Universidad de las Ciencias Informáticas
Facultad 3

Título: Procedimiento para el análisis de factibilidad de cambios en el proceso de desarrollo de software

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias Informáticas

Autor(es): Rosa Nelvia Álvarez Chang

Yeslaine Cortina Blanco

Tutor(es): Ing. Miguel Ángel Martínez Acosta

Junio 2006

DECLARACIÓN DE AUTORÍA

Nosotras declaramos que somos las únicas autoras de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autores:	Tutor:
Yeslaine Cortina Blanco	Miguel Ángel Martínez Acosta
Rosa Nelvia Álvarez Chang	

Pensamiento

Lo fundamental es hacer algo nuevo cada día y luego perfeccionar lo que se ha hecho el día anterior.
Ché

Agradecimientos

A mi abuelita Julia por demostrarme que soy importante para ella y compartir mi gran sueño.

A mi tía, por tener tanta confianza en mí, por haber estado en momentos muy difíciles de mi vida y por ser la madre que necesité. "Te quiero no es suficiente".

A mis loquitas: Ane, yuny, Yari, lin y también a luna por ser las mejores amigas de este mundo y aguantarme todas mis malcriadeces. Gracias por hacer que los días fueran mejores a pesar de todo y gracias por comprenderme.

A mi hermanita Yordanis, a Migue y Maura, no se que me habría hecho sin ustedes.

A tito que llegó para convertirse en un gran amigo.

A Javier Morales (el loko), Didier, Osmany, Henrry , Javier Ramírez , Mindrey y Yoa por haber hecho una diferencia en mi vida y al resto de mis amistades que han contribuido a la realización de este sueño.

Yeslaine Cortina Blanco

Agradecimientos

Agradezco a mi mami, por estar siempre, por tus consejos, por tu protección, porque eres única.

A mi papá por ser mi respaldo, mi apoyo, por confiar en mí.

A mi hermanita, por ser lo mejor que tengo en la vida, por ser mi amiga, porque te quiero.

A mi abuelo Chino, a mi abuela Nelvia, por ser tiernos conmigo, por ser especiales.

A mis abuelos que ya no están (Chichí, María), que me hacen falta.

A Mamá Cuca, se que hubiera querido estar.

A mis tíos Isabelita, Dely, Fermín, Pedro. Por ser dulces, atentos, complacientes.

A todos mis primos. Espero estar para ustedes.

A Rolando, por ser mi respiro, mi luz, por estar en el momento justo.

A Mami Elba, A Papi Julio, A toda su familia, que ha sido la mía también.

A Orlando, gracias.

A Maité, Ovi, Javy, Magui, Gessy, Sony, Ricky, Edy, Sandra, Ingrid, Viole, mis amigos de siempre.

A Yuri, Yoa, Any, Lisyen, a todos los amigos de la UCI.

A todos los que contribuyeron a ser quien soy.

Rosy

Dedicatoria

A los que no están, A los que quisieron estar y no pudieron, A los que han sabido estar para mí.

….¨A quien quiero más que a nadie¨…...

Rosa Nelvia Alvarez Chang

Dedicatoria

Al esfuerzo de toda una vida.

A mis abuelos quienes les gustaría haberme visto graduada.

Yeslaine Cortina Blanco

Resumen

La ingeniería de software proporciona los métodos, herramientas y procedimientos para construir eficientemente un software. Aún así no existe un paradigma que defina como hacer un producto en específico por lo que durante el ciclo de desarrollo de este surgen muchos cambios. Estos cambios se pueden producir en cualquiera de los flujos de trabajo y sobre cualquiera de los elementos de configuración. Cuando el proyecto se encuentra en una fase inicial de desarrollo o se produce una afectación, en pocos y no complejos elementos de configuración, es menos engorroso determinar el impacto que este causará sobre el producto o sobre los demás ECS. En cambio, cuando estos son múltiples o el proyecto se encuentra en una fase avanzada, predecir la factibilidad del cambio a realizar se torna complicada. En la actualidad la facultad 3 carece de un procedimiento que ayude a prever el impacto de los cambios y que auxilie a tomar la decisión de si es o no conveniente realizarlo, la obtención de este es el propósito del presente trabajo., realizándose un análisis y valoración de las herramientas que brindan soporte al proceso de control de cambios a nivel mundial.

Palabras claves: Gestión de Configuración de Software, análisis de impacto de los cambios, elementos de configuración de Software, factibilidad de los cambios.

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

El desarrollo de la industria Nacional de software, debido a la gran perspectiva económica que tiene el país al respecto, es una tarea de prioridad. A pesar de todos los esfuerzos hechos los resultados no cubren las expectativas. Según (Lic. Ailyn Febles Estrada Dra. Sofía Alvarez Cárdenas): la productividad es baja, la cantidad real de recursos a consumir - en tiempo principalmente- es casi impredecible y el trabajo realizado casi nunca tiene la calidad y profesionalidad requerida. Los proyectos están excesivamente tarde y los beneficios que pudieran obtenerse al utilizar los mejores métodos e instrumentos en las distintas etapas no se detectan en este medio de desarrollo.

En la industria de software actual existe una tendencia al crecimiento del volumen y la complejidad de los productos, lo que exige una mayor preparación por parte de las empresas. La obtención de un producto con calidad implica en estos días mejoras en los métodos y procedimientos utilizados en el proceso de desarrollo de software, sin embargo entre los principales problemas detectados en la industria de software cubana se encuentra la aplicación inadecuada de las técnicas de Ingeniería de Software y la no existencia de procedimientos que permitan planificar y controlar los procesos, por lo que las etapas resultan muy difíciles o no son representativas de la realidad.

Esto es lo que (Lic. Ailyn Febles Estrada Dra. Sofía Alvarez Cárdenas) denominan **la falta de industrialización de la producción de software** es decir, la tendencia a no usar procedimientos estandarizados a la hora de producir software que permitan planificar y controlar y, que aunque no se considera la única deficiencia si se cree que es una de la más importantes.

El perfeccionamiento de la industria del Software sólo puede garantizarse si las empresas realizan sus procesos de manera que la productividad y la calidad sea alta.

Las Herramientas de Ingeniería de Software brindan apoyo a los métodos y constituyen un soporte al desarrollo. Estas herramientas pueden ser manuales, semiautomáticas o totalmente automatizadas. La tendencia actual es que este apoyo sea llevado a cabo por herramientas informáticas que brinden una solución integral al problema del desarrollo del software. (Rancán, 2003) Estas son las herramientas CASE (Computer Aided Sofware Engineering) que se puede traducir como Ingeniería de Software Asistida por Computadora. Existen en el mercado diferentes herramientas de este tipo que permiten automatizar parte o todo el proceso de desarrollo. Estas herramientas son útiles en la construcción de los artefactos que genera el proceso pero una complicación se produce cuando surge la variable ¨ cambio ¨. Cuando el cambio se produce en fases tempranas del proceso reajustar el proyecto puede no ser difícil, pero cuando se está en una fase avanzada es precisa la introducción de procedimientos y técnicas para la estimación de las implicaciones que esto traería consigo y mantener así la integridad del producto, ya que aunque las herramientas proporcionan como automatizar las actividades no solucionan todas los problemas que el cambio implica.

Una idea poco realista sería evitar el cambio pero, la primera ley de la ingeniería de software plantea: “Sin importar en cual momento del ciclo de vida estemos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida del proyecto”.

Independientemente de la metodología que se escoja a lo largo de su ciclo de vida los productos evolucionan, desde su concepción, la captura de requisitos inicial hasta la puesta en producción del mismo, y posteriormente desde el inicio del mantenimiento hasta su retiro se van realizando una serie de cambios, tanto en el código como en la documentación asociada.

Estos cambios se pueden producir en cualquiera de las fases de desarrollo del software y sobre los diferentes artefactos que se generan. La Gestión de Configuración del Software es la disciplina encargada del control de la evolución de los productos de software y de facilitar el mantenimiento del sistema, aportando información para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un cambio.

.Actualmente en la Universidad de las Ciencias Informáticas (UCI) la Gestión de Configuración se ha introducido como materia en el proceso docente educativo y su uso como parte del proceso de elaboración y mantenimiento de software está dando sus primeros pasos, introduciendo en cada proyecto productivo el rol de Gestor de Configuración en aras de garantizar un mayor nivel de calidad en la formación del nuevo ingeniero informático y que le permita a la nueva industria de software ser competitiva.

