Universidad de las Ciencias Informáticas.

Facultad 3.

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

Título:

La Gestión de Configuración y Cambio en el proyecto productivo Cálculo de Índices de Precios al Consumidor.

Autores: **Maykel Medrano Najara.**

Yanisleidy Barroso Benítez.

Tutor: **Ing. Rolando Pérez Pinto.**

Ciudad de la Habana, Junio de 2007

DECLARACIÓN DE AUTORÍA

Declaramos que somos únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso del mismo de la manera que estimen conveniente.

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

_____ _____

Firma del primer autor Firma del segundo autor

_____

Firma del tutor.

RESUMEN

La Oficina Nacional de Estadísticas (ONE) es la institución que se dedica en el país a llevar un registro de todas las estadísticas que son de interés para la planificación y control de la economía, los servicios, etc. Actualmente uno de los servicios que brinda la ONE al país es el Cálculo de Índices de Precios, el cual puede estar dirigido al consumidor, al comercio interior e industrial.

La ONE cuenta actualmente con un sistema integral para el cálculo del índice de precios al consumidor, pero el mismo ha quedado obsoleto. El proyecto IPC, surge con el propósito de desarrollar un sistema más flexible, que cumpla con las exigencias actuales.

El objetivo de este trabajo es el desarrollo y aplicación de la Gestión de Configuración para mantener la integridad del software, con el fin de evitar que en el proyecto IPC surjan problemas de gestión y control del proceso de desarrollo del software debido a la no utilización de la Gestión de Configuración. Para ello se ha hecho un estudio de las principales actividades, estrategias y procedimientos que constituyen esta disciplina con el objetivo de ver cuales se ajustan al proyecto.

ABSTRACT

The Statistical National Office (SNO) is the institution in the country that is devoted to carry out an updated registering with the records of all important statistics used in the planning and control of the economy and services. Nowadays one of the services brought by (SNO) to the country is the calculus of the prices index, which could be addressed to the customer, to the internal and industrial commerce.

At present (SNO) counts on a wholesome system for calculating the price index of the customer but this system has become old fashion. The project IPC has arisen with the purpose of developing a much more flexible system which could fulfill the present challenges on the matter.

The main objective of this paper is the development and application of the Configuration Management (CMU) so as to keep the software integrity that, at long last, will avoid the coming up of problems concerning the management and control of the process of the software development due to the under use of CM. The author of this paper carried out a study of the main activities, strategies and procedures that make up this discipline, aiming at testing which of them best suit the project.

DEDICATORIA:

Yanisleidy:

A mis padres porque todo lo que soy se los debo, por el amor y el apoyo que me han brindado siempre, mis logros y triunfos alcanzados siempre serán para honrarles y enorgullecerles.

A **Mike**, por todo su amor y comprensión.

Maykel:

A mis **padres**, por el cariño, el amor, la confianza y el ejemplo, que me han transmitido durante toda la vida.

A mi **hermano**… por "salvarme la vida".

A mi abuela "**paquita**", aunque no estés con nosotros, yo siempre te he tenido a mi lado.

A mi abuela Elsie.

A Martha Barreto, a quien considero una **abuela** más.

A **Mela**, por ser la sobrina más linda que tengo.

A mi **tío** y mi **tía**, por ser mis padres estos cinco años, por estar siempre pendientes de mí.

A mis **primas** (la rana y la gorda), gracias por no dejarse vencer ante las dificultades, siempre serán mis primas queridas.

A **Yani**, por ser compañera, amiga y novia durante estos cinco años.

AGRADECIMIENTOS:

A la Revolución Cubana por darnos el derecho a todos a tener una educación gratuita de alta calidad y por hacer de nosotros jóvenes integrales, comprometidos y concientes del momento que estamos viviendo.

A las personas que han contribuido en nuestra formación profesional y a aquellos que nunca han dejado de darnos su incondicional apoyo y confianza.

A los profesores Eugenia G. Muñiz y Pascual Verdecia por sus consejos y apoyo incondicional siempre que los hemos necesitado.

A Manuel Vázquez Acosta por la paciencia que ha tenido con nosotros en el proyecto.

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

A todos aquellos que se acercaron a preguntar, a indagar por el estado de la tesis, a los que nos dieron su apoyo... Gracias a todos!!!

ÍNDICE

INTRODUCCIÓN:

La Oficina Nacional de Estadísticas (ONE) es la institución que se dedica en el país a llevar un registro de todas las estadísticas que son de interés para la planificación y control de la economía, los servicios, etc.

La ONE, debe brindarle a la dirección del país reportes estadísticos de interés cada cierto tiempo. Para lograr este objetivo la ONE cuenta con una Oficina Municipal de Estadística (OME) en cada municipio del país que le brinda la información a la Oficina Territorial de Estadística (OTE) que se encuentra en cada provincia y estas a su vez son las que le proporcionan la información a la ONE creando un flujo de información en ambos sentidos (ascendente y descendente).

Actualmente uno de los servicios que brinda la ONE al país es el Cálculo de Índices de Precios, el cual puede estar dirigido al consumidor, al comercio interior, hotelero e industrial. Para llevar a cabo este proceso, se define una canasta básica la cual va a estar integrada por los bienes y servicios más representativos, es decir, aquellos en los que la población mayormente consume. Estos son determinados mediante el estudio de los mercados más concurridos para hacer los sondeos de precios (variación de precios). Todo esto se realiza mediante un sistema de captación de los datos, se definen formularios donde se toma el producto con su precio. Estos sondeos se realizan una vez al mes en las provincias y semanal (en el caso de los mercados agropecuarios). Toda esta información se digitaliza, se calcula la media aritmética de los precios y se envía a la ONE.

La ONE cuenta actualmente con un sistema integral para el cálculo del índice de precios al consumidor. Dicho sistema solamente tiene la posibilidad de captar el precio (que no es estable) por lo cual ha quedado obsoleto. Otras desventajas del mismo es que no permite introducir tarifas, hacer comparaciones entre años y la sustitución de productos.

El proyecto “Cálculo de Índices de Precios al Consumidor” (en lo adelante IPC) surge a raíz de estos problemas, con el propósito de desarrollar un sistema más flexible, que permita cambiar el precio de los productos o servicios, así como la sustitución de cualquiera de los

mismos, hacer comparaciones, ajustes, un sistema que determine si los productos son estacionarios o no etc. Dicho proyecto ha sido estructurado por roles, donde cada rol tiene sus tareas y actividades específicas a desarrollar, lográndose así una mayor organización y planificación del trabajo a realizar. Como parte de este equipo de desarrollo los autores son los encargados de llevar a cabo el rol de Gestión de Configuración del Software (GCS).

Situación problemática:

La inexistencia de una gestión de configuración en los procesos de desarrollo de software, deriva en la mala utilización del recurso humano altamente calificado, trayendo consigo atrasos en actividades de desarrollo y mantenimiento de los productos y un inadecuado y a veces inexistente control de los proyectos. Esto se ha convertido en los últimos años en un tema crítico de nuestra sociedad, la cual aspira obtener software de elevada calidad en el menor tiempo posible y minimizar su costo, sin embargo, los resultados alcanzados no cubren las expectativas esperadas debido precisamente a diversos problemas, entre los que se destacan:

No existe: (FEBLES. 2001)

- Relación entre los cambios en los requisitos de una versión del producto y los cambios físicos sobre el código.
- Control de las versiones de los módulos, ficheros, proyectos, etc.
- Procedimiento para llevar a cabo las actividades relacionadas con la configuración en el desarrollo del software.
- Un formato preestablecido para hacer pedidos de cambios.
- Un seguimiento de los cambios de cada versión del proyecto.
- Registro de quien hizo cada actividad relacionada con el proyecto.

Lo que provoca, entre otros, los siguientes conflictos:

- Concurrencia en el acceso a los elementos de configuración.
- Duplicación de esfuerzos para llevar el sistema a un estado anterior.

- Es imposible ver el estado del progreso del proyecto.

La necesidad de hacer más eficaz, eficiente y efectiva la producción de software en la UCI y en particular en el proyecto productivo IPC, dada la urgencia de garantizar disciplina en la ejecución de los procesos asociados al rol de Gestión de Configuración de Software se define el siguiente **Problema de investigación:**

La falta de aplicación de la Gestión de la Configuración durante el desarrollo del proyecto IPC impide identificar, organizar y controlar las modificaciones que sufre el software.

Para enfrentar este problema, se definió como **Objeto de la investigación**:
Proceso de desarrollo del software.

Dentro del **Campo de acción:**
Proceso de investigación, comportamiento y aplicación de la Gestión de Configuración y Cambios en el proyecto productivo “Cálculo de Índices de Precios al Consumidor”.
Para responder al problema de investigación, se definió el **Objetivo general**:
Aplicar un modelo que permita una apropiada Gestión de Configuración y Cambios al proyecto productivo “Cálculo de Índices de Precios al Consumidor”.

A partir de un análisis del objetivo general se derivaron los siguientes **Objetivos específicos:**

1. Adoptar un modelo de Gestión de Configuración y Cambios para el proyecto “Cálculo de Índices de Precios al Consumidor”.
2. Analizar los resultados luego de la aplicación del modelo adoptado.

Se formuló la siguiente **Hipótesis de investigación**:

Aplicando la Gestión de la Configuración se identifican, organizan y controlan las modificaciones que sufre el software durante el desarrollo del proyecto IPC.

Indicadores:

- Dominio y uso de la herramienta para el control de versiones por parte de los miembros del proyecto.
- Dominio y uso de la herramienta para el control de versiones por parte de los responsables del rol de GCC.
- Organización del expediente del proyecto.
- Documentación y control de los cambios a los ECS.
    - Estado de los pedidos de cambio.
- Estado de los ECS.

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

1. Examinar y evaluar las formas utilizadas por las industrias de software de éxito en el mundo para dar seguimiento a los aspectos relacionados con la GCS.
2. Seleccionar y definir los procesos para la GCS que serán puestos en práctica en el proyecto IPC.
3. Analizar las herramientas para la GCS que existen en el mercado y seleccionar cuales pueden ser utilizadas para la automatización del proceso.
4. Establecer y aprobar el Plan de Gestión de la Configuración.
5. Establecer y aprobar el Plan de Gestión de cambios que regirá durante todo este proceso.
6. Determinar que se va a considerar como Elemento de Configuración de Software (ECS), teniendo en cuenta que los mismos se puedan definir y controlar de forma separada.
7. Identificar las líneas bases (BASELINE) de la configuración del software.
8. Mantener la integridad y trazabilidad de la configuración a lo largo de todo el ciclo de vida del software mediante las herramientas para el control de versiones.
9. Controlar frecuentemente los cambios durante todo el proceso de desarrollo.

Entre los **Métodos de trabajo científico** utilizados se destacan los siguientes:

Métodos teóricos: El análisis documental: facilitó, el análisis y sistematización del tema, a partir de los documentos de autores nacionales y extranjeros. El método hipotético-deductivo para la elaboración de la hipótesis central de la investigación; el método sistémico para lograr que los elementos que forman parte del modelo sean un todo que funcione de manera armónica; el método histórico-lógico y el dialéctico para el estudio crítico del los trabajos anteriores, y para utilizar estos como punto de referencia y comparación de los resultados alcanzados.

Método lógico: El método analítico-sintético al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta.

Métodos empíricos: El método de la entrevista para obtener los problemas presentes en las empresas cubanas de desarrollo de software; el método experimental para comprobar la utilidad de los resultados obtenidos a partir del modelo definido.

Estructura de la tesis

La tesis quedó estructurada en tres capítulos. El Capítulo I, referido al marco teórico y referencial de la investigación donde se realiza un análisis del estado del arte en el tema de Gestión de Configuración.

En este capítulo se realiza un estudio de los siguientes aspectos:

1- Principales conceptos y valoraciones de autores sobre la Gestión de Configuración a nivel internacional, así como un análisis del estado del arte de proyectos similares en la Universidad de las Ciencias Informáticas.

2- Particularidades de la Gestión de Configuración: actividades del proceso de GCC.

3- Evaluación de las principales herramientas para la Gestión de Configuración dado un entorno de desarrollo determinado. Mejores herramientas para la Gestión de Configuración del proyecto teniendo en cuenta las características del producto y el grado de conocimiento de los miembros del proyecto. Selección de la herramienta.

4- Análisis del estado actual del proyecto, teniendo en cuenta:

- Aplicación de la Gestión de Configuración.
- Errores provocados por la gestión actual.

El Capítulo II.

En el mismo se propone un modelo que describe procesos y procedimientos necesarios que apoyen a la puesta en práctica de la GCS en el proyecto y que sirven de base para la obtención de los resultados esperados en el proyecto IPC.

Capítulo III.

Se hace un análisis de los resultados del proyecto una vez concluido el trabajo de diploma luego de haberse implantado en el proyecto IPC el modelo propuesto en el capítulo I.

CAPÍTULO I: Antecedentes de la Gestión de Configuración de Software.

En el presente capítulo se muestra un estudio sobre los principales conceptos y valoraciones de autores sobre la Gestión de Configuración a nivel internacional, así como un análisis del estado del arte de proyectos similares en la Universidad de las Ciencias Informáticas. Se realiza una investigación de las principales herramientas más utilizadas para el desempeño de este rol, finalizando con un análisis crítico del mismo en el proyecto productivo IPC.

1.1. Estudio del estado del arte.

1.1.1. Gestión de Configuración de Software. Definiciones.

Muchas han sido las personalidades e instituciones que han dedicado gran parte de sus estudios a la Gestión de Configuración del Software, por lo que a lo largo de la historia del desarrollo del software se pueden encontrar infinidad de documentos y definiciones que hacen referencia a este tan importante tema.

La definición más utilizada de Gestión de Configuración de Software es la del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) que establece: (FEBLES. 2001)

“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”.

Roger S. Pressman, en su libro “Ingenieria del SoftWare. Un enfoque práctico” dedica un capítulo al tema de la Gestión de la Configuración, en el cual además de realizar un análisis bastante exhaustivo de la misma, la define como: (PRESSMAN 2005)

“… conjunto de actividades diseñadas para controlar el cambio identificando los productos del trabajo que probablemente cambien, estableciendo relaciones entre ellos, definiendo mecanismos para gestionar distintas versiones de estos productos, controlando los cambios realizados, y auditando e informando de los cambios realizados. “

El SEI (Software Engineering Institute), como parte de la Universidad Carnegie Mellon, ha reconocido la importancia del estudio de la Gestión de Configuración en la actualidad, por lo que también han incursionado en este tema, entre sus estudios sobre el mismo se puede encontrar que el propósito de la Gestión de Configuración de Software es establecer y mantener la integridad de los productos, a través del ciclo de vida de los proyectos de software. (HARVEY 1986)

Para (RATIONAL 2003) la GC “describe la estructura del producto e identifica los elementos que lo constituyen y que son tratados como entidades que pueden ser puestas bajo control de versiones en el proceso de GC. La GC tiene que ver con la definición de la configuración, así como la construcción, el etiquetado y recolección de versiones de los artefactos”.

Watts Humphrey, autor de innumerables libros como “A discipline for software engineering” y “Managing the Software Process” en (HUMPHREY. 2000) define la Gestión de Configuración como:

“… todas las actividades usadas para administrar el contenido de un producto software desde el principio hasta el fin del proceso de desarrollo. El propósito del proceso de Gestión de Configuración de Software es asegurar que el contenido de un producto es conocido y está disponible todo el tiempo, que las funciones del producto son identificadas desde los requisitos a través del diseño de la implementación final, y que todos los volúmenes del producto se controlan propiamente y se protegen.”

De todas las definiciones revisadas durante el presente trabajo, (CMU 1996; FEBLES. 2001; HUFF agost 1994; HUMPHREY. 2000; PRESSMAN 2005; RATIONAL 2003) se ha considerado como la más completa y a su vez concisa la de Ivar Jacobson, Martín Griss y

Patrick Jonson en el libro “Software Reuse: Architecture, Process, and Organizartions for Busisness Success” en la que plantea: (JACOBSON IVAR 1997)

“Gestión de Configuración: Procesos 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”

1.1.2. Gestión de Configuración de Software. Valoraciones.

A continuación se brindan apreciaciones de diversos autores que permiten estimar cuanto se valora la GC, y qué nivel de importancia se le otorga dentro del desarrollo de software.

Para enormes y complejos productos, los sistemas de gestión de configuración de software pueden ser realmente sofisticados. Para los ingenieros de procesos de software en equipo, sin embargo, los sistemas de gestión de configuración de software están relacionados enormemente con el control de cambios. Los cambios desenfrenados pueden perder enormes cantidades de tiempo, corromper el diseño, causar errores que pueden ser utilizados, e incluso resultar en malos productos. (HUMPHREY. 2000)

En (RATIONAL 2004), se hace referencia a una frase escrita por Tom Milligan asesor de IBM Rational Software, donde plantea que: “La Gestión de Configuración de Software es el héroe no nombrado en el proceso de desarrollo de software”.

Para concluir no debe dejarse de mencionar la Ley Fundamental de la Gestión de Configuración (BROWN 1998), donde se define el papel de la Gestión de Configuración de Software en el proceso de desarrollo:

“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”

1.1.3. La Gestión de Configuración a nivel nacional e internacional.

La Industria del Software en el mundo se ha desarrollado considerablemente en los últimos años, sin embargo, los resultados logrados no cubren las expectativas esperadas, producto en gran medida a una mala utilización del recurso humano altamente calificado, provocando esto a su vez atrasos en actividades de desarrollo y mantenimiento de los productos, además de un inadecuado y a veces inexistente control de los proyectos. Esto se ha convertido en los últimos años en un tema crítico de nuestra sociedad moderna, la cual aspira obtener software de elevada calidad en el menor tiempo posible y al menor costo, aspiración por demás un poco distante debido básicamente a que la productividad que se alcanza, en general, es baja, la cantidad de recursos a consumir en tiempo principalmente es alta y el trabajo realizado casi nunca tiene la calidad requerida. Los proyectos se concluyen en fecha posterior a lo planificado y los problemas no se detectan a tiempo.

A pesar del interés gubernamental y del indiscutible desarrollo de la Ingeniería del Software en los últimos años, las estadísticas internacionales no son alentadoras. Informes de instituciones dedicadas al análisis de software muestran los siguientes indicadores: (FEBLES. 2001)

- 25% de los proyectos de software son abortados,

- Se liberan productos a sus clientes con remanentes del 15% de defectos,

- Muchas empresas gastan de 30% a 44% de su tiempo y dinero en volver a trabajar sobre software ya liberado.

- Se cumplen las planificaciones de tiempo solamente el 53% de las veces.

Cuando se profundiza en las raíces o causas de tales resultados afloran de manera reiterada la aplicación inadecuada de las técnicas de Ingeniería de Software, la no utilización de los roles y procesos apropiados para el desarrollo de las tareas de la empresa de software y el no utilizar modelos de calidad en ellas. (FEBLES. 2001)

El desarrollo de la industria del software en el mundo y en particular en Cuba exige la aplicación de procedimientos, estándares y modelos que garanticen la calidad del software elaborado. Cuando se construye software de computadora, los cambios son inevitables. Además, los cambios aumentan el grado de confusión entre los ingenieros de software que están trabajando en el proyecto. La confusión surge cuando no se han analizado los cambios antes de realizarlos, no se han registrado antes de implementarlos, no se les han comunicado a aquellas personas que necesitan saberlo o no se han controlado de manera que mejoren la calidad y reduzcan los errores.(PRESSMAN 2005)

Las dificultades relacionadas con la mala organización y funcionamiento de las pequeñas y medianas empresas de software que inciden en la calidad del producto final están relacionadas con la ausencia, entre otros, de modelos de referencia para la introducción y ejecución, de forma iterativa e incremental de los procesos de GCS, dirigidos a mejorar su competitividad en el mercado y disminuir los costos. El éxito de un proyecto y por lo tanto de una empresa de software depende de la correcta ejecución de cuatro tipos de funciones: (FEBLES. 2001)

1- Gestión del Proyecto.

2- Desarrollo Técnico.

3- Sistema de Calidad.

4- Sistema de Gestión de Configuración.

La ISO (Organización Internacional de Normalización) 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)

Específicamente la guía ISO 9000-3 entre los aspectos que tiene en cuenta se puede encontrar la Gestión de configuración de Software. El hecho de que una norma de calidad tan reconocida a nivel mundial incluya dentro de sus requerimientos, el de efectuar las labores de GCS, es plenamente consistente con la importancia que le otorgan los diversos autores analizados.

