFORMES***

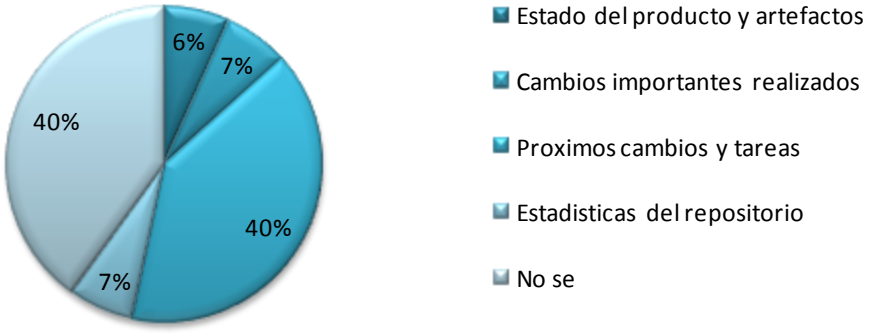



*ILUSTRACIÓN 27* ***USO DE HERRAMIENTAS DE CONTROL DE VERSIONES***

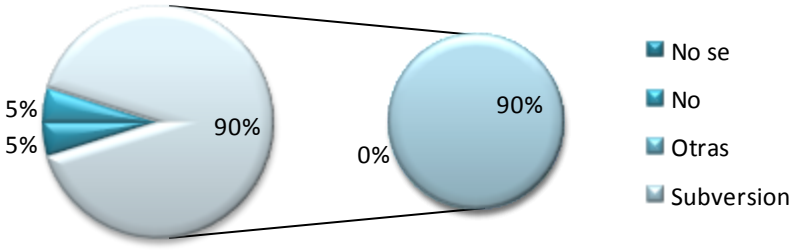



*ILUSTRACIÓN 28* ***JUSTIFICACION DEL USO DE SUBVERSION***

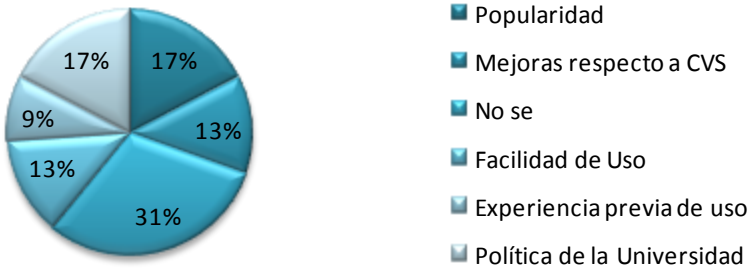


*ILUSTRACIÓN 29* ***EXISTENCIA Y FRECUENCIA DE CAMBIOS SIN COMENTARIO***

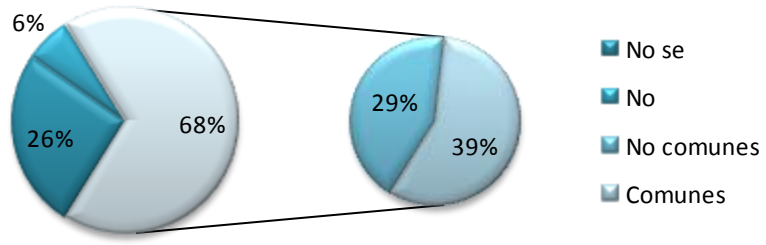



*ILUSTRACIÓN 30* ***USO ACTIVO DE HERRAMIENTAS DE GESTIÓN DE PROYECTO***

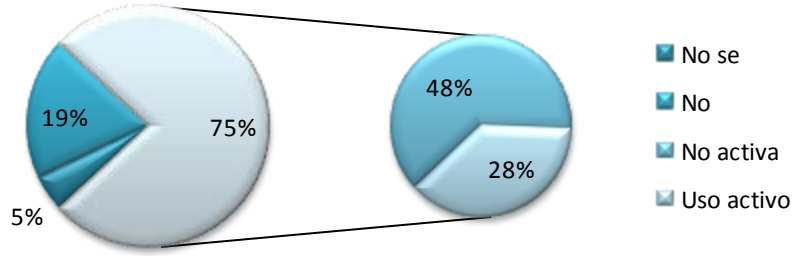
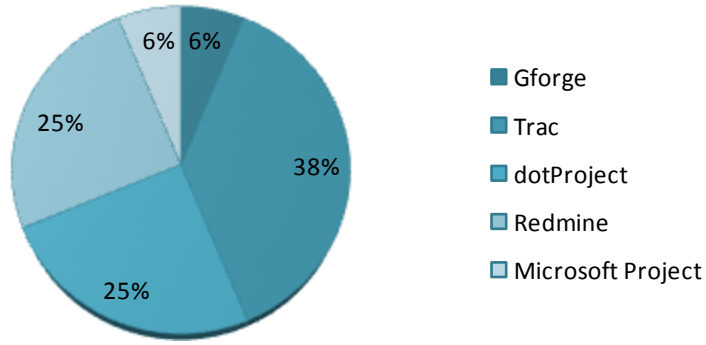



*ILUSTRACIÓN 31* ***SELECCIÓN DE HERRAMIENTAS DE GESTIÓN DE PROYECTO***




## CONCLUSIONES

La Gestión de la Configuración de Software, como se ha podido observar a lo largo de este primer capítulo, es una disciplina que se encuentra reconocida en los estándares y modelos de calidad más utilizados a nivel internacional como son las normas ISO, IEEE y el modelo CMMI. Autores de renombra en el mundo del software han situado a

la GCS como una disciplina esencial para el éxito de un proyecto software. Ella juega un papel primordial en el proceso de desarrollo de software, tiene como objetivo 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. Además es una actividad de protección que facilita el mantenimiento del sistema, ya que aporta información precisa para valorar el impacto de los cambios solicitados y reduce el tiempo de implementación de un cambio, tanto evolutivo como correctivo. Asimismo, permite controlar el sistema como producto global a lo largo de su desarrollo, obtener informes sobre el estado de desarrollo en que se encuentra y reducir el número de errores de adaptación del sistema, lo que se traduce en un aumento de calidad del producto, de la satisfacción del cliente y, en consecuencia, de mejora de la organización.

La definición de los procesos de GCS específicos para un proyecto dado y que estas sean, sino óptimas, al menos saludablemente eficientes es un proceso arduo, complicado, minucioso y que requiere de mejora constante.

En la universidad, los procesos de Gestión de la Configuración de Software se realizan en casi todos los proyectos, sin embargo se realiza como respuesta a políticas de la universidad que la marcan como requerida y no por reconocimiento real de su necesidad. Las herramientas y políticas definidas en la mayoría de los proyectos no cubren las necesidades de este o no llegan a los desarrolladores. Se puede calificar de manera general el estado de la GCS en la UCI como pobre o de baja eficiencia.

# Capítulo 2: Propuesta a Aplicar

## INTRODUCCIÓN

En este capítulo se desarrollará la propuesta de Plan de Gestión de la Configuración y Cambios que está dirigida al proyecto IMAC y alcanzará a todas las actividades y los implicados relacionados con el desarrollo del proyecto, los archivos que se generen por el equipo de desarrollo y las herramientas que se usan en él.

El propósito de este plan es describir las políticas de gestión de la configuración de software que se aplicarán para el transcurso del ciclo de vida del proyecto IMAC atendiendo a:

- Responsables e involucrados en las actividades de GCS.
- ECS a controlar.
- Bibliotecas de Software.
- Planificación de las tareas con los ECS.
- Establecimiento de las Líneas Base.
- Proceso de Control de Cambios.
- Generación de Informes de Estado.

Para el establecimiento correcto de estos elementos primero estudiaremos algunas características del proyecto IMAC, tanto de su equipo de desarrollo como de su organización lógica.

## CARACTERÍSTICAS DEL PROYECTO IMAC

Como se explicó en la introducción de esta investigación el proyecto IMAC es el primer proyecto de informatización de un ministerio completo en el país, específicamente del Ministerio de Auditoría y Control (MAC). En especial el producto SIGAC es la solución de gestión de la información que se utilizará en dicho ministerio y en todas sus sedes provinciales.

El producto solución va a ser utilizado en el día a día por los especialistas del MAC para procesar la información por todo el flujo de trabajo que tienen definido en el ministerio. Se definió por ende como “clientes” a estos especialistas.

Al ser un proyecto de colaboración y teniendo el MAC un departamento de informática se habilitó un compañero del MAC para realizar como intermediario entre el proyecto y los clientes. Este compañero aunque no forma parte del equipo de desarrollo si puede realizar revisiones de estado y además coordina el trabajo directo con los clientes.

Se debe tener en cuenta que existen una serie de parámetros en todas las áreas del proyecto cuya adopción es de tipo obligatoria pues forman parte de las estrategias de estandarización de la universidad, como pueden ser las plantillas del expediente de proyecto.

Existe además un conjunto de relaciones entre el proyecto y las entidades superiores que se deben tener también en cuenta para diseñar la estrategia a utilizar. Un ejemplo de ello es la necesidad de revisiones periódicas del estado del proyecto y de los ECS de este por la dirección de la facultad, por la dirección de calidad de la universidad. Otro ejemplo puede ser la necesidad de cooperación con otros proyectos de la universidad.

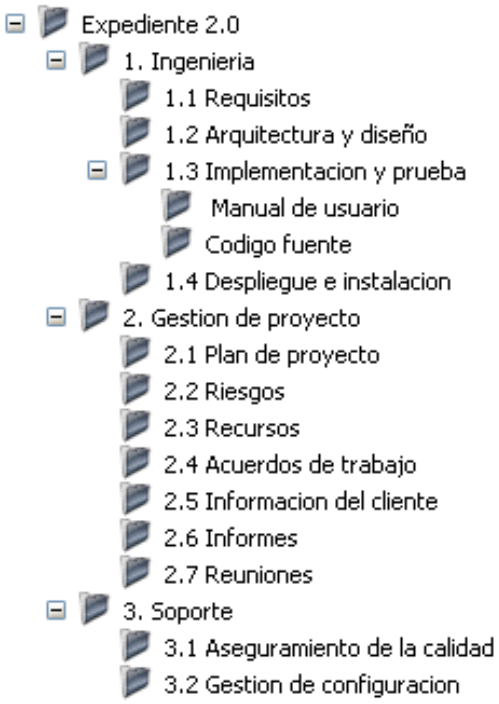
## METODOLOGÍA DE DESARROLLO Y DOCUMENTACIÓN GENERADA

La metodología escogida para aplicar de manera general es RUP aunque se usa IDEF0 para los diagramas de casos de uso del negocio por ser más fáciles de comprender por los clientes que hacen uso de éstos para ayudar al equipo de desarrollo a comprender el flujo de trabajo del negocio. Esto significa que el ciclo de vida estará sujeto a las fases y procesos que indica RUP.

Para generar la documentación y el expediente de proyecto, se usan las plantillas propuestas por la Dirección Técnica de la Infraestructura Productiva de la UCI en su versión número 2.0 que son los modelos establecidos por el Departamento de Calidad UCI como estándares para la documentación del expediente de proyecto y que fueron diseñadas tomando como modelo a RUP.

Esta Plantilla de Expediente de Proyecto brinda no solo una guía y un estándar para los documentos que se generan en el proyecto sino también una distribución estandarizada para el guardado de toda esta documentación. Las plantillas están distribuidas como sigue:

*ILUSTRACIÓN 32* ***DISTRIBUCIÓN DE LAS PLANTILLAS DEL EXPEDIENTE 2.0***




La plantilla de expediente de proyecto incluye una plantilla específica para el Plan de Gestión de la Configuración de Software. Esta plantilla de plan contiene los siguientes elementos a llenar:

- Gestión de configuración de software.
  - Organización de la gestión de configuración de software.
  - Responsabilidades.
  - Relación de la gestión de configuración con el ciclo de vida del proyecto.
    - Interfaces con otras organizaciones dentro del proyecto.
    - Responsabilidades con otras organizaciones dentro del proyecto.
- Actividades de gestión de configuración de software.
  - Identificación de la configuración.
    - Especificación de la identificación.
    - Líneas base del proyecto.
- Bibliotecas.
- Control de la configuración.
  - Procedimiento para procesar pedidos de cambios y su aceptación.
  - Comité de control de cambios.
  - Revisión de documentos.

    - Herramientas automatizadas para el Control de cambios.
- Estado de la configuración.
    - Reportes.
    - Proceso de entregas.
- Hitos.

## COMPONENTES DE LA SOLUCIÓN

Según las necesidades del proyecto y la arquitectura establecida la solución está dividida en los siguientes componentes:

- **Módulo de Acciones de Planificación:** Aplicación web desarrollada por el proyecto en PHP usando como base frameworks como el marco de trabajo del ERP UCI. Su principal misión es proveer de un sistema de gestión, seguimiento y control para los planes anuales de acciones de control del MAC.
- **Módulo de Acciones de Control:** Una de las necesidades del MAC es la gestión de contenido de documentos por lo que este módulo es una personalización de la herramienta de gestión documental Alfresco Labs para el flujo de trabajo del MAC.
- **Módulo de Integración:** Es un módulo de servicio desarrollado por el proyecto para permitir la gestión del contenido de Alfresco Labs desde el módulo de planificación.

## COMPOSICIÓN DEL EQUIPO DE DESARROLLO

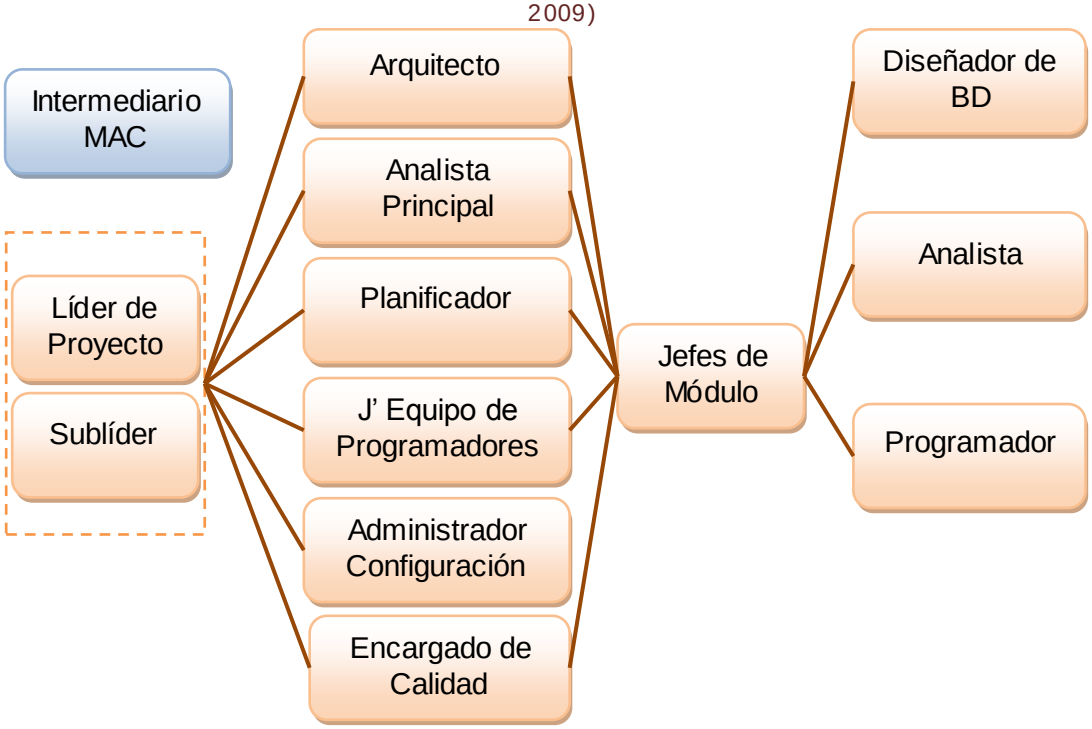
El equipo de desarrollo del proyecto está constituido por los siguientes roles:

- Líder de Proyecto.
- Sublíder.
- Arquitecto.
- Analista Principal.
- Planificador.
- Jefe de Equipo de Programadores.
- Diseñador de Base de Datos.
- Encargado de Calidad.
- Administrador de la Configuración.
- Jefe de Módulo.
- Analista.

- Programador.

La jerarquía de roles es como sigue:

*ILUSTRACIÓN 33* ***DISTRIBUCIÓN JERÁRQUICA DE ROLES EN IMAC*** (EQUIPO IMAC, 2009)




El equipo además está compuesto por 20 personas, algunos de ellos cumplen con varios roles cada uno tal y como se muestra en la tabla.

| Persona | Roles | Módulo |
|---|---|---|
| Tomás Rodriguez Barrios | Líder de proyecto | |
| Yasser Azan Basallo | Sublíder, Arquitecto, J' Módulo | Integración |
| Ariel Díaz Rodriguez | J' Equipo Programadores, J' Módulo | Planificación |
| Aymara Marin Diaz | Analista Principal | |
| Ivette Barrientos Núñez | J' Módulo | Acciones de Control |
| Cecilia Oro Rivera | Encargado de Calidad | |
| Leydis de la Luz Paredes | Encargado de Calidad | |
| Yunior A. Hdez. Andrade | Planificador, Administrador GCS | |
| Andrés B. Glez. Morales | Programador | Integración |
| Anay Díaz Estrada | Analista | Integración |
| Aliuska Dominguez Rosales | Programador | Planificación |
| Antonio Hdez. Dominguez | Diseñador de BD | |
| Annia Pimentel Rivero | Analista | Planificación, Acciones de Control |
| Alexander Raventos Liranza | Programador | Planificación |
| Leslye Bravo García | Analista | Acciones de Control |
| Luis E. Ledesma García | Analista | Acciones de Control |
| Liudmila Hidalgo Ricardo | Programador | Acciones de Control |
| René A. Piñeiro Hdez. | Analista, Programador | Planificación |

| | | |
|---|---|---|
| Rangel Hurtado Díaz | Programador | Acciones de Control |
| Sucel Ochoa Ochoa | Programador | Acciones de Control |
| Yumara García Rodriguez | Programador | Planificación |

Una característica que es importante resaltar es que del equipo de trabajo solamente los 5 profesores que cubren los roles principales y dos estudiantes más, poseen experiencia en el trabajo en proyecto anteriores. Los demás conocen el trabajo de proyecto solo por sus asignaturas.

Además, la mayoría de los integrantes del proyecto mantiene una concepción poco enfocada a la colaboración en su trabajo, característica común en las personas que se forman principalmente como programadores y que no han chocado aún con necesidades reales de colaboración y medidas establecidas para su facilidad.

Otra característica del equipo de desarrollo que es importante destacar es que los profesores que integran el equipo de dirección del proyecto son nóveles con uno o dos año de experiencia como profesionales solamente y que se estrenan como responsables máximos de un proyecto productivo.

Cabe destacar también que dichos profesores tampoco tienen más de un año de experiencia usando sistemas de gestión de la configuración y nunca han necesitado crear políticas de gestión de la configuración o evaluarlas.

## RECURSOS MATERIALES ASIGNADOS

Al proyecto se le asignaron 15 PCs en el laboratorio de proyecto número 301 del docente 1. Cada una de las PCs cuenta con las siguientes características:

- Placa: Asus P5LD2-VM
- Procesador: Intel Pentium IV a 3.0Ghz (2 CPU) 64bits
- RAM: DDR2 512Mb
- Video: Intel 82945G Express Chipset Family (on board) 128Mb
- Disco duro: 160Gb
- Red: Intel Tekoa PRO/1000 Adapter (100,0Mbps)

La distribución de uso de estas PCs incluye una para la instalación de pruebas de Alfresco Labs, gestor documental que formará parte de la solución que brindará el proyecto para el módulo de Acciones de Control. Otra para la instalación de un servidor central de bases de datos PostgreSQL y un servidor web Apache2 habilitado

para ejecutar código de PHP5 que servirá para los módulos de Planificación e Integración. El gestor de Alfresco Labs necesita ejecutarse en una PC exclusiva para él pues sus requerimientos mínimos son exactamente las capacidades de las PCs con que cuenta el proyecto y los intentos por instalar Alfresco Labs compartiendo los mismos recursos con el servidor Apache2 y PostgreSQL 8.3 fueron infructuosos. Ambas PCs funcionarán con sistema operativo Ubuntu Linux 9.04 en su versión para servidores.