Por lo que a raíz de los problemas existentes en la industria cubana que no están muy lejos de ser los de la industria de software actual se ha llegado a plantear la siguiente:

Situación problémica

Durante el proceso de desarrollo de software se realizarán cambios muchas veces dados por:

- Nuevos requisitos del negocio o condiciones que dictan los cambios en las características del producto o en las normas comerciales.
- Nuevas necesidades de los clientes que demandan la modificación de los datos producidos por un sistema basado en computadora.
- Cambios en las prioridades del proyecto o en la estructura del equipo de ingeniería del software.
- Restricciones presupuestarias o de planificaciones que provocan una redefinición del sistema o del producto.

Como se notificó anteriormente estas transformaciones siempre repercuten en la evolución del producto, por lo que es necesario definir si estos cambios son o no factibles, originando el siguiente problema:

Problema Científico

Existencia de deficiencia en el análisis de factibilidad y procedimientos de gestión de cambios en los proyectos, afectando el aseguramiento de la calidad, el ajuste a los cronogramas expuestos, la gestión de compromisos y costos del proyecto.

Objeto de estudio

Proceso de Gestión de Configuración de Software.

Objetivo General

Elaborar un procedimiento para determinar la factibilidad de cambios en los procesos de Desarrollo de Software.

Campo de acción

Identificación de la Configuración y proceso de control de cambios. Análisis de impacto de los cambios en el proceso de desarrollo de software.

Hipótesis

El uso de un procedimiento para determinar la factibilidad de cambios contribuye al aseguramiento de la calidad, ajuste a los cronogramas expuestos, la gestión de compromisos y estimación de costos del proyecto.

Variables

1. ***Independientes:***

Cambios en el proceso de desarrollo de software.

2. ***Dependientes:***

- Eficiencia en la toma de decisiones
- Aseguramiento de la calidad
- Ajuste a los cronogramas expuestos
- Estimación de costos.

Tareas a realizar

1. Estudiar de los conceptos fundamentales de Gestión de Configuración, y actividades de la Gestión de Configuración vinculadas al análisis de impacto de los cambios.
2. Hacer una valoración de las herramientas vinculadas a la administración de cambios.
3. Estudiar las técnicas que existen para el análisis de impacto de los cambios en el proceso de desarrollo de software, hacer un análisis crítico para determinar su utilización.
4. Estudiar los métodos de estimación para el análisis de costos y recursos con el fin de hacer una propuesta en el procedimiento.
5. Propuesta de un procedimiento para el análisis de impacto de los cambios en el proceso de desarrollo del software.
6. Evaluación de resultados a través del criterio de especialistas.

Métodos utilizados

Se utilizaron métodos científicos de investigación para este estudio, primeramente métodos teóricos:

Método Teórico-Histórico que analiza la trayectoria completa del fenómeno, su condicionamiento a los diferentes períodos de la historia, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales para la realización de un análisis y valoración de las herramientas que brindan soporte al proceso de control de cambios a nivel mundial así como un estudio de la Gestión de Configuración.

Métodos Teóricos-lógicos: Se basan en el estudio histórico del fenómeno, ponen de manifiesto la lógica interna de su desarrollo, de su teoría y haya el conocimiento mas profundo de su esencia. Estos métodos expresan en forma teórica la esencia del objeto, explican la historia de su desarrollo, reproducen el objeto en su forma superior y permiten unir el estudio de la estructura del objeto de investigación con su concepción histórica. Este permitió estudiar todos los conceptos relacionados con la Gestión de Configuración de Software.

Entre los métodos Teóricos Lógicos;

Método hipotético deductivo:

- Desempeña un papel esencial en el proceso de verificación de la hipótesis.
- Tiene un gran valor heurístico, pues permite adelantar y verificar nuevas hipótesis de la realidad.
- Permite inferir conclusiones y establecer predicciones a partir del sistema de conocimientos que ya se poseen.
- Se aplica en el análisis y construcción de muchas de las teorías científicas, permitiendo la sistematización del conocimiento científico al deducirlo de un número limitado de principios e hipótesis generales.

Se usaron además métodos particulares:

La entrevista que no es más que una conversación planificada entre el investigador y el entrevistado para hacer un estudio del análisis de impacto en la facultad 3.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción

En este capítulo se hace un estudio del estado del arte de la Gestión de Configuración a nivel mundial, del proceso de análisis de impacto en la Facultad 3 y de los conceptos fundamentales que se tratan a lo largo del documento. Se hace una valoración crítica de las diferentes definiciones de GCS así como el papel que juega dentro del proceso de desarrollo de software, definiendo sus objetivos, actividades fundamentales y relación directa con la calidad del software. Por último se hace un análisis de diferentes herramientas que apoyan las actividades dentro de la Gestión de Configuración.

1.2 Estudio en la facultad 3

Se hizo un estudio de cómo se hace actualmente el proceso de análisis de impacto en la facultad 3 para ello se hizo una entrevista que arrojó los siguientes resultados.

De los proyectos entrevistados solo en el 44 % se desempeña el rol de ingeniero de gestión de la Configuración.

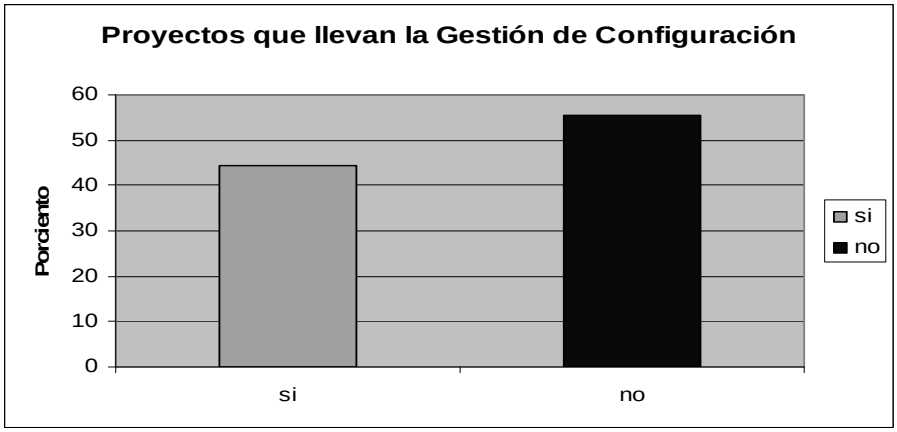


Figura 1. Proyectos que llevan la gestión de Configuración

Del total de proyectos se estimó la cantidad que actualmente realizan el análisis de impacto de los cambios de una forma u otra.

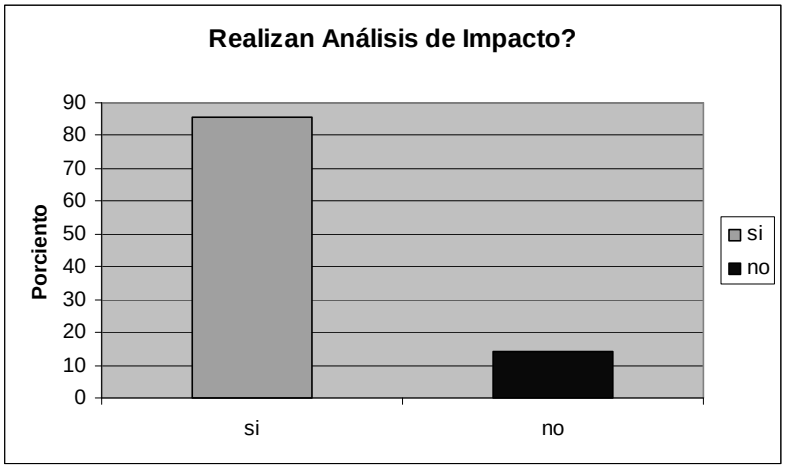


Figura 2. Proyectos que realizan análisis de impacto de los cambios.

De los proyectos que hacen análisis de impacto de los cambios se investigó la cantidad que usan o conocen alguna técnica para realizar este tipo de análisis.