El CMM (Modelo de Maduración de Capacidades) fue desarrollado inicialmente para los procesos relativos al software por la Universidad Carnegie-Mellon para el SEI. Es un modelo el cual brinda un procedimiento para pasar de un proceso inmaduro, a un proceso maduro. Precisamente la Gestión de Configuración desempeña un papel importante a la hora de conseguir incrementar el nivel de madurez del proceso software de una organización. En concreto se trata de una de las áreas clave, en el modelo CMM, necesarias para que una organización de desarrollo de software alcance el nivel 2 (repetible), de entre los 5 que plantea dicho modelo. (CMU 1996)

Según estudios realizados, en el nivel repetible o nivel 2 hasta el 2001 solo el 33% de las organizaciones habían incorporado a su sistema de trabajo la Gestión de Configuración. (FEBLES. 2001)

En (ESTRADA and CÁRDENAS 2005) se muestra como a través de 31 encuestas realizadas a desarrolladores y líderes de las empresas de la Industria Cubana de Software (ICSW), un 61%, no conocen que es GCS y en un 79% no se aplican ninguno de los procedimientos asociados a este proceso. Esta misma encuesta fue aplicada a estudiantes de tercer año que formaban parte de equipos de desarrollo de software en el Instituto Superior Politécnico “José Antonio Echeverría” (CUJAE) y los resultados fueron similares.

Como se ha visto hasta el momento en la bibliografía estudiada e información recopilada varios autores coinciden e insisten en considerar que la Gestión de Configuración es el proceso central en la producción de software el cual debe estar presente en todo el ciclo de vida, sin embargo, a pesar de todo lo planteado acerca de la importancia de la misma, según estadísticas internacionales y encuestas nacionales es un proceso no conocido, que no se concreta en las organizaciones y por lo tanto tampoco se ejecuta cabalmente trayendo como resultado pérdida de información, gasto innecesario de recursos, insatisfacción del cliente y desconocimiento de los directivos y líderes sobre el estado real de los proyectos.

Esto en parte también se debe a que en muchas ocasiones se ve la solución de la empresa de software en la aplicación de la tecnología más novedosa, y no en realizar las actividades relacionadas a la gestión del proyecto, la gestión de la configuración, la gestión de la calidad y las mediciones. (FEBLES. 2001)

Se considera que el desafío principal es lograr organizaciones de software maduras, las cuales se caractericen por una alta capacidad para administrar los procesos de software con un enfoque futurista, que respondan por la motivación, el compromiso, la participación y la destreza del personal, considerando estos elementos como parte de los valores por los que debe aspirar todo especialista de software.

Con una introducción ordenada de la Gestión de Configuración se logra que los implicados creen disciplina y no se opongan a la aplicación de procedimientos en su rutina de trabajo. Esto garantiza que los procesos además de ser bien definidos sean bien aplicados y consecuentemente adquieran un mayor rendimiento y calidad.

1.1.4. Análisis del estado del arte de proyectos similares en la Universidad de las Ciencias Informáticas.

El rol de Gestión de Configuración en la Universidad de las Ciencias Informáticas (en lo adelante UCI), hasta finales del año 2005 no se consideraba como un proceso concretamente formalizado, ya que en la mayoría de los proyectos productivos solamente se

controlaban las versiones, pero no existía dentro de los equipos de desarrollo un rol que se encargase de la Identificación de la Configuración, Control de Cambios, Planificación de la Gestión de Configuración y Generación de Informes de Estado.

Según los resultados obtenidos en la entrevista realizada a especialistas de la UCI, la cual puede consultarse en el anexo 1, del proceso de Gestión de Configuración de Software en la UCI se conoce que no están bien definidos los procesos que regulan la Gestión de Configuración y todo se hace un poco basado en la experiencia personal de los líderes de proyectos, de los especialistas en Gestión de Configuración y de un alto nivel de empirismo. Prácticamente no se hace gestión de cambios, no se les da seguimiento a los requisitos ni a los cambios de forma tal que se les pueda dar a los clientes una explicación detallada del estado del producto a partir de las solicitudes de cambio realizadas por el mismo.

Una vez que fueron aumentando en complejidad los productos a desarrollar, se fue haciendo más difícil a la hora de mantener la integridad de los componentes del producto software y facilitar la visibilidad del mismo, por lo que fueron al traste muchos de los proyectos.

Gracias a una encuesta realizada durante el curso 2005-2006 por la dirección de calidad de la UCI (ver anexo 2), a los proyectos existentes en todas las facultades de la misma, se cuenta con estadísticas que muestran de una forma real el estado actual de la Gestión de Configuración a nivel de facultad en la UCI. A continuación, en la figura 1 se muestra gráficamente el resultado de dicha encuesta por lo que se puede corroborar lo planteado anteriormente sobre el estado en que se encuentra la puesta en práctica del rol de Gestión de Configuración en la UCI.

Ya para mediados del curso 2005-2006 se fomentó en las facultades el trabajo con este rol y se comenzaron a establecer métodos de trabajo con el mismo, con el objetivo de además de disciplinar a los involucrados en el proceso de desarrollo, almacenar y disponer de los datos históricos necesarios para lograr un trabajo más predecible y eficiente.

Hasta ese entonces las herramientas más utilizadas eran Visual SourceSafe y Concurrent Versions System (CVS), aunque ya se estaban dando los primeros pasos en la utilización del Subversion, el cual aportaba mayores ventajas que el CVS. El Subversion en la mayoría de los casos se instalaba sobre la plataforma Windows y por medio de un servidor Apache o las aplicaciones clientes Tortoise o AnkhSVN se interactuaba con el repositorio.

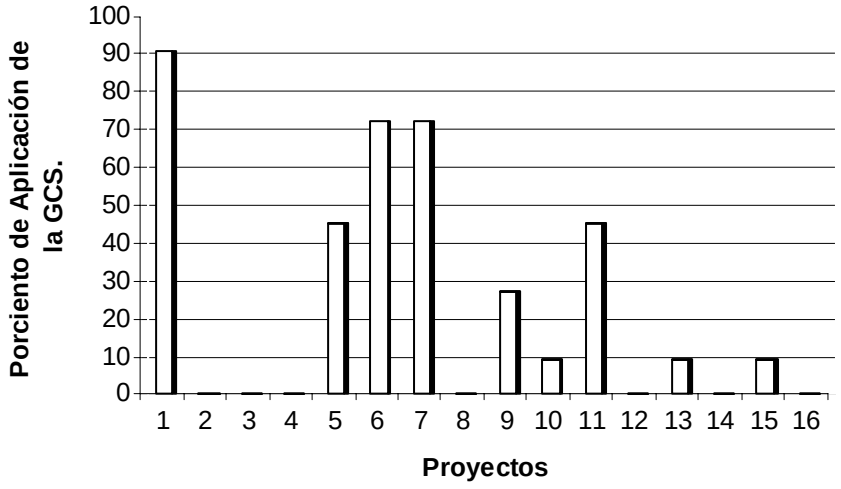

Figura 1 (Resultados de la encuesta realizada a los proyectos productivos de la facultad3 en la UCI)

1.2. Metodologías de desarrollo de software.

El mundo de la informática no para de hablar de procesos de desarrollo, el modo de trabajar eficientemente para evitar catástrofes que llevan a que un gran porcentaje de proyectos se terminen sin éxito.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software, estas van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué

papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (MURCIA 2006)

El propósito de una metodología de desarrollo es principalmente proveer normas de conducta que son aplicables a través de las herramientas, lenguajes y equipos. (KALLEBERG 2005)

¿Que metodología usar? ¿Cuál será la más idónea según las características del producto? Estas son las interrogantes que se plantea todo equipo de desarrollo de software antes de emprender una nueva tarea. No obstante, no tiene sentido ajustarse a un proceso al pie de la letra, sino que hay que adaptarlo a las necesidades y características del equipo de trabajo en el proyecto.

Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte se encuentran aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. (BECK 1999)

Por otra parte se puede encontrar la filosofía de las metodologías ágiles, esta se centra en otras dimensiones, como por ejemplo el factor humano o el producto software, la cual ofrece mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque ha mostrado su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. (BECK 1999)

Es importante destacar que la aplicación de una metodología no asegura, en sí misma, el éxito del proyecto. La aplicación de una metodología permite asegurar al menos que se ha realizado el suficiente esfuerzo para lograrlo, dentro de un marco teórico razonable.

1.2.1 Metodologías ágiles: Extreming Programing (XP)

Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales. A continuación se describe brevemente la Programación Extrema (eXtreme Programming, XP), metodología ágil más popular en la actualidad. (BECK 1999)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Este representante debería estar en condiciones de contestar rápida y correctamente a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones. (JOSÉ H. CANÓS 2006)

El ciclo de vida ideal de XP consiste de seis fases: (JOSÉ H. CANÓS 2006)

- Exploración.
- Planificación de la Entrega (Release).
- Iteraciones.
- Producción.
- Mantenimiento.
- Muerte del Proyecto.

XP define UserStories (historias de usuario) como base del software a desarrollar. Estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del software, que no solo se limitan a la Graphical User Interface (GUI), también pueden describir el modelo, dominio, etc. A partir de las UserStories y de la arquitectura perseguida se crea un plan de releases entre el equipo de desarrollo y el cliente. (KALLEBERG 2005)

Junto a los UserStories están los escenarios de pruebas que describen el escenario contra el que se comprueba la realización de las UserStories. UserStories y casos de pruebas son la base sobre la que se asienta el trabajo del desarrollador. La codificación del software en XP se produce siempre en parejas (dos programadores, un ordenador), por lo que se espera que la calidad del software se eleve en el momento de escribir el mismo. Al contrario que muchos otros métodos, el código pertenece al equipo en completo, no a un programador o pareja, de forma que cada programador puede cambiar cualquier parte del código en cualquier momento si así lo necesita, dejándose en todo caso las mejoras orientadas al rendimiento para el final. (KALLEBERG 2005)

En XP se programará solo la funcionalidad que es requerida para la release actual. Es decir, una gran flexibilidad y capacidad de configuración solo será implementada cuando sea necesaria para cumplir los requerimientos del release. Se sigue un diseño evolutivo con la siguiente premisa: conseguir la funcionalidad deseada de la forma más sencilla posible. Este diseño evolutivo hace que no se le de apenas importancia al análisis como fase independiente, puesto que se trabaja exclusivamente en función de las necesidades del momento.

1.2.2 Metodologías pesadas: Rational Unified Process (RUP).

El proceso de desarrollo lleva asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Este esquema "tradicional" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y

recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. (BECK 1999)

Rational Unified Process (RUP), es un proceso de ingeniería de software propuesto por Rational Software Corporation para la construcción completa del ciclo de ingeniería de software. Permite la productividad en equipo y la realización de mejores prácticas de software a través de plantillas y herramientas que lo guían en todas las actividades de desarrollo del software. Es un producto que unifica las disciplinas en lo que a desarrollo de software se refiere, incluyendo modelado de negocio, manejo de requerimientos, componentes de desarrollo, ingeniería de datos, manejo y configuración de cambios, y pruebas, cubriendo todo el ciclo de vida de los proyectos basado en la construcción de componentes y maximizando el uso del UML (Unified Modeling Language). (FOWLER 2003; JACOBSON IVAR 1997)

Los aspectos definitorios de RUP pueden resumirse en tres frases claves: *dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental*. Esto es lo que hace único al proceso unificado. Para hacer que estas ideas funcionen, se necesita un proceso polifacético, que tenga en cuenta ciclos, fases, flujos de trabajo, gestión de riesgo, control de calidad, gestión de proyectos y **Gestión de configuración**. (JACOBSON IVAR 1997)

Un proyecto, siguiendo RUP, se divide en cuatro fases: (JACOBSON *et al.* 2000)

Inicio:

Durante la fase de inicio se desarrolla una descripción del proyecto final a partir de una buena idea y se presenta el análisis del negocio para el producto. Esencialmente esta fase responde a las siguientes preguntas:

- ¿Cuáles son las principales funciones del sistema para sus usuarios más importantes?
- ¿Cómo podría ser la arquitectura del sistema?
- ¿Cuál es el plan del proyecto y cuanto costará desarrollar el producto?

Elaboración:

Se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema.

Construcción:

Se crea el producto, se añaden los músculos (software terminado) al esqueleto (la arquitectura).

Transición:

Cubre el período durante el cual el producto se convierte en versión beta. En esta versión un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Esta fase conlleva actividades como la fabricación, formación del cliente, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentren tras la entrega.

1.2.3 Justificación de la metodología seleccionada.

Luego de realizar un estudio de las metodologías más utilizadas para el desarrollo de software, se puede encontrar que entre ellas existen similitudes, por ejemplo, ambas presentan como base UseCases y UserStories, estas describen los requerimientos de la aplicación desde el punto de vista del usuario y definen los requisitos técnicos sin adentrarse en detalles de implementación. Estas dos metodologías se basan en un proceso iterativo, lo que permite acercarse poco a poco a la solución sin adentrarse demasiado en los detalles, aunque las iteraciones de XP tienen por lo general una duración menor que en RUP, puesto que la carga a llevar por los programadores a parte del desarrollo del propio software es menor.

No obstante RUP está pensado para proyectos y equipos grandes, en cuanto a tamaño y duración. XP se implementa mejor para proyectos cortos y equipos más pequeños. La diferencia fundamental entre ambos es que mientras lo métodos pesados intentan conseguir el objetivo común por medio de orden y documentación, los métodos ligeros (también llamados métodos ágiles) tratan de mejorar la calidad del software por medio de una

comunicación directa e inmediata entre las personas que intervienen en el proceso. Mientras que el RUP intenta reducir la complejidad del software por medio de la estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

En el desarrollo de un proyecto con XP es más importante la entrega al cliente del software que necesita que las funcionalidades que quedan por implementar. Por su parte, RUP es un proceso basado mucho en la documentación. Define en cada momento del ciclo de vida del proyecto, que artefactos, con que nivel de detalle, y por qué rol, se deben crear. También cuales artefactos son necesarios para poder realizar una actividad y que artefactos se deberán crear durante dicha actividad.

Con RUP se presentarán al cliente los artefactos del final de una fase y se valorarán las precondiciones para la siguiente y solo después de que el cliente acepte los artefactos generados se pasará a la siguiente fase. La calidad de los artefactos generados será probada durante la totalidad del ciclo de vida del proyecto a través de distintas medidas de calidad. En contrapartida, el aseguramiento de la calidad en XP no se basa en formalismos en la documentación, sino en controles propios y una comunicación fluida con el cliente. En XP el cliente recibe después de cada iteración un pedazo funcional del programa. A través de un ciclo de iteración corto (pocas semanas) el cliente es informado constantemente sobre la situación del proyecto y puede intervenir rápidamente si el desarrollo se aleja de sus necesidades.

En XP se conseguirá a través de la programación a pares que en la creación del código se puedan evitar errores y malos diseños ya que se controlará cada línea de código y decisión de diseño instantáneamente. En RUP se intentará reducir la complejidad del software a producir a través de una planificación intensiva. Así se intentará evitar que por la desaparición de algún miembro clave del equipo se pierda el conocimiento sobre la aplicación.

En todo caso, los métodos de desarrollo modernos ponderan la utilización frecuente de reuniones entre los miembros del equipo, que pueden ir desde diarias, como propone XP, a semanales o mensuales, de duración de minutos o de horas, según los objetivos de la reunión.

Luego de un profundo análisis, teniendo en cuenta las características antes mencionadas, se considera que la metodología que mejor se adapta a la situación concreta que se enfrenta el proyecto para obtener los resultados esperados, es la proporcionada por RUP. Otro factor que influye en esta decisión es el conocimiento que tiene cada integrante del equipo de desarrollo de la misma, puesto que ha sido la metodología estudiada en el plan de estudio de la carrera.

1.3. Particularidades de la Gestión de Configuración de Software.

Los procesos asociados a la GCS que aparecen en el estándar de la IEEE abarcan las siguientes actividades: (IEEE 1990; JIMÉNEZ 2001)

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

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

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

Por otra parte ISO sugiere para esta área los siguientes procesos: (ISO 1995)

- Identificación de la configuración.

- Control de cambios a la configuración.
- Informe del estado de la Configuración.
- Auditoría de la configuración.

Sin embargo se considera que las tres propuestas, individualmente son incompletas ya que los sistemas que automatizan la gestión de configuración suelen incluir el control de versiones, con el fin de mantener un registro histórico de las diferentes versiones por las que pasan los componentes de un producto, que permita la recuperación de cualquiera de ellas.

Por consiguiente, se adopta el modelo propuesto por la Dra. Ailyn Febles en (FEBLES. 2001), quedando los mismos de la siguiente forma:

- Identificación de la Configuración.
- Gestión de Cambios en la Configuración.
- Planificación de la Gestión de Configuración.
- Gestión de Versiones.
- Generación de Informes de Estado.

Como se puede observar, en el modelo propuesto no se incluye la Auditoría de la Configuración, a pesar de ser un proceso que es presente en IEEE, CMM e ISO, ya que la misma es una actividad de garantía de la calidad de software que ayuda a asegurar que se mantenga la calidad durante la realización de los cambios.(PRESSMAN 2005) Además, en (FEBLES. 2001) se considera que para la pequeña y mediana empresa (PYME) es mucho más factible y viable ubicar la auditoría de configuración fuera de la Gestión de Configuración y dentro de la Garantía de Calidad de la organización. Las actividades asociadas a la auditoría son extremadamente costosas, requiere de personal experimentado, y con un gran conocimiento del proceso de desarrollo. Sin embargo, debe ser realizada por personal ajeno al equipo de desarrollo técnico para mantener la objetividad de la auditoría. Lograr esto individualmente en cada proceso en las PYME no es factible.

Por estos motivos es que los autores, como encargados de la gestión de configuración del proyecto IPC, se encargarán de coordinar con el grupo de calidad de la facultad para que este sea quien realice la Auditoría de la configuración del proyecto.

La Gestión de Versiones, es referida indistintamente por algunos autores (FEBLES. 2001; JIMÉNEZ 2001) pero no es incluido como un proceso independiente en ninguno de los estándares analizados. Se decidió incluirlo por las siguientes razones:

Dentro del control de configuración los dos elementos fundamentales son los cambios y las versiones asociadas a los ECS. Ninguno puede ser obviado en la búsqueda de eficiencia y calidad.

Cuando se mezcla el control de versiones con el control de los cambios en un proceso único, sin delimitar fronteras y actividades específicas se diluyen las responsabilidades y no se ejecutan muchas tareas que son imprescindibles como la actualización de los repositorios, el control de acceso a los ECS, el seguimiento de componentes que no son ECS, etc.

El proceso de Planificación de la Gestión de Configuración aparece en la propuesta de CMM y en la IEEE pero no en la de la ISO. (FEBLES. 2001; JIMÉNEZ 2001) se incluye este proceso dentro del modelo por la importancia de la planificación para organizar tareas en tiempo, destinar recursos, asignar responsabilidades y posteriormente poder darle seguimiento a lo planificado. Para el control de la configuración es clave, pues es una actividad que se realiza durante todo el ciclo de vida del proyecto, incide en todos los componentes del proyecto e involucra a todos los miembros del equipo de trabajo.

Las áreas de procesos de Identificación de la Configuración, Gestión de Cambios en la Configuración y Generación de Informes de Estado son coincidentes en los tres estándares analizados ISO, IEEE y CMM. (FEBLES. 2001; JIMÉNEZ 2001)Por su importancia e incidencia dentro del proceso fueron seleccionados para formar parte del modelo por las siguientes razones:

1. Es importante que se establezca en la organización un método único para la identificación de los elementos de configuración y que sea conocido por todas las partes involucradas, para que todos los integrantes del proyecto se comprometan con sus objetivos y actúen uniformemente.
2. Cuando se construye un software los cambios son inevitables. Una de las principales amenazas para la calidad de software viene justamente de los cambios, fuente aparentemente benigna. No atender, analizar y ejecutar los cambios de forma organizada implica, entre otros, problemas de calidad, de satisfacción del cliente, pérdida de información, etc. (PRESSMAN 2005)
3. Una de las reglas de oro de CMM es, “mantenga informado a todos los interesados”. La generación y transmisión de informes es una manera de cumplir con esta regla.

1.3.1. Identificación de la Configuración.

A través de la identificación de la configuración se identifican y asignan nombres significativos y consistentes a todos y cada uno de los elementos que forman parte del producto software, en cada fase de su desarrollo, es decir, a cada uno de los Elementos de Configuración del Software (ECS). (JIMÉNEZ 2001)

Para poder llevar a cabo de forma exitosa el proceso de Identificación de la Configuración de Software, es necesario realizar un conjunto de actividades: (JIMÉNEZ 2001)

- 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.

1.3.1.1. Selección de los Elementos de Configuración:

“Los elementos que componen toda la información producida como parte del proceso de Ingeniería de Software se denominan colectivamente Elementos de Configuración de Software.” (PRESSMAN 2005)

Para llevar a cabo la selección de los ECS se debe tener en cuenta separar en ECS distintos las funcionalidades distintas, con el objetivo de minimizar el impacto de los cambios, ya que si un ECS incluye funciones muy diversas es probable que aumente el número de veces que será necesario repetir el proceso de reconstrucción del elemento y liberación de una nueva versión. (PRESSMAN 2005)

Además se deben tener presente los siguientes criterios: (JIMÉNEZ 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.
- 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.