También es necesario utilizar un PC para las bibliotecas de código y documentos del proyecto que se establezcan aunque, en dependencia de las características necesarias y las herramientas que se seleccionen para las bibliotecas, probablemente sea posible instalar estas en la misma PC de los servidores Apache2 y PostgreSQL 8.3.

Siguiendo estos parámetros se puede estimar que se utilizarán 2 PCs como servidores de servicios para el proyecto y las restantes 13 como estaciones de trabajo para los 21 desarrolladores.

Se hará referencia al servidor que ejecuta los servicios de Apache2, PostgreSQL y Subversion como “Server01” o “Servidor01” y al servidor que ejecuta los servicios de Alfresco Labs para las pruebas como “Server02” o “Servidor02”.

## FORMATOS DE DOCUMENTOS GENERADOS

La decisión tomada en conjunto para el desarrollo del proyecto fue la de realizarlo completamente sobre soporte libre por esto se instruyó la instalación de las PCs con sistema operativo Ubuntu 9.04 Jaunty Jacklope en el documento de la Arquitectura.

Sin embargo la mayoría de los clientes de la solución usan Windows XP en alguna de sus modificaciones y por tanto se hacía necesario las pruebas también bajo este sistema operativo. Además todas las plantillas del Expediente de Proyecto 2.0 de Calidad UCI están hechas para usarse con Microsoft Office. Por tanto es de esperar que coexistan en el proyecto los documentos con formato generado por herramientas libres y otros generados por herramientas privativas.

Según el documento de Arquitectura del proyecto los formatos de archivos cuya existencia se puede anticipar común en el proyecto son:

- .DOC Documento de texto con formato de Microsoft Word y OpenOffice.org Writer para la documentación del expediente de proyecto, asuntos legales, etc.

- .DOCX Documento de texto con formato y funciones especiales de Microsoft Word 2007 y OpenOfice.org 3.0
- .XLS Documento de hoja de cálculo de Microsoft Word y OpenOffice.org Calc. Para listados específicos del expediente de proyecto.
- .XLSX Documento de hoja de cálculo con funciones especiales de Microsoft Excel 2007 y OpenOffice.org Calc 3.0.
- .PHP Código para aplicaciones web ejecutadas en el lado del servidor. Para el código fuente de los módulos de Planificación e Integración.
- .ACP Archivo de salva de configuraciones y personalizaciones de Alfresco Labs.
- .SQL Scripts de salvas de las bases de datos necesarias.
- .PHTML Archivo de carga de interfaz del framework del ERP. Para módulo de Planificación.
- .HTML Archivo de código para la web. Para módulo de Planificación.
- .CSS Archivo de hoja de estilos en cascada para páginas web.
- .MPP Archivo de proyecto de Microsoft Project. Para la planificación del cronograma.
- .TXT Archivo de texto plano.
- .VPP Archivo de proyecto de Visual Paradigm Suite. Para los diagramas de ingeniería.
- .JPG Archivo de imagen comprimida Joint Photographic Expert Group. Para visualizaciones de algunos diagramas e imágenes diseñadas.
- .GIF Archivo de imagen para la Web. Para imágenes diseñadas.
- .PNG Archivo de imagen por capas. Para imágenes diseñadas.
- .JS Archivo de código de JavaScript. Para modulo de Panificación.
- .RAR Archivo de datos comprimidos de WinRAR. Para tags y algunos documentos de referencia muy grandes.
- .PDF Documento seguro de Adobe Acrobat. Para documentación de consulta e informes estáticos.
- .PPT Archivo de presentación con diapositivas de Microsoft PowerPoint o OpenOffice.org Impress. Para documentación de cursos, tutoriales, etc.
- .PPTX Archivo de presentación con diapositivas y funciones especiales de Microsoft PowerPoint 2007 ó OpenOffice.org Impress 3.0.
- .VSD Documento de diagrama de Microsoft Visio. Para diagramas específicos.
- .DOT Plantilla de documento de texto de Microsoft Word.

## MODELO DE GCS PARA IMAC

Para la propuesta de GCS para el proyecto IMAC se hace necesario definir la estrategia a seguir de manera que esta permita llevar a cabo los procesos de CGS con resultados satisfactorios según las características del proyecto definidas en el epígrafe anterior.

La estrategia consiste en adaptar los modelos definidos en IEEE e ISO y los puntos requeridos en la plantilla de Plan de Gestión de la Configuración del Expediente de proyecto de la Dirección de Técnica UCI para las necesidades y restricciones del proyecto. En general las actividades que se realizarán se resumen como sigue:

- Identificación de la Configuración.
  - Selección de los ECS.
  - Establecimiento de relaciones entre los ECS.
  - Definición del esquema de nombrado.
  - Organización de los ECS.
  - Definición de las bibliotecas de software.
- Gestión y Control de Cambios.
  - Definición del proceso de control de cambios.
    - Establecimiento de líneas base.
    - Cumplimiento de hitos.
  - Definición del mecanismo de solicitud de cambios.
- Generación de Informes de Estado.
  - Establecimiento de registros.
  - Generación de Informes de estado.
- Inspecciones de la Configuración.
  - Definición del proceso de auditoría de la configuración.

### IDENTIFICACIÓN DE LA CONFIGURACIÓN

#### SELECCIÓN DE LOS ECS

Para la selección de los ECS se siguieron los criterios documentados en (Antonio, 2001) que fueron explicados en el capítulo anterior.

Otros datos a tener en cuenta de cada ECS son:

- Disciplina a la que tributa.

- Fases en que se modifican.
- Roles que lo usan.

Los elementos de la configuración de software que se generarán según la metodología usada y las especificaciones de políticas de la universidad, que necesitamos controlar se pueden dividir en tres grandes grupos:

- Elementos del Expediente.
- Elementos de la Ingeniería.
- Elementos de la Solución.

En la siguiente tabla se detalla cada ECS con el grupo al que pertenece y los datos adicionales.

ELEMENTOS DEL EXPEDIENTE

| ECS | Descripción |
|---|---|
| **Visión** | Describe a alto nivel los principales aspectos relacionados con la solución en términos de necesidad del usuario final. |
| **Plan de Desarrollo de Software** | Describe de manera detallada como será el desarrollo del proyecto. |
| **Plan de Estimaciones** | Estimación del tamaño del producto y del esfuerzo necesario para el desarrollo. |
| **Cronograma de Proyecto** | Define las tareas a realizar, los recursos empleados en cada una y las fechas estimadas de realización. |
| **Planificaciones Individuales** | Describe las tareas y fechas de éstas para un desarrollador como parte de la estrategia de planificación. |
| **Ambiente de Desarrollo** | Describe los recursos materiales asignados al proyecto y sus usos. |
| **Plan de Capacitación** | Describe las actividades de preparación del personal. |
| **Roles y Responsabilidades** | Describe los recursos humanos asignados al proyecto, sus roles y las actividades de cada rol |
| **Tecnologías de Trabajo** | Define las herramientas de trabajo a usar. |
| **Levantamiento de Software** | Describe los recursos las herramientas de trabajo. |
| **Minutas de Reunión** | Resúmenes de acuerdos de reuniones. |
| **Lista de Riesgos** | Describe los riegos del proyecto y sus efectos |
| **Plan de Mitigación de Riesgos** | Describe las acciones para mitigar cada uno de los riesgos de la lista de riesgos. |
| **Plan de Aseguramiento de la Calidad** | Describe las acciones de revisión a realizar para asegurar la calidad del producto final. |
| **Plan de Medición** | Define las métricas y medidas a aplicar durante el ciclo de vida del proyecto. |
| **Glosario de Términos** | Define los términos necesarios para la comprensión |

| | |
|---|---|
| | del proyecto. |
| **Informes de Auditorías** | Lista las recomendaciones y no conformidades detectadas durante auditorías al proyecto. |
| **Listas de Chequeos** | Lista los puntos eficientes e ineficientes detectados durante las revisiones. |
| **Listas de Recomendaciones** | Lista las recomendaciones de mejorías y cambios detectados durante revisiones y pruebas. |
| **Convenio de Colaboración** | Términos del acuerdo de colaboración entre las entidades interesadas en el proyecto. |
| **Plan de Gestión de la Configuración** | Describe las políticas que se definen en esta investigación. |
| **Registro de ECS** | Lista los ECS del proyecto con sus respectivos identificadores significativos |
| **Registro de Componentes** | Lista de los componentes con su tipo e identificadores. |
| **Registro de LB e HT** | Lista de los hitos pasados y planificados y las líneas bases creadas con sus identificadores. |

ELEMENTOS DE LA INGENIERÍA

| ECS | Descripción |
|---|---|
| **Arquitectura del Software** | Describe la arquitectura de la solución. |
| **Arquitectura de la Información** | Aclara y justifica la arquitectura de la información usada. |
| **Estándares de Trabajo** | Define las normas y estándares de codificación, de ficheros, etc. |
| **Metodología de Desarrollo** | Define la metodología a utilizar. |
| **Descripción de los CU** | Describe y prioriza los Casos de Uso. |
| **Plan de Gestión de Requisitos** | Describe las directrices para requisitos y su trazabilidad en función del proyecto. |
| **Modelo del Sistema** | Explica los procesos resultantes de la automatización a partir de los CU y cada una de las interfaces. |
| **Requisitos Funcionales** | Describe los requisitos funcionales que debe cumplir el software. |
| **Requisitos No Funcionales** | Describe los requisitos no funcionales. |
| **Diagramas de Flujo** | Describe los flujos de trabajo de procesos del sistema por sus módulos. |
| **Diseño de Interfaces** | Describe las interfaces para cada CU del sistema. |
| **Modelo de Datos** | Describe los datos persistentes del proceso. |
| **Diseño de la Base de Datos** | Describe la base de datos física a generar. |
| **Modelo de Despliegue** | Describe los elementos del despliegue de la solución. |
| **Vista de Datos de la Arquitectura** | Modela desde el punto de vista arquitectónico las bases de datos del proyecto. |
| **Requerimientos del Hardware** | Describe el hardware necesario para desplegar la solución. |
| **Estrategia de Respaldo y Recuperación** | Describe los procesos de respaldo y recuperación de datos para la solución una vez instalada. |

| | |
|---|---|
| **Estrategia de Soporte Técnico** | Describe los procesos e involucrados en el soporte técnico de la solución. |
| **Planes de Pruebas** | Describe la estrategia de pruebas para cada componente del sistema. |

## ELEMENTOS DE DE LA SOLUCIÓN

| ECS | Descripción |
|---|---|
| **Personalización y Configuración de Alfresco Labs** | Contiene los datos de las modificaciones para Alfresco Labs requeridos para el módulo de Acciones de Control |
| **Fuente de Planificación** | Contiene el código fuente generado para el módulo de planificación dentro del marco de trabajo del ERP |
| **Fuente de Integración** | Contiene el código fuente generado para el módulo de Integración contenido en Planificación |
| **Manual de Usuario** | Describe las funcionalidades de la solución y las explica para el usuario final |

## RELACIONES ENTRE LOS ECS

Las relaciones entre los ECS han sido definidas siguiendo las recomendaciones de (Antonio, 2001) y analizando cada ECS para determinar, con el modelo recomendado, las relaciones que, de manera lógica, están creadas y necesitan ser controladas.

Existen una serie de relaciones de dependencias y derivación establecidas por los procesos de la metodología de desarrollo utilizada que pueden ser fácilmente detectadas. Por ejemplo, de los ECS de los flujos de trabajo de análisis y diseño se derivan los ECS de los flujos de implementación. Así mismo los ECS de los flujos de trabajo de análisis y diseño son derivados de los ECS de requerimientos. Otras relaciones son establecidas a simple vista como relaciones de composición. Por ejemplo el Plan de Desarrollo del Software esta compuesto por los documentos de Roles y Responsabilidades, Visión, etc. más otras informaciones propias.

*ILUSTRACIÓN 34* ***DEPENDENCIAS Y ASOCIACIONES ENTRE LOS ECS DE LOS FLUJOS DE SOPORTE***

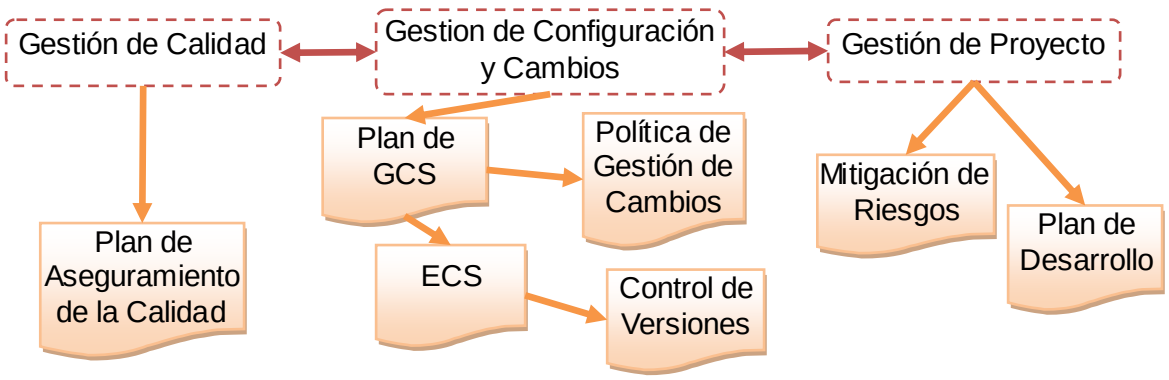
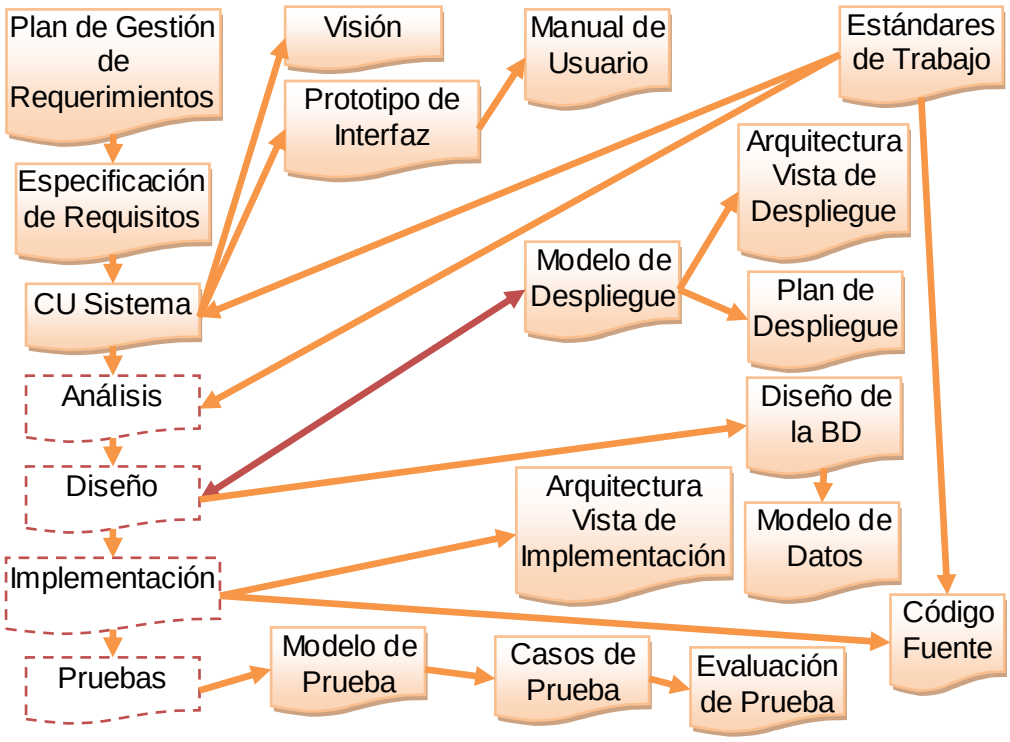


*ILUSTRACIÓN 35* ***DEPENDENCIAS Y ASOCIACIONES ENTRE LOS ECS DE LOS FLUJOS INGENIERILES***




Las siguientes tablas muestran las relaciones específicas encontradas entre los ECS seleccionados. Los ECS que no se encuentran en la lista no tienen relaciones con ningún otro.

| ECS | Relación con ECS |
|---|---|
| Plan de Desarrollo de Software | **Compuesto:** Visión, Roles y Responsabilidades, Plan de Estimación, Cronograma, Recursos materiales, Plan de riesgos, Plan de GCS, Plan de Aseguramiento de la Calidad, Metodologías |
| Tecnologías de Trabajo | **Depende:** Arquitectura |
| Plan de Mitigación de Riesgos | **Depende:** Lista de Riesgos |
| Plan de Aseguramiento de la Calidad | **Compuesto:** Plan de Medición |
| Listas de Recomendaciones | **Depende:** Lista de Chequeos |
| Plan de Gestión de la Configuración | **Depende:** Arquitectura, Metodologías, Tecnologías de Trabajo |
| Requisitos Funcionales | **Depende:** Plan de Gestión de Requisitos |
| Requisitos No Funcionales | **Depende:** Plan de Gestión de Requisitos |
| Diagramas de Flujo | **Deriva:** Descripción de CU |
| Diseño de la BD | **Deriva:** Modelo de Datos |
| Vista de Datos de la Arquitectura | **Depende:** Arquitectura, Modelo de Datos |
| Requerimientos del Hardware | **Depende:** Arquitectura, Tecnologías de Trabajo |
| Modelo de Despliegue | **Depende:** Arquitectura, Requerimientos de Hardware |

| | |
|---|---|
| Estrategia de Respaldo y Recuperación | **Depende:** Modelo de Despliegue |
| Estrategia de Soporte Técnico | **Depende:** Acuerdo de Colaboración |
| Planes de Pruebas | **Depende:** Cronograma, CU |
| Personalización de Alfresco | **Depende:** Requerimientos Funcionales, Requerimientos no funcionales |
| Código fuente | **Depende:** Arquitectura, CU Sistema, Diagramas de Flujo |
| Manual de Usuario | **Depende:** Interfaces |

## ESQUEMA DE NOMBRADO

Para el nombrado de ECS se usará un esquema referenciado para el tratamiento normal de los ECS, pero además se utilizará una codificación significativa para crear identificadores más cortos para los casos en que se necesite un identificador corto. Estos identificadores significativos cortos serán fáciles de recordar para hacer valer la significación pero además se mantendrá un registro de identificadores para evitar posibles confusiones y permitir el referenciado directo desde otros ECS.

En general el nombre de un ECS será escrito en forma de título, es decir con la letra inicial de cada palabra significativa en mayúscula y el resto en minúscula. Las palabras no significativas se escribirán completamente en minúscula. Los espacios en el nombre serán sustituidos por el carácter “guión bajo” (underscore), carácter ASCII 95 (_). Por ejemplo: *“Plan_de_Gestión_de_la_Configuración.doc”*.

Los identificadores significativos de cada ECS serán creados por el administrador de la configuración una vez pasado el ECS a revisión formal. Éstos identificadores constarán de las letras iniciales de las palabras significativas del nombre del ECS escritas en minúscula y tendrán como máximo 5 caracteres. Detrás de los 5 o menos caracteres se agregará un carácter guión bajo y a continuación el identificador de la línea base a la que pertenece. (Ver Establecimiento de las Líneas Base)

Cada ECS cuyo formato lo permita contará internamente con una tabla de versionado en la que constará cada evento realizado sobre el ECS con los datos adicionales a éste como son: Fecha del Evento, Tipo de Evento, Descripción del Evento, Número de la revisión y Autor(es). Los tipos de eventos pueden ser: Creación, Revisión y Actualización.