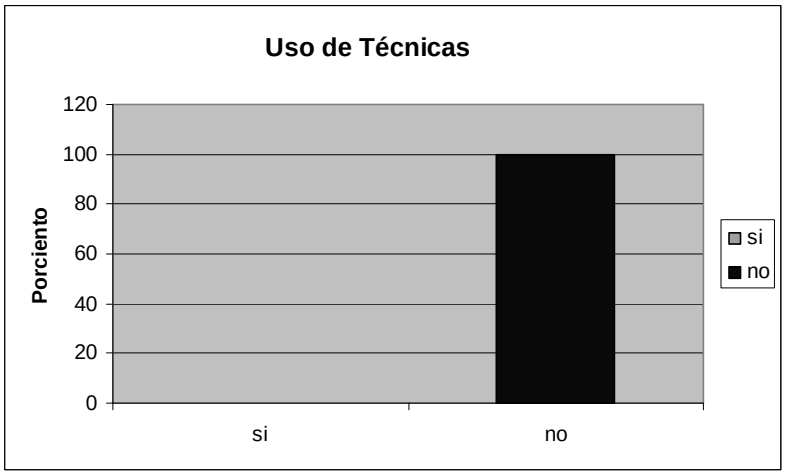


Figura 3. Conocimiento o uso de Técnicas.

De los proyectos que actualmente realizan el análisis de impacto de los cambios se determinó cuantos usan una estrategia.

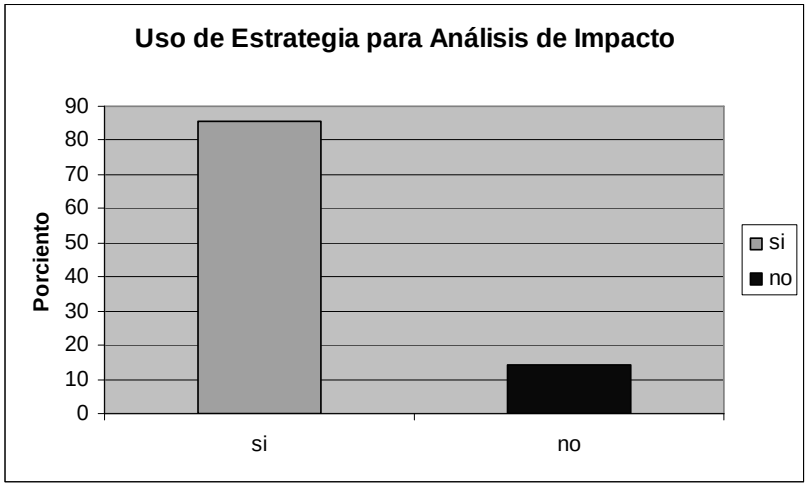


Figura 4. Estrategias usadas para el análisis de impacto de los cambios.

El uso de una estrategia debería aumentar el indicador de eficiencia en la toma de decisiones así que se analizó cuan eficiente son estas propuestas en los proyectos:

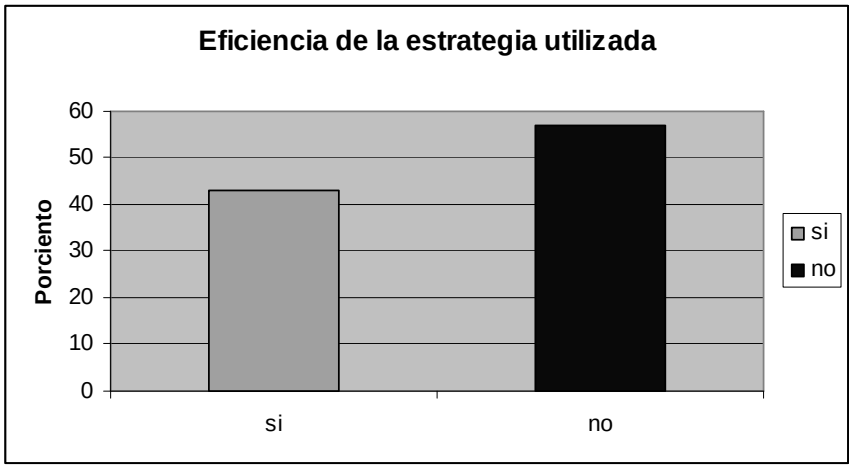
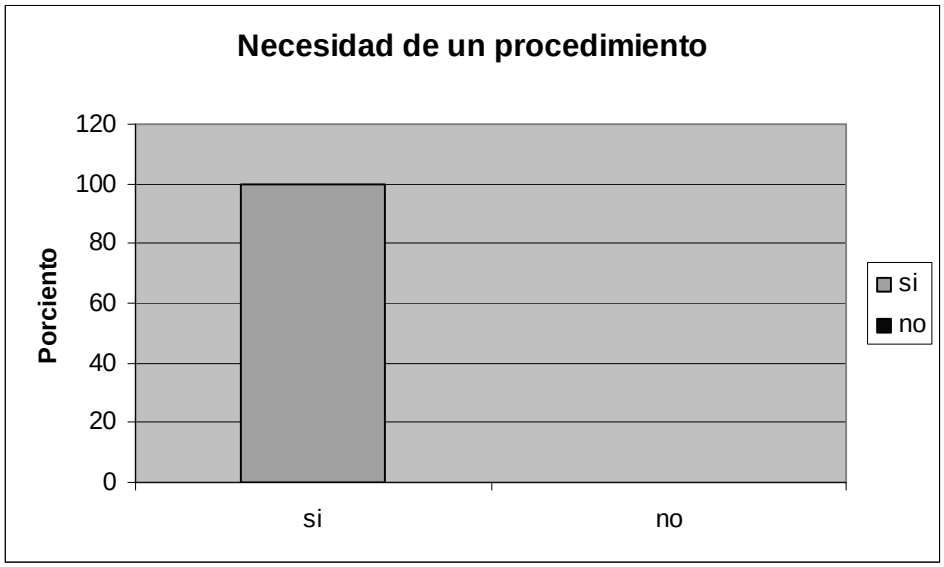


Figura 5. Indicador de eficiencia de las estrategias actuales.

Por ultimo se determinó en cuantos proyectos se piensa que es necesario el uso de un procedimiento estándar para la realización del análisis de impacto.



El resultado del análisis de las entrevistas nos demuestra que de la muestra escogida un porciento elevado realiza el análisis de impacto, de ellos algunos han creado su propia estrategia con un resultado poco eficiente y sin emplear técnicas o procedimientos efectivos por lo que el 100 % de los entrevistados está de acuerdo en la realización de un procedimiento que incluya todos los aspectos a tener en cuenta a la hora de hacer este tipo de análisis.

1.3 Conceptos fundamentales

El proceso de análisis de impacto de los cambios se enmarca dentro de lo que se podría llamar una disciplina de apoyo a los procesos de desarrollo de software (la Gestión de Configuración de Software), por lo que para su entendimiento es necesario tener un conocimiento básico de algunas definiciones que lleven a comprender la importancia del proceso antes mencionado.

1.3.1 Gestión de Configuración del software (GCS)

De manera general existe un consenso entre los autores que abordan el tema, de que la GCS es una disciplina de seguimiento y control. Haciendo una investigación de las definiciones expuestas por varios de ellos podemos encontrar que según (RANCÁN 2003): La **Gestión de Configuración** se puede definir como un área de la Ingeniería de Software cuya misión es controlar la evolución de un producto desarrollado, involucrando un conjunto de técnicas para gestionar con eficiencia los cambios que se realicen sobre ese producto a lo largo de su ciclo de vida.

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

Para Babich hay una conexión directa entre la GCS y la productividad.

Según (Antonio, 2001) es una "… disciplina, cuya misión es controlar la evolución de un sistema software" Esta definición es considerada bastante abarcadora pero muy sencilla (Martínez, 2006) y (Navarro., 2006).

Para (BROWN, 1998) es la que administra y controla el contenido, el cambio o el estado de la información compartida en un proyecto. El propósito fundamental de la GCS es establecer y mantener la integridad y el control en los productos software a lo largo del ciclo de vida del proyecto.

La más utilizada (Martínez, 2006) (Estrada, 2003) (Navarro., 2006) es la dada por IEEE en la que establece que (Society, 1990) (Institut, 1987) (BAMFORD, 1995):

"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."