La importancia de una adecuada selección de los elementos de configuración radica en que la elección de excesivos ECS puede provocar un número excesivamente elevado de especificaciones y documentos, que al final resultarían inmanejables. Además, otro inconveniente es que el mantener muchos ECS puede resultar muy costoso, sin embargo, el tener pocos puede provocar que no se tenga la suficiente visibilidad sobre el producto. (JIMÉNEZ 2001)

1.3.1.2. Definición de las relaciones en la configuración:

Se puede considerar que los ECS son objetos, y están conectados con otros ECS mediante relaciones. Esta información ayudará al personal de GCS a comprender dónde se sitúa un elemento con respecto al resto. Algunas de las relaciones que puede ser interesante mantener son las siguientes: (JIMÉNEZ 2001)

- **Equivalencia**: cuando determinado ECS está almacenado en varios lugares diferentes pero todas las copias corresponden al mismo ECS.
- **Composición**: ECS que esté compuesto por otros artefactos de la metodología o de otros módulos del producto.
- **Dependencia**: Relaciones entre ECS, fundamentalmente para facilitar el seguimiento de los requisitos.
- **Derivación**: A partir de qué se ha originado algo.
- **Sucesión**: Historia de los cambios sobre un ECS desde una revisión a otra.

Esta actividad es de gran importancia ya que gracias a estas relaciones, una vez solicitado un cambio sobre un ECS, puede determinarse fácilmente cuales otros ECS se verían afectados con dicho cambio, lo cual es un factor importante para determinar si es factible la autorización del cambio.

1.3.1.3 Definición de un Esquema de Identificación:

Es necesario decidir también cuál es el método que se va a utilizar para identificar de forma unívoca cada Elemento de Configuración del Software. Dicho en otras palabras, es necesario establecer un Esquema de Identificación que permita etiquetar cada uno de los Elementos de Configuración del Software. Sea cual sea, el esquema de identificación utilizado debe proporcionar al menos la siguiente información: (JIMÉNEZ 2001)

1. Número o código del Elemento de Configuración del Software.
2. Nombre del ECS.
3. Descripción del ECS.
4. Autor/es del ECS.
5. Fecha de creación.
6. Identificación del proyecto al que pertenece el Elemento de Configuración del Software.
7. Identificación de la línea base a la que pertenece.
8. Identificación de la fase y subfase en la que se creó.
9. Tipo de Elemento de Configuración (documento, programa, elemento físico (cintas, discos, etc.).
10. Localización.

Por otro lado, el esquema de identificación debe permitir diferenciar entre las diferentes versiones de un mismo Elemento de Configuración del Software, puesto que los ECS van a evolucionar a lo largo del ciclo de vida. Para ello, se suelen utilizar los siguientes campos: (JIMÉNEZ 2001)

11. Número de versión.
12. Fecha de la versión.

Toda esta información puede ir codificada y agrupada en un único código de identificación o puede ser referenciada como partes separadas.

1.3.1.4. Definición y establecimiento de Líneas Bases:

Para controlar los cambios sin impedir los cambios justificados se utiliza la Línea Base o “Baseline”, la cual se define como:

“Una especificación o producto que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios.” (PRESSMAN 2005)

La IEEE define una Línea Base como una especificación o producto que ha sido revisado formalmente y se ha aceptado, que más tarde servirá como punto de partida para el desarrollo, y que solo podrá ser cambiado a partir de un proceso formal de control de cambio. (IEEE 1990)

Analizando las definiciones expuestas anteriormente se puede llegar a la conclusión que existen un grupo de elementos comunes en las definiciones. La mayoría toman a las Líneas Base como puntos de partida para desarrollos futuros dentro del Proceso de Desarrollo de Software, son establecidas al cumplir con un hito en el proyecto y que una vez creadas, los cambios sobre los ECS que la conforman se realizan bajo un proceso formal de control de cambio.
Una línea base se puede establecer de dos formas: (FEBLES. 2001)

• Físicamente: Etiquetando cada Elemento de Configuración del Software y almacenándolos en un Archivo o Biblioteca de Proyecto.
• Lógicamente: Publicando un documento de identificación de la configuración, que referencia el estado actual del producto en dicho punto del proceso de desarrollo.

Mientras que la primera variante almacena físicamente el elemento, la segunda solo registra que versión de cada elemento forma parte de la misma. Al incluir el concepto versión, se crea una dependencia de la herramienta gestora de versiones empleada, pues la línea base sería un documento que hace referencia a versiones de ficheros almacenados en el gestor de versiones, con lo cual solo podrán ser recuperados a través de este. Y es la recuperación precisamente uno de los objetivos fundamentales de la línea base, para poder servir como punto de partida para el desarrollo futuro.

1.3.1.5 Definición y Establecimiento de bibliotecas de software:

Las Bibliotecas de Software son una colección controlada de software y/o de la documentación relacionada al mismo, diseñada para ayudar en el uso, el desarrollo y el mantenimiento del software. (JIMÉNEZ 2001; MARTÍNEZ and SOCA 2006)

En (JIMÉNEZ 2001) se definen un grupo básico de bibliotecas a ser establecidas, las cuales fueron tenidas en cuenta para el establecimiento de las bibliotecas en el proyecto IPC, entre las más importantes se encuentran:

Biblioteca de trabajo: Se establece al inicio del proyecto, y comprende el área de trabajo donde los analistas y diseñadores elaboran los documentos del proyecto y donde los programadores desarrollan el software, es decir, donde se realiza la codificación y pruebas unitarias entre otras. En esta biblioteca el control de cambios es informal.

Biblioteca de Soporte al Proyecto: En esta biblioteca se almacenan los ECS aprobados y transferidos desde la Biblioteca de Trabajo. Cuando un elemento pasa a esta biblioteca, se encuentra sujeto a un control de cambios interno y semiformal.

Biblioteca Maestra: Se usa para almacenar ECS liberados para su entrega al cliente o distribución en el mercado. Los elementos en la biblioteca maestra se encuentran sujetos a un control de cambios formal y estricto. 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 Maestra es individual para cada proyecto.

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

1.3.2. Gestión de cambios en la configuración:

A medida que progresa el proceso de desarrollo, el número de ECS crece rápidamente. La Especificación del Sistema da lugar al Plan del Proyecto y a la Especificación de Requisitos Software. A su vez estos dan lugar a otros documentos, etc. Si simplemente cada ECS produjera otros ECS, no habría prácticamente confusión. El problema aparece cuando entra en juego la variable CAMBIO.

"…El cambio se puede producir en cualquier momento y por cualquier razón. De hecho, la Primera Ley de la Ingeniería de Sistemas establece: Sin importar en que momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida." (PRESSMAN 2005)

El control de cambios es la actividad de Gestión de Configuración más importante y su objetivo es proporcionar un mecanismo riguroso para controlar los cambios, partiendo de la base de que los mismos se van a producir. Normalmente combina procedimientos humanos

y el uso de herramientas automáticas. El origen de los cambios es tan variado como los cambios mismos. Existen cuatro fuentes fundamentales de cambios: (JIMÉNEZ 2001)

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

En cualquier caso es necesario establecer de forma precisa, al comienzo de cada proyecto, cuál será el proceso de gestión de cambios que se va a utilizar. Para ello, será necesario definir: (JIMÉNEZ 2001)

- Políticas a nivel organizativo que promuevan las actividades de control de cambios.
- Los estándares que se van a adoptar y a los que será necesario ajustarse.
- Los procedimientos que se van a utilizar para poner en práctica las políticas de Gestión de Configuración.

Mientras que el objetivo de las políticas es definir el ‘por qué’ de la Gestión de Configuración, deben establecer un marco para definir el ‘qué’, el ‘cuándo’, el ‘cómo’ y el ‘quién’: (JIMÉNEZ 2001)

- Las responsabilidades.
- El contenido de las líneas base.
- Los tipos de cambios que se van a controlar.
- Las funciones del Comité de Control de Cambios (CCC).
- El flujo de documentación entre el solicitante de un cambio y el CCC.

- Los criterios para la valoración de las solicitudes de cambio, tanto de tipo técnico como de gestión.

Por lo general se establecen varios niveles de control de cambios: (JIMÉNEZ 2001)

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

Tanto en el proceso formal como en el semi-formal, aparece una nueva figura en la Organización, el Comité de Control de Cambios. El Comité de Control de Cambios es una persona o grupo encargado de tomar las decisiones finales acerca del estado y la prioridad de las peticiones de cambio. (JIMÉNEZ 2001)

El papel del CCC es tener una visión general, o sea, evaluar el impacto del cambio del ECS en cuestión. ¿Cómo impactará el cambio en el hardware? ¿Cómo impactará en el rendimiento? ¿Cómo alterará el cambio la percepción del cliente sobre el producto? ¿Cómo afectará el cambio a la calidad y a la fiabilidad? El CCC se plantea estas y otras muchas cuestiones. (PRESSMAN 2005)

El proceso de control de cambios:

No hay ningún estándar para el control informal o interno de los cambios, aunque sí hay algunas recomendaciones (IEEE STD 1042 Guide to Software Configuration Management).

En cuanto al control de cambios formal, se puede estructurar de muchas formas. Las etapas típicas de un proceso formal, es decir, el proceso que habría que seguir para hacer un cambio sobre una línea base son las que se muestran a continuación: (JIMÉNEZ 2001)

1. Iniciación del Cambio.
2. Clasificación y registro de la solicitud de cambio.
3. Aprobación o rechazo inicial de la solicitud de cambio.
4. Evaluación de la solicitud de cambio.
5. Presentación del Informe de Cambio al Comité de Control de Cambios.
6. Realización del cambio, entrando en un proceso de seguimiento y control.
7. Certificación del cambio.
8. Notificación del resultado a quien origina el cambio.

Los procesos de alta y baja de la Biblioteca de soporte al proyecto implementan dos elementos importantes del control de cambios: el control de acceso y el control de sincronización. (JIMÉNEZ 2001)

El control de acceso gobierna los derechos de los ingenieros de software a acceder y modificar objetos de configuración concretos. El control de sincronización asegura que los cambios en paralelo, realizados por personas diferentes no se sobrescriban mutuamente. (PRESSMAN 2005)

1.3.3. Planificación de la Gestión de Configuración:

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

El Plan de Gestión de Configuración debe contener los siguientes apartados, según el estándar de IEEE, aunque por lo general los estándares se adaptan a las necesidades de la organización. (JIMÉNEZ 2001)

1. Introducción:
    1.1. Propósito del Plan.
    1.2. Alcance.
    1.3. Definiciones y acrónimos.
    1.4. Referencias.
    1.5. Definición de alto nivel del proceso de GCS.
2. Especificaciones de gestión:
    2.1. Organización.
    2.2. Responsabilidades.
    2.3. Implementación del Plan de Gestión de Configuración.
    2.4. Políticas, directivas y procedimientos aplicables.
3. Actividades de Gestión de Configuración:
    3.1. Identificación de la Configuración.
    3.2. Control de la Configuración.
    3.3. Contabilidad del Estado de la Configuración.
    3.4. Auditoría de la Configuración.
4. Control de suministradores.
5. Recogida y retención de registros.

1.3.4. Gestión de versiones en la configuración:

Los cambios sobre un ECS en un momento dado del proceso de desarrollo conducen a la creación de una nueva versión de dicho ECS. El proceso de control de versiones consiste en mantener un registro histórico de las diferentes versiones por las que pasan los componentes de un producto, que permita la recuperación de cualquiera de ellas. (JIMÉNEZ 2001)

Una versión es una instancia de un ECS, en un momento dado del proceso de desarrollo, que es almacenado en un repositorio, y que puede ser recuperado en cualquier momento para su uso o modificación. (JIMÉNEZ 2001)

El control de versiones combina procedimientos y herramientas para gestionar las versiones de los objetos de configuración creados durante el proceso del software. Clemm describe el control de versiones como: (PRESSMAN 2005)
La gestión de configuración permite a un usuario especificar configuraciones alternativas de sistemas de software mediante la selección de las versiones adecuadas. Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar [y construir] una configuración describiendo el conjunto de atributos deseados.

1.3.5. Generación de Informes de Estado:

El objetivo de esta tarea es mantener a los usuarios, a los gestores y a los desarrolladores al tanto del estado de la configuración y su evolución. En definitiva, pretende dar respuesta a las preguntas “¿Qué ocurrió?”, y “¿Cuándo ocurrió?”. Esto se logra registrando toda la información precisa acerca de lo que va ocurriendo y generando los informes necesarios. Dicha tarea implica, por tanto, la realización de tres actividades básicas: (JIMÉNEZ 2001)

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

Por lo que los productos de esta actividad son fundamentalmente de dos categorías:

- Registros.
- Informes.

Cada vez que se le asigna una nueva identificación a un ECS o una identificación actualizada se genera una entrada para la generación de Informes de Estado la Configuración (IEC). Cada vez que el CCC aprueba un cambio, se genera una entrada en el IEC. Cada vez que se lleva a cabo una auditoría de la configuración, los resultados aparecen como parte de una tarea de generación de un IEC. (PRESSMAN 2005)

La importancia de esta tarea radica en: (JIMÉNEZ 2001)

- Mantener la continuidad del proyecto. Se trata de permitir que el proyecto siga adelante cuando, por ejemplo, el jefe de proyecto se encuentra ausente.

- Evitar la duplicidad del trabajo. Si no se guarda información acerca de lo que ya se ha hecho, se puede estar repitiendo el trabajo.

- 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.

La información a capturar proviene de las tres actividades de GCS ya tratadas (Identificación de la configuración, control de cambios en la configuración y control de versiones). Toda esta información es almacenada en una base de datos que se podrá utilizar, además de para generar los informes de Contabilidad de Estado pertinentes, para facilitar otras tareas como: (JIMÉNEZ 2001)

- Análisis post-mortem del proyecto.

- Realización de estimaciones para proyectos futuros.

1.4. Principales herramientas para la GC.

Las herramientas de control de versiones o gestión de revisiones ayudan a crear, identificar y almacenar nuevas versiones, al mismo tiempo que se mantienen las versiones anteriores.

El software de control de versiones permite conceder con facilidad acceso a los desarrolladores de una manera organizada, con lo cual pueden estar seguros de que disponen de la versión más reciente del código. Estos programas de control de versiones incluyen características que le permiten etiquetar un conjunto de secuencias como parte de una versión, lo que facilita la obtención de la copia más reciente con todos los cambios.

Existen muchas herramientas de control de versiones como son, Source Code Control System (SCCS), Revision Control System (RCS), Project Revision Control System (PRCS), Monotone, CMSynergy, Aegis, Arch, ClearCase, etc.

A continuación, las principales características de las Herramientas de Control de Versiones más utilizadas junto con las herramientas clientes de mayor uso en desarrollos .net, java etc.

1.4.1 Herramientas (Software Propietario)

El Visual Source Safe(VSS) es una herramienta de software que se utiliza para el control de versiones de un proyecto. La misma se basa en la concepción del modelo cliente – servidor y presenta dos módulos, el de cliente y el de administración.

Esta herramienta de software ayuda a administrar los proyectos, sin tener en cuenta el tipo de archivo (archivos de texto, archivos gráficos, archivos binarios, archivos de sonido o archivos de vídeo) guardándolos en una base de datos. Cuando se necesita compartir archivos entre dos o más proyectos, se puede realizar de manera rápida y eficiente. Cuando se agrega un archivo a VSS, se hace una copia de seguridad del archivo en la base de datos, de forma tal que esté disponible para las demás personas, y los cambios que se hagan en el archivo se guarden de forma que se pueda recuperar una versión anterior en

cualquier momento. Los miembros del equipo pueden ver la versión más reciente de cualquier archivo, hacer cambios y guardar una nueva versión en la base de datos. (UCI 2005)

La última versión de Visual SourceSafe cuenta con diversas características nuevas y mejoradas que simplifican la funcionalidad básica y mejoran el acceso a las bases de datos, así como su fiabilidad y rendimiento. A continuación un resumen de sus principales características incorporadas. (MICROSOFT 2005)

Para mejorar el rendimiento de las bases de datos, Visual SourceSafe tiene una mayor capacidad de almacenamiento de datos de 4 GB, a la vez que tiene límites de almacenamiento más elevados. La operación de recuperación de archivos es dos veces más rápida que en las versiones anteriores. Reduce la necesidad de analizar, administrar y reparar las bases de datos. Es compatible con las bases de datos de las versiones 4.0, 5.0 y 6.0.

Visual SourceSafe permite crear tareas personalizadas que recuperen y descarguen archivos de código fuente antes de crearlos mediante una generación automatizada. La última versión trata el contenido de archivo codificado en Unicode, incluidos los archivos XML, como texto, lo que permite a los desarrolladores almacenar, comparar y combinar estos archivos al igual que cualquier otro archivo de texto. Con el fin de permitir la realización de operaciones paralelas, Visual SourceSafe incluye un nuevo estilo de trabajo Copiar-Modificar-Combinar para varias desprotecciones de una base de datos.

Las interfaces de usuario tradicionales del Explorador y Administrador de Visual SourceSafe proporcionan menús y barras de herramientas compatibles con los elementos de la interfaz de usuario de Visual Studio o Microsoft Office. También presenta varios asistentes que guían al usuario a través de las operaciones de bases de datos.

Rational ClearCase:

“IBM Rational ClearCase” es una de las herramientas incluidas en la Suite Rational 2003. Ofrece una gestión de los activos de software para equipos de desarrollo de mediano y gran tamaño.(IBM 2005)

Entre las funcionalidades de Rational ClearCase se encuentran:

• Gestión del ciclo de vida y control de los activos de desarrollo de software.

• Control integrado de versiones.

• Gestión de Línea Base.

• Integración con WebSphere Studio, Eclipse y Microsoft® .NET.

• Integración con Rational ClearQuest.

Al igual que Rational ClearQuest, Rational ClearCase se integra con el resto de la Suite Racional. Así como la alta generalidad con que cuenta en tareas como la generación de informes, que por el grado de flexibilidad y parametrización que posee, tiene una curva de aprendizaje alta, requiriendo de grandes esfuerzos para su asimilación.

1.4.2 Herramientas (Software Libre).

Source Forge:

Es un ambiente de desarrollo colaborativo, herramientas heterogéneas y procesos con un entorno integrado para la administración de proyectos, administración de cambios y capacidades de colaboración. (SOURCEFORGE 2007)

Entre las funcionalidades de Source Forge se pueden encontrar:

Herramientas de proyectos:

• Seguimiento de problemas y cambios.

• Administración de tareas.

• Sistema de liberaciones (release) de ficheros.

Colaboración:

• Administración de documentos.

• Forum de discusión y noticias.

• Lista de correos.

Interoperabilidad:

• Microsoft Project.

• Microsoft Office.

• Sistema de Control de Versiones (Subversion, CVS, Rational ClearCase).

Subversion:

Es un sistema de control de versiones libre y de código fuente abierto, el cual puede ser usado para administrar cualquier conjunto de ficheros.

A continuación se muestra algunas de las principales características de Subversion: (COLLINS-SUSSMAN *et al.* 2004)

- Implementa un sistema de ficheros versionado “virtual” que sigue los cambios sobre árboles de directorios completos a través del tiempo. Ambos, ficheros y directorios, se encuentran bajo el control de versiones.

- Permite añadir, borrar, copiar, y renombrar ficheros y directorios. Y cada fichero nuevo añadido comienza con un historial nuevo.

- Una colección cualquiera de modificaciones o bien entra por completo al repositorio, o bien no lo hace en absoluto. Esto permite a los desarrolladores construir y enviar los cambios como fragmentos lógicos e impide que ocurran problemas cuando sólo una parte de los cambios enviados lo hace con éxito.

- Posee una noción abstracta del acceso al repositorio, facilitando a las personas implementar nuevos mecanismos de red. Subversion puede conectarse al servidor HTTP Apache como un módulo de extensión. Esto proporciona a Subversion una gran ventaja en estabilidad e interoperabilidad, y acceso instantáneo a las características existentes que ofrece este servidor autenticación, autorización, compresión de la conexión, etcétera. También tiene disponible un servidor de Subversion independiente, y más ligero. Este servidor habla un protocolo propio, el cual puede ser encaminado fácilmente a través de un túnel SSH.

- Muestra las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de texto (legibles para humanos) y ficheros binarios (ilegibles para humanos). Ambos tipos de ficheros son almacenados igualmente comprimidos en el repositorio, y las diferencias son transmitidas en ambas direcciones a través de la red.

Subversion, utiliza el modelo Copiar – Modificar- Fusionar, el mismo es una alternativa al bloqueo que consiste en que cada usuario crea una copia local en su máquina del fichero o del Proyecto, luego se trabaja en paralelo donde cada uno hace sus cambios independientemente sin esperar uno por el otro y después todas las copias privadas se fusionan en una nueva versión final. Esto permite que varias personas trabajen en un mismo archivo simultáneamente, de manera que ambos bajan el archivo a su carpeta de trabajo, modifican y después fusionan estos cambios.