Cuando se necesite sacar un ECS fuera del ámbito del control formal del proyecto y de manera independiente se agregará al nombre de éste el nombre del proyecto en mayúsculas y el número de la revisión exacta que tenía el ECS cuando se extrajo del control formal del proyecto separados entre si y del nombre del ECS por un guión

(hyphen), carácter ASCII 45 (-). Por ejemplo: *"Plan_de_Gestión_de_la_Configuración-IMAC-2867"*

Para el nombrado de directorios, se utilizará un esquema idéntico al de los ECS puntuales sin embargo no se aplicará ninguna codificación para estos. Si se necesita hacer referencias a directorios se harán utilizando el nombre completo de éste y su ubicación. Los nombres de directorios se escogerán atendiendo a los ECS que van a contener, al paquete que representa o a la función, proceso, o flujo al que tributan los ECS contenidos en él. Los espacios en los nombres de directorios deberán ser sustituidos por guiones bajos (_).

Para el nombrado de componentes se utilizará un esquema de codificación significativa doble en el cual cada componente tendrá una codificación larga para su nombre y una codificación corta para identificaciones cortas que sean necesarias y referencias. Para la codificación larga se utilizará el nombre del grupo de casos de uso del sistema cuya funcionalidad engloba este componente o el nombre de la funcionalidad misma antecedido por el tipo de componente en minúscula y un guión bajo. Para el tipo de componente se usarán identificadores de 3 caracteres que estarán recogidos en un registro. Algunos de los tipos de componentes con sus identificadores son: "Modulo del Proyecto" (mod), "Tabla de la Base de Datos" (tbl), etc. Un ejemplo de nombre de componente puede ser: *"mod_Planificación"*, *"tbl_usuario"*. Para la codificación corta de componentes se usará el tipo de componente en minúscula seguido de hasta 6 caracteres de cada palabra que forme parte del nombre de la funcionalidad o grupo de funcionalidades que engloba. Por ejemplo: *"modAccContr"*, *"modPlanif"*.

Para el nombrado de entregables se utilizará un esquema de codificación significativa diferente a las de los ECS puntuales. Los entregables se brindarán comprimidos en formato RAR a menos que sea absolutamente necesaria la entrega en otro formato. El nombrado de estos archivos entregables comprimidos comenzará por el nombre del proyecto en mayúsculas, un guión bajo y el identificador del tipo de entregable también en mayúscula. Los tipos de entregable y sus identificadores serán: "de documentación" (DOC), "de expediente" (EXP), "de aplicación" (APL) y "entregables compuestos" (SIS). Seguidamente se le agregará el número de la última revisión que tenga el ECS contenido en el entregable con el número mayor de revisión en el momento de la creación del entregable y la fecha de creación de éste separados del resto por guiones. La fecha al final del nombre será un solo número de 8 cifras en el cual los primeros 4 números serán el año, los 2 siguientes indicarán el mes y los 2 últimos el

día. El en caso del mes y el día se usarán ceros para rellenar los faltantes que pudieran generar meses y días cuyo número es de una sola cifra. Un ejemplo de nombre de entregable puede ser: *"IMAC-DOC-2867-20090520"*.

Para el nombrado de líneas base e hitos se utilizarán cadenas sin especificaciones que serán determinadas por el líder el Comité de Control de Cambios (CCC) pero agregándole al principio el número de orden de éste antecedido por "LB" para las líneas base ó "HT" para los hitos y la fecha de creación con el mismo formato de fecha usado para los entregables todos separados por guiones. Por ejemplo: *"HT017-20081016-Primer ciclo completo"*, *"LB004-20080220-Todos los requerimientos"*. Para referenciar a hitos o a líneas base se puede usar la primera parte del nombrado de estos, que para el ejemplo anterior sería *"HT017"*, *"LB004"* que, por conformar una codificación no completamente significativa se guardará en un registro de líneas bases e hitos.

## DEFINICIÓN DE LAS BIBLIOTECAS DE SOFTWARE

Para el proyecto se establecerán las siguientes bibliotecas de software:

- **Biblioteca Maestra:** En esta biblioteca se almacenarán entregables, las líneas base aprobadas y establecidas y las liberaciones (releases) del sistema. Los elementos almacenados en esta biblioteca se encontrarán comprimidos en formato RAR y se mantendrá un control de cambios formal sobre ellos brindando acceso de lectura a todos los implicados en el proyecto y de escritura solo a los miembros del comité de control de cambios (CCC).
- **Biblioteca de Soporte:** Almacenará elementos de desarrollo que sirvan de ayuda al proyecto y ramas paralelas que puedan separarse de la línea principal de desarrollo. El acceso de lectura será para todos y la escritura solo a quien interese.
- **Biblioteca de Trabajo:** Comprende el área de trabajo de cada grupo de desarrollo de manera organizada. Estará dividida en dos partes: un área para cada grupo de desarrolladores por rol, equipo de trabajo o módulo del proyecto según se necesite, en la que se establecerá un control de cambios semi-formal y otra área individual para cada desarrollador con acceso total para él y control informal de cambios.

- **Biblioteca de Salvas:** Almacenará salvas periódicas de las demás bibliotecas en formato comprimido RAR. Sobre esta biblioteca no se establecerá control de cambios y el acceso a ella estará limitado exclusivamente al Administrador de la Configuración del Proyecto.

Teniendo en cuenta que los implicados en el proyecto son exclusivamente:

1- El coordinador del MAC y los clientes que este invite los cuales requieren ver el funcionamiento de la solución
2- Los grupos de la dirección administrativa de la universidad como los grupos de calidad de la facultad y de la IP y otros implicados en la producción en la facultad los cuales requieren ver el funcionamiento de la aplicación, el funcionamiento del proyecto y el cumplimiento del equipo de desarrollo
3- Y los desarrolladores del proyecto que requieren modificar los ECS de sus respectivas áreas y ver los de las otras áreas.

Y además atendiendo a las políticas de la universidad que establecen confidencialidad para los ECS de proyecto de modo que se asegure cierto nivel de seguridad sobre ellos, se decidió usar un control automatizado de versiones que funcione en modo centralizado, que permita la asignación de permisos de acceso a usuarios, que permita el control a nivel de línea de archivo para asegurar el control necesario sobre el código fuente y que provea facilidades para el acceso y visualización suficientes para los recursos y las necesidades del proyecto.

Según los estudios realizados en el primer capítulo de esta investigación y las tendencias de los proyectos de la universidad se decidió usar Subversion como herramienta de control automático de versiones.

Como Subversion permite el establecimiento de permisos a usuario por directorio y la actualización parcial de contenido se pueden establecer las bibliotecas Maestra, de Soporte y de Trabajo en un mismo repositorio de Subversion pues bastaría con configurar permisos especiales para los subdirectorios que representarán cada biblioteca y asignar a cada desarrollador o grupo de desarrolladores la dirección completa del directorio sobre el que necesitan trabajar. Con esto además se obtiene un beneficio agregado y que es la posibilidad de hacer seguimiento de todos los cambios ocurridos sobre los ECS a la vez, incluso se puede hacer seguimiento de cambios en las áreas donde el control de cambios es informal; opción que puede ayudar a la gerencia a determinar la cantidad de trabajo por día que realiza cada

desarrollador basado en los cambios sobre sus áreas de trabajo dentro del control automático de versiones.

Para un acceso más cómodo Subversion permite la comunicación sobre protocolo HTTP usando WebDav. Sobre este punto y basándose en experiencias prácticas se decidió instalar Subversion en la misma PC que el servidor PostgreSQL y Apache2, beneficiándose de este último para la comunicación HTTP. Los pasos para la instalación y configuración de repositorio de Subversion con comunicación por HTTP y autenticación contra el controlador de dominio UCI sobre el sistema operativo Ubuntu 9.04 se encuentran en los anexos de este trabajo.

Para los usuarios que necesiten acceso de lectura al repositorio descritos anteriormente como implicados de tipo 1 y 2 se podrá usar el acceso vía HTTP usando un navegador web cualquiera como Mozilla Firefox 3.04 o Internet Explorer 7. Para los implicados de tipo 3, o sea los desarrolladores que requieren acceso de lectura a todo el repositorio y de escritura a sus áreas de trabajo se usará:

- Desde Windows XP
  - **TortoiseSVN:** Disponible gratuito en http://tortoisesvn.tigris.org/. Esta herramienta se integra con el Explorador de Windows para brindar un acceso realmente sencillo de usar. Tiene ciertos problemas con renombrar y mover archivos dentro del repositorio pero estos se ven opacados por su facilidad de uso.
- Desde Ubuntu 9.04
  - **svnclient:** Disponible en el repositorio de Ubuntu en el mismo paquete de Subversion Es una aplicación de consola brindada por los propios desarrolladores de Subversion. Es un poco engorroso de usar precisamente por la interacción por comandos pero implementa todas las funcionalidades de la comunicación con el servidor Subversion a la perfección.
  - **RapidSVN:** Disponible en el repositorio de Ubuntu. Es una herramienta específica con interfaz gráfica que permite trabajo bastante sencillo con el repositorio. Es bastante más cómodo que el cliente de consola pero ha presentado problemas en operaciones de renombrar y mover archivos dentro del repositorio.

Una breve presentación de guía sobre el uso de estos clientes de Subversion para el proyecto IMAC se encuentra en los anexos de este trabajo.

## DISTRIBUCIÓN DE LOS ECS

Uno de los puntos importantes a tener en cuenta para la organización de los ECS, además de las relaciones entre ellos y su esquema de nombrado, es la distribución que estos ocuparán.

Los ECS serán agrupados en directorios siguiendo un orden similar al descrito en la plantilla de expediente de proyecto propuesta por la Dirección Técnica de la IP de la universidad (DT-IP). Los nombres de directorios que se proponen serán ligeramente modificados para ajustarse al esquema de nombrado de directorios establecido con anterioridad, se les eliminará la numeración en el nombre y se les sustituirán los espacios por guiones bajos. A ésta propuesta se le agregarán además algunos directorios que son necesarios por IMAC. Algunos de ellos con su dirección dentro de la propuesta de expediente de la DT-IP son:

- **Planificación** *(/Gestión_de_Proyecto/Recursos/)*: Alberga el cronograma de trabajo del proyecto.
- **Individuales** *(/Gestión_de_Proyecto/Recursos/Planificación/)*: Alberga las planificaciones individuales de cada desarrollador.
- **Cronogramas Antiguos** (/Gestión_de_Proyecto/Recursos/Planificación/): Alberga cronogramas que han sido descartados por una u otra razón y que ya no se usarán más que para referenciar hechos antiguos que no se generaron.
- **Base de Datos** (/Ingeniería/Arquitectura_y_Diseño/): Alberga los modelos de datos, modelos de BD, diagramas ER, diccionario de datos, etc.
- **Mod_Acciones_de_Control** (/Ingeniería/Implementación_y_Pruebas/Código_Fuente/): Alberga los archivos de personalización y configuración de Alfresco Labs para el gestor documental necesario en el módulo de Acciones de Control.
- **Mod_Planificación** (/Ingeniería/Implementación_y_Pruebas/Código_Fuente/): Alberga los códigos fuentes generados del módulo de Planificación.

Se debiera establecer el nivel de profundidad en el control que se le quiere dar a cada ECS con el objetivo de evitar controlar los ECS a un nivel superior al que ellos requieren. El nivel al que se controla un ECS es directamente inverso la fortaleza del control. Si se establece el control de un ECS a un nivel superior al que requiere pudiera perderse información necesaria a la hora de realizar verificaciones de los

cambios. Si, por el contrario, se establecen niveles por inferiores a los necesarios para un ECS se estarán dedicando recursos extras para algo que no es necesario mantener con un control tan estricto.

Los ECS que constituyen parte del código fuente de la solución son archivos de texto plano que requieren un control a nivel de línea dentro del archivo. Esto sucede porque cuando se trabaja con código fuente los cambios muy pequeños en el texto pueden representar grandes cambios en la funcionalidad y ser, por lo general, muy difíciles de encontrar. Entre estos ECS están los archivos con extensión .PHP, .HTML, .CSS, .PHTML, .SQL, .ACP, algunos archivos .TXT, .XML,

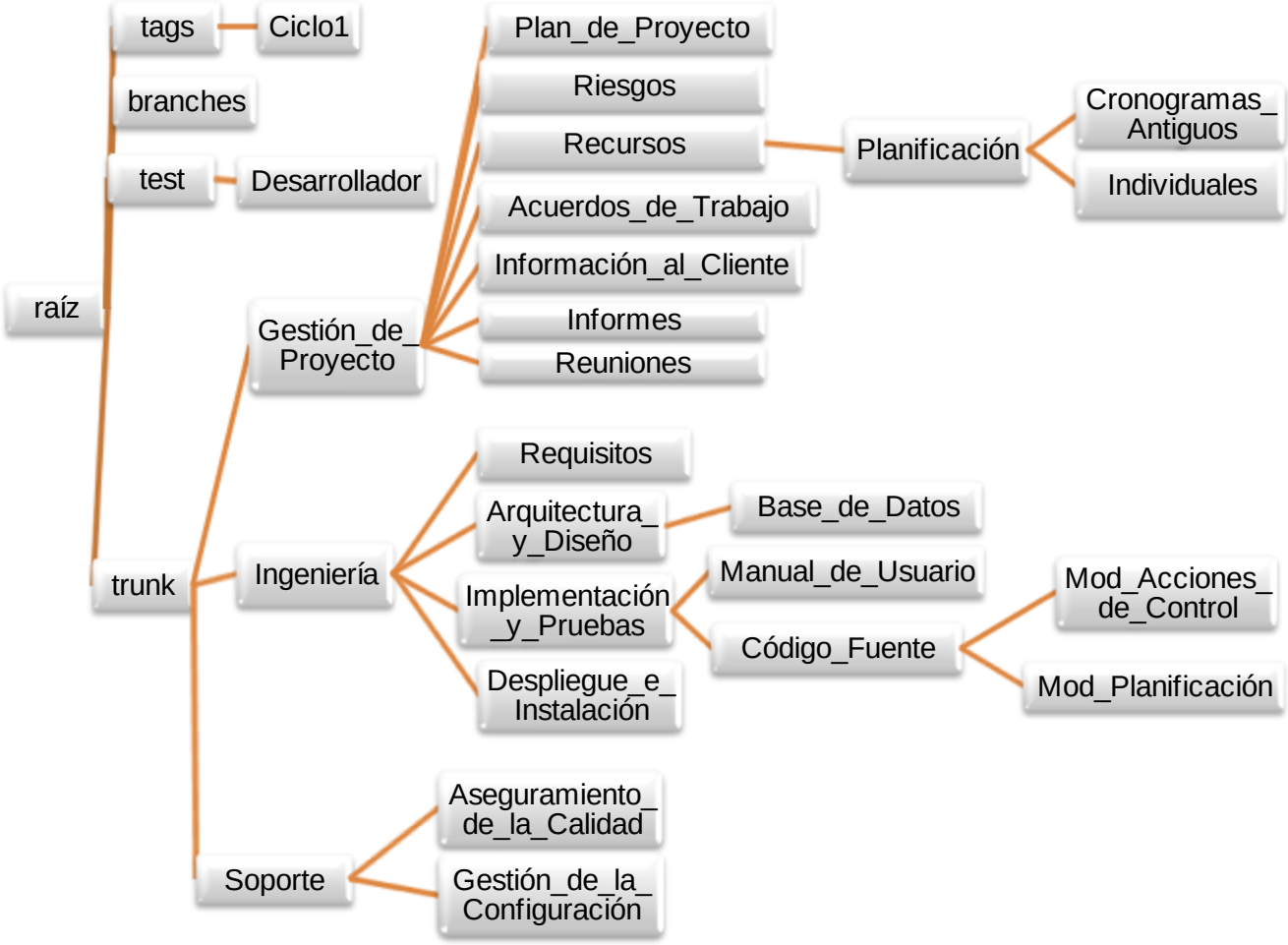
Por otra parte, los ECS que forman parte del expediente de proyecto, de la documentación de ingeniería, las imágenes, los entregables comprimidos y otros ECS que son generados como archivos binarios por herramientas, no necesitan un control a nivel de línea pues un solo elemento del contenido importante de estos ECS puede estar contenido en varias líneas del archivo. Es por ello que para estos ECS con un control a nivel de archivo completo es suficiente.

Teniendo en cuenta esta división por nivel de control requerido se debiera dividir en dos partes la distribución de los ECS y reagruparlos siguiendo las especificaciones antes planteadas e incluyendo este nuevo elemento organizativo.

Además, el control de algunos ECS a nivel de archivo hace lógico pensar en una solución de gestión documental para ellos, que ya se trata en el proyecto y que es el mismo Alfresco Labs. Sin embargo, como se describió en el epígrafe “Recursos Materiales Asignados” de este mismo capítulo, el gestor documental se instalará para realizar pruebas de funcionamiento, configuración y personalización de Alfresco Labs, lo que lo convierte en una instancia muy mutable y riesgosa. También se explicó que, debido a sus requerimientos mínimos de funcionamiento, cada instancia de Alfresco Labs requeriría un PC completamente dedicado para él, lo que comprimiría aun más el horario de trabajo de las solamente 13 restantes estaciones de trabajo. Como resultado se decidió mantener todos los ECS bajo el nivel de control más estricto necesario en aras de mantener la mayor cantidad posible de PCs como estaciones de trabajo.

Teniendo en cuenta además la creación de las bibliotecas definidas anteriormente bajo un mismo repositorio se propone la siguiente organización dentro de este:

*ILUSTRACIÓN 36* ***DISTRIBUCIÓN DEL REPOSITORIO PARA IMAC***




Los directorios tags, branches, test y trunk responderán a cada una de las bibliotecas que son necesarias.

- **tags:** Representa la Biblioteca Maestra.
- **branches:** Representa la Biblioteca de Soporte.
- **test:** Representa el área individual de cada desarrollador dentro de la Biblioteca de Trabajo.
- **trunk:** Representa la Biblioteca de Trabajo de los grupos de usuarios.

Para establecer los permisos de acceso a las diferentes áreas del repositorio se han creado los siguientes grupos de usuarios:

- **Todos:** Incluye todos los desarrolladores del proyecto. Tienen acceso de solo lectura a todo el repositorio y escritura en el directorio *Individuales* y en sus directorios propios dentro de del directorio *test*.

- **Organizad:** Líder, Sublíder, Analista Principal, Arquitecto, Jefe del Equipo de Programadores y Administrador de la Configuración. Tienen acceso de escritura a todo el repositorio.
- **Calidad:** Encargados de Calidad del proyecto. Tienen acceso de escritura al directorio de *Aseguramiento de la Calidad*.
- **Datos:** Diseñadores de BD. Tienen acceso de escritura al directorio de *Base de Datos*.
- **Analista:** Analistas del proyecto. Tienen acceso de escritura al directorio de *Requisitos*, *Arquitectura y Diseño*, *Manual de Usuario* y algunos de ellos, por necesidades especiales, a *Despliegue e Instalación*.
- **ProgAccCntrl:** Programadores del módulo de Acciones de Control. Tienen acceso de escritura a *Mod_Acciones_de_Control*.
- **ProgPlanif:** Programadores del módulo de Planificación. Tienen acceso de escritura a *Mod_Planificación*.
- **ProgIntegr:** Programadores del módulo de Integración. Tienen acceso de escrita a partes específicas dentro de *Mod_Planificación*.
- **AuditExter:** Incluye los usuarios de los implicados de tipo 1. Tienen acceso de solo lectura al directorio *tags*.
- **AuditInter:** Incluye los usuarios de los implicados de tipo 2. Tienen acceso de solo lectura a todo el repositorio.

Observe que los grupos a partir del segundo y con excepción de los dos últimos, además de sus permisos propios, heredan los permisos definidos para el grupo **Todos**.

## HERRAMIENTA DE GESTIÓN

Para lograr un acceso más fácil a las estadísticas del repositorio y contribuir al control simple pero robusto se decidió instalar una herramienta de gestión de proyecto que permita realizar trabajo colaborativo centralizado en un solo proyecto, control de incidencias, gestión de recursos, publicación, que se integre con el repositorio de Subversion, que sea accesible vía web, que permita generar reportes variados según las necesidades, que sea extensible de manera relativamente fácil y que sea gratuito.