Sin embargo, se considera que esta definición es incompleta (Estrada, 2003) (Navarro., 2006). Porque faltan elementos como el control del proceso y el control del esfuerzo de los desarrolladores (Brad, 2000).

Otra definición es propuesta por la Norma ISO 9000-3:1991 (BAMFORD, 1995), donde se establece que la Gestión de Configuración de Software provee mecanismo para identificar, controlar y dar seguimiento a cada una de las versiones de los elementos que conforman al producto software.

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

Haciendo una valoración crítica de las definiciones antes expuestas se puede llegar a la conclusión de que la mayoría de las ellas centran su importancia en la identificación, organización y control del software como producto obviando otros elementos importantes, por lo que se coincide con el criterio de (Estrada, 2003) (Navarro., 2006) que de todas las definiciones revisadas (Antonio, 2001) (Brad, 2000) (Rancán, 2003) (BROWN, 1998) (A.BABICH, 1986) (Rational, 2003) se ha considerado la más completa la de Ivar Jacobson, Martín Griss y Patrick Jonson en el libro "Software Reuse: Architecture, Process, and Organizartions for Busisness Success" en la que plantean (jacobson , 2007)

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

de los elementos; controla, almacena, maneja y libera los elementos asociados al producto de software"

1.3.2 Papel de la GCS dentro del proceso de desarrollo de Software.

El éxito de un proyecto depende en gran medida de cuatro tipos de funciones (ANTONIO 2001):

- Gestión del Proyecto.
- Desarrollo Técnico
- Sistema de Calidad.
- Sistema de Gestión de Configuración.

La gestión de configuración facilita el mantenimiento del sistema, aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de un cambio, tanto evolutivo como correctivo. Así mismo, permite controlar el sistema como producto global a lo largo de su desarrollo, obtener informes sobre el estado de desarrollo en que se encuentra y reducir el número de errores de adaptación del sistema, lo que se traduce en un aumento de calidad del producto, de la satisfacción del cliente y, en consecuencia, de mejora de la organización. Por lo que no podemos verlo como un aspecto independiente de la IS (Rancán, 2003) ya que se encuentra muy ligado a áreas entre las que se destacan:

- El mantenimiento del producto software.
- La calidad del producto.
- El entorno de desarrollo.
- El modelo de proceso.
- La organización que desarrolla el producto

Existen varias metodologías para el proceso de desarrollo. Hemos elegido los procesos de desarrollo que siguen la metodología RUP. RUP define cuatro fases, seis flujos de trabajo y tres de apoyo:

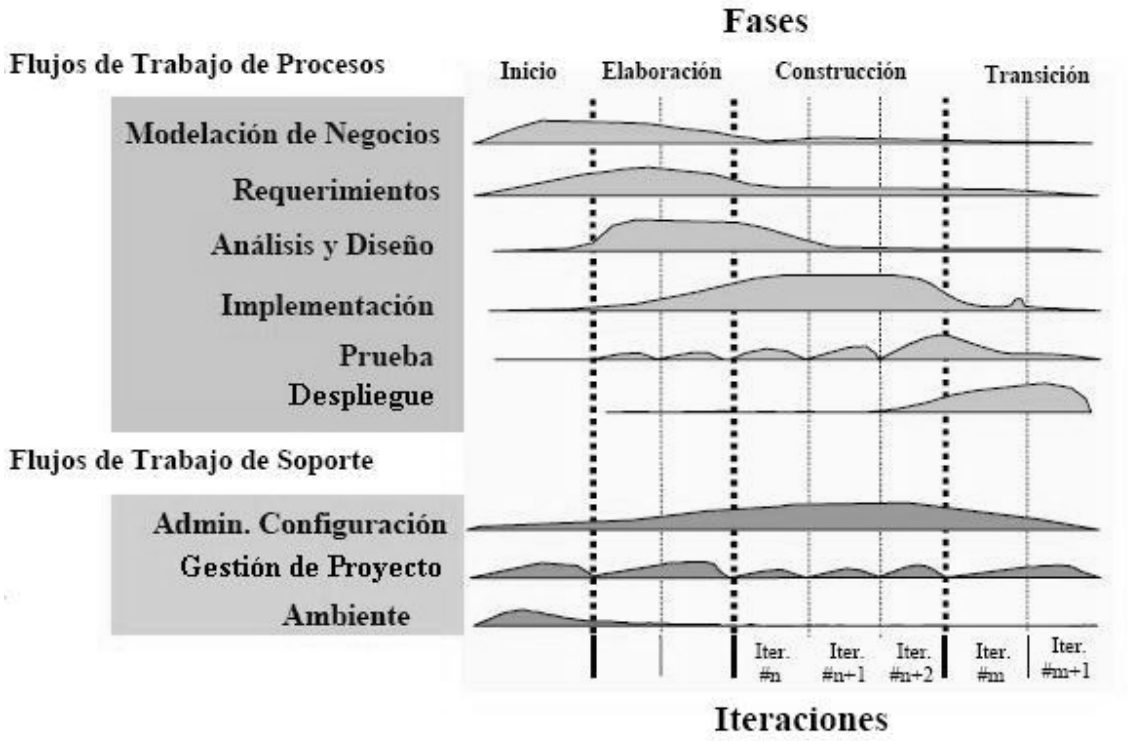


Figura 6. RUP en dos dimensiones.

1.3.3 Objetivos

El objetivo de la Gestión de Configuración es establecer y mantener la integridad y coherencia del producto software a fin de facilitar el seguimiento de los cambios que sobre él se implementan, asegurando la posibilidad de realizar auditorias de control sobre la evolución de las diferentes configuraciones. Para dar cumplimiento a estos se han definido una serie de actividades.

1.3.4 Actividades fundamentales.

De acuerdo a lo especificado por la IEEE, la Gestión de Configuración contempla las siguientes actividades (Rancán, 2003) encaminadas al cumplimiento de los objetivos:

Identificación de la configuración: Consiste en identificar la estructura del producto, sus componentes y tipo, haciéndolos únicos y accesibles mediante algún procedimiento.

Control de cambios en la configuración: Consiste en controlar las versiones de un producto y los cambios que sobre él se producen a lo largo de su ciclo de vida.

Generación de Informes de Estado: Consiste en la producción de informes sobre el estado de los ECS de un producto y de las solicitudes de cambio realizadas sobre ellos.

Auditoria de Configuración: Consiste en la validación de la integridad de un producto, manteniendo la consistencia entre sus componentes.

1.4 Configuración del software. Consideraciones importantes.

Según (Pressman, 2002) el resultado del proceso de ingeniería de software es una información que puede dividirse en tres amplias categorías:

- Programas de computadoras (Tanto en forma de código fuente como ejecutable).
- Documentos que describen los programas de computadoras (tantos técnicos como de usuario)
- Datos (Contenidos en el programa o externos a él).

Al conjunto de toda la información y productos utilizados o producidos en un proyecto como resultado del proceso de Ingeniería de Software se le denomina CONFIGURACIÓN DEL SOFTWARE (Antonio, 2001)**.**

1.4.1 Elementos de configuración

A cada uno de los componentes de la configuración del software se le va a llamar ELEMENTO DE CONFIGURACIÓN DEL SOFTWARE (ECS).

Definimos como un *Elemento de Configuración* a una unidad física y/o lógica parte de un conjunto mayor de elementos, producida o adquirida, que por sus características es distinguible de las demás y cuya evolución interesa administrar.

Un elemento de configuración del software (SWCI) es un conjunto de productos de trabajo documentados que se han creado en los procesos del ciclo de vida, o que se emplean en los mismos.

Por producto de trabajo se entiende a un elemento tangible que es el resultado de determinadas actividades o tareas de desarrollo: planes de pruebas, documentos de requisitos, documentos de diseño, código, manuales de usuario, etc.

Los elementos de configuración del software son cualquier parte del desarrollo o del producto entregable y necesitan poder identificarse, almacenarse, cambiarse, revisarse o mantenerse de forma independiente.

Estos elementos no comprende sólo los productos de software que se entregan al cliente, también incluyen los elementos que son necesarios para crear esos productos. Todos los artefactos producidos durante el proceso de ingeniería de software no tendrán que constituir necesariamente elementos de configuración, solo serán marcados como tal aquellos artefactos que deseemos poner bajo el procedimiento de control de cambio.