Otra de las características que presenta es que no sólo se integra con el Sistema Operativo Windows sino también con Linux, plataforma de Software Libre a la cual la Universidad desea migrar cuando las condiciones estén creadas y además, una de las políticas de desarrollo del software que ha establecido la universidad es que todos los proyectos trabajen con esta herramienta por ser más robusta y perteneciente a la Comunidad de Software libre.

Algunas de las herramientas clientes más utilizadas para el control de revisiones desde Subversion se muestran a continuación, las mismas permiten realizar las operaciones más comunes de control de versiones directamente desde el entorno de desarrollo en que se este trabajando.

AnkhSVN:

Es un agregado para Microsoft Visual Studio.NET del sistema de control de versiones en Subversion. Este le permite realizar las operaciones de control de versión más comunes directamente dentro del VS.NET IDE. No todas las funcionalidades brindadas por SVN están aun disponibles, pero si están implementadas la mayoría de las operaciones que brinda el flujo de trabajo diario. (COLLABNET 2006a)

Subclipse:

Es un agregado de Eclipse que añade la integración de Subversion al Eclipse IDE. Subclipse requiere un adaptador cliente de Subversion para comunicarse con el repositorio. Subclipse incluye un adaptador cliente 100% Java llamado SVNKit.

Para usuarios Windows, Subclipse también incluye la biblioteca JavaHL de Subversion la cual se comunica con su repositorio usando las bibliotecas y API estándares de Subversion. (COLLABNET 2006b)

TortoiseSVN: (COLLABNET 2006c)

Es un software realmente fácil para el control de versiones, entre sus principales características se encuentran:

Fácil de Usar:

- Todos los comandos están disponibles desde el explorador de Windows.
- Solamente se muestran los comandos que tengan sentido para el archivo/carpeta seleccionado.
- Puede verse el estado de un fichero directamente en el explorador de Windows.
- Permite mover archivos mediante clic derecho/arrastrando en el explorador de Windows.

Se brindan todos los protocolos de Subversion:

Potente interfaz Commit:

- Integra corrector ortográfico para los mensajes de los cambios.
- Auto llenado de caminos y palabras claves del fichero modificado.
- Permite dar formato al texto con caracteres especiales.

Permite realizar un gráfico de todas las revisiones, estadísticas de todas las revisiones realizadas durante el proyecto.

Fácil comparación entre dos ramas.

También dispone de un gran número de Herramientas Provechosas como son:

TortoiseMerge: muestra los cambios que un usuario realiza en sus ficheros y ayuda a resolver conflictos.

TortoiseBlame: para saber quien hizo un cambio en particular y por que.

TortoiseIDiff: para ver los cambios que ha sufrido un determinado fichero.

SubWCRev: para el número de revisión, la fecha, etc. en el fichero de código

También se encuentra **Trac**, una herramienta open source de interfaz Web, la cual integra herramientas básicas de apoyo a la Gestión de la Configuración, a la Administración y planificación de procesos, el cual contiene un sistema de seguimiento (Tracking System). Trac cuenta con plena integración a Subversion, contenido orientado a componentes, contiene una wiki integrada, línea de tiempo (Time Line), diagrama de Gantt para la planificación, interfaz de administración para el control de versiones y permite notificar cambios vía correo electrónico. Trac ha sido desarrollado en Python, es multiplataforma y tiene seguridad integrada. (EDGEWALL 2006)

Gracias al Trac se puede lograr una mejor gestión de los componentes a desarrollar, se puede mantener un seguimiento de las tareas y los defectos reportados sobre los componentes, una mejor comunicación entre todos los desarrolladores y permite obtener estadísticas del avance del proyecto, basadas en el análisis del cumplimiento de las tareas asignadas.

Como se ha visto en el análisis de las principales características y funcionalidades de estas herramientas para la Gestión de Configuración de Software, si bien es cierto que todas dan soporte a muchas de las actividades incluidas en la GCS, dejan a un lado cosas muy importante como la identificación de los elementos de configuración y la administración de líneas base, además que en muchas toman solo como GCS el control de versiones.

Actualmente en nuestro país existe una propuesta arquitectónica de una herramienta de apoyo a la Gestión de Configuración nombrada ConfigCASE 3.0 con soporte para los procesos de Identificación de la Configuración y Control de Cambios a la Configuración, también admite la abstracción del repositorio de versiones que permite realizar de forma automática actividades sobre el mismo evitando su uso manual por parte de los miembros del proyecto. En caso de ser aprobado el desarrollo del ConfigCASE 3.0, sería de gran ayuda para el desempeño de los procesos de Gestión de Configuración, puesto que incluye casi la totalidad de las actividades contenidas en la misma. (MARTÍNEZ and SOCA 2006)

1.4.3 Justificación de la Selección.

Para el caso del proyecto “Cálculo de Índices de Precios al Consumidor” se considera que la herramienta para el Control de Versiones más conveniente a usar es el Subversion, primero que todo porque la misma es Open Source (Código Abierto). Además, la ONE es una institución gubernamental, y como es estrategia del país migrar al software libre, puede que ellos lo hagan en cualquier momento.

Aunque se desarrolle el producto con Visual Studio. NET y lo más lógico sería utilizar el Visual Source Safe, realizar el control de versiones con Subversion no presentaría ningún inconveniente puesto que se cuenta con la herramienta cliente Ankhsvn, la cual permite realizar las operaciones de control de versión más comunes directamente desde el VS.NET.

Otro motivo por el que se decide la herramienta a utilizar es que la dirección de producción de la facultad 3 de la UCI ha llevado a cabo un número de acciones elementales para que los proyectos tengan un medio más efectivo de llevar a cabo el rol de Gestión de Configuración, y precisamente la herramienta de control de versiones elegida ha sido Subversion.

La dirección de producción de la facultad, a tono con las políticas globales de la Universidad ha preparado sistemas de control de versiones para que todos los proyectos gestionen su código fuente y ha elegido un sistema que permite el intercambio y la planificación de las tareas del proyecto.

A diferencia de los servidores actuales que se encuentran en la Infraestructura Productiva (IP), se ha optado por distribuir y delegar la responsabilidad de estos sistemas por todos los laboratorios de producción de la facultad. Actualmente, todos los laboratorios de producción tienen una máquina para la gestión de código fuente, la configuración y la planificación de los proyectos existentes en dicho laboratorio, por lo que no existe un cuello de botella para modificar los permisos de los repositorios. En la figura 2 se muestra gráficamente la distribución de los servidores de Control de Versiones en los laboratorios de la facultad 3 de la UCI.

Para eliminar el cuello de botella que supone que el administrador sea quien pueda establecer los permisos en los repositorios, se diseñó un esquema elemental que permite que cada líder de proyecto maneje su repositorio. Además, se hace esto sin tener que dar privilegios de administración a los líderes de proyectos.

En cada servidor existe un repositorio especial, llamado svn-meta, que solamente utilizan los líderes de proyectos. Este repositorio no es para gestionar códigos de ninguno de los proyectos, sino para que los líderes puedan manejar los permisos en sus proyectos.

La idea consiste en crear una carpeta para cada proyecto, y dentro de estas los ficheros que enumeran los usuarios y sus permisos en el repositorio real.

Cuando un líder de proyecto necesita cambiar los permisos de su proyecto, sólo tiene que modificar esos ficheros y subirlos para el repositorio especial svn-meta. El servidor detecta esa operación y realiza las operaciones necesarias para que los permisos sean aplicados.

Las características del sistema son:

- Debian como sistema operativo.
- SVN para el control de versiones.
- Trac para la documentación y planificación.

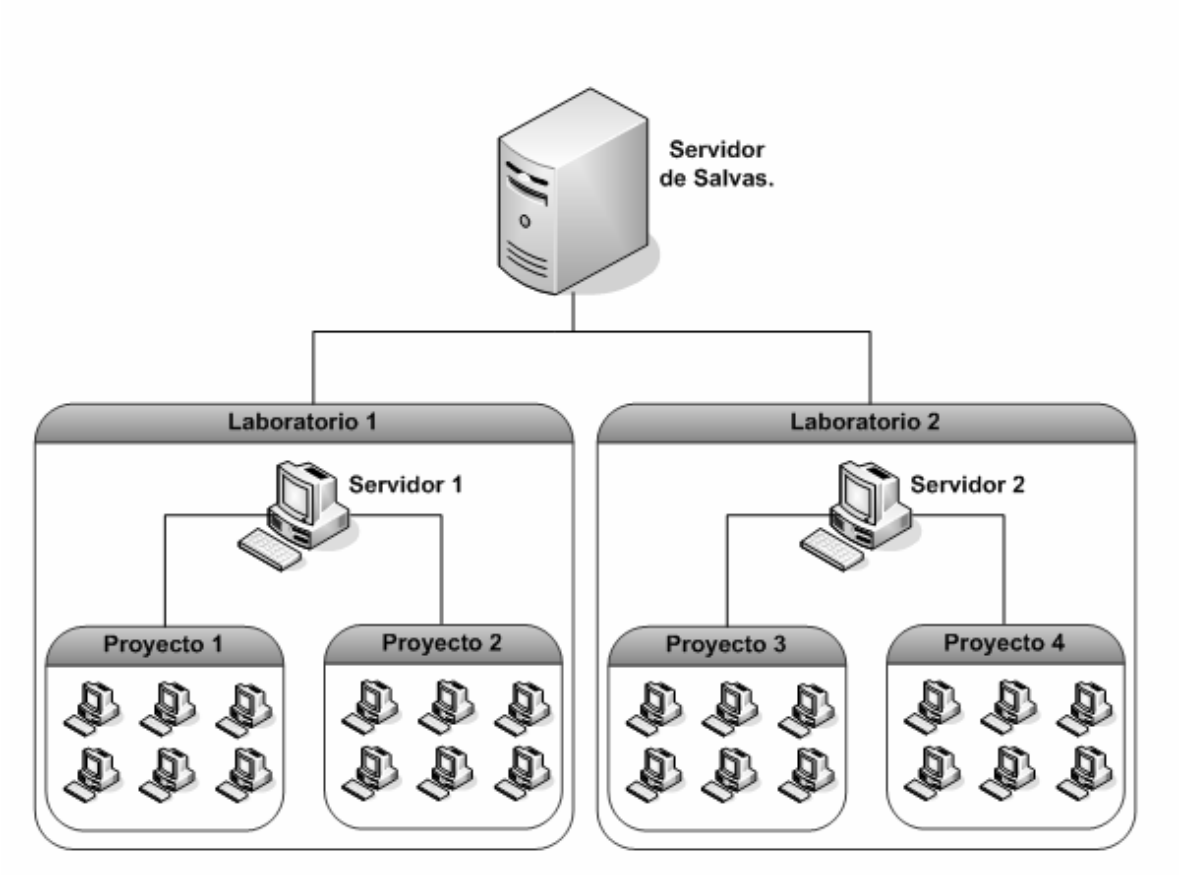

Figura 2: Distribución de los servidores de Control de Versiones en los laboratorios de la facultad 3 de la UCI

El Debian es un sistema Linux muy robusto y en el cual instalar y configurar el resto de los servicios no resulta difícil. Además, permite automatizar la realización de backups, usando la herramienta rsync.

El SVN es un sistema de control de versiones bastante flexible y eficiente en términos de rapidez y espacio. Además, los servidores de Gforge de la Universidad están basados en SVN y por tanto una migración, o un proceso de backup a estos servidores es muy simple de realizar.

El Trac, no tiene todas las facilidades del Gforge, pero se considera que es suficiente para las actuales necesidades de los proyectos. Es además muy fácil de instalar, configurar y salvaguardar. No depende de ningún componente externo para almacenar su información, tiene una interfaz directa con el SVN, Wiki integrada, línea de tiempo, diagrama de Gantt para la planificación, permite notificar cambios vía correo electrónico y posee una wiki versionada, lo cual permitirá almacenar el expediente de proyecto en la misma e ir controlando las versiones de este.

Estos esquemas presentan varias ventajas, la primera es que, al tener estos servidores en el mismo laboratorio donde se desarrollan los proyectos, cualquier desconexión en la red que aísle al laboratorio de otros puntos de la universidad, no afecta a los proyectos porque la red dentro del laboratorio se mantiene y por tanto se puede tener acceso a los servicios.

En segundo lugar, no se congestiona la red, ni una única máquina (o un número más limitado de ellas), porque se distribuye mucho mejor la carga de trabajo. En tercer lugar, la máquina puede ser utilizada, porque se ha instalado el Debian con un entorno gráfico, de esta forma el encargado de mantener el servidor puede estar utilizando el ordenador o en caso de que no se sienta cómodo con el Debian, puede utilizar de forma remota otra computadora con Windows 2003, de esta forma no se pierde un puesto de trabajo. En cuarto lugar, la realización de backups a servidores de salva se puede configurar en pocos minutos y una vez programados, estos backups se realizan automáticamente. Y por último, no hay dependencia de un administrador para la operación de los proyectos, sino que se delegan las responsabilidades de administración sin comprometer la seguridad del servidor.

No obstante quedan algunos temas por resolver, ya que como estos servidores están en Linux, no existe una forma “fácil” de añadirlos al dominio o al DNS lo que obliga a lidiar con números IP o a poner un servidor de DNS en cada laboratorio. Sin embargo, es posible fijar un nombre para estos servidores, tanto en una PC con Windows como en Linux, basta con editar el fichero que se encuentra en system32\drivers\etc\host (win) o etc\host (linux), y añadir la línea siguiente:

10.32.18.29 ipc.uci.cu.

Esto hará que la PC en la cual ha sido modificado el fichero no tenga que consultar al DNS de la UCI para saber que ipc.uci.cu tiene el IP 10.32.18.29.

Otro problema es la obligatoriedad de utilizar permisos de lectura para todos los usuarios (*=r) en el fichero dav_svn.authz (aunque esta sea una práctica deseable para tener compartidos los componentes en desarrollo y desarrollados), puesto que la configuración actual de los repositorios obliga a que esta línea esté presente en cada repositorio, porque el Apache media la autentificación y está configurado de forma que las operaciones de lectura no requieren de autentificación, pero como el Subversión no da autorización a los usuarios no autentificados, si no se pone esta línea, no habría forma para que los miembros del equipo crearan su copia local.

Como el expediente de trabajo se encuentra en el trac, no es posible (por el momento) versionar los diagramas localizados en los documentos producidos, ya que los mismos son imágenes.

1.5. Análisis del estado del proyecto en sus inicios en cuanto a la Gestión de Configuración.

El proyecto productivo IPC comenzó a desarrollarse a principios del año 2007. El sistema a desarrollar tiene como antecedente un viejo sistema de cálculo de índice de precios el cual ha quedado obsoleto y no posee documentación de su desarrollo, lo que implica prescindir de elementos (componentes, módulos) ya logrados y ventajosos que servirían como entrada y facilitarían el proceso de la Gestión de Configuración. Basándose en dichas circunstancias se tomaron como primeros elementos de configuración la escasa documentación facilitada por la ONE. Dicha documentación está compuesta por la ayuda del sistema software anterior y otros documentos que proporcionan una visión general inicial para lograr un mayor entendimiento del proceso a desarrollar.

Luego de un profundo análisis y tomando en cuenta los lamentables resultados a los que han llegado los proyectos estudiados en el epígrafe 1.1.4, debido a la falta de la aplicación de gestión de configuración y con el fin de no caer en los mismos, se abogó porque en el proyecto IPC fuese puesto en práctica el rol de gestión de configuración desde sus inicios.

No obstante, a pesar de la aplicación de los procesos de GC, el proyecto IPC no ha estado ajeno a la presencia de dificultades surgidas al comienzo de este, las cuales pueden ser fácilmente observadas tomando en cuenta algunos indicadores que permitan llevar un seguimiento y monitoreo de las mismas.

Se presentaron algunas dificultades en cuanto al dominio y uso de las herramientas de control de versiones por parte de los encargados de la gestión de configuración, ya que en un comienzo fue utilizado Subversion corriendo en la plataforma Windows, pero siguiendo la estrategia de la facultad y dadas las ventajas que esta opción brinda se determinó montar el Subversion en un servidor Debian, esto trajo consigo todo un proceso de adaptación al Debian puesto que surgieron dificultades a la hora de otorgar permisos y configurar el trac. Por otro lado los integrantes del equipo de desarrollo solo poseían conocimientos básicos del trabajo con la herramienta de control de versiones, por lo que fue necesario un proceso de familiarización con la misma a través de talleres, cursos de capacitación y videos instructivos, publicados en el trac del proyecto. En cuanto a la utilización de la herramienta, se puede decir que ha sido deficiente, ya que se ha creado un poco de rechazo ante esta nueva forma de trabajo, a pesar de que en reiteradas ocasiones se les ha explicado a los miembros del equipo de desarrollo la importancia de su uso, quedando demostrado que aun no se cuenta con un alto nivel de compromiso por parte de los integrantes del proyecto. Una forma de corroborar el problema anteriormente planteado es mediante los indicadores que se muestran a continuación:

A. Porcentaje de utilización del repositorio por los integrantes del proyecto.
B. Número de modificaciones sobre un ECS en un período de tiempo.
C. Número de commits realizados al repositorio en un período de tiempo.

Como se puede apreciar, la figura 3, muestra el porcentaje de utilización del repositorio por los integrantes del proyecto, en la misma se aprecia como el 50% de los integrantes del equipo de desarrollo no han realizado ningún tipo de modificación en el repositorio. Por otra parte, en la figura 4, se puede apreciar, como en el rango de una semana solo se modificó el ECS:"Especificaciones de CU del sistema" dos veces (el cual trascurrido ese tiempo aun continuaba sin completar). También, si se analiza la figura 5 se puede ver como durante una semana, solo dos usuarios realizaron modificaciones sobre el repositorio.

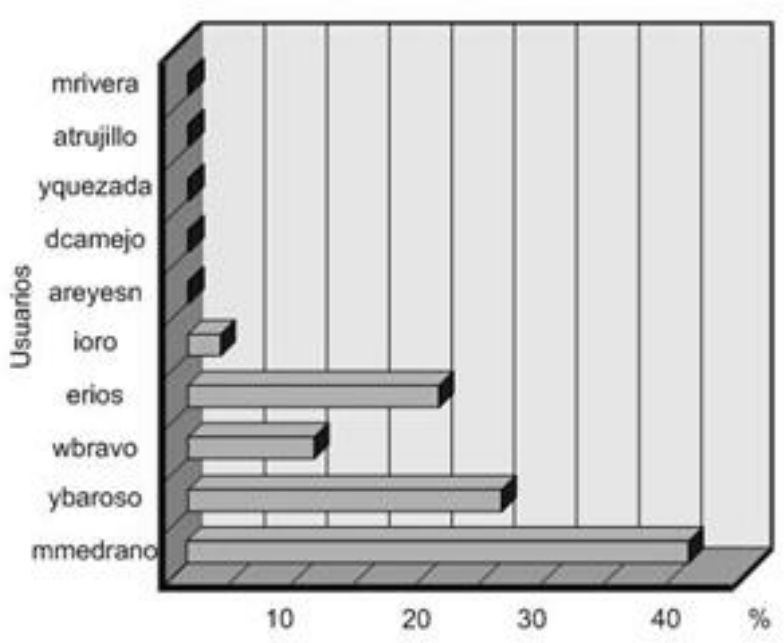

Figura 3: Porcentaje de utilización del repositorio por los integrantes del proyecto.

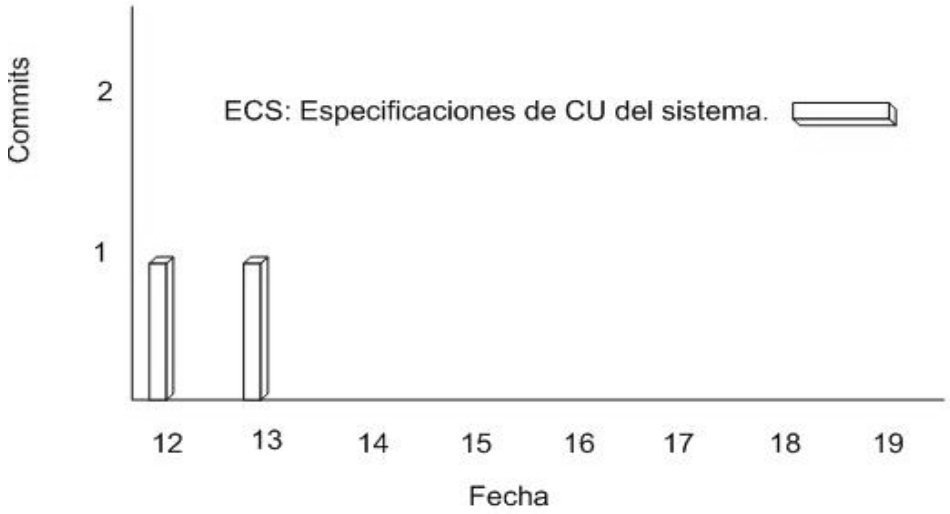

Figura 4: Número de modificaciones sobre un ECS en un período de tiempo (12/01/2007~19/01/2007).