De las herramientas que se exponen en el capítulo 1 de este trabajo, dos cumplen con estos requisitos que hemos definido para la herramienta de gestión de proyecto necesaria: Trac y Redmine. Sin embargo Redmine tienen algunos elementos que influencian negativamente en su selección. Primero, su flexibilidad sin entrar en código

es inferior a la de Trac como puede fácilmente observarse comparando la cantidad de plug-ins que ofrecen ambas comunidades de soporte. Además el lenguaje Ruby y la dependencia del motor intermedio “Ruby on Rails” lo hacen mas difícil de estudiar en con el objetivo de implementar futuras funcionalidades necesarias. Por último, Redmine no se encuentra en los repositorios de Ubuntu 9.04 por lo que es necesario descargarlo desde el sitio de sus desarrolladores en internet el cual solo provee del código fuente para su instalación manual y un instalador expreso muy bueno para sistemas vírgenes pero de muy poca flexibilidad.

Trac, por otra parte, usa Python, lenguaje de programación de amplia aceptación y cuyas dependencias son instaladas por defecto cuando se instala Ubuntu 9.04. La instalación de Trac puede hacerse a través de los estables que están dentro del repositorio de Ubuntu 9.04, el cual contiene Trac y todas sus dependencias, o también puede descargarse como archivo en formato comprimido TAR.GZ e instalarse como módulo del Python ya existente con solo 1 comando en la consola.

Por las razones expuestas en el capítulo 1 de este trabajo y las expuestas en los dos párrafos anteriores se decidió usar Trac como herramienta de gestión de proyecto, instalándose en la PC que ejecuta el servidor de Subversion, el servidor Apache2 y servidor PostgreSQL beneficiándose de estos. En los anexos de este trabajo se encuentra una guía paso a paso para la instalación y configuración de Trac.

## ESTRATEGIA DE SALVA Y RECUPERACIÓN

La Biblioteca de Salvas establecida para salvaguardar copias periódicas de la información contenida en las demás bibliotecas estará constituida por dos directorios físicos dentro del sistema de archivos de las 2 PCs usadas como proveedoras de servicios para el proyecto, los servidores del proyecto.

El acceso local a las PCs servidoras estará limitado al Líder del Proyecto, al Sublíder, al Administrador de la Configuración. Al Servidor02, que se usa para las pruebas de Alfresco Labs, tendrán acceso local también los analistas encargados de dichas pruebas.

Como toda la información de las bibliotecas esta contenida en un solo repositorio, las salvas se realizarán mediante un solo archivo de exportado del repositorio completo. Todos los días a las 0000h se realizará de manera manual la creación del exportado del repositorio a un archivo local dentro del directorio definido para la Biblioteca de Salvas conteniendo el exportado incremental para ese día.

La idea de realizar el exportado de manera incremental responde a que el tamaño del repositorio puede crecer mucho. La salva incremental se realizará usando el comando de Subversion para ello: ***"svnadmin dump /RUTA/REPOSITORIO > /RUTA_S/SALVA --incremental"*** donde RUTA es la dirección local donde se encuentra el repositorio, REPORITORIO es el nombre del directorio de este, RUTA_S es la dirección local del directorio de salvas y SALVA es el nombre del archivo de salva incremental. Este proceso se realizará en el Servidor01 y el archivo de salva incremental será copiado después de su creación diaria al otro directorio de la biblioteca de salvas que se encontrará en el Servidor02. Esta actividad será responsable del Administrador de la Configuración y será supervisada semanalmente por el Líder y el Sublíder del proyecto.

El proceso de recuperación en caso de pérdida de datos será responsabilidad del Administrador de la Configuración que realizará esta tarea bajo supervisión del Líder y Sublíder del proyecto. Se realizará a partir de las salvas usando *"**svnadmin load /RUTA/REPOSITORIO/ < /RUTA_S/SALVA**"*. Des pues de esta actividad se realizará una revisión de la copia local de cada desarrollador para prevenir pérdida de datos por concepto de sincronización de las versiones poseídas por los desarrolladores y la versión actual en el repositorio.

## GESTIÓN Y CONTROL DE CAMBIOS

### COMITÉ DE CONTROL DE CAMBIOS

El Comité de Control de Cambios (CCC) es la entidad encargada de tomar las decisiones importantes sobre los cambios durante el ciclo de vida del proyecto. El CCC deberá valorar la posible aceptación o rechazo de los cambios propuestos sobre los ECS que se encuentran bajo control formal de cambios. El CCC estará integrado por miembros permanentes y miembros especiales.

Los miembros permanentes serán los roles más importante en la dirección del proyecto, cuya decisión tendrá el mayor peso. Entre ellos se encontrará:

- Líder del proyecto.
- Sublíder.
- Planificador.
- Arquitecto.
- Analista Principal.
- Jefe del Equipo de Programadores.

- Jefe del Equipo de Aseguramiento de la Calidad.
- Administrador de la Configuración.

Las opiniones de todos los miembros permanentes del CCC tendrán igual peso en las decisiones aunque el líder de proyecto puede acotar dicha equidad si la situación lo requiere.

Los miembros especiales serán aquellos roles cuya decisión es importante solo en determinados asuntos. Entre ellos se encontrará:

- Jefes de Módulo.
- Diseñador de Base de Datos.
- Equipo de Aseguramiento de la Calidad.

## DEFINICIÓN DEL PROCESO DE CONTROL DE CAMBIOS

En todo momento el área individual de la biblioteca de trabajo representará trabajo no validado de los desarrolladores y el área de trabajo de equipo de la biblioteca de trabajo representará el trabajo colaborativo pre-validado de los integrantes del equipo de desarrollo desde la última línea base establecida y aprobada hasta la actualidad.

De manera general los ECS se encontrarán en el momento de su creación y mientras no formen parte de ninguna línea base en la biblioteca de trabajo, ya sea en el área individual del desarrollador que lo ha creado o en el área del equipo de trabajo a que su creador pertenece. Mientras el ECS no pertenezca a ninguna línea base se considerará bajo control informal permitiendo al desarrollador hacerle todos los cambios justificados que estime conveniente. Cuando los ECS son pre-revisados pasan a control semi-informal de cambios, si se encontraban en el área de trabajo individual son pasados al área de trabajo del equipo y una vez allí los cambios importantes, de gran envergadura o que tengan repercusión en otros ECS deben tramitarse a través del Líder de proyecto. Una vez que los ECS son incluidos en una línea base pasan al control formal de cambios y los cambios en ellos deben ser estrictamente controlador por el CCC.

Los cambios en los ECS que se encuentren bajo control formal o semi-formal solo ocurrirán para satisfacer el cumplimiento de tareas asignadas que impliquen estos cambios. Estas tareas pueden aparecer de dos maneras:

- **Planificadas:** Son los cambios para los que se han planeado con anterioridad su ocurrencia mediante tareas en el cronograma de proyecto con fecha tentativa y que tributan a un hito planificado.
- **Por Incidencia:** Son los cambios que no se encuentran con anterioridad en el cronograma de proyecto y que responden a tareas asignadas a partir de solicitudes de mejora o reportes de errores.

## PROCESO DE CAMBIO PLANIFICADO

El proceso de cambio planificado comenzará con la planificación de macro-tareas por parte de los responsables de equipo, el líder de proyecto y el planificador. Esta macro-planificación del trabajo se dividirá en etapas cuyo final conformará hitos planificados. Para cada hito se definirá el conjunto de ECS, funcionalidades o tareas que deben estar cumplidas, funcionales o completos según el caso y una fecha tentativa. Esta primera planificación será revisada por el líder de proyecto y una vez aceptada pasará a conformar el cronograma de proyecto quedando este desde el inicio bajo control semi-formal por parte del planificador.

Cada macro-tarea asignada a un rol, a un desarrollador, a un grupo de desarrolladores o a un grupo de roles deberá ser desglosada en tareas simples por cada uno de los implicados en ella, definiendo para cada tarea simple: una fecha tentativa y una duración estimada. Si la macro-tarea es asignada a un grupo de roles o a un grupo de desarrolladores el desglose lo debe hacer el jefe o responsable de dicho equipo, desglosando en sub-tareas para cada integrante de manera que este pueda luego desglosar la sub-tarea en tareas simples propias. El aviso de asignación de tareas, sub-tareas o tareas simples se realizará a través de los tickets de tipo “Tarea” de Trac.

El listado de tareas simples y sub-tareas a controlar en caso de ser responsable de equipo se incluirá en la Planificación Individual de cada desarrollador y se mantendrá bajo control semi-formal de cambios. Los cambios hechos sobre la planificación individual deberán ser revisados por el Planificador del proyecto para verificar que están acordes a la planificación general del proyecto y aceptados o denegados por este según se requiera. Si la planificación individual es aceptada el planificador incluye estos desgloses en el cronograma del proyecto. Si es rechazada se envía una notificación al responsable de la tarea o lista de tareas con la razón del rechazo y este debe re-planificarlas.

Cuando se cumpla completamente una tarea y se suban al repositorio del equipo de trabajo los cambios que finalizan la ejecución de esta tarea se debe cerrar el ticket referente a la tarea en Trac y actualizar la planificación individual. El jefe de equipo o quien haya enviado la tarea deberá verificar que esta se ha cumplido. Si la tarea no se cumplió correctamente el revisor deberá reabrir el ticket relacionado y notificar al desarrollador. Si la tarea se cumplió correctamente el o los ECS modificados pasarán a ser revisados por los encargados de calidad del proyecto para ser incluidos bajo el control formal de cambios y posteriormente en la próxima línea base.

### PROCESO DE CAMBIO POR INCIDENCIA

El proceso de cambio por incidencia comienza con el levantamiento de una incidencia por alguno de los implicados. Esta puede tener dos variantes:

- Por corrección de defecto
- Por mejora de elemento

Cualquier implicado puede levantar una incidencia pero por lo general las incidencias levantadas por los implicados de tipo 1 son siempre clasificadas como “corrección de defecto” aunque no lo sean. Las incidencias levantadas por el equipo de desarrollo y los implicados de tipo 2 suelen ser más precisas en su clasificación. Además es común el levantamiento de incidencias repetidas, erróneas o con datos insuficientes para su procesamiento. Es por ello que al levantarse una incidencia el equipo de desarrollo debe analizarla para establecer validez y su clasificación correcta antes de pasar a procesarla.

Para levantar una incidencia el implicado que vaya a realizarlo deberá crear un nuevo ticket de “mejora” o de “defecto” según sea el caso en el ambiente Trac. Si la incidencia es levantada por un cliente a través del coordinador del MAC, un implicado de tipo 2 u otra persona que no tiene acceso al Trac, no posee permisos en este para levantar incidencias o no se encuentra en posición de acceder al Trac en el momento en que va a hacer el levantamiento, este debe llenar un informe de incidencia y hacerlo llegar al líder de proyecto para que este lo tramite el Trac. Este informe deberá contener la siguiente información:

- Fecha de incidencia.
- Implicado que realiza el levantamiento.
- Descripción del incidente.
- Proceso de duplicado del incidente. (Si se puede repetir)

- Tipo de prueba que se realizaba. (Si es aplicable)
- Observaciones.

Cada informe de incidencia deberá ser archivado y procesado como si la incidencia hubiera llegado a través de un ticket de Trac.

El procesamiento de las incidencias lo realiza el comité de control de cambios. Este comienza por decidir si el cambio implicado en la incidencia puede llevarse a cabo. Para ello el CCC debe tener en cuenta aspectos como:

- Valor del cambio para el proyecto.
- Tamaño.
- Complejidad.
- Impacto sobre el rendimiento del producto (uso de memoria y CPU)
- Recursos disponibles para efectuar el cambio (humanos y materiales)
- Relación con otros cambios ya aprobados y en progreso.
- Tiempo estimado para completar el cambio.
- Relación con las políticas de la empresa (satisfacción del cliente, competitividad, etc.)
- Existencia de alternativas.

Si el CCC decide no proceder con el cambio se envía un informe de denegación al implicado que realizó el levantamiento de la incidencia explicando las razones por las cuales no se procede.

Si el CCC decide proceder con el cambio, este genera las macro-tareas y asigna los responsables necesarios para el proceso. Desde ese momento el cambio se tramita como un cambio planificado pero se mantiene abierta la incidencia que lo generó hasta el último paso del proceso planificado de cambios en el que los ECS generados son revisados y pasados a control semi-formal o formal según se entienda, entonces la incidencia es cerrada.

Si el proceso de cambios generado a partir de una incidencia provoca cambios sustanciales que modifiquen una parte importante del proyecto o que el CCC considere necesario, se generará un hito relacionado con el cierre de la incidencia en cuestión.

## CREACIÓN DE LÍNEAS BASE

Las líneas base del proyecto contendrán los archivos que han sido revisados formalmente por el equipo de aseguramiento de la calidad y han recibido el visto bueno del CCC.

Se establecerá una línea base al final de cada una de las fases de RUP y al final de cada iteración. Al comienzo de cada fase se tomará como línea base la que fue generada al final de la fase anterior y se comenzará a trabajar sobre esta. Las líneas base se planificarán para realizarse:

- Al final de la primera iteración de la fase de inicio. **LB1**.
- Al final de la última iteración de la fase de inicio. **LB2**.
- Al final de la fase de elaboración. **LB3**.
- Al final de la segunda iteración de la fase de construcción. **LB4**.
- Al final de la fase de construcción. **LB5**.
- Al finalizar las pruebas de aceptación en la fase de transición. **LB6**.
- Al finalizar el piloto en la fase de transición. **LB7**.

Las líneas base contendrán: los ECS de las líneas base anteriores y que se han refinado o se mantienen vigentes y un conjunto de ECS propios según la fase de creación que tributen a los siguientes aspectos:

- **Inicio:** Visión, Recursos, Cronograma inicial, Aspectos Legales, Arquitectura general, Plan de Desarrollo del Software, Gestión de la Configuración, Metodología de desarrollo, Requerimientos, Casos de Uso del Negocio, Estándares.
- **Elaboración:** Plan de Aseguramiento de la Calidad, Modelo de Pruebas, Casos de uso del Sistema, Componentes, Diseño del Sistema, Prototipo de Interfaz, Modelo de Datos, Modelo del Despliegue, Modelo de Implementación, Manual del Usuario.
- **Construcción:** Código Fuente, Modelos Físicos de las BD.
- **Transición:** Listados de deficiencias, Resultados de las pruebas, Acciones Correctivas.

Las creaciones de líneas base constituirán hitos en el ciclo de desarrollo del software. Si en las fases de Inicio y Elaboración aparecieran cambios no planificados cuya importancia o magnitud fuese importante, el CCC pudiera decidir la creación de una nueva línea base. No así en las fases de Construcción y Transición en las cuales cambios de tal magnitud deberán esperar a la próxima versión.

## HITOS

Los hitos son momentos dentro de ciclo de vida del proyecto en los que se han completado una serie de actividades y funcionalidades o se han obtenido una serie de logros. Cada hito tiene asociado una serie de ECS y/o tareas cuyo completamiento indica el cumplimiento del hito asociado. En general pueden establecerse hitos para denotar el completamiento de cualquier tarea o grupo de ellas. Para cada hito se debe especificar:

- Nombre del Hito.
- Fecha tentativa.
- Tareas, actividades o elementos que deben completarse para alcanzarlo.
- Fecha real de completamiento.

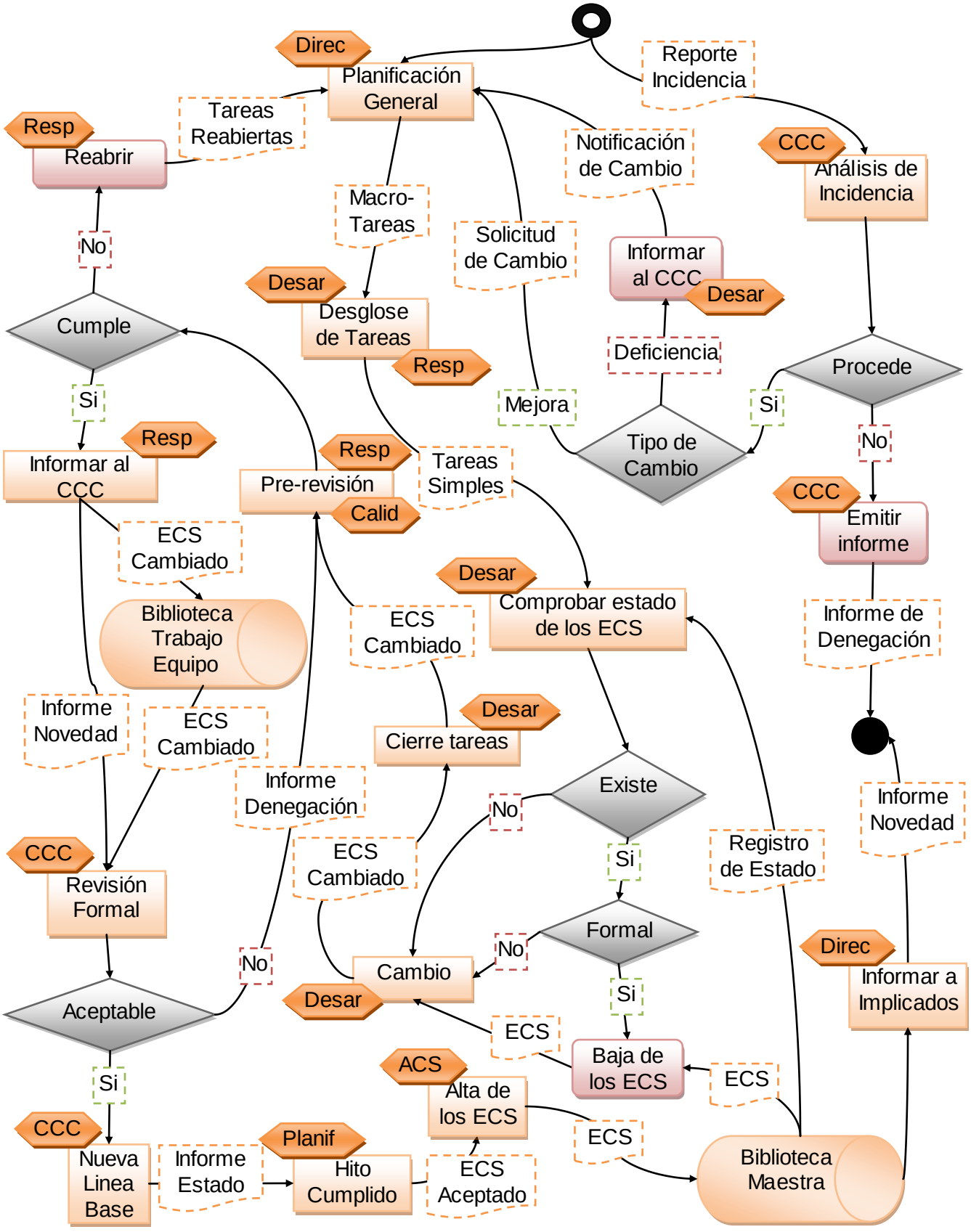
Existen dos tipos de hitos:

- **Planificados:** Son aquellos hitos que se definen con anterioridad cuando se realiza la planificación del proyecto y se les espera completarlos para una fecha tentativa. Estos hitos por lo general marcan pautas en el ciclo de desarrollo. Ejemplos de estos son los hitos definidos para completarse con la creación de las Líneas Base establecidas o los hitos de fin de fase.
- **No Planificados:** Son hitos que no se encontraban en la planificación inicial y que fueron incluidos durante el transcurso del proyecto. Por lo general estos hitos provienen de cambios importantes surgidos de incidencias. Por ejemplo los hitos que representan fusiones con otras ramas cuya existencia o funcionalidad se desconocía.

Todos los hitos se son definidos por el comité de control de cambios y son agregados al cronograma y controlados por el planificador del proyecto.

En el diagrama se muestra el proceso de control de cambios tanto formal como semi-formal en sus dos variantes de inicio, planificado o por incidencia. Los hexágonos representan la entidad que realiza el proceso descrito. Estas son: **Direc:** Equipo de dirección del proyecto, **CCC:** Comité de Control de Cambios, **Resp:** Responsable de revisión del proceso que se describe o Jefe de equipo, **Planif:** Planificador del proyecto, **ACS:** Administrador de la Configuración, **Calid:** Equipo de Aseguramiento de la Calidad y **Desar:** Desarrollador.