Ejemplos de elementos de configuración del software

Se pueden considerar como ECS, entre otros, a los siguientes componentes:

- Especificaciones del Sistema.
- Estimaciones y Planes.
- Especificación de requisitos software.
- Diseño arquitectónico.
- Diseño detallado.
- Prototipos generados.
- Código fuente.
- Documentación relacionada con la determinación de los factores de riesgo y su gestión a efectos de minimizar sus consecuencias.
- Programas ejecutables y librerías asociadas.
- Manuales del usuario, de operación e instalación.
- Documentación relacionada con cursos de formación en el uso del producto.
- Plan de pruebas.
- Casos de Prueba y resultados obtenidos.
- Estándares y procedimientos de Ingeniería de Software utilizados.
- Informes de incidencia.
- Pedidos de mantenimiento.
- Ordenes de cambio.
- Documentación del Software y Hardware utilizados como herramientas de desarrollo.
- Diseño de bases de datos.
- Bases de Datos.
- Información del entorno de desarrollo y de implantación.
- Contenidos iniciales de las bases de datos.

1.4.2 Versión

Los ECS pueden ser modificados según el procedimiento establecido, como resultado del cambio y una vez aprobado, se obtiene una nueva versión del mismo.

VERSIÓN es una instancia de un elemento de configuración, en un momento dado, que es almacenada en un repositorio, y que puede ser recuperada en cualquier momento para su uso o modificación. (Antonio, 2001)

1.4.3 Revisión Técnica Formal

El objetivo de las RTF es conseguir un trabajo técnico de una calidad uniforme detectando errores durante el proceso, con el fin de hacer más manejable el trabajo técnico, señalar las necesidades de mejora, verificar que durante el proceso se están alcanzando los requisitos, los atributos de calidad, el uso de estándares definidos, evaluando además, los artefactos generados por cada disciplina o flujo de trabajo que una vez aprobado formarán parte de la línea base del proyecto (Antonio, 2001).

Cada RTF se centra en una parte específica y pequeña del software según la fase e iteración en que se lleve a cabo y según el flujo de trabajo a verificar. El centro de atención de las RTF es un producto de trabajo (ECS).

1.4.4 Línea Base o Baseline

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

1.5 Roles propuestos por RUP

RUP propone una serie de roles para el flujo de apoyo Gestión de Configuración que pueden ser adaptados en dependencia del proyecto.

Administrador de la configuración responsable de:

- Crear y Configurar el ambiente para la gestión de la configuración.
- Ejecutar las auditorias a la configuración.
- Realizar los reportes a las auditorias de la configuración.
- Elaborar el documento Plan de Gestión de la Configuración.
- Administrar la BD (Base de Datos) o repositorio del proyecto.
- Crear la unidad de despliegue.

Presidente del Comité de Control de Cambio responsable de:

- Establecer el proceso de control de cambio.
- Revisar las solicitudes de cambio.
- Chequear si la solicitud esta duplicada o ha sido rechazada previamente.

Integrador responsable de:

- Crear el área de trabajo para realizar la integración.
- Crea las líneas base.
- Promueve la línea base. (Le modifica el estado).

Analista de Prueba responsable de:

- Verificar los cambios.

Cualquier Rol:

- Crea su espacio de trabajo.
- Realiza los cambios.
- Entrega los cambios implementados.
- Actualiza su área de trabajo.
- Presenta la solicitud de cambio.

- Actualiza la solicitud de cambio.

1.6 Gestión de Configuración y Calidad.

La experiencia de los últimos años ha mostrado la estrecha relación entre la calidad de los productos de software y los procesos utilizados para construirlos. La GCS es una actividad de garantía de calidad que se aplica en todas las fases del proceso de ingeniería. Según (ISO-12207) la GC forma parte del conjunto de procesos del ciclo de vida del software.

Para la Gestión de Calidad de Software a nivel mundial se han seguido principalmente dos tendencias:

- Seguir las reglas implantadas por las oficinas internacionales de estandarización para los productos y servicios a través de las normas ISO y la IEEE.
- Seguir las creadas específicamente para el mundo del software como CMMI y SPICE.

El **modelo CMM** (Capacity Maturity Model) describe un marco de referencia para el desarrollo y mantenimiento de software en las organizaciones así como la contratación de software (Wiegers, 1998). Este modelo organiza los pasos para evolucionar los procesos de software en cinco niveles, se define que todas las empresas están por defecto en el nivel 1 de CMM. Para lograr el nivel 2 es necesario cumplir un conjunto de exigencias y entre ellas juega un papel fundamental la GC.

La familia de **normas ISO 9000** es un conjunto de normas internacionales y guías de calidad que han obtenido una reputación mundial como base para establecer sistemas para gestión de calidad. La norma de apoyo ISO 10007:1995, proporciona directrices para asegurarse de que un producto complejo sigue funcionando cuando se cambian los componentes individualmente (ISO-9000), definiendo el objetivo principal de la GCS como: Documentar y proveer visibilidad de los productos de software y del estado de progreso en la satisfacción de los requerimientos funcionales y físicos. (ISO-10007, 1995)

El **estándar IEEE** 1074-1995 para el Desarrollo de Procesos del Ciclo de Vida del Software, establece el Proceso de Gestión de Configuración del Software como uno de los Procesos Integrales. Estos son los Procesos necesarios para completar exitosamente las actividades del proyecto, y son utilizados para asegurar la finalización y calidad de las funciones del proyecto. Este proceso consta de las siguientes actividades:

1. Planificar la Gestión de Configuración.
2. Desarrollar la Identificación de la Configuración.
3. Realizar el Control de la Configuración.
4. Realizar la Contabilidad de Estado

Estas actividades coinciden con algunas de las definidas en el proceso de Gestión de Configuración.

1.7 Herramientas de apoyo a la Gestión de Configuración.

Una buena metodología depende del uso de herramientas que permitan automatizar y facilitar las actividades de cada etapa del ciclo de vida del producto permitiendo crear y modificar fácilmente lo que se está haciendo. Las herramientas CASE surgen con esta necesidad y con el objetivo de apoyar cada una de las distintas etapas del ciclo de vida de desarrollo generando gráficamente la mayoría de los conceptos relacionados a cada etapa del ciclo de desarrollo (Análisis, diseño…et.).Estas herramientas han sido clasificadas de acuerdo a diferentes criterios (Instituto -Nacional-Estadística -Informática, 1999):

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.
- Paso del ciclo de vida

La última clasificación define las herramientas que se usan en un paso determinado del ciclo de vida (vertical CASE) como las herramientas para requisitos y las herramientas para la codificación; y las que son comunes a través de todo el ciclo de vida (horizontal CASE) como las herramientas para la documentación y la gestión de configuración.

Inicialmente se automatizaban de forma aislada cada una de las etapas de desarrollo pero después aparecieron las I-Case (Integrated CASE o CASE integrado) que abarcan todas las fases del ciclo de vida del desarrollo de sistemas y se aplican a todas las etapas de forma integral. A continuación se hace una valoración de las herramientas CASE para la Gestión de Configuración que existen:

1.7.1 Kanav

La Plataforma Kanav es una Suite de Herramientas diseñadas para integrar y gestionar las áreas claves involucradas en la producción de software. Esta brinda soporte, permiten la definición, el control y el mejoramiento del Proceso de Desarrollo de Software, propiciando una gestión de los proyectos y de los productos, durante todo su ciclo de vida.

Está compuesta por cuatro módulos interrelacionados:

- K-Process
- K-Request
- K-Project
- K- Process Navigator

De forma general esta plataforma contiene un conjunto de funcionalidades que dan soporte al proceso de desarrollo de software. Específicamente:

K-process: Orientado a la gestión de procesos facilitando la posibilidad del control de las diferentes versiones y la definición de las adaptaciones del proceso a diferentes tipos de proyectos.

K-Request: Orientado a la Gestión de Requerimientos.

K-Project: Orientado a la Gestión de Proyectos que permite la creación y planificación de proyectos a partir de los tipos de proyectos definidos en el proceso así como el control del avance del mismo. Facilita la administración y el versionamiento de todos los componentes y brinda todo tipo de métricas.

K- Process Navigator: Contiene documentación para cubrir las áreas de Gestión de requerimientos, planificación de proyectos de software, seguimiento y supervisión de proyectos de software, aseguramiento de la calidad del software, gestión de configuración de software, gestión de subcontratación de software (kanav) (innevo) .