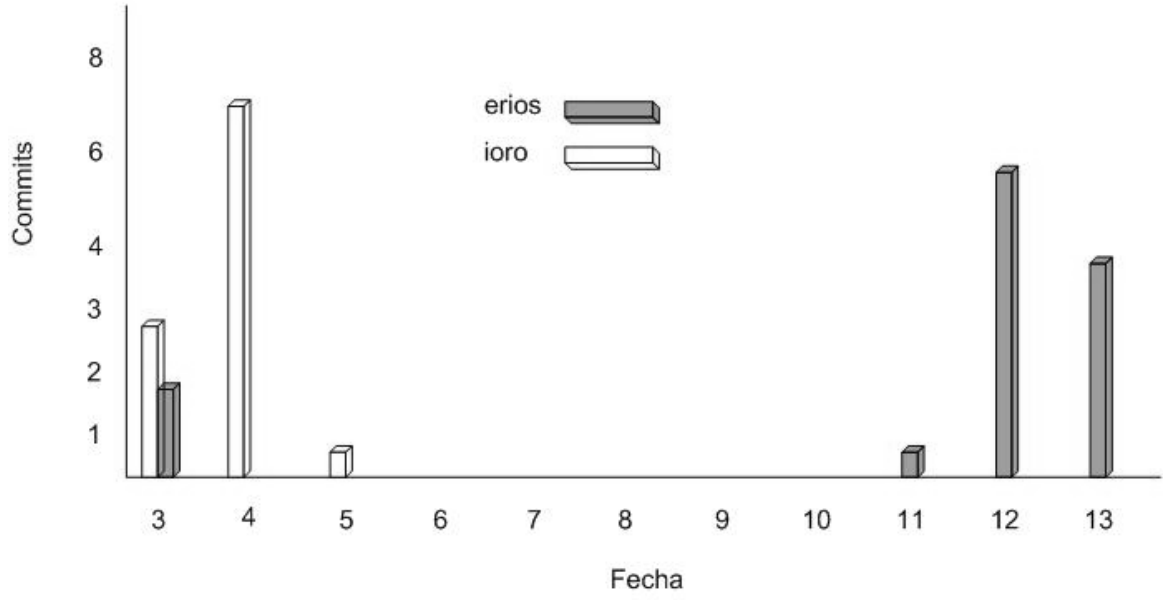

Figura 5: Número de commits realizados al repositorio en un período de tiempo (3/01/2007~13/01/2007).

La escasa existencia de plantillas que guiaran y sirvieran de soporte, con el fin de lograr un expediente de trabajo organizado, con un formato uniforme y que a su vez agilizaran el proceso de desarrollo, fue otro de los problemas que afectaron el avance del proyecto, es decir no se contaba con la cantidad necesaria para documentar los artefactos generados durante el proceso de desarrollo de software y las pocas que se poseían eran las generadas por RUP, las cuales se encuentran en idioma ingles.

Faltaba por refinar la estructura organizativa del expediente del proyecto dentro del trac, además de verse directamente afectado ya que variaba constantemente su estructura a medida que se le añadían plantillas, trayendo esto consigo su inestabilidad y el desconocimiento del mismo por parte de los desarrolladores, los cuales además manifestaban rechazo por la utilización del trac para almacenar el expediente del proyecto ya que no poseían experiencia en la utilización del mismo.

La documentación de los cambios estaba presentando serias dificultades puesto que los integrantes del equipo de desarrollo, en su mayoría no conocían los procedimientos formales que debían seguirse para llevar a cabo cualquier cambio y otros, simplemente no seguían estos procedimientos, pasando por alto algunos cambios de poca repercusión, pero que no fueron documentados.

Por tanto, se concluye que existió una poca utilización de los ECS existentes en el repositorio, por parte de los integrantes del equipo de trabajo. Además, como consecuencia directa, todos los problemas antes mencionados trajeron consigo pérdida de información al no estar la misma centralizada y a disposición de todos los integrantes del equipo de desarrollo. La inexistencia de una correcta planificación del trabajo para realizar las tareas provocaba un gasto innecesario de recursos. La falta de un conocimiento pleno de la herramienta de control de versiones por parte de los integrantes del equipo de desarrollo propiciaba la duplicación de esfuerzos para llevar el sistema a un estado anterior. El desconocimiento de los directivos y líderes, sobre el estado real del proyecto tornaba

prácticamente imposible ver el progreso del mismo, impidiendo que este pudiese ser planificado, supervisado (monitoreado), y controlado correctamente.

CAPÍTULO II: Aplicación de la Gestión de Configuración de Software en el proyecto.

El presente Capítulo muestra los procedimientos y mecanismos utilizados en cada una de las actividades de la GC y la forma de organización del contenido del proyecto.

De forma específica, los aspectos fundamentales del capítulo, se centran con mayor prioridad en:

2.1. Aplicación del modelo propuesto al proyecto.
2.2. Organización y contenido del proyecto.

2.1. Aplicación del modelo propuesto al proyecto IPC.

Para lograr un correcto desempeño del rol de Gestión de Configuración se han trazado los siguientes objetivos:

- Establecer y mantener la integridad de los productos generados durante el proceso de desarrollo del proyecto software a lo largo de todo el ciclo de vida del producto.
- Evaluar y controlar los cambios sobre ellos, es decir, controlar la evolución del sistema software.
- Facilitar la visibilidad y la información sobre el producto.

Para materializar dichos objetivos se hizo necesaria la realización de los procesos que constituyen la propuesta del modelo presentado en el capítulo 1:

- **Identificación de la Configuración:** Se identificó la estructura del producto, sus componentes y el tipo de estos, haciéndolos únicos y accesibles de alguna forma.
- **Gestión de Cambios en la Configuración:** Se controlaron las versiones y entregas del producto y los cambios que se produjeron en él a lo largo del ciclo de vida.

- **Planificación de la Gestión de Configuración:** Se definieron las políticas, estándares y procedimientos que se van a utilizar para gestionar la configuración en el transcurso del proyecto.
- **Gestión de Versiones:** Se mantuvo un registro histórico de las diferentes versiones por las que pasaron los componentes del producto, que permitiría la recuperación de cualquiera de ellas.
- **Generación de Informes de Estado:** Se elaboraron informes acerca del estado de los componentes del producto y de las solicitudes de cambio, recogiendo estadísticas acerca de la evolución del producto.

Interrelación entre subprocesos de la GCS:

Los subprocesos que componen la GCS se encuentran fuertemente relacionados entre sí, conclusión a la que se arriba al estudiar la descripción de dichos procesos efectuada por diversos autores y normas (FEBLES. 2001; IEEE 1998; ISO 1995; JIMÉNEZ 2001). La figura 6 puede servir para constatar este hecho, a partir de tres de los procesos aplicados (Identificación de la Configuración, Control de Versiones y Control de Cambios).

A continuación se desglosan las actividades de cada uno de los procesos que constituyen el modelo propuesto, las cuales fueron aplicadas durante todo el proceso de desarrollo.

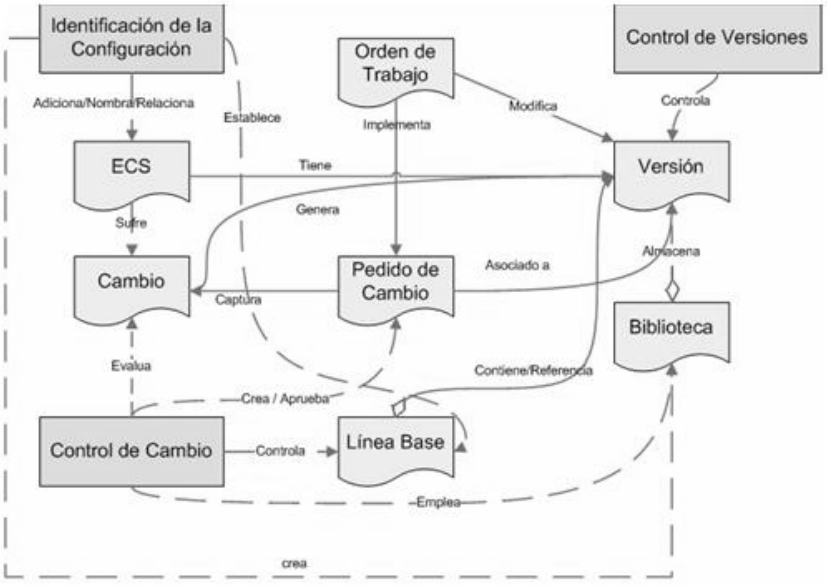

Figura 6: Relación entre subprocesos de la Gestión de configuración.

2.1.1. Identificación de la Configuración.

A través de la identificación de la configuración se identificaron y asignaron nombres significativos y consistentes a todos y cada uno de los elementos que formaron parte del producto software, en cada fase de su desarrollo, es decir, a cada uno de los Elementos de Configuración del Software (ECS). A continuación se muestra el procedimiento a seguir para realizar el proceso de Identificación de la Configuración en el proyecto IPC, el mismo también se encuentra gráficamente en la figura 7:

1. Se determina los componentes que serán ECS y cuando pasan al control de configuración.
2. Se establece el esquema de identificación de los ECS determinados.
3. Se establece las relaciones que van a tener los ECS.
4. Se proponen componentes para convertirse en ECS. (esta actividad es realizada por los desarrolladores)

5. Se autoriza a que el componente pase a ser ECS.
6. Se establecen las relaciones del nuevo ECS con el resto de los ECS del repositorio.
7. Se coloca la versión del ECS en el repositorio del proyecto. La versión pasa al estado estable o base**.**
8. Se actualiza la Línea Base del proyecto para cada modificación del repositorio.
9. Se genera un documento que contiene la identificación de la configuración, que corresponde al estado actual de la Línea Base en dicho punto del proceso de desarrollo.
10. Se publica este documento y se distribuye.

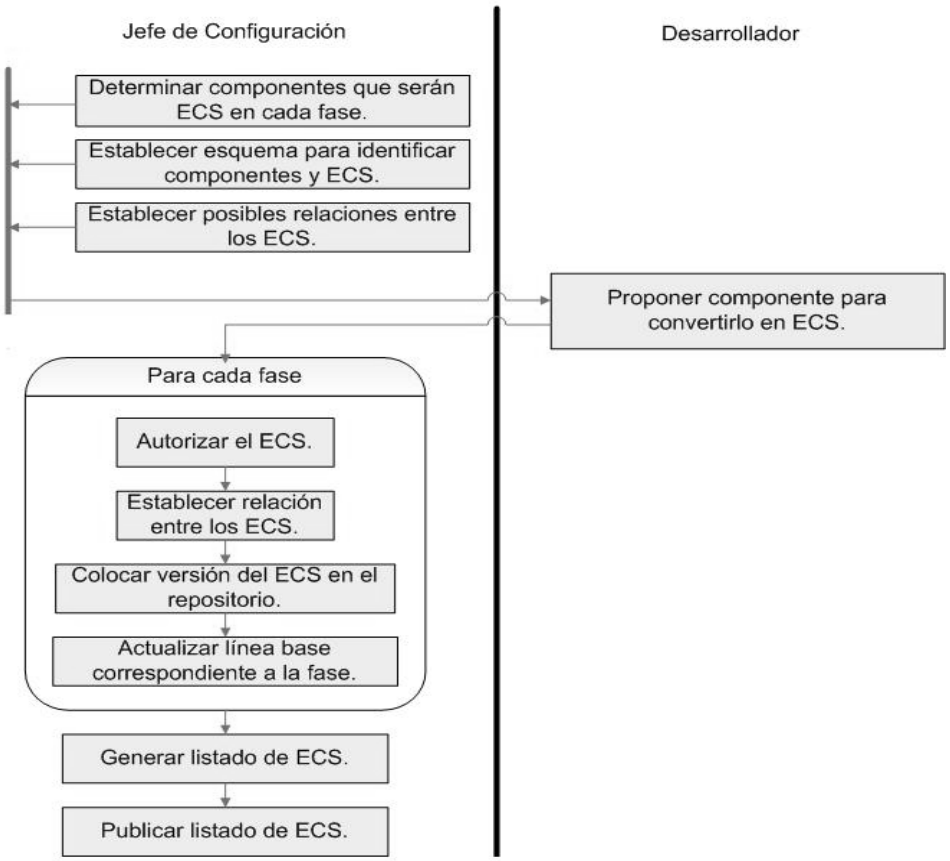

Figura 7: Procedimiento a seguir para realizar el proceso de Identificación de la Configuración en el proyecto IPC

2.1.1.1. Selección de los Elementos de Configuración:

Los ECS establecidos en el proyecto IPC fueron los siguientes:

Entregados por la ONE:

Formato duro:

1- Encuesta de ingresos y gastos.

2- Tratamiento de problemas estacionales.

3- Índice de Precios al Consumidor (IPC).

4- Cuatro opciones para calcular los movimientos de precios.

5- Índices de precios y ajustes de calidad.

6- Metodología del cálculo del IPC.

7- IPC en moneda nacional según grandes grupos y subgrupos de consumo.

8- Canasta del IPC.

9- Sondeos de precios en el mercado informal, Mayo 2006.

10-Tabla de IPC, Mercado informal, Octubre 2006.

11-Modelos 7001, 7002, 7003 y 7004 del SIEN.

12-Sistema integral para el cálculo del IPC.

Formato digital:

1. Metodología del Índice de Precios al Consumidor.doc
2. Estructura de los gastos.doc
3. CANASTA.DBF

4. CLASIF.DBF
5. CODEST.DBF
6. DPA.DBF
7. ENCUESTA.DBF
8. A070103.DBF
9. DV070103.DBF
10. ES070103.DBF
11. SP070103.DBF

Documentación producida durante los flujos ingenieriles:

Inicio	**Requerimientos**
	1. Especificaciones de requisitos de software. 2. Modelo del Negocio. 3. Especificaciones de CU del sistema.
Elaboración	**Diseño**
	1. Modelo de diseño. 2. Realización de Casos de Uso de Diseño. 3. Modelo datos. 4. Diseño de la Base de datos. 5. Descripción de la arquitectura (vista del modelo de diseño). 6. Modelo de despliegue. 7. Descripción de la arquitectura (vista del modelo de despliegue).
Construcción **Transición**	**Implementación**
	1. Modelo de implementación. 2. Descripción de la Arquitectura. (vista modelo

	implementación). 3. Código fuente del programa. - Código objeto y ejecutable. - Manual de usuario.
	Pruebas
	1. Casos de Uso de Prueba. 2. Evaluación de pruebas. 3. Plan de pruebas.

Documentación producida durante la Gestión de configuración:

- Plan de Gestión de la Configuración.
- Elementos de Configuración de Software.
- Plan de Gestión de Cambios.
- Control de versiones.

Documentación producida durante la Gestión de Proyecto:

- Plan de Iteración.
- Plan de Aceptación del Producto.
- Plan de Administración de Riesgos.
- Plan de desarrollo del Software.

2.1.1.2. Definición de las relaciones en la configuración:

Para tomar una decisión sobre la implementación o rechazo de un cambio, el administrador solicitará mostrar el grafo de relaciones, del mismo se puede mostrar el estado de los ECS que dependen del cambio y las personas a quienes están asignados, lo cual da mayor idea del impacto que podrá tener el cambio a partir de la cantidad potencial de personas involucradas.

Se consideró que las relaciones que mayormente se establecen, entre los ECS del proyecto IPC, y que resulta factible capturar, son las de dependencia y composición, ya que son las más propias a establecerse entre los ficheros fuentes y ejecutables entregables de las aplicaciones.

Grafo de relaciones entre ECS.

El grafo de relaciones entre ECS es una estructura de datos que almacena ECSs (nodos) y relaciones entre estos (arcos) en un formato de Grafo dirigido y etiquetado, las relaciones tendrán etiquetas en dependencia del tipo, se propone emplear “utiliza”, como marca para las relaciones de dependencia y “contiene” para las relaciones de composición. Para el grafo se propone una representación visual, compatible con el lenguaje UML, en el que los ECSs serán representados como componentes UML y las relaciones como dependencias de UML, etiquetadas con el valor “utiliza” o “contiene” según corresponda la relación.(ver figura 8)

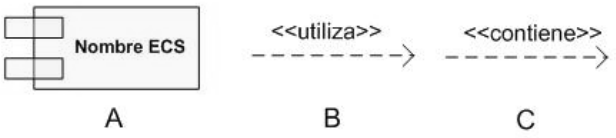

Figura 8: Elementos empleados en el Grafo de Relaciones. A) ECS en formato visual icono. B) Relación de dependencia etiquetada “utiliza”. C) Relación de composición etiquetada “contiene”.

Con el objetivo de lograr una visión general y menos engorrosa, ya que se cuenta con una cantidad considerable de ECS, se agruparon dichos elementos en paquetes que subdividen el grafo en otros más pequeños que colaboran entre si, estos paquetes representan los principales flujos de trabajo utilizados en el proyecto IPC (Ver figura 9). Esta figura muestra las relaciones entre los paquetes establecidos, como se puede apreciar, la Gestión de Proyectos se aplica a los paquetes restantes, al igual que en el caso de la Gestión de Configuración, o sea, que no son procesos aislados, sino que están conectados entre si.

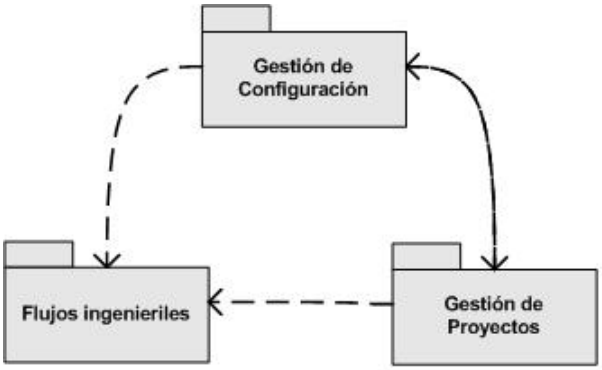

Figura 9. Distribución de los ECS y sus relaciones en paquetes funcionales. Dichos paquetes presentan relaciones entre ellos.

La distribución por paquetes ha quedado de la siguiente forma:

1. Flujos ingenieriles:

Este paquete agrupa todas las relaciones existentes entre los ECS que están vinculados con los distintos flujos ingenieriles definidos por RUP, entre los cuales se encuentran los ECS definidos en el levantamiento de requisitos, diseño, implementación, prueba, como se muestra en la figura 10.

2. Gestión de Proyecto:

Contiene las relaciones constituidas entre los ECS que intervienen en el proceso de desarrollo del Proyecto (ver figura 11). Primeramente es establecido el Plan de Iteraciones el cual distribuye el trabajo por iteraciones y por consiguiente indica la documentación que se ha de llenar, los ECS que surgen, a los que hay que darle refinamiento y seguimiento. Este plan está relacionado con el Plan de desarrollo del Software, que describe como se debe construir el producto, este a su vez está relacionado con el Plan de Aceptación del Producto que establece las pautas para que el producto sea aceptado y el Plan de Administración de Riesgos que se establece para controlar los riesgos relacionados con el Software.