*ILUSTRACIÓN 37* ***PROCESO DE CONTROL DE CAMBIOS EN IMAC***




SUBIENDO CAMBIOS AL REPOSITORIO

Al subir cambios al repositorio desde el cliente para Windows XP se debe prestar atención a la lista de modificaciones que se va a subir. Un error muy común es el de subir archivos temporales como caché de imágenes (*Thumbs.db*), archivos de autoguardado (~$*.*), etc. Estos archivos temporales no son ECS y por lo tanto no forman parte del repositorio del proyecto.

Cuando se sube un cambio al repositorio se debe asignar un comentario significativo para este, que describa lo mejor posible el cambio que se está subiendo. Para ello se utilizará una construcción como sigue:

- [verbo en participio] [ECS modificado por la acción del verbo] [detalles de la modificación en caso necesario].

Los verbos en participio denotarán la acción realizada sobre el ECS y pueden ser: “agregado”, “creado”, “modificado”, “actualizado”, “eliminado”, “renombrado” o “movido”. Por ejemplo: “Actualizado Cronograma de Proyecto. Cerradas las tareas que han sido culminadas del 20 al 25 de febrero de 2009.”

En caso de haber más de un ECS modificado en el cambio que se sube se pondrá cada modificación con el formato descrito en forma de lista con un asterisco delante. Por ejemplo:

“* Actualizado Cronograma de Proyecto. Cerradas las tareas que han sido culminadas del 20 al 25 de febrero de 2009.

* Agregado Plantilla de Planificación Individual. Plantilla para establecer el nuevo sistema de planificación“

Se agregará al repositorio un gancho pre subida que no permitirá configurado para no aceptar la subida de cambios sin texto de comentario. Los pasos para la configuración de este gancho está en los anexos de este trabajo.

El repositorio Subversion y el ambiente Trac estarán además integrados para modificar los datos de Trac desde los comentarios de subida de cambios de Subversion. Esto se logra instalando el plug-in “RepositoryHookSystemPlugin” disponible en el sitio de la comunidad de extensiones de Trac. (The TracHacks Community) El gancho para lograr esta integración se encuentra en los anexos de esta investigación.

Se podrán cerrar y referenciar automáticamente desde los comentarios de Subversion los tickets de Trac agregando una línea al final del comentario de subida del cambio con el siguiente formato:

- [comando] #X & [comando] #Y & #Z

Donde "X", "Y" y "Z" son números de tickets existentes en Trac, "&" es la conjunción lógica "y" y "comando" puede ser:

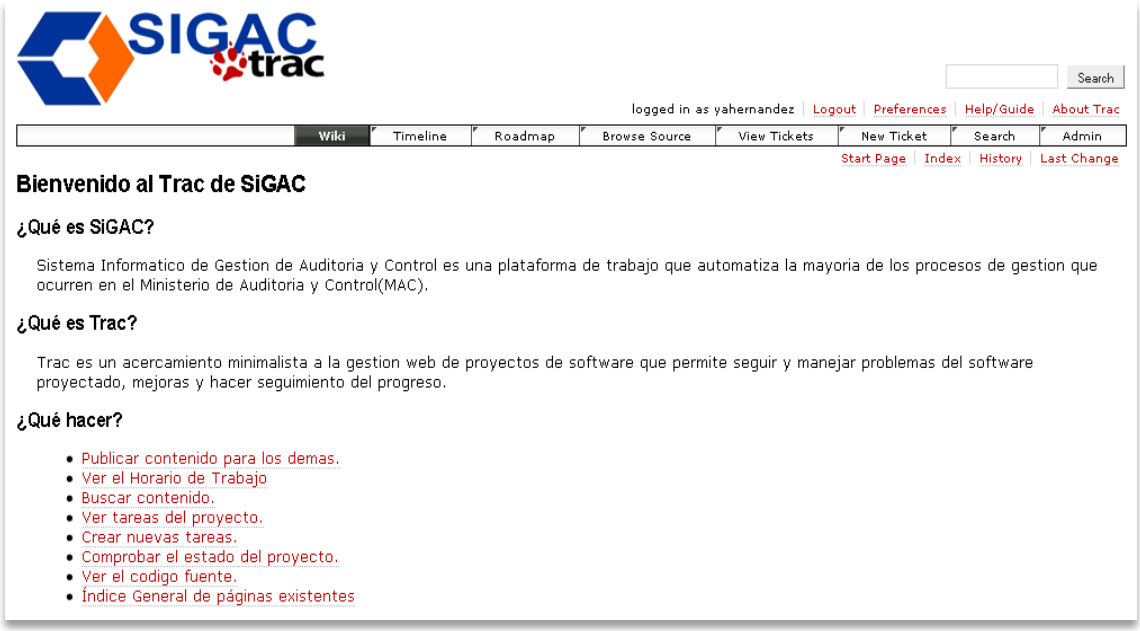
- **fix:** para cerrar un ticket
- **see:** para referenciar un ticket

Por ejemplo, si se agrega al final de un comentario de subida de cambios al repositorio el siguiente texto: ***"fix #3 & #6, and see #2"*** al subirse el cambio se cerrarán los tickets número 3 y número 6 y se agregará una nota al ticket número 2 con el texto del comentario de esta subida de cambios.

## GESTIONANDO CON TRAC

Trac obtiene mucha de la información que gestiona directamente del repositorio de Subversion brindándola de manera lógica y concisa al usuario. Sin embargo existe información que Trac necesita que sea gestionada manualmente, como la gestión de incidencias y tareas, la gestión de hitos y la configuración y actualización periódica del flujo de trabajo de los tickets.

*ILUSTRACIÓN 38* ***PANTALLA PRINCIPAL DEL TRAC DE IMAC***



La pantalla principal de Trac muestra por defecto la página de inicio de la wiki integrada (WikiStart). En dependencia de la plantilla instalada Trac puede verte ligeramente diferente, más o menos colorido y con las barras y elementos en diferentes posiciones de la pantalla. Sin embargo siempre estarán presentes los elementos principales que son:

- **La barra de usuario (UserBar):** Contiene la interfaz de autenticación (Login), el acceso al panel de preferencias (Preferences), el acceso a la ayuda (Help/Guide) y el panel de créditos e información de contacto (About Trac).
- **La barra de navegación principal (MainNav):** Contiene el acceso a los componentes de Trac. En dependencia de los permisos asignados al usuario actual y de las extensiones instaladas esta barra contendrá más o menos elementos pero los principales son:
    - **Wiki:** Sistema de publicado de información en la wiki integrada.
    - **Timeline:** "Linea de tiempo", contiene el listado de actividades realizadas sobre el repositorio, la wiki, el sistema de tickets y el sistema de hitos.
    - **Roadmap:** "Camino por recorrer", contiene los hitos definidos con sus respectivas fechas tentativas, su estado de completamiento y los tickets asociados a él.
    - **Browse Source:** Contiene una interfaz web del repositorio con la que se puede navegar por los directorios del repositorio y por las versiones de cada directorio.
    - **View Tickets:** Brinda acceso al sistema de reportes.
    - **New Ticket:** Es un acceso directo a la interfaz de creación de nuevos tickets.
    - **Search:** Brinda acceso a la interfaz de búsqueda con la que se pueden realizar búsquedas sobre los tickets, la wiki, los hitos y hasta dentro del repositorio.
    - **Admin:** Brinda acceso al panel de administración de Trac. Este elemento solo se muestra para usuarios con derechos administrativos sobre Trac.
- **La barra de navegación secundaria (SecNav):** Esta barra cambia sus opciones según el módulo o componente de Trac en el que se esté navegando e inclusive puede desaparecer para módulos que no la necesiten. Por lo general contiene funciones para avanzar y retroceder en sistemas continuos. Por ejemplo permite moverse entre periodos de tiempo del timeline.

La gestión de incidencias y tareas se realiza en Trac a través de tickets. Un ticket de Trac es un elemento básico de información que Trac obtiene a través de los usuarios que interactúen con él y que debe ser mantenido y gestionado. La opción de “Tipos de Ticket” del panel de administración permite configurar Trac para gestionar cualquier elemento diferenciadamente a través de tickets.

Para crear nuevos tickets se selecciona la opción “New Ticket” de la barra de navegación principal. Con esto se accede a la ventana de nuevo ticket. Al llenar el formulario y presionar sobre el botón “Create Ticket” se creará un nuevo ticket con los datos que se hayan seleccionado. Quizás la parte mas importante es la caja combinada denominada “Type”; en ella se selecciona el tipo de ticket a crear, que puede ser “Tarea” para crear tareas, “Defecto” para crear reportes de defectos o “Mejora” para solicitar mejoras.

Algunas cajas de texto son llenadas automáticamente por Trac con los valores predeterminados que estas poseen. Por ejemplo la caja “Reporter” que representa quien realiza el reporte de ticket es llenada automáticamente por Trac con el nombre del usuario actualmente identificado. Todas las cajas que Trac llena con valores predeterminados pueden ser modificadas antes de enviar el ticket.

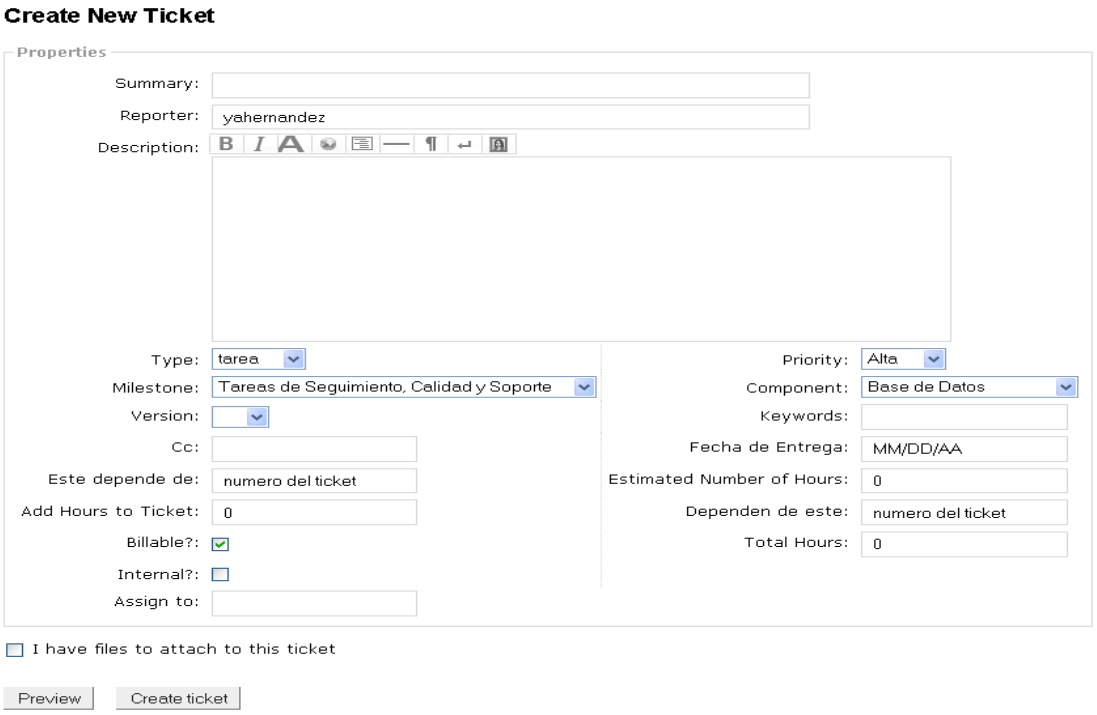
Las cajas combinadas, sin embargo, poseen valores estáticos que se llenan por selección de la lista de valores permitidos. Por ejemplo la caja “Priority” que representa la prioridad que se le quiere asignar al ticket.

Los valores de las cajas combinadas se pueden configurar en el panel de administrador de Trac, actividad que pertenece a las responsabilidades del Administrador de la Configuración en coordinación con el equipo de directivo del proyecto.

También pueden generarse atributos personalizados para los tickets que constituyan nuevas cajas de texto o combinadas a llenar para la creación de nuevos tickets. Estas cajas pueden representar cualquier información que se requiera guardar para todos los tickets y pueden ser configuradas como obligatorias u opcionales.

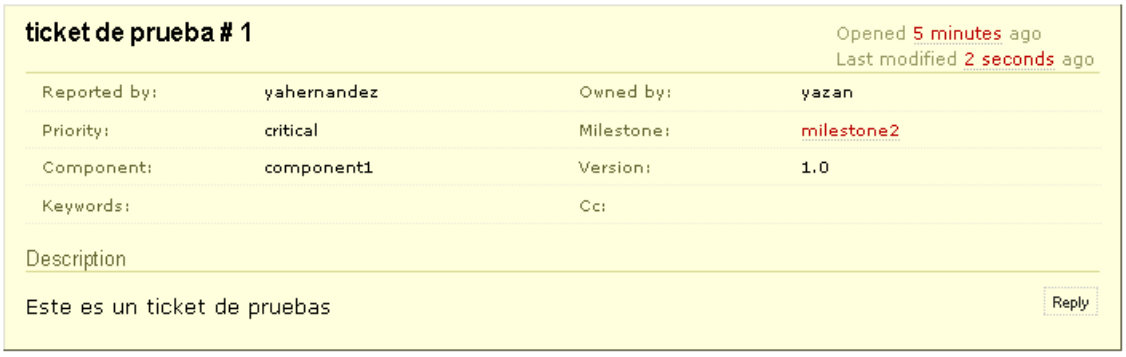
Para solicitar la creación de nuevos valores para los atributos de cajas combinadas y nuevos atributos personalizados para los tickets se creará un ticket de tarea para el Administrador de la Configuración quien, de conjunto con el CCC, valorará la propuesta.

*ILUSTRACIÓN 39* ***INTERFAZ DE CREACIÓN DE TICKETS DE TRAC***




Es importante observar que cuando se crea un ticket el ambiente Trac le asigna un número identificativo con el que se puede referenciar al ticket. Este identificador puede observarse en el título de la página una vez creado el nuevo ticket y en el resumen del ticket que aparece sobre un recuadro amarillo en la parte superior de la página.

*ILUSTRACIÓN 40* ***RESUMEN DE TICKET DE TRAC***



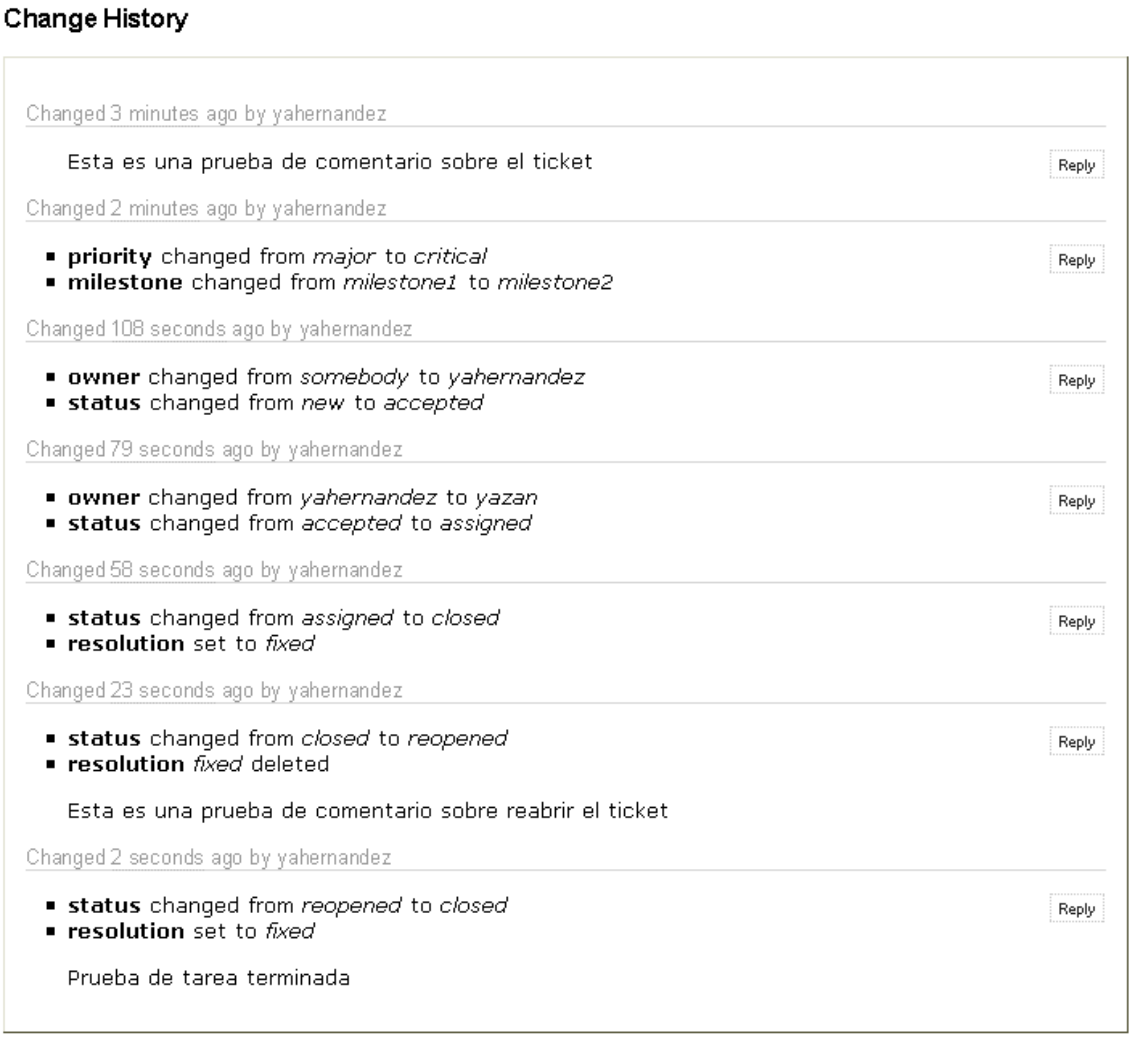

Al final de la página del ticket se encuentra el formulario para cambiar los estados del ticket. Este formulario es de importancia vital, pues controla el estado del ticket por el flujo de trabajo pre-configurado para él por el administrador.

Las opciones que posee este formulario dependen del estado actual del ticket y del flujo de trabajo establecido. Normalmente para un ticket recién abierto las opciones serían Asignarlo, Aceptarlo, Resolverlo o Dejarlo abierto. Para un ticket asignado se tendría Reasignarlo (a otro desarrollador), Resolverlo, Dejarlo asignado. Para un ticket

cerrado se tendría Reabrirlo, Dejarlo cerrado. Lógicamente estos estados dependen de la configuración del flujo de trabajo establecido para los tickets del tipo específico.

La otra parte más importante del ticket es el historial de cambios. Trac guarda un historial de todas las acciones realizadas sobre un ticket. Este historial puede verse debajo del resumen del ticket.

*ILUSTRACIÓN 41* ***HISTORIAL DE UN TICKET DE TRAC***




Las demás ventanas de los componentes de Trac son bastante auto explicativas.

## REPORTES DE ESTADO

El conocimiento del estado de los ECS es vital para el desarrollo del proyecto. Es por ello que se llevarán los siguientes registros y se crearán los siguientes informes en aras de mantener siempre informados a los implicados sobre el estado de la configuración.