Existen una serie de herramientas que permiten a los programadores de un proyecto centralizar y coordinar, estas son las gestoras de versiones. Entre ellas se pueden destacar las siguientes:

- Subversion
- CVS (Concurrent Versions System)
- RCS (Revision Control System)

1.7.2 Subversión

Es una herramienta para el sistema de control de versiones open-source, brinda entre sus funcionalidades: la recuperación de viejas versiones de los ficheros, ver su historial, etiquetar versiones, ramificar, unir, y efectuar retorno a versiones anteriores.

1.7.3 CVS

Herramienta para el desarrollo de versiones de ficheros de forma concurrente Es un sistema de almacén de ficheros (repositorio) centralizado en un servidor .Entre sus funcionalidades más importantes están:

- **Actualizar**: Descarga cambios en el CVS a nuestra copia local.

- **Estado**: Compara la copia local con la copia del CVS.
- **Entrega**r: Entregar archivos al repositorio.
- **Añadir**: Agrega archivos locales o directorios al repositorio.
- **Eliminar**: Elimina los archivos seleccionados del repositorio.

Para entender CVS es conveniente conocer un poco el sistema UNIX y tener algunas nociones de teleinformática, lo que dificulta su uso además CVS no es por sí mismo una herramienta que sirve para hacer la gestión de configuración, tan sólo gestiona el almacén de ficheros.

1.7.4 RCS

Funcionalidades más importantes:

- Inicializar los ficheros de repositorio.
- Guardar revisiones.
- Recuperar la última revisión del repositorio.
- Guardar un fichero sin perderlo.
- Ver el log de cambios.
- Cambios realizados al fichero.
- Borrar y añadir nuevos ficheros al repositorio.

1.7.5 Source Forge

Source Forge es un ambiente de desarrollo colaborativo herramientas heterogéneas y procesos con un entorno integrado para la administración de proyectos, administración de cambios y capacidades de colaboración (Martínez, 2006)

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).

1.7.6 Microsoft Visual SourceSafe. VSS

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

MS Visual SourceSafe permite:

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

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

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

1.7.7 Controla.

CONTROLA, una herramienta de apoyo al proceso de desarrollo de Software para las pequeñas compañías, con enfoque en la administración de requisitos.

Permite la identificación de los requisitos cerca del *stakeholder* (cliente/usuario del sistema), sus detalles y la administración de los cambios a través de rastreabilidad y control de versiones. La herramienta está siendo usada por los usuarios a lo largo de Brasil, entre ellos los estudiantes de graduación y postgrado, los maestros y pequeñas compañías de desarrollo del software. Entre sus principales funcionalidades están:

- Herramientas de estimación de plazos del proyecto basadas en los Casos de Uso.
- Registro de los usuarios, miembros de equipo de desarrollo y clientes del sistema, así como su jerarquía para solicitación y aprobación de requisitos.
- Relaciones de dependencia entre los requisitos para los diferentes grupos de artefactos a través de las matrices de rastreabilidad.
- Evaluación del impacto de alteraciones en el sistema a través de la rastreabilidad.

La principal dificultad de esta herramienta es que centra sus funcionalidades en la administración de requisitos, y aunque permite que sean definidas matrices y rastreabilidad

hace un análisis de impacto solamente cuando los cambios se hacen a nivel de requerimientos.

1.7.8 Rational ClearCase

Rational ClearCase es una de las herramientas incluidas en la Suite Rational 2003 que entre sus principales funcionalidades se encuentran:

- Ofrece control de versiones, administración del espacio de trabajo, administración de la construcción y configuración del proceso.
- Trabaja con Rational ClearQuest para integrar la administración en la configuración de software con el rastreo de defectos y errores.
- Permite el Unified Change Management: El proceso de Rational basado en actividades para la administración del cambio.
- Incluido en Rational Suite para administración de requerimientos a través del ciclo de vida.
- Se integra con IDEs y herramientas de desarrollo. WebSphere Studio, Eclipse y Microsoft® .NET

Rational ClearCase rastrea cambios en cada archivo y directorio, manteniendo una completa y documentada historias de versiones de código fuente, binarios, ejecutables, documentación, suites de pruebas, librerías y artefactos Web.

Unify Change Management (UCM): define un proceso consistente para la administración de cambios basado en actividades que los equipos pueden aplicar a sus proyectos de desarrollo en el momento. Hecho posible gracias al Rational ClearCase y Rational ClearQuest, UCM es un componente clave del Rational Unified Process, un framework comprensible para entregar las mejores prácticas en el desarrollo de software. UCM simplifica el desarrollo al incrementar el nivel de abstracción para administrar los cambios en términos de actividades o componentes, al contrario de rastrear archivos individuales manualmente. Con el UCM, las actividades son automáticamente asociadas con su set de cambios, el cual encapsula todas las versiones de

artefactos en un proyecto usado para implementar la actividad. UCM asegura que los desarrolladores estén trabajando con la información más actualizada cuando inician la implementación.

1.7.9 Rational ClearQuest

Entre las actividades que realiza Rational ClearQuest se encuentran:

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

Una de las principales ventajas de Rational ClearQuest, como se puede apreciar en lo señalado anteriormente, es su integración con el resto de las herramientas de la Suite de Racional (Rational, 2003).

Sin duda una de las mayores dificultades de los productos Rational es su elevado costo. Rational ClearQuest cuesta $1,579.50. (Shopper, 2007)

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..

1.7.10 ConfigCASE

ConfigCASE es un software de apoyo al desarrollo del proceso de GCS y una plataforma para el control del flujo de trabajo. Consta de seis módulos:

- **Módulo General**: Contiene toda la información de la empresa, grupos de proyectos, desarrolladores, clientes, etc.
- **Módulo de Metodologías**: Permite establecer las metodologías a seguir en la ejecución del proyecto, sus fases y etapas.
- **Desarrollador**: Controla el proceso de resolución de los cambios en el proyecto. Permite además generar peticiones de cambios y defectos.
- **Generador de Peticiones de Cambios**: Permite a cualquier especialista o directivo de la empresa generar peticiones de cambios.
- **Controlador de Peticiones de Cambios**: Permite al Jefe de Proyecto asignar y evaluar las peticiones de cambio.
- **Controlador de Órdenes de Trabajo**: Permite al Jefe de Proyecto asignar y evaluar Órdenes de Trabajo durante todo el ciclo de vida del proyecto.
- **Métricas**: Módulo que permite medir el proceso y determinar posibles mejoras en su ejecución.

A pesar de la existencia de un gran número de herramientas CASE para la GCS muchas de ellas (subversion, Visual SourceSafe) solo se enfocan en alguna actividad del proceso de gestión de configuración lo que determina que se tengan que usar varias para dar soporte por separado a una actividad determinada, el resultado de esto es que al no existir un nivel de integración resulte muy costoso realizar procesos como el de análisis de impacto. Por otra parte herramientas como

el Clearquest o el Clearcase requieren de un nivel de aprendizaje y herramientas como Kanav y Controla tienen el inconveniente de ser propietarias con un elevado costo de adquisición que oscila (ClearCase y ClearQuest) en el orden de los 9000 dólares. ConfigCase es una herramienta que posee un alto nivel de integración y a pesar de que automatiza procesos como la asignación del trabajo, control de cambios, control de errores, control de clientes y versiones de software así como un conjunto de métricas para estos procesos deja fuera actividades de la identificación de configuración y no define un esquema de identificación para todas las relaciones que se establecen entre los elementos de configuración de software.

La utilización de un esquema de identificación para todas las relaciones que se establecen entre los elementos de configuración sería de gran utilidad a la hora de la toma de decisiones en un análisis de impacto ya que nos daría una información que podría ser un diagrama o grafo de relaciones en el que se visualizara todos los elementos y resultaría mas fácil identificar los ECS impactados por el cambio.

La creación de este tipo de grafo llevó al estudio de modelos que permitieran la toma de decisiones en grafos de dependencias.

1.8 Otro modelo

Enhanced Telecom Operations Map (eTOM) realizó un modelo que representa los componentes de la infraestructura de IT, basado en la utilización de Abstract System Dependence Graph (ASDG) en su libro (Sergio R. Machuca, 2002).