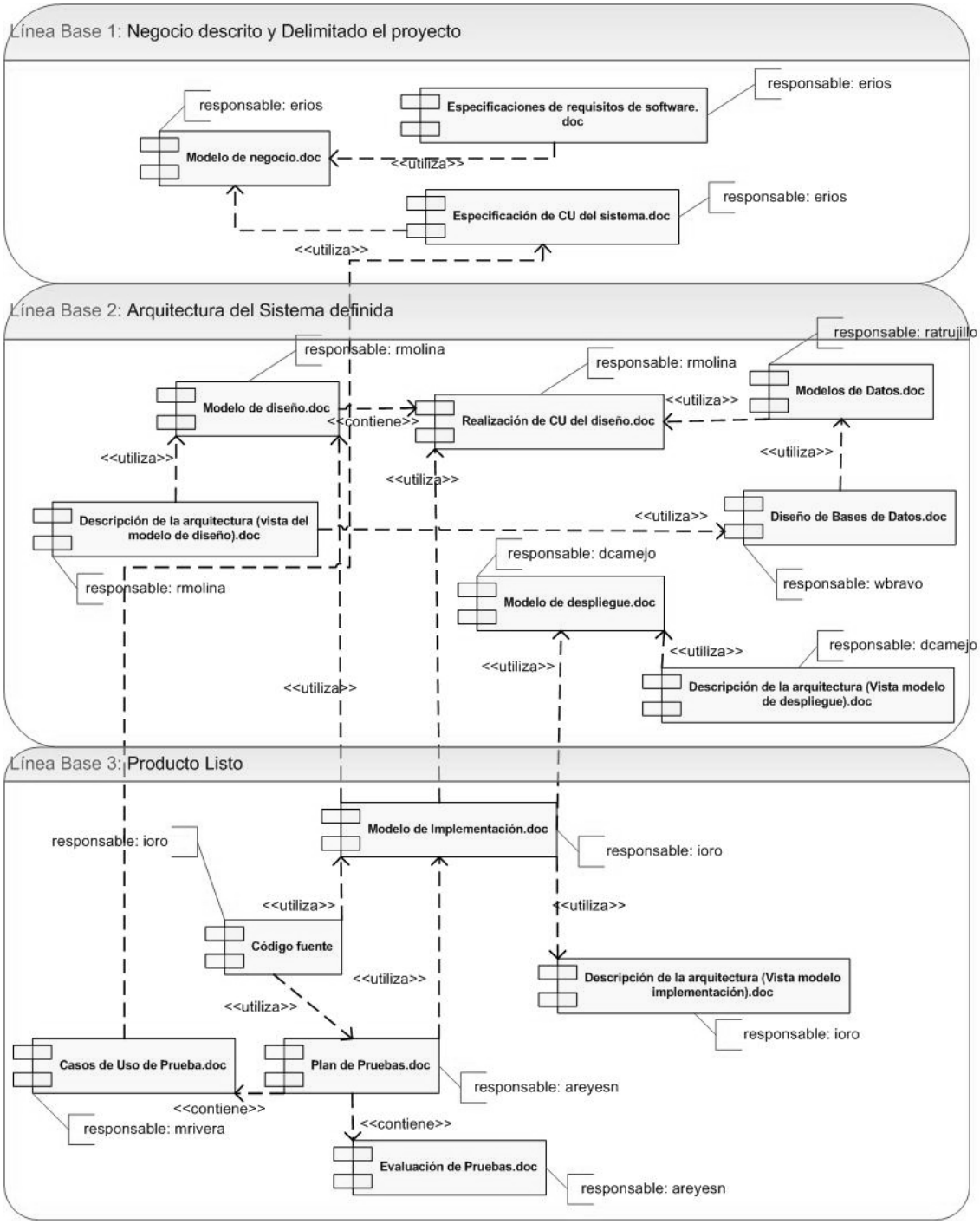

Figura 10: Dependencias entre los ECS del paquete de Flujos ingenieriles.

Estos planes dependen unos de otros, sin un plan de desarrollo de software no hay producto que pueda ser aceptado, sin un plan de iteraciones el trabajo es incoherente por lo tanto el software puede estancarse y sin un buen control de los riesgos no se puede asegurar el producto.

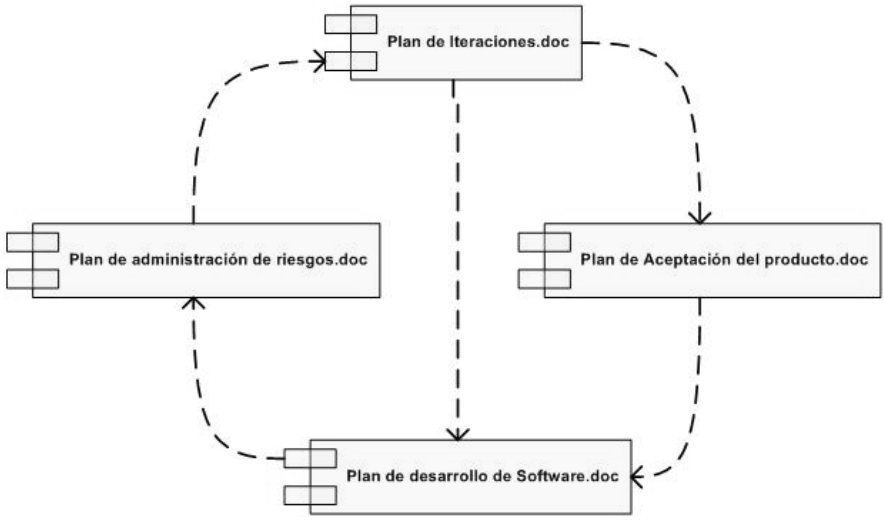

Figura 11. Descripción de las relaciones de dependencias entre los ECS seleccionados en el proceso de Gestión de Proyecto.

3. Gestión de Configuración de Software:

Este paquete recoge las relaciones entre los ECS definidos en la Gestión de Configuración del Software (ver figura 12). Se establece el Plan de Gestión de Configuración del Software, este menciona los ECS y con ellos como entrada establece la línea base del proyecto, recoge información de la herramienta de control de versiones y gestiona los cambios realizados o por realizar a los ECS, entre otras actividades. Este plan depende para su gestión del documento de Elementos de Configuración de Software, estos elementos pueden sufrir cambios a lo largo de todo el ciclo de desarrollo del proyecto por lo que se hace necesario establecer un Plan para la Gestión de los Cambios. Un ECS no solo puede cambiar sino que también puede contar con varias versiones, la información de las versiones y de los cambios van a dar al Plan de GCS.

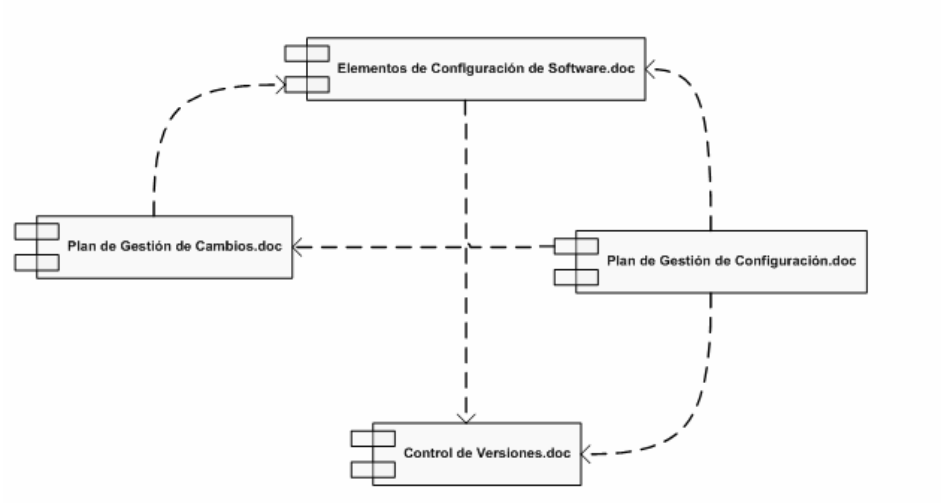

Figura 12. Descripción de las relaciones que se establecen entre los distintos ECS determinados en el proceso de Gestión de Configuración del Software.

Estos grafos de relaciones han sido publicados en el trac del proyecto, lo cual permite que todos los miembros del proyecto tengan visibilidad en cuanto a la repercusión de un cambio sobre un ECS determinado.

2.1.1.3. Definición de un Esquema de Identificación:

Fue necesario decidir también cuál sería el método a utilizar para identificar de forma unívoca cada Elemento de Configuración del Software, es decir se estableció un Esquema de Identificación que permitió etiquetar cada uno de los ECS. El esquema de identificación obtenido proporciona la siguiente información:

Código del ECS: [Proy] - [Tipo ECS] _ [#ECS].

Versión: Número de la Versión.

Línea Base: Identificación de la línea base a la que pertenece.

Nombre del ECS: Nombre del elemento de configuración de software.

Descripción del ECS: Breve descripción del ECS.

Autor/es del ECS: Nombre del autor/es del ECS.

Datos de los autores: Dirección de correo electrónico de los autores.

Fecha de Creación: dd/mm/aaaa.

Tipo de ECS: Tipo de ECS.

Localización: Dirección donde se puede consultar el ECS.

Los posibles tipos de los ECS pueden ser:

- BD: Base de datos.
- Doc: Textos.
- Zip: Ficheros compactados.
- F: Fuentes.
- R: Ficheros del Rational.
- Ex: Ejecutable.

Los números de las versiones comienzan por la 1.0 e incrementará según se desarrolle el producto.

A la hora de definir el código de identificación de los ECS se optó por el sistema de codificación significativo, debido a que es muy ventajoso, ya que no es necesario mirar en el registro de asignación de códigos para saber de que ECS se trata.

Este esquema es utilizado por el documento **“**Elementos de Configuración de Software”, el cual forma parte de los artefactos a entregar, el mismo puede ser consultado en el anexo 3.

2.1.1.4. Definición y establecimiento de Líneas Bases:

En el proyecto IPC se establecieron líneas base al finalizar determinadas fases del ciclo de vida de desarrollo, con el objetivo de identificar los resultados de las tareas realizadas durante las fases y asegurar que se han completado las mismas.

La primera línea base es “Negocio descrito y Delimitado el proyecto”, como su nombre lo indica, en ella se describe el negocio y se delimita el proyecto describiendo sus alcances. Entre los ECS que la componen se encuentran:

1. Especificación de requerimientos de software.
2. Modelo del Negocio.
3. Especificaciones de CU del sistema.

La siguiente línea base, “Arquitectura del Sistema definida”, define la Arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. Entre los ECS que la componen se encuentran:

1. Modelo de diseño.
2. Realización de Casos de Uso de Diseño.
3. Modelo datos.
4. Diseño de la Base de datos.
5. Descripción de la arquitectura (vista del modelo de diseño).
6. Modelo de despliegue.
7. Descripción de la arquitectura (vista del modelo de despliegue).

La siguiente línea base “Producto Listo” obtiene como resultado un producto listo para su utilización que está documentado y tiene un manual de usuario, la misma incluye los siguientes elementos de configuración de software:

1. Modelo de implementación.

2. Descripción de la Arquitectura. (vista modelo implementación).
3. Código fuente del programa.
   - Código objeto y ejecutable.
   - Manual de usuario.
4. Casos de Uso de Prueba.
5. Evaluación de pruebas.
6. Plan de pruebas.

2.1.1.5. Definición y Establecimiento de bibliotecas de software:

Se estableció una **biblioteca de trabajo**, en la cual se almacenan los ECS que se están desarrollando por los trabajadores del proyecto. En esta biblioteca los ECS no están aún en estado estable, y los cambios son informales. Una vez que se termine de implementar la versión del ECS, la misma debe ser probada, para que así se esté seguro de que los cambios se realizaron de manera correcta. Esto conlleva a que se mueva la versión del ECS a la **biblioteca de integración**, donde será probada por un trabajador del proyecto. Si la prueba resultó exitosa, y se concluyó la implementación de la versión del ECS, este puede moverse a la **biblioteca de Soporte del proyecto**, donde empezará a estar bajo un proceso de control de cambio interno y semi-formal. Esto quiere decir que la versión del ECS presentará un mayor nivel de estabilidad. A esta biblioteca, solo tienen permisos de escritura los miembros del Comité de Control de Cambios, mientras que tienen permisos de lectura el resto del equipo de desarrollo. Una vez que se esté seguro que la versión de un ECS, tenga calidad para ser entregado a un cliente o para que se pueda distribuir en el mercado, dicha versión del ECS se mueve para la **biblioteca maestra**, la cual se creó con similares permisos de lectura-escritura a la Biblioteca de Soporte al Proyecto, donde se encuentran los ECS bajo un proceso de control de cambio formal y estricto. Como **Biblioteca de Backup** los responsables de la gestión de configuración efectúan salvas del repositorio con frecuencia de dos días hacia un disco duro externo, con el objetivo de disminuir la probabilidad de pérdida

de la información. No obstante, en un futuro, el repositorio realizará salvas automáticas hacia servidores de salvas establecidos por la dirección de producción de la facultad.

2.1.2. Control de cambios en la configuración:

En el proyecto productivo IPC se establecieron varios niveles de control de cambios:

- Control de cambios informal.
- Control de cambios semi-formal.
- Control de cambios formal.

Tanto en el proceso formal como semi-formal surge una nueva figura en el proyecto, el Comité de Control de Cambios (CCC), el cual está integrado por Disnier Alberto Camejo Domínguez, actual jefe de proyecto, quien será el encargado de: comprobar la correcta utilización de los procedimientos de control de cambios por parte de los implicados en los mismos y ser partícipe durante la toma de decisiones referidas a un cambio. Además, al ser este el responsable principal de la arquitectura del sistema también será el encargado de: estimar técnicamente las solicitudes de cambio, determinando el impacto sobre el resto de los componentes del sistema y teniendo en cuenta el costo que implique la realización del mismo.

Maykel Medrano Najara y Yanisleidy Barroso Benítez, quienes como responsables de la GCC serán los encargados de: recibir y priorizar las peticiones de cambios, aprobar o rechazar los cambios que no necesiten pasar por el CCC (cambios locales). Informar y hacer seguimiento de las solicitudes aprobadas, hacer reportes del estado de los cambios, finalmente publicar los cambios y hacer llegar la nueva versión del producto.

No obstante se estableció que el resto del equipo de desarrollo debe estar presente a la hora de decidir sobre la realización o no de un cambio, aportando las ventajas y desventajas que

el mismo traería consigo. De esta forma se tendría un mayor número de argumentos y se fomentaría aun más el trabajo en equipo.

Ya que no existe ningún estándar para el control informal o interno de los cambios, se realizaron algunas recomendaciones como son:

- Consultar con aquellos que estén vinculados de forma directa al ECS que va a ser modificado, o informar a los mismos luego de realizar dicho cambio.
- Mantener las “buenas prácticas de programación” durante la realización de algún cambio sobre el código.

A continuación se muestra el proceso a seguir para hacer un cambio formal sobre una línea base en el proyecto IPC, dicho proceso se encuentra representado gráficamente en la figura 13.

1. Se informa una incidencia producto de un problema detectado o de un cambio en los requerimientos (surge de esta forma el informe de incidencia).

2. Los encargados de la Gestión de Configuración evalúan el informe de incidencia y realizan el informe de cambio, el cual contiene información respecto al esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funciones del sistema y el coste estimado del cambio.

3. Se genera la Solicitud de Cambio y se envía, junto con el Informe de Cambio, al Comité de Control de Cambio.

4. Si se considera que el cambio es beneficioso se genera una Orden de Cambio, que describe el cambio a realizar, las restricciones que se deben respetar y los criterios de revisión. En caso contrario la Solicitud de cambio pasa a denegada y se continúa con la actividad 10.

5. Se asigna la Orden de Cambio a alguno de los ingenieros de software para que este se encargue de llevarlo a cabo.
6. Dar de baja al objeto a cambiar en la Biblioteca de Soporte al Proyecto.
7. Se realiza el cambio, entrando en un proceso de seguimiento y control.
8. 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.
9. El objeto se devuelve a la Biblioteca de Soporte al Proyecto.
10. Se notifica el resultado a quien originó el cambio y al resto del equipo de desarrollo.

En un proceso semi-formal, se considera que para el caso del proyecto IPC no hay necesidad de generar el informe de incidencia y el informe de cambio, ya que el equipo de desarrollo es pequeño y está en constante intercambio de ideas, pero sí debe realizarse la solicitud de cambio, la orden de cambio, la evaluación del cambio y su seguimiento.

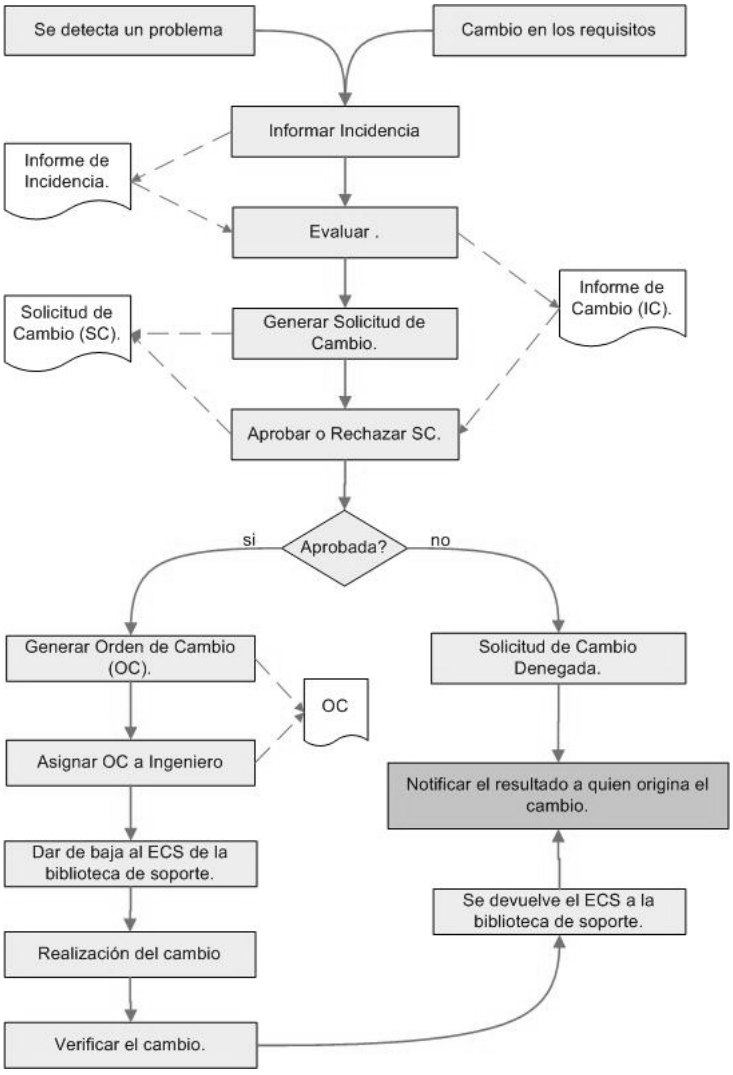

Figura 13: Diagrama del proceso a seguir para hacer un cambio sobre una línea base en el proyecto IPC:

Al definir este proceso fue necesario también definir mecanismos para:

1. Solicitar cambios sobre los Elementos de Configuración.
2. Analizar y evaluar el impacto de las solicitudes de cambio.
3. Aprobar o rechazar las solicitudes de cambio.
4. Controlar la realización de los cambios aprobados.

Mecanismo definido para solicitar cambios sobre los ECS:

En el proyecto IPC se elaboró un formulario para llevar a cabo el proceso de solicitud de cambio sobre los ECS, el mismo se muestra en el anexo 4. También se hace necesario solicitar una mejora, informar de un problema o deficiencia detectada durante una auditoría, una prueba o el uso del sistema. Por esta razón se tuvo en cuenta la creación de un formulario para estos casos, dicho formulario se nombró "Informes de Incidencias" y puede ser consultado en el anexo 5.

Mecanismo definido para analizar y evaluar el impacto de las solicitudes de cambio:

Una vez solicitado un cambio, se hace necesario analizar y evaluar el impacto del mismo sobre el resto de los ECS, los posibles efectos secundarios que pueda ocasionar, la repercusión sobre otras funciones del sistema, el esfuerzo técnico requerido, cantidad de personas involucradas en el cambio, costo y tiempo de desarrollo que implicaría la aceptación de esta petición de cambio. Todos estos criterios para la evaluación de las solicitudes de cambio son tenidos en cuenta por los gestores de configuración.

Los autores se apoyan en las técnicas de estimación de COCOMO para calcular la cantidad de personas involucradas en el cambio, el costo y el tiempo que implicaría llevar a cabo el mismo. El impacto del cambio sobre el resto de los ECS se determina mediante las relaciones que existen entre los ECS (ver epígrafe 2.1.1.2).

Mecanismo definido para aprobar o rechazar las solicitudes de cambio:

Los resultados de la evaluación del impacto de las solicitudes de cambio son presentados como un informe de cambios al Comité de Control de Cambios, quien es el encargado de generar para cada cambio aprobado, una orden de cambio. La orden de cambio describe el

cambio a realizar, las restricciones que se deben respetar y los criterios de revisión y de auditoría.

Algunos de los criterios tenidos en cuenta por el CCC para aprobar o rechazar las solicitudes son los siguientes:

- Valor del cambio para el proyecto.

- Retorno de la inversión.

- Tamaño.

- Complejidad.

- Impacto sobre el rendimiento del producto.

- Recursos disponibles para efectuar el cambio.

- Relación con otros cambios ya aprobados y en progreso.

- Relación con las políticas de la empresa (satisfacción del cliente, etc.)

- Existencia de alternativas, etc.

Mecanismo definido para controlar la realización de los cambios aprobados:

Una vez aprobado un cambio debido a un problema se debe realizar un seguimiento del mismo. Para el proyecto IPC se elaboró una tabla con ayuda de la herramienta Microsoft Excel, por medio de la cual se controlan los cambios aprobados, la misma presenta los siguientes campos:

- Problema.
- Descripción.
- Prioridad.

- Estado.
- Causa.
- Solución.
- ECS Afectados.
- Notificante.
- Responsable.
- Fecha Notificación.
- Fecha Resolución.
- Fase en que se originó.
- Fase en que se detectó.

2.1.3. Plan de Gestión de Configuración:

A continuación se muestra la estructura del Plan de Gestión de Configuración del proyecto IPC, el mismo se muestra íntegramente en el anexo 6.