## REGISTROS

El proyecto mantendrá los siguientes registros:

- **Registro de elementos de la configuración:** Lista cada ECS del proyecto que se encuentre bajo control formal o semi-formal, con su estado actual y el tipo de control que se realiza sobre él.
- **Registro de líneas base e hitos:** Lista todas las líneas base que se han establecido en el proyecto con sus respectivos nombres, fecha de establecimiento y ECS que la componen y los hitos que se han cumplido con su fecha, nombre y descripción. Este registro se llevará de manera semiautomática usando "camino por recorrer" del ambiente Trac y el registro real dentro del repositorio.
- **Registro de Componentes:** Lista todos los componentes de la arquitectura de la solución con sus respectivos identificadores.
- **Registro de Cambios:** Lista todos los cambios realizados a los ECS del repositorio con sus tareas o incidencias a las que responde, fecha de realización del cambio, ECS afectados, versiones afectadas y el desarrollador que lo realizó. Este registro se llevará de manera automática usando la "línea de tiempo" del ambiente Trac.
- **Registro de Incidencias:** Listado de todas las incidencias ocurridas durante el proceso de desarrollo con su tipo, fecha de levantamiento, implicado la levantó, e historial evolutivo de la incidencia. Este registro se llevará automáticamente usando el "sistema de tickets" y el "sistema de reportes" del ambiente Trac.
- **Registro de Liberaciones y Variantes:** Lista las liberaciones realizadas y las variantes creadas con su nombre, código, fecha de liberación, ECS que la componen con su versión, cambios incorporados y cambios pendientes.
- **Actas de reuniones:** Contiene todas las actas de reuniones del equipo directivo y/o del CCC con su fecha, asistentes, propósito, deficiencias discutidas y disposiciones tomadas.

La actualización y/o modificación de estos registros es responsabilidad del Administrador de la Configuración de conjunto con otros roles que necesite en cada caso. Por ejemplo el Arquitecto del proyecto para el Registro de Componentes.

## INFORMES

Se mantendrá un sistema de información constante para los desarrolladores usando el "sistema de reportes" y el "sistema de wiki" del ambiente Trac. En la página de inicio

de wiki de Trac se mantendrá un listado de las últimas noticias del proyecto. La creación y actualización de estas será responsabilidad del Administrador de la Configuración aunque cualquier miembro del equipo directivo podrá enviar nuevas noticias.

El informe más actualizado a que se tendrá acceso siempre está en el sistema de reportes de Trac, en el cual se mantendrán una serie de reportes pre-configurados cuyo resultado el ambiente Trac calculará automáticamente con los datos actuales. Algunos de estos reportes son: listado de incidencias levantadas, listado de tareas abiertas, listado de tareas del usuario actual, listado de tareas terminadas, etc.

Para solicitar la creación de un nuevo reporte pre-configurado se creará un ticket de tarea en el ambiente Trac asignado directamente al Administrador de la Configuración con la solicitud incluyendo la descripción detallada del reporte que se necesita. El Administrador de la Configuración procederá a la creación de este luego de la aprobación por el Líder del proyecto.

Además de manera semanal y con fecha de salida para el jueves de cada semana se generará un informe general que incluirá la información siguiente en el período de jueves a jueves: Estado de los cambios, Resumen de incidencias, Tareas atrasadas, Resumen de Auditoría Semanal, Otras noticias. Este informe semanal es responsabilidad del Administrador de la Configuración respaldado por el Jefe del Equipo de Aseguramiento de la Calidad y el Líder del Proyecto.

También se crearán informes bajo demanda en situaciones especiales como:

- Al incluirse un Hito no planificado.
    - Informe de estado de los cambios, de tareas pendientes, de incidencias abiertas, de estado de la planificación. Lo reciben los implicados de tipo 2 y 3.
- Al completarse un Hito no planificado.
    - Informe de estado de los cambios. Lo reciben los implicados de tipo 2 y 3.
- Al establecerse una línea base no planificada.
    - Informe de estado de los cambios y de estado de los ECS. Lo reciben los implicados de tipo 3.
- Al planificarse una visita de los clientes.

    - Informe de tareas pendientes, de incidencias abiertas. Lo reciben los implicados de tipo 3.
- Al generarse una liberación.
    - Informe de novedad y de incidencias abiertas. Lo reciben los implicados de tipo 1, 2 y 3.
    - Informe de estado de los cambios. Lo reciben los implicados de tipo 3.
- En las situaciones en las que el CCC y/o el equipo directivo del proyecto lo estime necesario.

Estos informes especiales los realizará el CCC, los incluirá en el directorio de reportes, creará una nueva noticia en el sistema de wiki del ambiente Trac y además lo enviará por correo a los implicados designados.

## AUDITORÍAS DE LA CONFIGURACIÓN

Las revisiones dentro del proyecto serán rigurosas aunque muy comunes. En general el colofón de todos los cambios será el paso por 3 revisiones a diferentes escalas. Como se pudo observar con anterioridad en el epígrafe de “Definición del proceso de cambios” de este capítulo, a todos los cambios realizados les sigue una pre-revisión técnica por parte del responsable o encargado del equipo de realización del cambio para comprobar su consistencia con respecto a otros ECS, las omisiones o posibles efectos secundarios. El gestionado de esta revisión se realizará íntegramente sobre los comentarios del sistema de tickets del ambiente de Trac para que se guarden los resultados de estas pero a la vez se agilice el proceso.

Después de ello el equipo de Aseguramiento de la Calidad realiza una revisión del conjunto de ECS modificados para asegurar que estos cumplen con las especificaciones requeridas para el cambio que se ha solicitado sobre ellos y confirmar su calidad para el paso de estos al control semi-formal. Al concluir esta revisión el encargado deberá emitir un resumen de esta que se deberá guardar entre los informes de estado del proyecto. El resumen contendrá:

- ECS revisados.
- Encargado de la revisión.
- Resultado de la revisión.
- Recomendaciones.

Para mantener el historial de cambios, los resúmenes de revisiones deberán adjuntarse al ticket de Trac que generó la revisión.

El último paso de revisiones se realizará cuando el CCC decide el establecimiento de una nueva línea base. En este momento el CCC realiza una revisión formal de los ECS que se van a incluir en esta asegurándose de que tienen cumplen con las características y la calidad necesaria para pasar conformar una línea base y pasar al control formal.

Cada semana el CCC realizará una auditoría general al proceso para determinar:

- Si se han llevando a cabo las revisiones técnicas de forma correcta y con calidad.
- Si se han seguido los procedimientos descritos para el trabajo definidos en el Plan de GCS.
- Si se han actualizado los ECS que se afectan con los cambios.

Estas auditorías semanales terminarán con un resumen de lo encontrado en ellas que se guardará con los informes de estado y se distribuirá entre los implicados de tipo 3.

Además, cada vez que los implicados de tipo 2 lo decidan se realizarán auditorías al proyecto por parte del grupo de calidad de la facultad o del grupo de calidad UCI según sea el caso. En estas auditorías se comprobará la correcta realización del proceso de control de cambios y revisiones definido. Las recomendaciones y no conformidades encontradas en la auditoría se recogerán en un informe que se guardará para la realización de actividades de acciones correctivas por cada uno de los equipos del proyecto.

## CONCLUSIONES

Han quedado definidos y descritos los procesos de Gestión de la Configuración de Software para el proyecto IMAC según los aspectos que se consideran de importancia por los diferentes estándares y autores destacados en el tema.

Las políticas definidas para el proyecto se han definido ajustándose a las características del proyecto IMAC, teniendo en cuenta los acuerdos con los clientes, las disposiciones de los organismos administrativamente superiores y las necesidades del equipo de desarrollo.

Se espera que, con la puesta en práctica de estas políticas, se logre una estabilidad en el proceso de desarrollo que tribute a aumentar la eficiencia, eficacia y calidad general de este.

# CAPÍTULO 3: RESULTADOS OBTENIDOS

## INTRODUCCIÓN

La propuesta descrita en el capítulo anterior cubre, adaptándose a las características del proyecto IMAC, todos los aspectos importantes de la Gestión de la Configuración que fueron descritos en el primer capítulo sobre el análisis de documentos de destacados autores y estándares internacionales como parte de las actividades de GCS.

Sin embargo, ¿Cómo puede asegurarse que la propuesta descrita en el segundo capítulo de esta investigación es realmente eficiente y eficaz? La respuesta a esta interrogante se encuentra en el desarrollo de este capítulo.

## ¿CÓMO VALIDAR?

Una buena manera de validar la propuesta hubiese sido un informe de resultados de la aplicación controlada de la propuesta en el proyecto productivo para el que fue diseñada. Este informe contendría una descripción del estado del proyecto antes de la aplicación de la propuesta, una tabulación de elementos surgidos de la aplicación misma de la propuesta que incluyera puntos fuertes y puntos problemáticos con el fin de mejoras futuras y al final una descripción del estado del proyecto después de aplicada la propuesta. Con ese tipo de estudio se pudiese haber validado sin lugar a dudas la propuesta descrita.

Este tipo de validación se intentó realizar en el proyecto desde la primera versión de Plan de Gestión de la Configuración propuesto para el expediente de proyecto. Sin embargo las políticas definidas en dicho plan se desestimaron sin razón descrita o aparente resultando en la no aplicación de dichas propuestas. Después de diversas conversaciones y mejoras realizadas al plan se comenzaron a aplicar algunos de los puntos establecidos en el plan, como la organización del repositorio y el establecimiento lógico de las bibliotecas de software, pero dichas políticas se aplicaron descontroladamente y por muy poco tiempo lo cual imposibilita la validación inclusive de estas pocas políticas aplicadas.

La propuesta de plan se mejoró desde su versión inicial hasta la actual usando métodos analíticos y deductivos basados en la observación de eventos ocurridos

durante la parte del ciclo de desarrollo que se ha cumplido y analizando estos bajo la idea de cómo hacer para prevenir futuros eventos negativos del mismo tipo y cómo propiciar la ocurrencia futura más frecuente de eventos positivos.

Aún así la propuesta debe ser validada de algún modo. Para ello, como consecuencia de la no existencia de datos históricos sobre los cuales trabajar, se propone la utilización del método Delphi, método de validación basado en el método de expertos y ampliamente utilizado en validación de trabajos investigativos.

## EL MÉTODO DELPHI

El método Delphi se inspira en el oráculo Delphos, de la antigua Grecia al que se acudía para hacer preguntas al Dios a través de una sacerdotisa. A pesar de ser ambiguo en las respuestas, era el que tenía mejor reputación por la certeza de sus predicciones. Fue ideado a comienzos de los 50 en el Centro de Investigación estadounidense RAND Corporation (Centro de investigación y desarrollo de las fuerzas armadas) por Olaf Helmer y Theodore J. Gordon, como un instrumento para realizar predicciones sobre un caso de catástrofe nuclear. Desde entonces, ha sido utilizado frecuentemente como sistema para obtener información sobre el futuro.

El Delphi es uno de los métodos de pronosticación más confiables, constituye un procedimiento para confeccionar un cuadro de la evolución de situaciones complejas, a través de la elaboración estadística de las opiniones de un grupo de expertos en el tema tratado. Permite rebasar el marco de las condiciones actuales más señaladas de un fenómeno y alcanzar una imagen integral y más amplia de su posible evolución, reflejando las valoraciones individuales de los expertos que pueden estar basadas en un análisis lógico, como en su experiencia intuitiva (El método Delphi, 2007)

Para la aplicación del método se siguieron 3 etapas fundamentales:

- Elección de los expertos.
- Elaboración del cuestionario de validación de la propuesta.
- Desarrollo práctico y evaluación de los resultados.

### ELECCIÓN DE LOS EXPERTOS

Se ha considerado como experto una persona capaz de ofrecer valoraciones conclusivas sobre la Gestión de la Configuración de Software y de hacer recomendaciones sobre este tema con determinado nivel de competencia.

Especialmente en cuanto a la Gestión de la Configuración asociada a proyectos productivos en el entorno de la Universidad de las Ciencias Informáticas.

La selección de los posibles candidatos se realizó bajo los criterios siguientes:

- Graduado de nivel superior.
- Vinculación al desarrollo de proyectos informáticos.
- Mínimo de un año de experiencia.
- Conocimientos sobre el proceso de Gestión de la Configuración de Software.
- Experiencia práctica en el establecimiento de políticas y/o herramientas para la GCS en proyectos.

Los posibles candidatos se buscaron en la UCI preguntando en diversos departamentos, equipos de trabajo y grupos por indicaciones sobre personal que cumpliera con los criterios establecidos para la selección. Entre los diversos departamentos donde se buscó candidatos están:

- Grupo de Gestión de la Configuración del equipo de la Dirección Técnica de Calidad UCI.
- Proyectos cuyos resultados, según la encuestas sobre GCS cuya aplicación se describe en el capítulo 1, estaban por encima de la media.
- Polos productivos que realizan tareas de organización y estandarización de sus procesos en la UCI.

La cantidad de candidatos propuestos teniendo en cuenta el total de lugares consultados es relativamente pequeña. Tan solo se pudieron identificar 7 candidatos a expertos en la materia. Esto quizá se ajuste a la situación que arrojó la encuesta descrita en el primer capítulo pudiendo establecerse un posible vínculo entre la situación negativa general de la gestión de la configuración en los proyectos de la UCI y la poca cantidad disponible de personal calificado en el tema.

A los candidatos encontrados se les aplicó la encuesta de autoevaluación que se muestra en los anexos de este trabajo, con el fin de determinar su coeficiente de competencia. Los datos de esta encuesta de autoevaluación se procesaron para decidir que candidatos integrarían el panel de expertos.

Los datos de estos resultados se encuentran en el epígrafe “Evaluación de los Resultados” más adelante en este capítulo.

## ELABORACIÓN DEL CUESTIONARIO

El cuestionario de validación aplicado a los expertos seleccionados para el panel del presente trabajo de diploma se encuentra en los anexos de esta investigación.

El mismo consta de tres preguntas enfocadas a conocer la opinión del experto para cada punto de la propuesta, así como su opinión sobre la validez del trabajo en general y las recomendaciones o sugerencias que pudiese realizar.

La pregunta número 1 consiste en una tabla en cuya primera columna se encuentran los elementos puntuales de cada parte de la propuesta realizada. Entre ellos:

- Selección de los Elementos de la Configuración del Software (ECS).
- Definición de relaciones entre los ECS.
- Esquema de nombrado.
- Definición de las Bibliotecas de Software.
- Distribución de los ECS.
- Herramientas de Gestión.
- Estrategia de Salva y Recuperación.
- Definición del Comité de Control de Cambios.
- Políticas para la Solicitud y el Control de Cambios.
- Definición de Registros.
- Descripción de Informes.
- Definición de las Auditorías de la Configuración.

En la segunda columna el encuestado responderá calificando el elemento de la propuesta que corresponde por cada fila con las siguientes categorías de acuerdo a su opinión:

- Muy Adecuado (**MA**)
- Bastante Adecuado (**BA**)
- Adecuado (**A**)
- Poco Adecuado (**PA**)
- No Adecuado (**NA**)

La pregunta número 2 consta de tres incisos en los cada no de los cuales el encuestado deberá responder marcando en casillas de “Si”, “No” o “No sé”. Los incisos a contestar preguntan si, a consideración del encuestado, los elementos de la propuesta en general son:

a) Necesarios.
b) Suficientes.
c) Óptimos.

La pregunta 3 es de tipo abierta, lo que la convierte en poco eficiente, pero es necesaria para conocer elementos que escapen a la concepción de la propuesta y del cuestionario de validación y que el experto encuestado pueda aportar.

## EVALUACIÓN DE LOS RESULTADOS

Durante la identificación de candidatos se pudieron encontrar solamente 7 personas que, por consenso general se identificaban como posibles expertos en el tema. A estos se le explicó en que consistía el proceso de validación y se le consultó para obtener su consentimiento para la participación en el panel de evaluación. De los 7, accedieron a participar en el proceso solo 6 individuos. Ellos fueron los siguientes:

- Eduardo García Hernández, Ingeniero Informático, Especialista de Gestión de la Configuración de la Dirección Técnica de la Infraestructura Productiva de la UCI.
- Marcos Ortiz Valmaseda, Instructor no graduado, Gerente del proyecto O2PMIgration.
- Angel Goñi Oramas, Ingeniero Informático Adiestrado, Líder del proyecto NOVA, Facultad 10.
- Abel Meneses Abad, Ingeniero Informático, Instructor, Asesor de Investigaciones, Scrum Master del proyecto Unicornios, Facultad 10.
- Karel Perez Ramirez, Ingeniero Informático, Jefe y Organizador del Polo de Realidad Virtual, Facultad 5.
- Osmany Valdez Puga, Ingeniero Informático, Jefe de Asignatura Programación Facultad 5, Consultor de Ontoloxia del Centro de Consultoría.

## AUTOEVALUACIÓN DEL COEFICIENTE DE COMPETENCIA

A los 6 se les aplicó la encuesta de autoevaluación que se mencionó anteriormente. A la tabla “Grado de Influencia” se le aplicó entonces los valores que propone Delphi (El método Delphi, 2007):

| FUENTES DE ARGUMENTACION | Grado de influencia de cada una de las fuentes en sus criterios. | | |
|---|---|---|---|
| | A (alto) | M (medio) | B (bajo) |
| Análisis teóricos realizados por usted | 0.3 | 0.2 | 0.1 |

| | | | |
|---|---|---|---|
| Su experiencia obtenida | 0.5 | 0.4 | 0.2 |
| Trabajos de autores nacionales | 0.05 | 0.05 | 0.05 |
| Trabajos de autores extranjeros | 0.05 | 0.05 | 0.05 |
| Su propio conocimiento del tema | 0.05 | 0.05 | 0.05 |
| Su intuición | 0.05 | 0.05 | 0.05 |

Obteniéndose los siguientes resultados de cada experto:

| | | Preg 1 | Preg 2 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **No** | **Nombre del Experto** | **CON** | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** | **Ka** | **Kc** | **K** | **Competencia** |
| 1 | Eduardo García Hernández | 7 | 0,3 | 0,5 | 0,05 | 0,05 | 0,05 | 0,05 | 1 | 0,7 | 0,85 | ALTO |
| 2 | Marcos Ortiz Valmaseda | 3 | 0,3 | 0,4 | 0,05 | 0,05 | 0,05 | 0,05 | 0,9 | 0,3 | 0,6 | MEDIO |
| 3 | Angel Goñi Oramas | 7 | 0,1 | 0,5 | 0,05 | 0,05 | 0,05 | 0,05 | 0,8 | 0,7 | 0,75 | MEDIO |
| 4 | Osmany Valdez Puga | 6 | 0,2 | 0,4 | 0,05 | 0,05 | 0,05 | 0,05 | 0,8 | 0,6 | 0,7 | MEDIO |
| 5 | Abel Meneses Abad | 7 | 0,2 | 0,5 | 0,05 | 0,05 | 0,05 | 0,05 | 0,9 | 0,7 | 0,8 | ALTO |
| 6 | Karel Perez Ramirez | 7 | 0,3 | 0,4 | 0,05 | 0,05 | 0,05 | 0,05 | 0,9 | 0,7 | 0,8 | ALTO |

El grado o coeficiente de competencia de todos los candidatos es suficiente para usarlos como expertos por lo que se pasó a la aplicación del cuestionario de validación a los 6 individuos.