Proponen un grafo abstracto de dependencia de sistema (ASDG) que puede ser construido usando un subconjunto de información del SDG. ASDG consiste en vértices que representan componentes, por ejemplo funciones y variables globales.

Para las conexiones de red se define un grafo estándar en el cual los nodos son los componentes del ASDG y las aristas son las propias conexiones. Los componentes quedan agrupados en comunes (hardware, lógicos, eléctricos, recursos humanos) y de red.

Si se afecta a un componente común, se construye un árbol de alcance a partir el nodo que representa el componente, que incluya todos los nodos alcanzados por las relaciones de dependencia con el mismo. El árbol incluye todos los componentes impactados, los administradores y los usuarios. Para construir el árbol se realiza una recorrida simple del grafo, utilizando un algoritmo basado en Dijkstra, para cada nodo afectado.

En el caso en que se afecte un componente de red, el análisis es diferente. En este caso se debe verificar que para todo par de componentes, que residen en equipos distintos y que existe una dependencia entre ellos, exista un camino de Red (que no pase por los componentes de red afectados) entre los componentes de hardware donde reside cada uno. En caso de no existir se debe incluir en el árbol el componente dependiente en la relación y continuar con el árbol de alcance. Para verificar la existencia de un camino entre componentes se utiliza una variación de Floyd.

Se ha realizado el desarrollo de un sistema de gestión de cambios de infraestructura que cuenta con una implementación del CMDB según los lineamientos de ITIL. La herramienta permite registrar todos los componentes y sus relaciones (de red y dependencias) y realizar análisis de impacto ante cambios a alguno de los componentes de la infraestructura.

Este tipo de modelo tiene el inconveniente de que trata todos los nodos del grafo con el mismo nivel de complejidad por lo que no es útil para la toma de decisiones si usamos como nodos los elementos de configuración de software.

1.9 Conclusiones

El estudio e investigación de los conceptos fundamentales de la Gestión de Configuración da una medida de la importancia que se atribuye a esta disciplina dentro del proceso de desarrollo de software. El análisis de las herramientas existentes no apoya todo el proceso de GCS. Como resultado de la gran cantidad de información y las relaciones complejas involucradas, rastrear y evaluar el efecto de un cambio puede ser costoso ya sea por el tiempo que consume o por los errores que produce. La proposición de un procedimiento que defina las actividades, personas,

tareas involucradas en este proceso así como herramientas a utilizar contribuiría a una mejor práctica de la Gestión de Configuración y por tanto una mayor acercamiento a los estándares de calidad.

CAPÍTULO 2 FACTIBILIDAD DE CAMBIOS.

2.1 Introducción

El objetivo principal del análisis de impacto o factibilidad de cambios es calcular el alcance del cambio propuesto y documentar el resultado. Esto incluye la identificación de impacto potencial en los sistemas, en el hardware, el software, la documentación, los datos, las personas etc. además del alcance de los cambios. Esta fase involucra también la estimación de los recursos requeridos y el costo de cada cambio. Dependiendo de los resultados de este análisis, puede tomarse una decisión: hacerse o rechazar el cambio. El análisis de impacto es la fase en el proceso de gestión en la que se enfocará este estudio.

El cambio es inherente en el desarrollo de la mayor parte del software. Los requisitos del software cambian e incluso evolucionan a medida que van formulándose. Estos cambios pueden afectar las funcionalidades del sistema y las metas comerciales de la organización para la que el software se desarrolla.

Por estas razones es importante que los cambios sean traceados cuidadosamente, y sus efectos analizados en el funcionamiento del sistema.

La repercusión de un solo cambio propuesto puede no ser muy significante pero lo complejo de las relaciones lleva al fenómeno conocido como la onda efecto o propagación de impacto. Esto ocurre cuando un cambio se propone para un elemento de un sistema pero debido a las relaciones entre los artefactos un impacto lleva a otro, este a otro y así sucesivamente.

2.2 Identificación de la configuración.

La eficiencia de un análisis de impacto depende en gran medida de la identificación de la configuración que se realice en el proyecto

Identificación de la configuración: Consiste en identificar y asignar nombres significativos y consistentes a todos y cada uno de los elementos que forman parte del producto software, en cada fase de su desarrollo, es decir, a cada uno de los Elementos de Configuración del Software (Antonio, 2001):

En esta tarea se definen un conjunto de actividades:

1. Establecimiento de una jerarquía preliminar del producto software.
2. Selección de los Elementos de Configuración.
3. Definición de las relaciones en la configuración.
4. Definición de un Esquema de Identificación.
5. Definición y Establecimiento de líneas base.
6. Definición y Establecimiento de bibliotecas de software.

2.2.1 Jerarquía preliminar del producto software

En esta actividad se obtiene una primera visión de la estructura y los elementos que tendrá el sistema software.

2.2.2 Selección de los Elementos de Configuración.

La importancia de la elección de los ECS radica en que tener demasiados puede provocar un número elevado de especificaciones y documentos que al final resulta inmanejable. Sin embargo, el tener pocos ECS puede hacer que no se tenga la suficiente visibilidad sobre el producto. En su libro Gestión de Configuración de Productos Software en etapa de desarrollo (Rancán, 2003) propone toda una metodología para la aplicación de la Gestión de Configuración de los procesos de software en desarrollo y de técnicas a utilizar para obtener los ECS necesarios. Una buena práctica en esta actividad es seguir determinados criterios de selección entre los que se proponen:

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 EC con otros deberían ser simples. Hay que minimizar el acoplamiento entre ECS.

Singularidad del componente con respecto al resto.

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

Si se trata de elementos reutilizados.

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

2.2.3 Definición de las relaciones en la configuración.

Esta es una tarea clave para el proceso de análisis de impacto debido que aquí es donde se establece que relaciones mantener entre lo ECS y el grado de dependencia que realmente entre ellos existe, entre ellas se pueden definir las siguientes:

Equivalencia: Por ejemplo, cuando un determinado ECS que es un programa está almacenado en tres lugares diferentes (un disco maestro, una copia de seguridad en cinta y el diskette del programador), pero todas las copias corresponden al mismo programa.

Composición: Por ejemplo, el ECS “especificación de diseño” estará compuesto de otros ECS, como el “modelo de datos” o el “diseño del módulo N”, para cada uno de los módulos que componen el producto software.

Dependencia: Cualquier otro tipo de relaciones entre ECS, fundamentalmente en la documentación, y sobre todo para facilitar la trazabilidad de los requisitos. Así, por ejemplo, los diagramas de actividades del negocio dependen del diagrama de casos de uso del negocio.

Derivación: A partir de qué se ha originado algo. Por ejemplo, el código objeto del código fuente, o una determinada traza de ejecución de un determinado caso de prueba con un determinado programa ejecutable.

Sucesión: En la historia de cambios sobre un elemento desde una revisión a otra. Puede ser muy útil definir un Grafo de Evolución para cada ECS. Este grafo describe la historia de cambios de un objeto y su transición de unas versiones a otras.

Variante: Variación sobre un determinado elemento, con la misma funcionalidad, pero que, por ejemplo, funciona más rápido.

2.2.4 Definición de un Esquema de Identificación.

Esta tarea es la encargada de definir la información que debe identificar a cada elemento de configuración. Esta información debe incluir:

1. Número o código.
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 ECS.
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.

11. Número y fecha de versión.
12. Elementos de Configuración relacionados directamente.

2.2.5 Definición y Establecimiento de líneas base.

Las líneas bases se establecen en hitos del proceso de desarrollo de software con los objetivos de: Identificar los resultados de las tareas realizadas durante la fase y asegurar que se ha completado la fase.

Una línea base se puede establecer de dos formas:

- 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 identifica el estado actual del producto en dicho punto del proceso de desarrollo.

2.2.6 Definición y Establecimiento de bibliotecas de software.

Una biblioteca de software es una colección controlada de software y/o documentación relacionada cuyo objetivo es ayudar en el desarrollo, uso o mantenimiento del software. (Antonio, 2001). Se definen las siguientes bibliotecas de trabajo:

- Biblioteca de trabajo.
- Biblioteca de integración.
- Biblioteca de soporte al proyecto.
- Biblioteca de producción.
- Biblioteca maestra.
- Repositorio de software.
- Biblioteca de backup.