1. Introducción.

    - Propósito
    - Alcance
    - Resumen

2. Gestión de Configuración de Software.

    - Organización, responsabilidades e interfaz.
    - Herramientas, entorno e infraestructura.

3. Programa de la Gestión de la Configuración.

    Identificación de la configuración

    - Métodos de identificación.
    - Líneas base del proyecto.

Control de configuración y de cambios.

- Proceso de solicitud y aceptación de cambios.
- Comité de Control de Cambios (CCC).

Estado en la configuración.

- Almacén del proyecto y procedimiento de liberación.
- Reportes.

4. Hitos.

5. Entrenamientos y recursos.

2.1.4. Control de versiones en la configuración.

El control de versiones en el proyecto IPC se llevó a cabo mediante la herramienta Subversion (svn), la misma fue instalada en un servidor dentro del propio laboratorio del proyecto, siguiendo la propuesta de la dirección de producción de la facultad, con el fin de evitar que cualquier desconexión en la red que aísle al laboratorio de otros puntos de la universidad afecte al proyecto, ya que la red dentro del laboratorio se mantiene y por tanto se puede tener acceso al servicio, además, que cada líder de proyecto pueda administrar su repositorio sin necesidad de terceras personas, entre otras.

Características del servidor:

- Debian como Sistema Operativo.
- SVN para la gestión de código.
- Trac integrando Gestión de Configuración y Gestión de Proyecto.

Subversion utiliza un modelo del tipo *copiar-modificar-mezclar*. El sistema de control de versiones a menudo ayuda con la mezcla, pero en última instancia el propio desarrollador es el responsable de hacer que esto suceda correctamente (COLLINS-SUSSMAN *et al.* 2004). Por este y otros motivos, en el proyecto IPC se han establecido una serie de “buenas

prácticas", con el objetivo de minimizar los problemas y confusiones que conllevan a una pérdida de tiempo, entre las mismas se pueden apreciar:

- Antes de comenzar a trabajar en la carpeta de trabajo local, realizar un "update" sobre la misma para que esté actualizada con las últimas versiones existentes en el repositorio.
- Siempre, al realizar un "commit" poner un mensaje, logrando así una mayor visión entre las diferentes versiones presentes en el repositorio.
- Evitar realizar muchas modificaciones sobre un ECS sin antes mezclarlos con los existentes en el repositorio, puesto que mientras más los desarrolladores se alejan de la versión original, mayor es la dificultad para lograr una mezcla exitosa.

2.1.5. Generación de Informes de Estado:

Registros:

Los registros que se realizaron durante el proyecto IPC son los siguientes:

Registro de Solicitudes de cambios: El tipo de información que se mantiene acerca de cada solicitud de cambio es la recogida a través del formulario de Solicitud de Cambio. Este registro se muestra en el anexo 7.

Registro de Cambios: Muestra la información recogida a través del Informe de Cambio, la Orden de Cambio y el proceso de Gestión de Problemas (ver anexo 8).

Registro de Incidencias: El tipo de información que se mantiene acerca de cada incidencia es la recogida a través del Informe de Incidencia. Este registro se muestra en el anexo 9.

Registro de Modificaciones sobre documentación: cuyo formato se puede apreciar en el anexo 10.

Registro de Releases y Variantes: Su objetivo es describir la composición y estado de una versión liberada del producto, el mismo se puede apreciar en el anexo 11.

Informes:

Los informes que se realizaron durante el proyecto IPC son los siguientes:

1. **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.

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

3. **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.

4. **Informe de modificaciones:** Es un resumen de las modificaciones que se han efectuado en el producto software durante un determinado período de tiempo.

2.2. Organización y contenido del proyecto.

El proyecto IPC tiene dos módulos: Interfaz de Usuario (IPC-GUI) y Acceso a Datos (IPC-DB). Además, el producto que se le va a entregar al cliente "Sistema para el Cálculo de Índice de Precios al Consumidor" (IPC); por lo que se establecieron 3 carpetas en el repositorio: IPC-GUI, IPC-DB e IPC, cumpliendo esta última, la función de Biblioteca Maestra del proyecto. Cada uno de estos elementos de configuración tiene una vida relativamente independiente. Esto no significa que no haya dependencias entre esos módulos, sino que esa dependencia es a nivel de interfaz (API) y lo interno del módulo puede progresar sin afectar a los otros módulos.

Cada una de estas carpetas contiene 3 subcarpetas: trunk, tags, y branches. (ver figura 14)

En el trunk es donde ocurre el desarrollo de la iteración actual para ese módulo, dicha carpeta cumple la función de biblioteca de trabajo. En la carpeta tags, se almacena el código

de cada versión al final de la iteración, la misma es utilizada como Biblioteca de soporte, a la cual solo tendrán permiso de escritura los miembros del Comité de Control de Cambio. En las carpetas branches se hacen desarrollos y mantenimientos de versiones anteriores, también se pueden realizar variantes alternativas de módulos que no deben mezclarse directamente en el trunk. En la figura 15 se puede apreciar el flujo normal de los datos en la estructura utilizada.

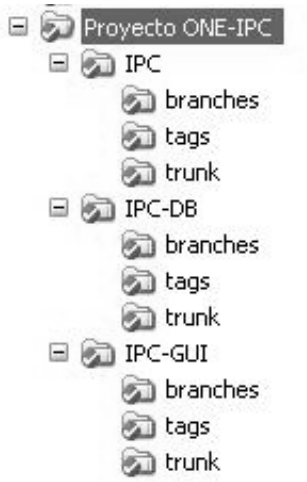

Figura 14: Estructura del repositorio.

Una vez terminada una iteración, se realizará una copia de la versión entregada hacia la carpeta de trabajo del encargado de la realización de las pruebas, con el fin de realizar las mismas, si estas resultan exitosas, se mueve dicha copia hacia la carpeta tags. El nombre de la copia debe ser correspondiente con la numeración de versión. De esta manera se tendrá la carpeta IPC-DB/tags/version-1 que contiene todo el código del módulo IPC-DB en la versión 1; al igual que IPC-GUI/tags/version-1, etc. De esta forma, se puede ahora seguir modificando las carpetas trunk en cada módulo para programar lo necesario para la segunda iteración. Si el cliente solicita un arreglo menor de la versión 1, no es conveniente usar el código del trunk que está en constante modificación, y puede que aún no funcione bien. Pero se cuenta con el código de la versión 1 en los tags. Por tanto es fácil obtenerlo, ver el

problema reportado y arreglarlo. El procedimiento, sin embargo, no es modificar directamente el código dentro del tags/version-1, sino hacer una copia de mantenimiento dentro de la carpeta branches. Se copia el código de tags/version-1 para branch/version-1.1, y es en esta carpeta donde son efectuadas todas las correcciones que han sido solicitadas por el cliente. Una vez arreglados todos los problemas, son realizadas nuevamente las pruebas (todas las iniciales: es decir, se cumplen los requisitos para la versión 1, y las necesarias para confirmar que se han corregido los errores). Se crea el tags/version-1.1 a partir del branch de mantenimiento, se le entrega el producto al cliente, y se elimina el branches/version-1.1. Si aparecen más errores se debe hacer el mismo proceso para la versión 1.2.

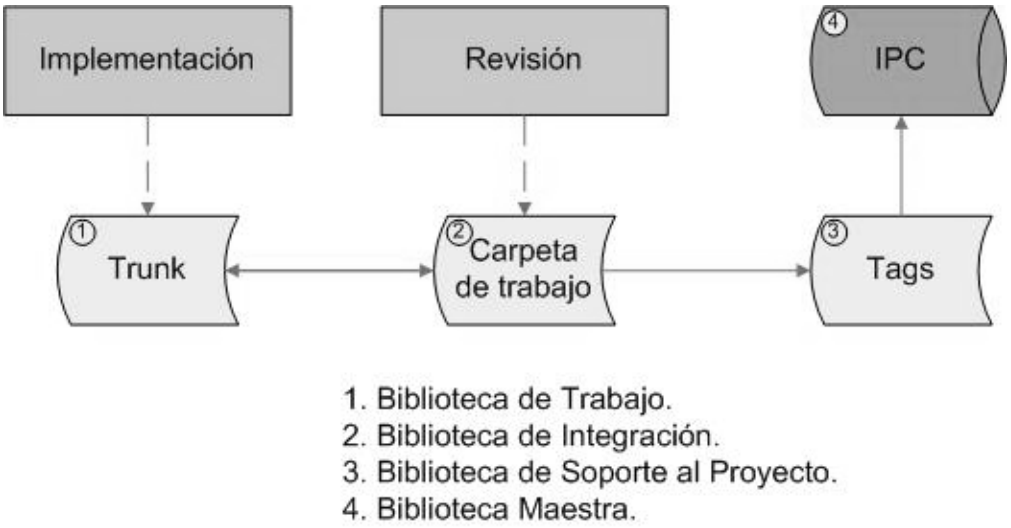

Figura 15: Flujo normal de los datos en la estructura utilizada del repositorio.

Todo esto puede ocurrir mientras se está desarrollando la segunda versión del producto (segunda iteración del proyecto), que incluye los nuevos requisitos, y no hay riesgo de mezclar el código nuevo todavía inestable y en progreso, con el código de la versión anterior. En cuanto al expediente del proyecto se optó por utilizar la wiki del trac para plasmar dicho contenido, ya que cuando se almacena un documento en formato binario dentro del Subversion, se pierde la facilidad que ofrece este de intentar mezclar los cambios que hagan

varias personas sobre el documento. La estructura del expediente de trabajo quedaría como se muestra en la figura 16.

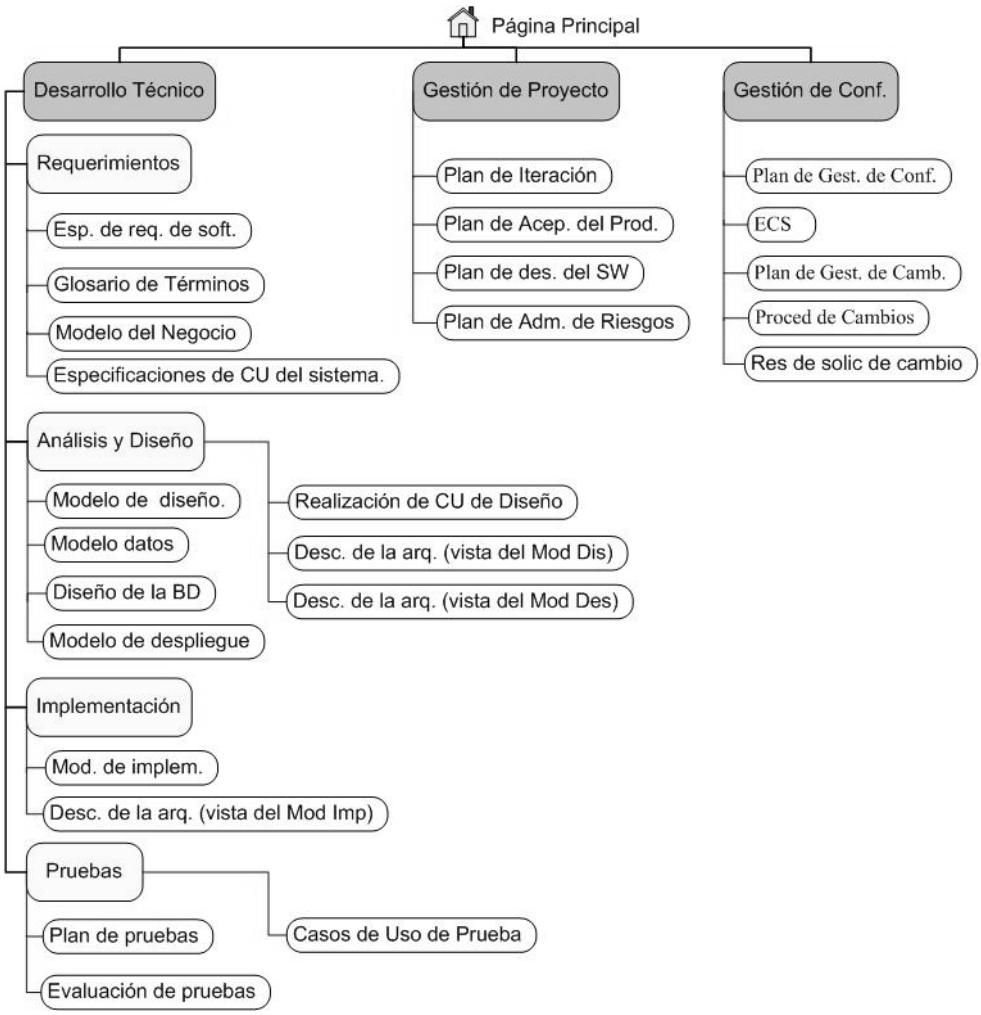

Figura 16: Mapa del trac (estructura del expediente de trabajo).

CAPÍTULO III: Análisis de los resultados obtenidos.

En el presente capitulo se verifica si fueron cumplidos los objetivos trazados en el primer capítulo, hasta el momento de culminar la redacción del documento. Para ello se realiza un análisis de los resultados obtenidos en cada una de las actividades aplicadas al proyecto IPC. Además, se comprueba si los artefactos obtenidos han cumplido su función. Concluyendo con una comparación entre el estado inicial del proyecto y el actual alcanzado.

Los objetivos que se proponen cumplir con la aplicación de la GCS en el proyecto productivo IPC son, como se pudo apreciar en el capítulo 2, establecer y mantener la integridad de los productos generados durante el proceso de desarrollo del proyecto software a lo largo de todo el ciclo de vida del producto, evaluar y controlar los cambios sobre ellos, es decir, controlar la evolución del sistema software y facilitar la visibilidad y la información sobre el producto. Estos se alcanzan realizando cada uno de los procesos de la Gestión de Configuración propuestos en el modelo a aplicar con sus correspondientes actividades. A continuación se brindan los resultados obtenidos de cada una de las actividades desarrolladas a lo largo de todo el ciclo de desarrollo del proyecto.

3.1. Identificación de la Configuración: Resultados.

A través de la identificación de la configuración se identificaron y asignaron nombres significativos y consistentes a todos y cada uno de los elementos que formaron parte del producto software, en cada fase de su desarrollo, es decir, a cada uno de los Elementos de Configuración del Software (ECS).

Se estableció un procedimiento para realizar el proceso de Identificación de la Configuración, el cual sirvió de guía a los miembros del proyecto a la hora de solicitar algún ECS.

Para llevar a cabo la selección de los ECS se separaron en ECS distintos las funcionalidades distintas, de esta forma se logró minimizar el impacto de los cambios, lo que permitió realizar una adecuada selección de los elementos de configuración estableciendo qué tipos de ECS

conformarán el proyecto, o al menos cuáles de ellos resulta de interés controlar, lográndose así que elementos no relevantes queden excluidos del control de configuración desde las etapas tempranas. Aunque siempre se podría modificar esta selección.

Se obtuvo el artefacto Elementos de configuración de software, uno de los principales resultados de esta actividad mediante el cual fue posible identificar de forma unívoca cada uno de los ECS y relacionar los mismos con sus responsables, además de la ubicación donde se encuentran, es decir permitió etiquetar cada uno de los Elementos de Configuración del Software. Otro de los resultados relevantes está vinculado a la actividad definición de las relaciones en la configuración, en la cual se realizó un grafo de relaciones entre los ECS seleccionados. El mismo ha proporcionado una mayor visión de la repercusión de cualquier cambio, ya que puede determinarse fácilmente cuales otros ECS se verían afectados con dicho cambio y las personas a quienes estarían asignados estos, ofreciendo mayor idea del impacto que podrá tener el cambio a partir de la cantidad potencial de personas involucradas. Todos estos factores han sido de gran importancia a la hora de determinar si es factible la autorización del cambio.

La definición y establecimiento de líneas base ha jugado un papel fundamental puesto que las mismas se tomaron como puntos de partida para desarrollos futuros, además de que sirvieron para comprobar el cumplimiento de determinados hitos durante el proceso de desarrollo de software y, una vez alcanzados los mismos, los cambios sobre los ECS que la conforman se realizaron bajo un proceso formal de control de cambios.

Por otro lado la definición y establecimientos de bibliotecas software fueron de gran ayuda, puesto que se obtuvo una mayor organización en cuanto al nivel de acceso de cada miembro del proyecto a los diferentes ECS, facilitándose de esta forma la tarea de Control de Cambios y la contabilidad de estado.

3.2. Gestión de los Cambios de la Configuración: Resultados.

Con la realización de esta actividad, se proporcionó un mecanismo riguroso para controlar los cambios, dentro del cual se logró la formación del Comité de Control de Cambios, el cual está integrado por el jefe de proyecto y los dos gestores de configuración. No obstante la determinación de que el resto del equipo de desarrollo estuviese presente a la hora de decidir sobre la realización o no de un cambio proporcionó grandes ventajas, ya que de esta forma se logró que cada uno de ellos participase en la toma de decisiones sobre la aprobación o no de las Solicitudes de Cambios y confecciones de ordenes de trabajo, además de formar parte del estudio que se hace de los posibles impactos que se pueden generar. Lográndose de esta forma que cada integrante del proyecto comprendiera la importancia de generar Solicitudes de Cambios con la calidad requerida, así como clasificarlas según su nivel de prioridad, mostrar los ECS que serían utilizados en estos cambios y dar una breve descripción sobre que se lograría si estos cambios se realizaban. De esta forma se le facilitaba al CCC decidir si era factible e importante para el Proyecto realizar o no los cambios generados y hacer las órdenes de trabajo. Así todo el trabajo se lograría de manera simultánea, logrando la integración del esfuerzo de trabajo realizado por los desarrolladores con el del CCC.

También se establecieron los niveles de control de cambio que se tendrían en cuenta durante el desarrollo del software y los procedimientos para realizar los mismos, de esta forma los desarrolladores cuentan con una guía a seguir una vez que surja la necesidad de realizar un cambio sobre algún ECS, evitando que se lleve a cabo un indisciplinado y caótico desarrollo del software. Dentro de los procedimientos establecidos también se definieron mecanismos para solicitar cambios sobre los Elementos de Configuración de Software, analizar y evaluar el impacto de las solicitudes de cambio, aprobar o rechazar las solicitudes de cambio y para controlar la realización de los cambios aprobados. Estos mecanismos contienen toda la información necesaria para la toma de decisiones durante el proceso de control de cambios, por lo que ayudaron en gran medida durante la puesta en práctica del mismo, ya que organizan todo el proceso, definiendo las pautas para el correcto desempeño de este.

3.3. Plan de Gestión de la Configuración: Resultados.

Mediante este plan se definieron políticas, estándares y procedimientos que fueron utilizados en el transcurso del proyecto IPC. Con la aplicación del mismo se permitió organizar las tareas en tiempo, destinar recursos, asignar responsabilidades y posteriormente poder darle seguimiento a lo planificado.

Para el control de la configuración fue clave, pues es una actividad que se realiza durante todo el ciclo de vida del proyecto, incide en todos los componentes del proyecto e involucra a todos los miembros del equipo de trabajo.

3.4. Gestión de Versiones en la Configuración: Resultados.

El proceso de Control de Versiones está inmerso en cada una de las actividades de Gestión de Configuración que se han desarrollado en el Proyecto. Mientras se aplicaba la Gestión de Configuración de Software al proyecto IPC, teniendo como base el Sistema de Control de Versiones Subversión, se obtuvieron resultados tales como versiones estables a partir de la estrategia definida.

Gracias a que Subversión presenta un modelo de versionado del tipo Copiar-modificar-fusionar en ningún momento se dieron problemas de pérdidas de tiempo puesto que todos los desarrolladores lograron trabajar simultáneamente y luego fusionar el resultado final.

Cada desarrollador pudo tener su copia local del repositorio y trabajar con ella desde cualquier sitio de la universidad. Los permisos se ordenaron por carpetas, lográndose de esta forma un control del acceso de cada desarrollador a las diferentes bibliotecas que se encuentran dentro del repositorio.

Tanto para los programadores, como el resto de los desarrolladores no fue difícil adaptarse al uso del Subversión a pesar de valerse de Microsoft Visual Studio.Net para realizar la implementación del software, gracias a la utilización del AnkSvn, el cual pudo integrarse al Subversion y realizar todas las operaciones básicas de trabajo con el repositorio. También

fue de gran ayuda la utilización del tortoise, por medio del cual se lograron utilizar todos los comandos necesitados para interactuar con el Subversion desde la interfaz de Windows.