## CUESTIONARIO DE VALIDACIÓN DE LA PROPUESTA

Los resultados de la aplicación del cuestionario de validación a los 6 expertos, divididos por preguntas, fueron los siguientes:

### PREGUNTA 1

Tabla de frecuencias absolutas:

| No | Elementos | MA | BA | A | PA | NA | Total |
|---|---|---|---|---|---|---|---|
| 1 | Seleccion de los ECS | 5 | 1 | 0 | 0 | 0 | 6 |
| 2 | Relaciones entre los ECS | 3 | 3 | 0 | 0 | 0 | 6 |
| 3 | Esquema de Nombrado | 5 | 1 | 0 | 0 | 0 | 6 |
| 4 | Bibliotecas de Software | 2 | 2 | 2 | 0 | 0 | 6 |
| 5 | Distribucion de los ECS | 4 | 2 | 0 | 0 | 0 | 6 |
| 6 | Herramientas de Gestion | 3 | 3 | 0 | 0 | 0 | 6 |
| 7 | Estrategia de Salva y Recuperacion | 2 | 4 | 0 | 0 | 0 | 6 |
| 8 | Definicion del CCC | 6 | 0 | 0 | 0 | 0 | 6 |
| 9 | Solicitud y Control de Cambio | 4 | 2 | 0 | 0 | 0 | 6 |
| 10 | Registros | 2 | 3 | 0 | 0 | 1 | 6 |
| 11 | Informes | 4 | 2 | 0 | 0 | 0 | 6 |
| 12 | Auditorias de la Configuracion | 4 | 1 | 1 | 0 | 0 | 6 |

**MA:** Muy adecuado, **BA:** Bastante adecuado, **A:** Adecuado, **PA:** Poco adecuado, **NA:** No adecuado.

Tabla de frecuencias absolutas acumuladas:

| No | Aspectos | MA | BA | A | PA | NA |
|---|---|---|---|---|---|---|
| 1 | Seleccion de los ECS | 5 | 6 | 6 | 6 | 6 |
| 2 | Relaciones entre los ECS | 3 | 6 | 6 | 6 | 6 |
| 3 | Esquema de Nombrado | 5 | 6 | 6 | 6 | 6 |
| 4 | Bibliotecas de Software | 2 | 4 | 6 | 6 | 6 |
| 5 | Distribucion de los ECS | 4 | 6 | 6 | 6 | 6 |
| 6 | Herramientas de Gestion | 3 | 6 | 6 | 6 | 6 |
| 7 | Estrategia de Salva y Recuperacion | 2 | 6 | 6 | 6 | 6 |
| 8 | Definicion del CCC | 6 | 6 | 6 | 6 | 6 |
| 9 | Solicitud y Control de Cambio | 4 | 6 | 6 | 6 | 6 |
| 10 | Registros | 2 | 5 | 5 | 5 | 6 |
| 11 | Informes | 4 | 6 | 6 | 6 | 6 |
| 12 | Auditorias de la Configuracion | 4 | 5 | 6 | 6 | 6 |

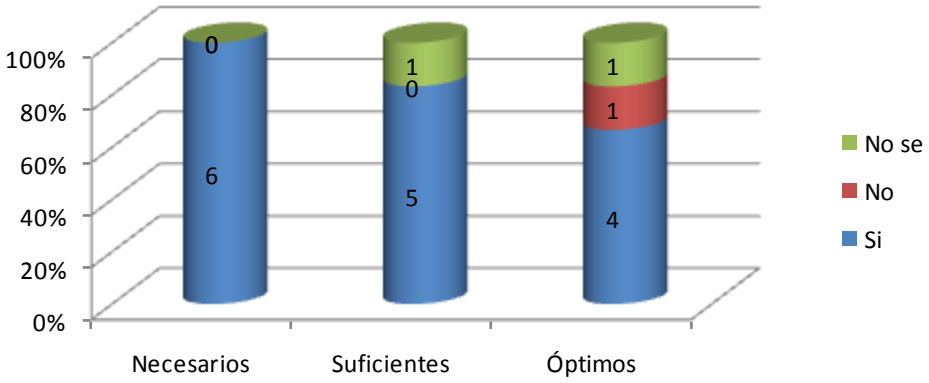
Tabla de frecuencias relativas acumuladas

| No | Aspectos | MA | BA | A | PA | NA |
|---|---|---|---|---|---|---|
| 1 | Seleccion de los ECS | 0,8333 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 2 | Relaciones entre los ECS | 0,5000 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 3 | Esquema de Nombrado | 0,8333 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 4 | Bibliotecas de Software | 0,3333 | 0,6667 | 0,9999 | 0,9999 | 0,9999 |
| 5 | Distribucion de los ECS | 0,6667 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 6 | Herramientas de Gestion | 0,5000 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 7 | Estrategia de Salva y Recuperacion | 0,3333 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 8 | Definicion del CCC | 0,9999 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 9 | Solicitud y Control de Cambio | 0,6667 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 10 | Registros | 0,3333 | 0,8333 | 0,8333 | 0,8333 | 0,9999 |
| 11 | Informes | 0,6667 | 0,9999 | 0,9999 | 0,9999 | 0,9999 |
| 12 | Auditorias de la Configuracion | 0,6667 | 0,8333 | 0,9999 | 0,9999 | 0,9999 |

Tabla de puntos de corte

| | | Total= | 12 | | | | | N = | 2,09 |
|---|---|---|---|---|---|---|---|---|---|
| No | Aspectos | MA | BA | A | PA | Suma | P | N-P | |
| 1 | Seleccion de los ECS | 0,97 | 3,72 | 3,72 | 3,72 | 12,12 | 3,03 | -0,94 | Muy adecuado |
| 2 | Relaciones entre los ECS | 0,00 | 3,72 | 3,72 | 3,72 | 11,16 | 2,79 | -0,69 | Muy adecuado |
| 3 | Esquema de Nombrado | 0,97 | 3,72 | 3,72 | 3,72 | 12,12 | 3,03 | -0,94 | Muy adecuado |
| 4 | Bibliotecas de Software | -0,43 | 0,43 | 3,72 | 3,72 | 7,44 | 1,86 | 0,24 | Muy adecuado |
| 5 | Distribucion de los ECS | 0,43 | 3,72 | 3,72 | 3,72 | 11,59 | 2,90 | -0,80 | Muy adecuado |
| 6 | Herramientas de Gestion | 0,00 | 3,72 | 3,72 | 3,72 | 11,16 | 2,79 | -0,69 | Muy adecuado |
| 7 | Estrategia de Salva y Recuperacion | -0,43 | 3,72 | 3,72 | 3,72 | 10,73 | 2,68 | -0,59 | Muy adecuado |
| 8 | Definicion del CCC | 3,72 | 3,72 | 3,72 | 3,72 | 14,88 | 3,72 | -1,62 | Muy adecuado |
| 9 | Solicitud y Control de Cambio | 0,43 | 3,72 | 3,72 | 3,72 | 11,59 | 2,90 | -0,80 | Muy adecuado |
| 10 | Registros | -0,43 | 0,97 | 0,97 | 0,97 | 2,47 | 0,62 | 1,48 | Bastante Adecuado |
| 11 | Informes | 0,43 | 3,72 | 3,72 | 3,72 | 11,59 | 2,90 | -0,80 | Muy adecuado |
| 12 | Auditorias de la Configuracion | 0,43 | 0,97 | 3,72 | 3,72 | 8,84 | 2,21 | -0,12 | Muy adecuado |
| Suma | | 6,09 | 35,84 | 41,88 | 41,88 | 125,68 | | | |
| P.de corte | | 0,51 | 2,99 | 3,49 | 3,49 | | | | |

## PREGUNTA 2




## PREGUNTA 3

De los 6 expertos encuestados 3 respondieron que la propuesta les parecía correcta y que no tenían nada que modificar, eliminar o agregar pues la propuesta les parecía bastante completa.

Dos expertos sin embargo sugirieron el uso de una herramienta de control de versiones con enfoque distribuido, específicamente propusieron el uso de Git o Bazaar pues según argumentan, estas presentan un enfoque más acorde con las políticas de software libre que defiende la universidad.

La sugerencia de la herramienta distribuida, aunque es válida, agrega un conjunto de complicaciones en cuanto a la productividad pues, Git por ejemplo, posee muy pobres interfaces gráficas y un concepto difícil de manipular por un equipo de desarrollo relativamente novel en el desarrollo de proyectos y en los temas de gestión de proyecto y gestión de la configuración.

Un experto argumentó que la estrategia de salva y recuperación, si bien estaba muy bien definida y funcional, era también un poco primitiva. Sugería el estudio y uso de una herramienta de salvas automáticas como Bácula.

Esta sugerencia es muy válida y deberá ser estudiada con cuidado para su posible aplicación.

## CONCLUSIONES

La propuesta de Plan de Gestión de la Configuración ha sido evaluada por un panel de 6 expertos que representan aproximadamente el 85,7% de la población de posibles expertos en gestión de la configuración encontrados en la universidad y seleccionados con un nivel de competencia de medio a alto.

Según las estadísticas de las respuestas de los cuestionarios de los expertos, la propuesta es bastante adecuada.

Hubo una recomendación del panel de expertos con respecto a la estrategia de salva y recuperación que debe ser evaluada y estudiada para su aplicación.

# CONCLUSIONES GENERALES

Al finalizar este trabajo se consideran cumplidos los objetivos trazados, lo cual puede apoyarse en el arribo a las siguientes conclusiones:

- Se identificaron los ECS que forman parte del proyecto y se establecieron las relaciones entre ellos.
- Se establecieron bibliotecas de software para el proyecto, así como políticas para su explotación.
- Se definió una estructura organizativa para los ECS en las bibliotecas de software.
- Se definió el proceso de control de cambios y solicitud de cambios de la configuración con sus actividades, artefactos, métodos e implicados.
- Se definieron las políticas de revisión y las actividades de seguimiento de defectos y cambios.

Además vale mencionar que, de manera general:

- Se desarrolló íntegramente el modelo que permite establecer la gestión de la configuración de software para el proyecto productivo IMAC siguiendo modelos y estándares internacionales.
- La propuesta fue validada por un panel de expertos por lo que debe contribuir a organizar, gestionar y controlar la información que se genera durante el proceso de desarrollo del proyecto IMAC.

# RECOMENDACIONES

Para completar este trabajo se proponen las siguientes recomendaciones:

- Aplicar la propuesta al proyecto productivo IMAC para mejorar los resultados obtenidos al menos durante la última etapa del desarrollo.
- Realizar estudios para lograr una integración completa de las tareas de los roles del proyecto sobre las herramientas de gestión de proyecto propuestas.
- Continuar perfeccionando la propuesta actual para elevar su eficiencia y eficacia.
- Publicar las investigaciones realizadas para fomentar el conocimiento general sobre la gestión de la configuración de software.

# BIBLIOGRAFÍA

Alfresco Software. (2009). *Open Source Enterprise Content Management System (CMS) by Alfresco*. Recuperado el 08 de Junio de 2009, de http://www.alfresco.com/index-a1.html

Álvarez Chang, R. N., & Cortina Blanco, Y. (2006). *Procedimiento para el análisis de factibilidad de cambios en el proceso de desarrollo de software.* UCI, La Habana, Cuba.

Álvarez, V. I. (1997). *Proceso general de la investigación.* Santa Clara, Cuba: Departamento Sicología, Facultad de Ciencias Sociales, Universidad Central de las Villas.

Antonio, A. d. (2001). *La gestion de la Configuracion del Software.*

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

Bach, J. (1998). *The Highs and Lows of Change Control.*

Bersoff, E. (1980). *Software Configuration Management.*

Brown, N. (1998). *Little Book of Configuration Management.*

Burrows, C. (1999). *Configuration Management, coming of age in the year 2000.* Crosstalk Magazine.

CNTI. (s.f.). *MeRinde*. Recuperado el 12 de Mayo de 2009, de Gobierno Bolivariano de Venezuela: http://merinde.rinde.gob.ve/

Dribin, D. (28 de Diciembre de 2007). *Choosing a Distributed Version Control System*. Recuperado el 2009 de Mayo de 11, de Dave Dribin's Blog: http://www.dribin.org/dave/blog/archives/2007/12/28/dvcs/

Dribin, D. (30 de Diciembre de 2007). *Why I Chose Mercurial*. Recuperado el 2009 de Mayo de 11, de Dave Dribin's Blog: http://www.dribin.org/dave/blog/archives/2007/12/30/why_mercurial/

Edgewall Software. (s.f.). *The Trac Project*. Recuperado el 08 de Mayo de 2009, de http://trac.edgewall.org/

EETimes. (17 de Mayo de 2007). *Rapid Subversion Adoption Validates Enterprise Readiness and Challenges Traditional Software Configuration Management Leaders*. Recuperado el 2 de Diciembre de 2008, de http://www.eetimes.com/press_releases/bizwire/showPressRelease.jhtml?articleID=608063&CompanyId=2

*El método Delphi*. (2007). Recuperado el 25 de 05 de 2007, de http://www.gtic.ssr.upm.es/encuestas/delphi.htm

Ema, E. (2002). *La Evolución desde la Gestión Convencional hasta la Gestión Moderna del Software (CMM y CMMI).*

Equipo IMAC. (2009). *Expediente de Proyecto.* UCI, La Habana, Cuba.

Equipo MeRinde. (s.f.). *Gestionar los Cambios*. Recuperado el 12 de Mayo de 2009, de MeRinde: http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=247&Itemid=249

Equipo MeRinde. (s.f.). *Repositorio de Versiones*. Recuperado el 12 de Mayo de 2009, de MeRinde: http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=118&Itemid=300

Febles, A. (2004). *MConfig.PM, Modelo de referencia para la Gestión de Configuración en la pequeña y mediana empresa de software.* La Habana, Cuba: CUJAE.

Febles, A. (2002). *Memorias de la Convención Informática.* La Habana, Cuba: CUJAE.

García, L. (2001). *Gestión de Configuración de software (Programa Curricular).* Villa Clara, Cuba: UCLV.

Gervás, P. (2002). *La gestion de la Configuracion del Software.* UCM.

Humphrey, W. (1989). *Administrando el Proceso de Software.*

IBM. (n.d.). *Teologic Synergy: Highlights*. Retrieved Enero 10, 2009, from Teologic is now IBM: http://www.telelogic.com/products/synergy/highlights.cfm

IEEE. (1990).

IEEE 730. (1998). *IEEE Standard for Software Quality Assurance Plans Std.*

ISO 10007. (15 de Abril de 1995). *ISO Guidelines for configuration management: Quality management* .

Jacobson, I., Griss, M., & Jonson, P. (2000). *Software Reuse: Architecture, Process, and Organizartions for Busisness Success.*

MaNGOS Community. (s.f.). *Announcements & News*. Recuperado el 12 de Mayo de 2009, de MaNGOS Community Forums: http://getmangos.com/community/forumdisplay.php?f=13

MaNGOS Community. (02 de Marzo de 2009). *GetMangosWiki*. Recuperado el 12 de Mayo de 2009, de http://getmangos.com/wiki/

MaNGOS Community. (s.f.). *MaNGOS Community Forum*. Recuperado el 11 de Mayo de 2009, de MaNGOS: The open source MMORPG server: http://getmangos.com/community/index.php

NASA. (1995). *Software Configuration Management Guidebook.* Software Assurance Technology Center.

OForge. (03 de Febrero de 2009). *Optaros OForge*. Recuperado el 11 de Mayo de 2009, de http://code.optaros.com/trac/oforge/wiki

Paulk, M. C. (1993). *Capability Maturity Model for Software, Version 1.1.* Carnegie Mellon University.

Pressman, R. (2005). Ingeniería del Software. Un enfoque práctico.

Pressman, R. S. (2002). *Ingeniería del Software. Un enfoque práctico.*

Rational Unified Proccess. (2003).

*Redmine*. (s.f.). Recuperado el 2009 de Mayo de 08, de http://www.redmine.org/

Software Engineering Institute. (s.f.). *CMMI Web Site*. Recuperado el 23 de febrero de 2009, de http://www.sei.cmu.edu/cmmi/

SourceForge, Inc. (s.f.). *dotProject en Sourceforge.net*. Recuperado el 08 de Mayo de 2009, de http://sourceforge.net/projects/dotproject/

SWEBOK. (2004). *Guide to the Software Engineering Body of Knowledge.* California, EUA.

The GanttProject Team. (s.f.). *GanttProject*. Recuperado el 08 de Mayo de 2009, de http://ganttproject.biz/

The TracHacks Community. (s.f.). *TracHacks: Plugins, macros, etc*. Recuperado el 2009 de Mayo de 08, de http://trac-hacks.org/

WorldPress. (2009). *MARKMCB*. Recuperado el 06 de Mayo de 2009, de 3 Reasons to Switch to Git from Subversion: http://markmcb.com/2008/10/18/3-reasons-to-switch-to-git-from-subversion/

# ANEXOS

## ANEXO 1: PASOS PARA LA INSTALACIÓN Y CONFIGURACIÓN DE SUBVERSION SOBRE HTTP AUTENTICANDO CONTRA DOMINIO UCI EN UBUNTU 9.04

Se asume que se tiene configurado con anterioridad un repositorio de Ubuntu 9.04 actualizado y que ya se ha instalado Apache2. Los comandos y textos entre comillas, en cursiva y negrita deberán ser escritos tal y como se exponen, con todos sus espacios y sin las comillas.

1- Instalación de los componentes necesarios. Abra una consola y escriba: *"**sudo apt-get install subversion subversion-tools libapache2-svn**"*.

2- Habilitar los módulos requeridos de Apache2. Escriba en la consola: "***a2enmod authnz_ldap***". Deberá recibir un mensaje anunciando que los módulos "***ldap***" y "***authnz_ldap***" se han activado.

3- Crear el directorio para el repositorio. Escriba en la consola: *"**mkdir /RUTA/DIRECTORIO**"*. Al escribir el comando reemplace la palabra RUTA por la ruta desde el directorio raíz al directorio donde quiere crear físicamente el repositorio y reemplace la palabra DIRECTORIO por el nombre que quiere asignarle a su repositorio. Recomendamos para mayor facilidad que el nombre escogido para el directorio esté escrito en minúscula y no contenga espacios. Debe asegurarse de tener derechos de escritura en la localización escogida.

4- Crear el repositorio. Escriba en la consola: *"**svnadmin create /RUTA/DIRECTORIO**"* al escribir el comando sustituya la palabra RUTA y la palabra DIRECTORIO por los datos que utilizó en el paso anterior.

5- Permitir a Apache2 manipular el repositorio. Escriba en la consola: *"**chown -R www-data:www-data /RUTA/**"*. Puede que requiera permisos de super-usuario para realizar esta operación.

6- Crear el archivo de permisos. En un directorio donde Apache2 pueda leer cree un archivo de texto y edítelo usando un editor de textos (como "nano", "vim", "mcedit" o "gedit"). Dentro de este archivo escriba el texto siguiente respetando mayúsculas, minúsculas, espacios y saltos de línea:

```
[groups]
GRUPO1 = USUARIO1,USUARIO2,…
GRUPO2 = USUARIO3,USUARIO4,…
GRUPO3 = USUARIO5,USUARIO6,…

[/]
@GRUPO1 = rw
USUARIO7 = r

[DIRECTORIO:/]
@GRUPO2 = rw

[DIRECTORIO:/RUTA_L]
@GRUPO3 = rw
USUARIO8 = w
```

Reemplace la palabra DIRECTORIO por el mismo dato que utilizó en el paso 4. Reemplace las palabras GRUPO# por los nombres que quiere ponerle a los grupos de usuarios a crear, sin espacios ni caracteres especiales. Recomendamos que use los nombres de los roles de su metodología. Reemplace las palabras USUARIO# por los nombres de usuario que pertenecerán a esos grupos y que requieren algún tipo de acceso al repositorio. Reemplace la palabra RUTA_L por la ruta exacta dentro del repositorio del directorio al que quiere asignar permisos. Observe que cuando se asignan permisos “r” significa lectura, “w” significa escritura y “rw” significa lectura y escritura. Observe también que [/] indica la raíz de todos los repositorios, o sea indica el dato RUTA del paso 4. Observe además que los permisos se heredan de directorios padres a directorios hijos si no se especifican permisos específicos en los hijos.

7- Salve los cambios en el archivo y cierre el editor de texto.