La utilización del sistema de control de versiones propuesto por la dirección de producción de la facultad fue de gran importancia para la obtención de buenos resultados ya que gracias a que se mantuvo el servidor en el mismo laboratorio donde se desarrolla el proyecto, no se vio afectado el proceso de desarrollo por fallos en la red puesto que la misma dentro del laboratorio se mantiene y por tanto se puede tener acceso al servicio. Otro resultado fue que se ganó en tiempo ya que fue eliminada la dependencia de un administrador para la operación de los proyectos, de esta forma, a la hora de otorgar permisos sobre el repositorio, los encargados de la gestión de configuración realizaban la tarea directamente, sin depender de terceras personas. También vale destacar que la principal ventaja que ha proporcionado la propuesta realizada por la facultad es la facilidad con que se pueden realizar salvas automáticas hacia servidores de salvas, disminuyéndose considerablemente de esta forma la probabilidad de pérdida de la información.

La decisión de que todo el expediente de proyecto se encontrase publicado en el trac, fue de gran importancia ya que se logró una mayor visión de la estructura de este, la información estaba disponible en una web para cualquier persona ajena al proyecto que se interesase por el funcionamiento del mismo y se limitaba el repositorio solo al código fuente de la aplicación, de esta forma se logró un mayor rendimiento de este, puesto que cuando se almacena un documento en formato binario dentro del Subversion, se pierde la facilidad que ofrece este de intentar mezclar los cambios que hagan varias personas sobre dicho documento.

3.5. Generación de Informes de Estado: Resultados.

Esta actividad ha estado presente durante todo el proceso, cada vez que se le asignaba una nueva identificación a un elemento de configuración o que se actualizaba, se creaba una entrada para la generación de Informes de Estado de la Configuración (IEC). Cada vez que el Comité de Control de Cambios aprobaba un cambio, se generaba una entrada en el IEC.

Los resultados alcanzados con la generación de Informes de Estados han sido muy satisfactorios ya que se ha logrado que cada integrante del proyecto, así como usuarios y directivos se mantengan al tanto del estado de la configuración y su evolución, es decir se ha logrado divulgar en tiempo todas las informaciones pertinentes (las condiciones en que se encuentran las versiones, los cambios generados por quien, porqué y cuando) hacia todas las partes involucradas. De esta manera se evitó la duplicación de esfuerzos para llevar el sistema a un estado anterior, que se cayeran en los mismos errores una y otra vez, se ayudaron a encontrar las causas de los fallos y brindar la posibilidad de ver el estado del progreso del proyecto, así como mantener la continuidad del mismo en caso de que surgiera algún contratiempo.

3.6. Análisis crítico una vez aplicado el modelo propuesto.

Actualmente en el proyecto no existe dificultad en cuanto al dominio y uso de las herramientas de control de versiones tanto por parte de los encargados de la gestión de configuración como por el resto de los integrantes del equipo de desarrollo, puesto que a medida que el proyecto ha ido avanzando, los miembros del mismo se han familiarizado con este, dándole un mayor uso, de esta forma se han percatado de lo cómodo y amigable que resulta el trabajo con esta herramienta de control de versiones.

El expediente de proyecto, fue conformado y publicado en el trac, de esta forma se ha logrado una mayor visión de la estructura del mismo, mediante un mapa del sitio, y a pesar de ser una estrategia del proyecto que en sus inicios impactó a los miembros del mismo, actualmente ha sido aceptada y llevada a cabo por la totalidad del equipo de desarrollo, no obstante el trac del proyecto no se ha explotado en su totalidad, puesto que la planificadora no utiliza las funcionalidades que para este rol ofrece el sitio.

En estos momentos se puede decir que cada miembro del proyecto es capaz de generar Solicitudes de Cambios con la calidad requerida, así como clasificarlas según su nivel de prioridad, mostrar los ECS que se verían implicados en estos cambios y dar una breve descripción sobre los beneficios que proporcionaría la realización de estos cambios, lo que

demuestra que cada miembro del equipo de desarrollo está conciente de la importancia que tiene el correcto desempeño de esta actividad, lo que indica que se ha creado una cultura de esta disciplina en los miembros del equipo de desarrollo. Gracias a esto se consiguió realizar una correcta documentación y control de los cambios durante el proceso de desarrollo del software, lográndose además una mayor visión del Comité de Control de Cambio a la hora de tomar una decisión sobre la aceptación o no de dichas solicitudes y una mejor programación de las tareas por parte de la persona encargada de la planificación.

También, gracias a la aplicación del indicador Estado de los Pedidos de Cambio, se ha logrado determinar en que medida el equipo de proyecto está dando respuesta a las solicitudes de cambio, detectar la aceptación del equipo a las peticiones de cambio, así como cuantas peticiones de cambio ya han sido resueltas y entregadas. Además, brinda la posibilidad de determinar el número de peticiones de cambio que se encuentran en cada estado de los posibles. Al graficar el resultado de este indicador en la fase de construcción del proyecto IPC, se obtienen los resultados que se muestran en la figura 17. Como se puede apreciar, el equipo de desarrollo está dando respuesta a la mayoría de las solicitudes de cambio, ya que de cinco solicitudes, fueron aprobadas tres, una se encuentra en desarrollo y solo una fue rechazada.

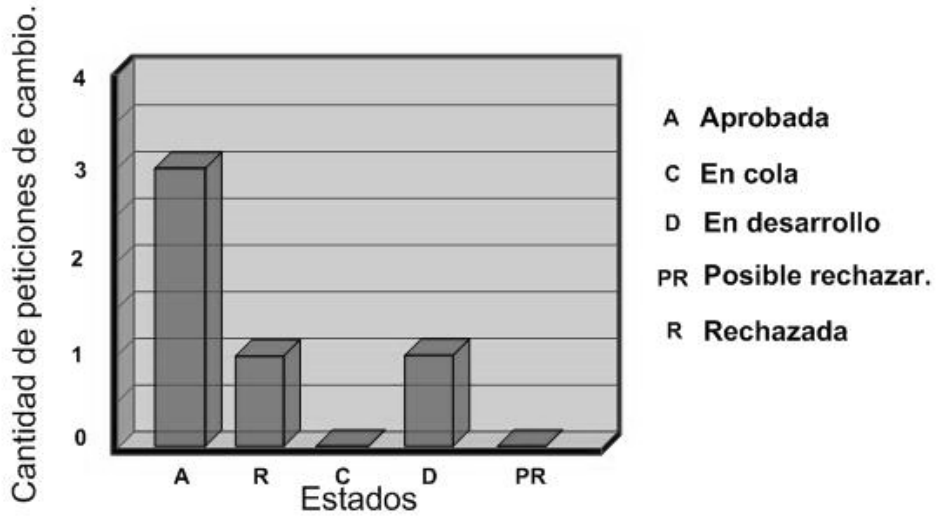

Figura 17: Gráfico de estado de las solicitudes de cambio.

Conocer el estado de los Elementos de Configuración de Software es de máxima prioridad tanto para los encargados de la Gestión de Configuración, como para el líder del proyecto. Este fue otro de los indicadores que se tuvo en cuenta ya que brinda la posibilidad de tener elementos a la hora de asignar una tarea asociada a un Elemento de Configuración de Software o saber cuantos de estos están siendo modificados (en uso).

Para este indicador se grafica el total de ECS en cada estado y se plasman en un gráfico de pastel de forma tal que la información para el análisis sean encontrados de forma cómoda y clara como se muestra en la figura 18.

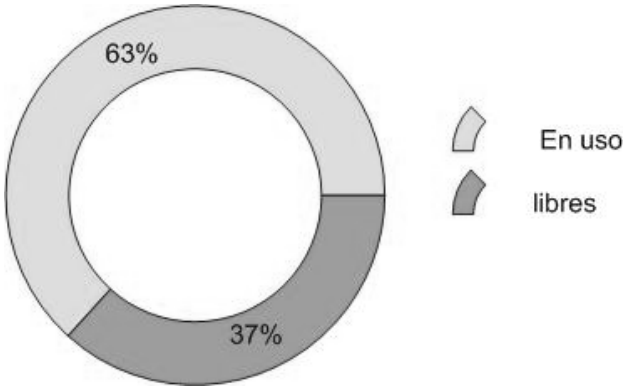

Figura 18: Gráfico del estado de los elementos de configuración.

CONCLUSIONES.

- La Gestión de Configuración de Software juega un papel primordial en el proceso de desarrollo de software. Es una disciplina que se encuentra presente en los estándares y modelos de calidad más utilizados a nivel mundial como son CMMI, IEEE e ISO.
- El modelo sienta las bases para poder garantizar mejor comunicación entre los involucrados, calidad en la ejecución de los procesos y en el producto final.
- La aplicación práctica del modelo ha permitido definir los procedimientos de control de configuración de software en el proyecto IPC, crear disciplina de trabajo asociada a los cambios, las versiones y los ECS, además de almacenar los resultados para la toma de decisión en futuras tareas o proyectos.
- La Identificación de la Configuración logró dar una dimensión del Proyecto seleccionando el número adecuado de elementos de configuración a controlar, así como establecer sus relaciones, accesibilidad e identificación unívoca.
- Se logró mantener a los usuarios, desarrolladores y directivos del proyecto al tanto del estado de la configuración y su evolución mediante los Informes de Estados.
- Se controlaron los cambios producidos a lo largo del ciclo de vida del proyecto con el nivel de responsabilidad y calidad requerida a través de la formación del CCC, establecimiento de los niveles de control de cambio, Solicitudes de Cambios, confecciones de órdenes de trabajo y definiciones de mecanismos a seguir.
- Mediante el uso de la herramienta Subversion se obtuvieron versiones estables del producto en todo su ciclo de vida y se gestionó el expediente del proyecto mediante el uso del trac.
- Se definió la Gestión de Configuración de Software como disciplina de gestión y control de proyectos productivos, así como también se seleccionaron y justificaron los procesos puestos en práctica en el proyecto IPC a través un estudio exhaustivo del estado del arte.

- La elaboración de un conjunto de mecanismos, procedimientos y plantillas fue un factor indispensable para el buen desenvolvimiento y desarrollo de cada uno de los procesos concebidos según las características específicas del Proyecto, obteniéndose un grupo de entregables fundamentales para el futuro desarrollo de la disciplina en el software.

Luego de observar los resultados logrados en cada una de las actividades realizadas y hacer un análisis comparativo a través de indicadores entre la situación de la GC del proyecto en un comienzo y la lograda hasta el momento de redacción del presente trabajo, se puede decir que se le ha dado cumplimiento a los objetivos propuestos. No obstante teniendo en cuenta que es una disciplina que apenas se ha comenzado ha explotar tanto en la universidad como en el país, queda mucho camino por recorrer, muchos aspectos en los cuales profundizar y perfeccionar, ¿y por que no?, aportar y adaptar a las condiciones y necesidades económicas de nuestro país. La Industria Cubana del Software se ha propuesto alcanzar un lugar relevante dentro del mercado mundial. Esta meta se vería seriamente comprometida si desde horas tempranas no se empieza a dedicar esfuerzo a la realización de actividades de GCS. Esto es solamente un paso de avance, pero en la medida que se gane en experiencia se alcanzará una GCS de elevado nivel y con ello el éxito de la industria cubana del software.

RECOMENDACIONES.

1. Aplicar el modelo y herramientas propuestas para la gestión de configuración en todos los proyectos productivos y evaluar sus resultados.
2. Enriquecer el modelo con la inclusión de métricas que permitan mejorar de forma incremental los procesos y transmitan mayor cantidad de información, así como poder tomar acciones correctivas a tiempo por parte de los directivos y los lideres de proyectos, llevando a escalas superiores los resultados internos de los equipos de desarrollo.
3. Implementar una herramienta la cual permita realizar grafos de relaciones, donde se muestren de forma automática la historia de los cambios y las relaciones entre los diferentes ECS sujetos a cambios en el proyecto.
4. Profundizar en el estudio de la herramienta para lograr resolver las desventajas que presenta actualmente la misma, cómo son:
    - Configurar el apache para el usuario del dominio.
    - La obligatoriedad del uso de lectura para todos los usuarios que accedan al repositorio.
    - Buscar una solución para poder realizar un verdadero control de versiones sobre los diagramas que se encuentran en el expediente de trabajo.
5. Definir e implementar un sistema de indicadores que permita calcular el impacto del cambio en un proyecto de forma automática.

BIBLIOGRAFÍA.

Bibliografía referenciada:

BROWN, N. Little Book of Configuration Management, AIRLIE, 1998.

CMU. The Strategy for CMM v2 Revision and Release [technical report], Carnegie Mellon University, 1996.

COLLABNET. AnkhSVN: A Subversion addin for Microsoft Visual Studio .NET, 2006a. [Disponible en: http://ankhsvn.tigris.org/

---. Subclipse 2006b. [2007]. Disponible en: http://subclipse.tigris.org/

---. TortoiseSVN, 2006c. [2007]. Disponible en: http://tortoisesvn.tigris.org/

COLLINS-SUSSMAN, B.; B. W. FITZPATRICK, et al. Control de versiones con Subversion. 2004. 286 p.

EDGEWALL. Trac. Integrated SCM & Project Management., 2006. [2007]. Disponible en: http://trac.edgewall.org/

ESTRADA, A. F. and S. Á. CÁRDENAS. La Gestión de Configuración y el desarrollo de Sofeware en las universidades. Una experiencia práctica.: Revista Cubana de Educación Superior, 2005. No. 1.

FEBLES., A. Case Corporativo para el proceso de control de cambios. La Habana, 2001. p.

HARVEY, K. Summary of the SEI Workshop on Software Configuration Management. Software Engineering Institute Carnegie-Mellon University, 1986.

HUFF, C. Spinning a Web: Publishing the SEI Software Configuration Management Research on the World Wide Web, Software Engineering Institute Carnegie-Mellon University, agost 1994.

HUMPHREY., W. S. Introduction to the Team Software Process (sm)". Addison-Wesley, 2000. p.

IBM. Rational ClearCase, International Business Machine., 2005. [Disponible en: http://www.ibm.com/software/awdtools/clearcase/

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

---. "IEEE Standard for Software Quality Assurance Plans". Std. 730-1998, IEEE Computer Society, 1998.

ISO. ISO 10007 Quality management – Guidelines for configuration management., ISO, 1995.

JACOBSON IVAR, M. G. A. P. J. Software Reuse: Architecture, Process, and Organizartions for Busisness Success. Addison- Wesley, 1997. p.

JIMÉNEZ, A. D. A. La Gestión de la Configuración de Software. chile, 2001.

MARTÍNEZ, R. D. and E. G. SOCA. ConfigCASE 3.0 HERRAMIENTA DE APOYO A LA GESTIÓN DE CONFIGURACIÓN. PROPUESTA ARQUITECTÓNICA., Instituto Superior Politécnico "José Antonio Echeverría", 2006. p.

MICROSOFT. Que es nuevo en Visual SourceSafe 2005. [2007]. Disponible en: http://msdn2.microsoft.com/en-us/library/ms181369(VS.80).aspx

PAULK, M. C. Capability Maturity Model for Software, Carnegie Mellon University, 1993.

PRESSMAN, R. S. Ingenieria del SoftWare. Un enfoque practico. 5. La Habana, Editorial Felix Varela, 2005. p.

RATIONAL, Ed. IBM RedBooks. Software Configuration Management A Clear Case for IBM Rational ClearCase and ClearQuest UCM, 2004. 0738491594

---. Rational Unified Process, IBM, 2003.

SOURCEFORGE. VA Software Corporation, 2007. [Disponible en: http://www.sourceforge.org

UCI. Manual VSS, 2005. [2006]. Disponible en: http://is.uci.cu/bibliografia/063.doc

Bibliografía consultada:

APPLETON, B. and S. P. BERCZUK. *Software Configuration Management Patterns. Effective Teamwork, Practical Integration.*, Addison-Wesley, 2002.

BACH, J. The highs and lows of change control. (computer software management) *(computer software management)*, 1998, v31(n8): p113(113).

BAR, M. and K. FOGEL. *Open Source Development with CVS*. DUNTEMANN, J., 2003. 3.

BOZNAK, R. G. A paradigm shift in product development management. (putting greater emphasis on product definition) *(putting greater emphasis on product definition)*, 1991, v23(n3): p14(13).

DART, S. *Concepts in Configuration Management Systems*, Software Engineering Institute Carnegie-Mellon University.

---. *Spectrum of Functionality in Configuration Management Systems*, Software Engineering Institute Carnegie-Mellon University, dic 1990.

DART, S. A. *The Past, Present, and Future of Configuration Management*, Software Engineering Institute Carnegie-Mellon University, 1992.

DER, A. V. *Software Configuration Management. ICSE Workshops SCM 2001 and 2003 Toronto, Canada, May 2001 and Portland, OR, USA, May 2003. Selected Papers.*, 2003.

ELAM, D. Document Management Explained *AIIM E-DOC*, 2005, 19(3): 16-16.

ESTÉVEZ, I. P. *MÉTRICAS PARA EL CONTROL DE PROYECTOS DE SOFTWARE* Habana, INSTITUTO SUPERIOR POLITÉCNICO “JOSÉ ANTONIO ECHEVERRÍA”, 2002. p.

ESTRADA, A. F. and S. Á. CÁRDENAS. *La Gestión de Configuración y el desarrollo de Software en las universidades. Una experiencia práctica.*: *Revista Cubana de Educación Superior*, 2005. No. 1.

FEBLES, A. and I. P. ESTÉVEZ. *Medir el proceso de control de configuración, ¿una utopía para la Industria Nacional de Software?*: *Revista Ingeniería Informática*, 2003.

FORTE, G. Configuration management. (five program development, computer-aided software engineering and text management software packages that handle configuration management) (Tools Fair: Out of the Lab, Onto the Shelf) (Product Announcement) *(five program development, computer-aided software engineering and text management software packages that handle configuration management) (Tools Fair: Out of the Lab, Onto the Shelf) (Product Announcement)*, 1992, v9(n3): p79(71).

FOWLER, G. S.; J. E. HUMELSHINE*, et al.* Tools and techniques for building and testing software systems, 1992, v71(n6): p46(16).

HAMILTON, D. Guaranteed to Get Attention *School Library Journal*, 1987, 33(8): 53.

HEYDON, A.; R. LEVIN*, et al. Software Configuration Management using Vesta*, 2006.

HUMPHREY, W. S. *Managing the Software Process*. Addison- Wesley, 1989. p.

---. *Personal Software Production*. Addison Wesley, 1998. p.

HUMPHREY., W. S. *A discipline for software engineering*. Addison- Wesley, 1995. p.

HUNT, A. and D. THOMAS Three legs, no wobble.(Three methods to ruin a software project) *(Three methods to ruin a software project)*, 2004, 21(1): 18(22).

JONASSEN, A. M. *Configuration Management Principles and Practice*, Addison-Wesley, 2003.

KAN, S. H. *Metrics and Models in Software Quality Engineering*. Addison- Wesley, 2000. p.

KEYES, J. *Software Configuration Management*, CRC Press, 2004.

LLOPIS, N. Perforce 2004.2.(Product/Service Evaluation) *(Product/Service Evaluation)*, 2004, 11(11): 10(13).

LOURIDAS, P. Version control.(concurrent versions systems) *(concurrent versions systems)*, 2006, 23(1): 104(104).

MACVITTIE, L. VERSION CONTROL, WITH INTEGRITY -- MKS' Source Integrity VCS stands out for its comprehensive platform support, ease of management and strong security, but we'd like a license to kill ... its license.(Software Review)(Evaluation) *(Software Review)(Evaluation)*, 2001: 69.

NAVARRO, J. A. F. *Entorno Unificado para la Gestión de Configuración de Software.* Habana, INSTITUTO SUPERIOR POLITÉCNICO “JOSE ANTONIO ECHEVERRIA”, 2006. p.

NUSENOFF, R. E. and D. C. BUNDE. *A Guide Book and Spedsheet Tool for a Corporate*

Metrics Program. Holanda, 1993. p.

PERSSON, A. P. I. C. U. A. A. *Implementing and Integrating Product Data Management and Software Configuration Management.*, Artech House, 2003.

RAMIREZ, K. Promoting and Managing Projects using Visual SourceSafe.(Microsoft Visual SourceSafe)(Product Support) *(Microsoft Visual SourceSafe)(Product Support)*, 2000, 15(5): 116.

SHAH, S. and R. PLESS Protect your source code: keep track with visual SourceSafe.(Feature: development) *(Feature: development)*, 2006, 8(4): 19(14).

SOURCEFORGE. *VA Software Corporation*, 2007. [Disponible en: http://www.sourceforge.org

SPINELLIS, D. Version control systems.(advantages of the systems in software development) *(advantages of the systems in software development)*, 2005, 22(5): 108(102).

STEINHOLTZ, B. and K. WALDEN Automatic identification of software system differences, 1987, v13(n4): p493(495).

STEVENS, C. A. and S. GAMBRELL Managing change with configuration-value management, 1993, v25(n5): p54(55).

WILLIAMS, C. C. and J. K. HOLLINGSWORTH Automatic mining of source code repositories to improve bug-finding techniques, 2005, 31(6): 466(415).

WILSON, M. Visual SourceSafe and ".NET" interoperability.(Tech Tips) *(Tech Tips)*, 2003, 2(7): 30(31).