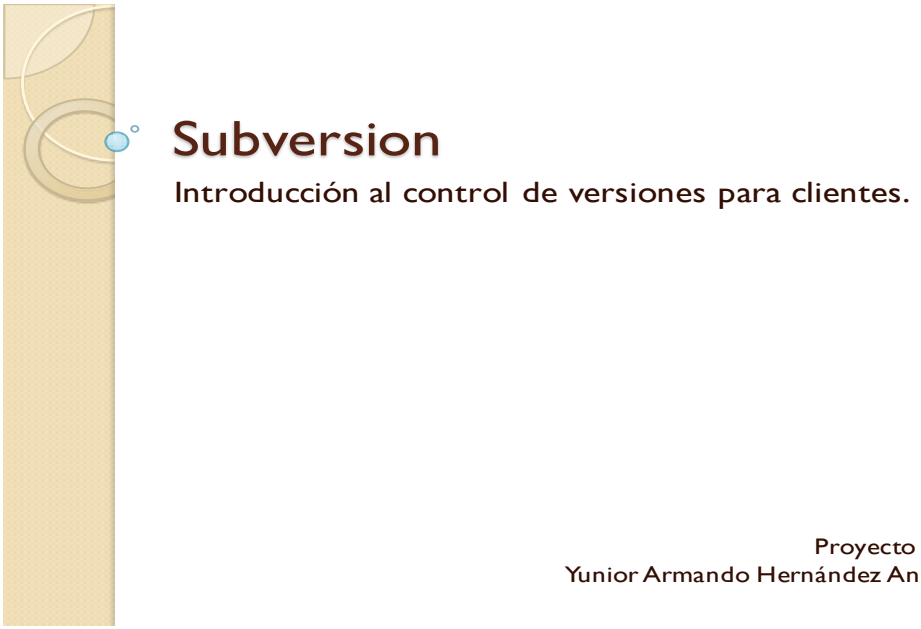
8- Configurar acceso desde Apache2 usando WebDav. Con un editor de texto abra el archivo ***“/etc/apache2/mods-avaliable/dav_svn.conf”***. Al final del archivo agregue el siguiente texto respetando mayúsculas, minúsculas, espacios y saltos de línea:

<Location /**LINK**>
DAV svn
SVNParentPath "**/RUTA/**/"
SVNPathAuthz on
AuthType Basic
AuthBasicProvider ldap
AuthzLDAPAuthoritative off
AuthName "**TEXTO**"
AuthLDAPURL "ldap://10.0.0.3:389/OU=UCI Domain Users, DC=uci, DC=cu,?sAMAccountName?sub?(objectClass=*)"
AuthLDAPBindDN "CN=ad search,OU=Systems, OU=UCI Domain Impersonals, DC=uci, DC=cu"
AuthLDAPBindPassword uF2SODWAHiW0eJboFFQEAvVzJ
AuthSVNAccessFile /**RUTA_P**/**ARCHIVO**
Require valid-user
<LimitExcept GET PROPFIND OPTIONS REPORT>
Require valid-user
</LimitExcept>
</Location>

Reemplace la palabra LINK por un nombre significativo para el acceso HTTP a la raíz de repositorios. Esto significa que para acceder al repositorio que Ud. creó en el paso 4, llamado DIRECTORIO, la URL a entrar sería http://DIRECCION_IP/LINK/DIRECTORIO/. Reemplace la palabra RUTA por el mismo dato asignado en el paso 4. Reemplace la palabra RUTA_P por la ruta desde el directorio raíz hasta el directorio donde guardó el archivo de permisos creado en el paso 6 y reemplace la palabra ARCHIVO por el nombre completo de ese mismo archivo. Reemplace la palabra TEXTO por un texto de autenticación. Por ejemplo: "*Bienvenido al Repositorio X. Autentíquese con su usuario de dominio UCI*".

9- Salve los cambios en el archivo y cierre el editor de texto.
10- Reiniciar Apache2. Escriba en la consola: "sudo /etc/init.d/apache2 restart". Observe que requiere permisos de super-usuario para realizar esta operación.
11- Para probar el funcionamiento correcto del repositorio acceda vía HTTP a este usando la URL http://DIRECCION_IP/LINK/DIRECTORIO.

## ANEXO 2: PRESENTACIÓN INSTRUCTIVA SOBRE EL USO DE CLIENTES DE SUBVERSION




## ANEXO 3: CONFIGURACIÓN DE UN GANCHO PRE SUBIDA PARA PREVENIR CAMBIOS SIN COMENTARIO EN SUBVERSION

Se asume que posee un repositorio de Subversion configurado siguiendo los pasos definidos en el anexo 1.

1- Antes de realizar modificaciones haga una copia de seguridad del archivo ***"/RUTA/hooks/pre-commit.tmpl"*** donde RUTA es la ruta completa desde la raíz hasta el directorio del repositorio de Subversion.

2- Con un editor de texto (como "nano", "vim", "mcedit" o "gedit") abra el archivo ***"/RUTA/hooks/pre-commit.tmpl"***, sustituya su contenido por el siguiente texto:

```
#!/usr/bin/python
import sys, os, string
SVNLOOK='/usr/bin/svnlook'
def main(repos, txn):
    log_cmd = '%s log -t "%s" "%s"' % (SVNLOOK, txn, repos)
    log_msg = os.popen(log_cmd, 'r').readline().rstrip('\n')
    if len(log_msg) < 10:
        sys.stderr.write ("Por favor, inserte un comentario no menor a 10 caracteres describiendo sus cambios.\n")
        sys.exit(1)
    else:
        sys.exit(0)
if __name__ == '__main__':
    if len(sys.argv) < 3:
        sys.stderr.write("Usage: %s REPOS TXN\n" % (sys.argv[0]))
    else:
        main(sys.argv[1], sys.argv[2])
```

3- Salve los cambios en el archivo y cierre el editor.

4- Renombre el archivo *"**/RUTA/hooks/pre-commit.tmpl**"* que editó en el paso 2 eliminándole la extensión. Quedará: *"**/RUTA/hooks/pre-commit**"*.
5- Permitir la ejecución del archivo. En la consola escriba: *"**chmod 777 /RUTA/hooks/pre-commit**"*.

## ANEXO 4: PASOS PARA LA INSTALACIÓN Y CONFIGURACIÓN DE TRAC SOBRE HTTP CON POSTGRESQL, SUBVERSION Y AUTENTICANDO CONTRA DOMINIO UCI

Se asume que se ha ejecutado la instalación de Subversion tal y como se explica en el primer anexo, que se ha instalado, configurado y se encuentra funcionando un gestor PostgreSQL.

1- Instalación de los componentes necesarios. Escriba en la línea de comandos: *"**sudo apt-get install trac python-psycopg2 python-genshi libapache2-mod-python**"*. Observe que requiere permisos de super-usuario para realizar esta operación. Con esto instalará los componentes de necesarios para la ejecutar Trac en la versión estable de estos que contiene el repositorio de Ubuntu 9.04. Si quiere instalar la versión más reciente de Trac puede descargarlo en formato comprimido TAR.GZ desde http://trac.edgewall.org/ así como la última versión del motor de estilos genshi. Debe tener en cuenta que cada versión de Trac requiere una versión mínima de Python. Por ejemplo en el momento en que se escribe la última versión de Trac disponible es la 0.11.4 que requiere Python 2.5 o superior. Normalmente no deben existir problemas con esta dependencia pues, con un repositorio de Ubuntu bien configurado, Python se actualiza automáticamente. Para instalar Trac usando el comprimido TAR.GZ:
    a. Instalar las otras dependencias desde el repositorio. Escriba en la consola: *"**sudo apt-get install python-psycopg2 libapache2-mod-python**"* y alternativamente incluya *"**python-genshi**"* si no piensa instalar genshi desde un comprimido también.
    b. Descomprimir el TAR.GZ en cualquier directorio donde pueda escribir
    c. Instalar usando Python. Acceda desde la consola al directorio donde descomprimió los datos del TAR.GZ. Escriba en la consola: *"**python setup.py install**"*.
2- Activar el módulo de Python para Apache2. Escriba en la consola: *"**a2enmod python**"*. Puede requerir permisos de super-usuario para ello.

3- Crear el directorio de ambientes de Trac. Escriba en la consola: *"**mkdir /RUTA/DIRECTORIO**"*. Al escribir el comando reemplace la palabra RUTA por la ruta desde el directorio raíz al directorio donde quiere crear físicamente el directorio de ambientes de Trac y reemplace la palabra DIRECTORIO por el nombre que quiere asignarle a su directorio de ambientes. Recomendamos para mayor facilidad que el nombre escogido para el directorio esté escrito en minúscula y no contenga espacios. Debe asegurarse de tener derechos de escritura en la localización escogida.

4- Preparar la base de datos de Trac. Mediante un cliente de PostgreSQL cualquiera (como "pgsql" o "pgAdmin3"), cree un rol de login para Trac (un usuario) con derecho a conectar al gestor de BD. Asegúrese de asignar una contraseña o clave para el rol de login. Cree luego una base de datos para Trac y asígnele como dueño al rol de login que creó anteriormente. Recomendamos que use nombres en minúscula y sin caracteres especiales para el usuario y para la base de datos. Revise que el usuario que creó tiene acceso a dicha base de datos.

5- Crear el ambiente de Trac. Acceda por la consola al directorio que creó para los ambientes de Trac y escriba: *"**trac-admin /RUTA/DIRECTORIO/ initenv**"* al escribir el comando sustituya la palabra RUTA por la ruta desde el directorio raíz al directorio que creó para los ambientes de Trac y sustituya la palabra DIRECTORIO por un nombre significativo para su Trac. Recomendamos que el nombre significativo se escriba en minúsculas y sin caracteres especiales o espacios. Debe asegurarse de tener derechos de escritura en la localización escogida.

6- Ofrecer los datos. Al realizar el paso anterior se ejecutará un asistente que lo guiará a través del proceso. Dicho asistente le pide datos como:

   a. Nombre del Proyecto: Escriba una cadena de nombrado tal y como desea que se muestre en el título de su Trac, sin restricciones de formato.

   b. Cadena de conexión a la Base de Datos: Aquí escriba el texto siguiente: *"**postgres://USUARIO:CLAVE@IPBD:PUERTO/BD**"* al escribir el comando sustituya las palabras USUARIO por el nombre del rol de login que creó en el paso 3, CLAVE por la contraseña que le asignó a dicho rol de login, IPBD por la dirección IP del equipo que ejecuta el gestor PostgreSQL o escriba simplemente *"**localhost**"* si se encuentra en la misma PC donde esta instalando Trac (recomendado), PUERTO por el

puerto por el que escucha el gestor PostgreSQL o simplemente elimine la parte de ***":PUERTO"*** si PostgreSQL está escuchando por el puerto predeterminado y finalmente sustituya BD por el nombre de la base de datos que creó para Trac en el paso 3.

c. Tipo de repositorio. Escriba ***"svn"*** o déjelo en blanco para usar Subversion.

d. Localización física del repositorio. Escriba la ruta física completa de la localización del repositorio de Subversion del proyecto para el cual se instala este ambiente Trac.

7- Permitir a Apache2 acceder al ambiente Trac. Escriba en la consola: ***"chown -R www-data:www-data /RUTA/"***.

8- Crear directorio para caché. Acceda mediante la consola a un directorio que sea accesible por Apache2 y escriba: ***"mkdir CACHE"*** sustituyendo la palabra CACHE por un nombre significativo para el caché de Trac.

9- Permitir a Apache2 escribir en el directorio de caché. Escriba en la consola: ***"chown -R www-data:www-data /RUTA/CACHE"*** sustituyendo RUTA por la ruta completa desde la raíz hasta el directorio de caché y CACHE por el nombre de directorio que creó para el caché de Trac en el paso 8.

10- Configurar el acceso a través de Apache2. Con un editor de texto (como "nano", "vim", "mcedit" o "gedit") abra el archivo ***"/etc/apache2/httpd.conf"***. Necesita derechos de super-usuario para ello. Al final del archivo agregue el siguiente texto respetando mayúsculas, minúsculas, espacios y saltos de línea:

```
<Location "/LINKTRAC/">
   SetHandler mod_python
   PythonInterpreter main_interpreter
   PythonHandler trac.web.modpython_frontend
   PythonOption TracEnvParentDir /RUTA/
   PythonOption TracUriRoot /LINKTRAC/
   PythonOption TracEggCache /CACHE/
   Order deny,allow
   Allow from all
</Location>
<LocationMatch "LINKTRAC/[^/]+/login">
   AuthType Basic
   AuthBasicProvider ldap
   AuthName "TEXTO"
   AuthLDAPURL "ldap://10.0.0.3:389/OU=UCI Domain Users, DC=uci,
DC=cu,?sAMAccountName?sub?(objectClass=*)"
   AuthLDAPBindDN "CN=ad search,OU=Systems, OU=UCI Domain
Impersonals, DC=uci, DC=cu"
   AuthLDAPBindPassword uF2SODWAHiW0eJboFFQEAvVzJ
   require valid-user
</LocationMatch>
```

Sustituya las palabras LINKTRAC por un nombre significativo para acceder vía HTTP a la raíz de ambientes Trac. Esto significa que para acceder vía HTTP al ambiente Trac que Ud. creó en el paso 5 llamado DIRECTORIO deberá usar la URL http://DIRECCION_IP/LINKTRAC/DIRECTORIO. Reemplace la palabra RUTA por la ruta completa desde la raíz hasta el directorio de ambientes Trac creado en el paso 3, CACHE por la ruta completa hasta el directorio de caché de Trac que creó en el paso 8 y TEXTO por un texto de autenticación. Por ejemplo: "*Bienvenido al Trac del proyecto X. Autentíquese con su usuario de dominio UCI*"

11- Pruebe el funcionamiento de su ambiente Trac accediendo vía HTTP con cualquier navegador (Mozilla Firefox o Internet Explorer) a la URL http://DIRECCION_IP/LINKTRAC/DIRECTORIO.

12- Modificar los permisos de seguridad de Trac por primera vez. Desde la consola escriba: *"**trac-admin /RUTA/AMBIENTE**"*, sustituya la palabra RUTA por la ruta completa desde la raíz hasta el directorio de ambientes de Trac y AMBIENTE por el nombre del directorio específico del ambiente Trac que creó en el paso 5. Se ejecutará un intérprete de comandos de Trac. Escriba *"**permission add USUARIO TRAC_ADMIN**"*, sustituya la palabra USUARIO por un nombre de usuario de dominio UCI (en minúsculas) que Ud. quiera que posea todos los derechos sobre este ambiente Trac. Salga del intérprete de Trac escribiendo *"**exit**"*. Ahora podrá usar el usuario de dominio que puso anteriormente para administrar el ambiente Trac desde el panel de administración de Trac en la web.

## ANEXO 5: GANCHO PARA INTEGRAR EL ESTADO DE TRAC CON LOS COMMITS DE SUBVERSION

Este gancho se configura en el archivo post-commit del directorio hooks de su repositorio de la misma manera que el proceso descrito en el anexo 3.

El texto del gancho está en el archivo adjunto a continuación:

trac-post-commit-hook

## ANEXO 6: ENCUESTA DE AUTOEVALUACIÓN PARA DETERMINAR EL COEFICIENTE DE COMPETENCIA

Nombre y Apellidos: _____

Centro de Trabajo: _____

Departamento: _____

Labor que realiza: _____

Años de Experiencia: _____

Categoría docente: _____ Categoría Científica: _____

1- Indique el grado de conocimiento que posee en materia de Gestión de la Configuración en una escala de 1 a 10 proporcional el grado de conocimiento al valor numérico seleccionado:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

2- Marque con una cruz (X) las fuentes que le han servido para argumentar el conocimiento que tiene Ud. de Gestión de la Configuración. Marque con dos cruces (XX) la que más ha influido.

| **Fuentes de Argumentación** | **Grado de Influencia** | | |
|---|---|---|---|
| | **Alto** | **Medio** | **Bajo** |
| Análisis realizado por Ud. | | | |
| Experiencia | | | |
| Trabajos de autores nacionales. | | | |
| Trabajos de autores extranjeros | | | |
| Su propio conocimiento del tema. | | | |
| Su intuición. | | | |

## ANEXO 7: CUESTIONARIO DE VALIDACIÓN APLICADO AL PANEL DE EXPERTOS

Compañero(a):

En la ejecución del presente trabajo de diploma, deseamos someter a la valoración de un grupo de expertos, la propuesta de Plan de Gestión de la Configuración establecido para el proyecto productivo Informatización del Ministerio de Auditoría y Control (IMAC).

Para ello necesitamos primeramente conocer el grado de dominio que Ud. posee sobre Gestión de la Configuración de Software y con este fin deseamos que se sirva de responder la encuesta “Grado_de_Competencia.xlsx” que se adjunta y a

continuación proceder a responder las preguntas que se encuentran más abajo en este documento.

La propuesta del presente trabajo de diploma, creada a través del estudio del estado del arte, las tendencias actuales, la situación de la Gestión de la Configuración de Software (GCS) en la Universidad de las Ciencias Informáticas (UCI) y las características específicas del proyecto, incluye:

- Selección de los Elementos de la Configuración del Software (ECS).
- Definición de relaciones entre los ECS.
- Esquema de nombrado.
- Definición de las Bibliotecas de Software.
- Distribución de los ECS.
- Herramientas de Gestión.
- Definición del Comité de Control de Cambios.
- Políticas para la Solicitud y el Control de Cambios.
- Definición de Registros.
- Descripción de Informes.
- Definición de las Auditorías de la Configuración.

Para responder a las siguientes preguntas Ud. deberá haber leído el segundo capítulo de la presente tesis.

1. Según los detalles de la propuesta valore el grado de factibilidad de cada elemento proporcionado de acuerdo con la siguiente escala: Muy Adecuado (**MA**), Bastante Adecuado (**BA**), Adecuado (**A**), Poco Adecuado (**PA**), No Adecuado (**NA**).

| Nº | Elemento | Valoración |
|---|---|---|
| 1 | Selección de los Elementos de la Configuración del Software (ECS). | |
| 2 | Definición de relaciones entre los ECS. | |
| 3 | Esquema de nombrado. | |
| 4 | Definición de las Bibliotecas de Software. | |
| 5 | Distribución de los ECS. | |
| 6 | Herramientas de Gestión. | |
| | Estrategia de Salva y Recuperación. | |

| | | |
|---|---|---|
| 7 | Definición del Comité de Control de Cambios. | |
| 8 | Políticas para la Solicitud y el Control de Cambios. | |
| 9 | Definición de Registros. | |
| 10 | Descripción de Informes. | |
| 11 | Definición de las Auditorías de la Configuración. | |

2. Determine si los elementos descritos son:

Necesarios: Si ____, No ____, No sé ____

Suficientes: Si ____, No ____, No sé ____

Óptimos: Si ____, No ____, No sé ____

3. Exprese otros criterios, observaciones o recomendaciones que pudieran servir para perfeccionar la propuesta realizada en el trabajo de diploma. (¿Qué modificar? ¿Qué agregar? ¿Qué eliminar? ¿Está correcta completamente? Etc.)

# Glosario de Términos

**Alfresco Labs:** Herramienta de gestión documental desarrollada por Alfresco Software. (Alfresco Software, 2009)

**BSD:** Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Es una licencia de software libre permisiva. Tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

**CASE:** Computer Aided Software Engineering. (Ingeniería de Software Ayudada por Computadora) Es una clasificación de herramientas dedicadas a los diferentes procesos de la ingeniería de software. Entre ellas: Visual Paradigm, Rational Rose, Planner, Microsoft Project, etc.

**DT:** Dirección Técnica, se refiere generalmente a la dirección técnica de la IP.

**ECS:** Elemento de la Configuración del Software.

**Framework:** Del inglés, marco de trabajo. Conjunto de herramientas reutilizables por separado o en conjunto con el objetivo de crear nuevas herramientas.

**GPL:** Licencia Pública General de GNU, es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

**HTTP:** Protocolo de Transferencia de Hipertexto. Es el protocolo más común de para la transferencia web.

**iCal:** Uno de los formatos estándar de calendarios electrónicos.

**IDE:** Integrated Development Environment, (Ambiente Integrado de Desarrollo)

**IMAC:** Proyecto de Informatización del Ministerio de Auditoría y Control.

**IP:** Infraestructura Productiva, nombre costo con el que se conoce a la infraestructura que soporta la producción en la UCI.

**MAC:** Ministerio de Auditoría y Control

**Middleware:** Del inglés, software intermedio. Es una clasificación dada a un conjunto de librerías que funcionan como “intermediarios” reutilizables para facilitar trabajos engorrosos.

**Plug-in:** Del inglés: “complemento”, es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica.

**RSS:** Es una familia de formatos de fuentes web codificados en XML. Se utiliza para suministrar a suscriptores de información actualizada frecuentemente. El formato permite distribuir contenido sin necesidad de un navegador, utilizando un software diseñado para leer estos contenidos RSS.

**SIGAC:** Sistema Informático de Gestión de Auditoría y Control la solución propuesta como parte del proyecto IMAC para gestionar la información del flujo de trabajo del MAC.

**SCV:** Sistema de Control de Versiones, también conocidos como Gestores de Código Fuente (GCF, SCM en inglés). Herramientas dedicadas a la gestión de cambios sobre archivos.

**UCI:** Universidad de las Ciencias Informáticas

**WebDav:** Es un protocolo, o mas precisamente una extensión de un protocolo que proporciona funcionalidades para crear, cambiar y mover documentos en un servidor remoto.