2.3 Control de cambios en la configuración.

En un proyecto pequeño es relativamente fácil controlar los cambios pero cuando un equipo de desarrollo se enfrenta a grandes proyectos de ingeniería de software la práctica incontrolada de este puede terminar en un desastre por eso el control de cambios es la actividad más importante de la Gestión de Configuración ya que es la encargada de definir un procedimiento para el proceso de control de cambios. Dentro del proceso antes mencionado se encuentra enmarcado el análisis de impacto por lo es objetivo del presente trabajo el estudio de esta actividad.

Según (Rancán, 2003) se definen dos tipos de cambios:

- Corrección de un defecto: Los clientes tienden a clasificar todos los cambios en esta categoría.
- Mejora del sistema: Los programadores, sin embargo, los suelen clasificar aquí.

2.3.1 Niveles de control de cambios.

Se establecen tres niveles de control de cambios:

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 Elementos de Configuración del Software

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.

2.3.2 Proceso de control de cambios. Análisis de impacto.

No existe ningún procedimiento para el control de cambio informal, no siendo así para el formal que se puede hacer de diversas formas dependiendo de las características específicas de cada proyecto. La siguiente figura muestra la estructura a seguir para el proceso de control de cambio según (Pressman, 2002):

Análisis del proceso:

1. Se hace una petición de cambio y se evalúa para calcular el esfuerzo técnico, los posibles efectos secundarios, el impacto global sobre otras funcionalidades del sistema y otros objetos de la configuración (ECS), para ello es necesario haber realizado desde los inicios del proyecto una buena gestión de requisitos.
2. Los resultados de la evaluación se presentan en un formato de plantilla de solicitud de cambio al Comité de Control de Cambio para que sea evaluada por este, para ello el comité revisa la solicitud de forma preliminar para determinar si es una solicitud valida, si lo es, se determina si la solicitud esta dentro o fuera del alcance de la liberación actual basándose en la prioridad, calendario, recursos, nivel de esfuerzo, riesgo, severidad y cualquier otro criterio relevante que el grupo establezca. La salida del análisis de la solicitud será la solicitud aceptada o denegada sobre la base de que es una solicitud no valida, está duplicada o está fuera del alcance (pospuesta)
3. Si es denegada, la solicitud se cierra y se le informa al usuario.
4. Si esta pospuesta, cuando se inicie la próxima iteración todas las solicitudes en este estado serán analizadas y si entran en el alcance de la iteración son incluidas en la iteración.

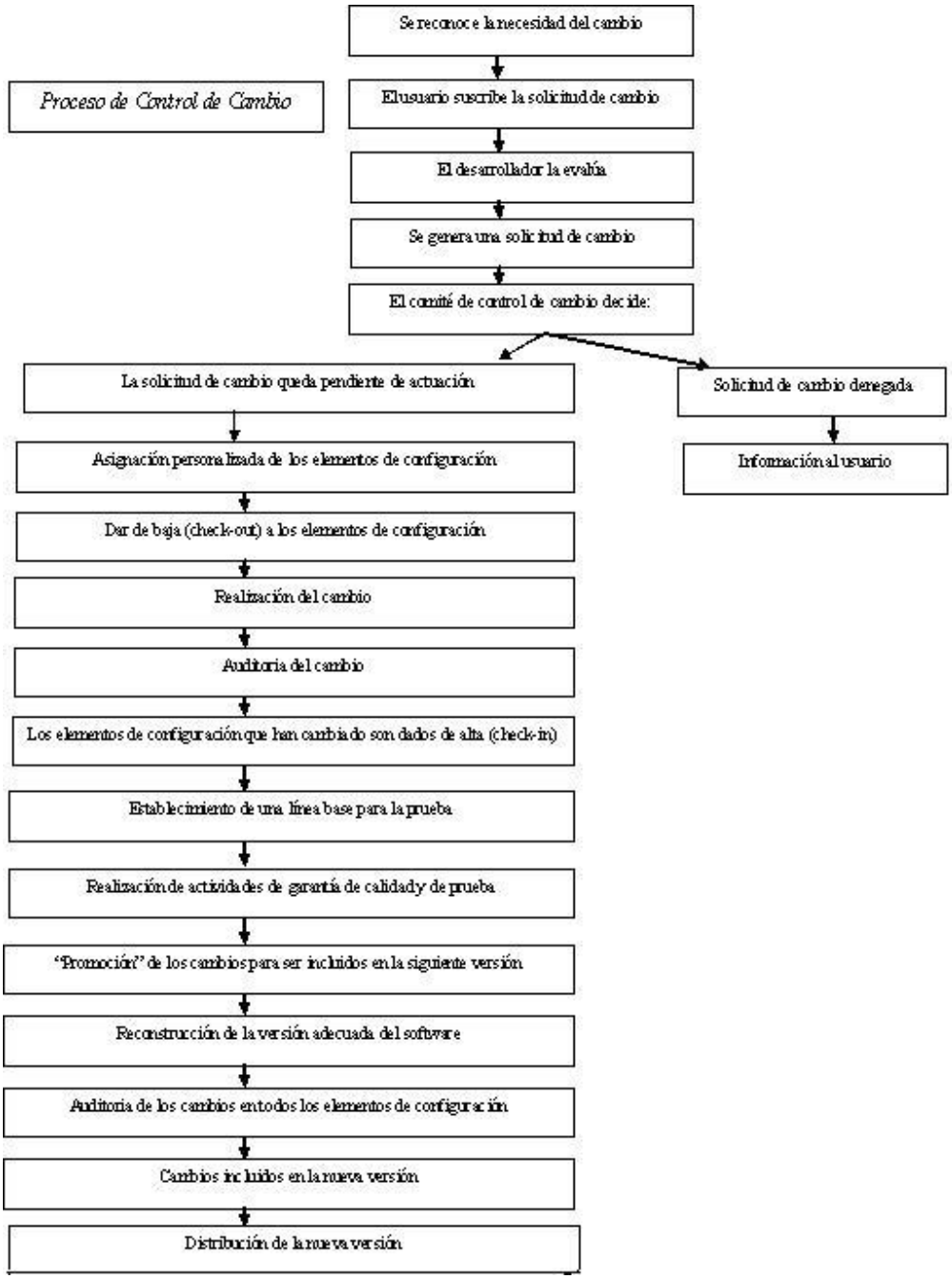


Figura 7. Proceso de control de cambios.

5. Si es aceptada, la solicitud se puede abrir generándose una orden de trabajo.
6. El Administrador del Proyecto asignará el trabajo al miembro del equipo apropiado, dependiendo del tipo de solicitud (Ej. Solicitud de mejora, defecto, cambio en la documentación, prueba, etc.) y realiza cualquier actualización que sea necesaria en el calendario del proyecto.
7. El administrador de la configuración extrae los ECS involucrados en el cambio.
8. Los miembros del equipo asignados al cambio realizan las actividades necesarias dentro del proceso de ingeniería de software (requerimiento, análisis y diseño, implementación, etc.) para atender y resolver la solicitud de cambio. Estas actividades pudieran incluir todas las revisiones normales y actividades de prueba según se describe dentro del proceso normal de desarrollo. Una vez concluido se marca la solicitud como resuelta.
9. Se realizan las revisiones técnicas formales necesarias para aprobar los cambios a los ECS que han sido modificados debido al cambio solicitado (Actividades de garantía de la calidad)
10. Una vez que han sido aprobados se define la línea base promovida como estable y una vez que se realicen las pruebas de integración necesarias entonces de define como liberada.
11. Se realiza la auditoria a la configuración para comprobar que los cambios se han implementado según lo acordado.
12. El administrador de la configuración del software incluye los cambios y construye la nueva versión con los cambios incorporados generando un documento de Notas de la liberación (release note) para que sea instalado en el cliente.
13. Se distribuye la nueva versión a los clientes

Al definir el proceso de control de cambio también debe definirse cual será el mecanismo que se debe seguir para analizar y evaluar las solicitudes de cambio (Impacto del cambio).

2.4 Proceso de análisis de impacto.

Existen definiciones diferentes de análisis de impacto de cambio. Pfleeger y Bohner definen el análisis de impacto de cambio como “la evaluación de los riesgos asociados con el cambio, incluso las estimaciones de los efectos en los recursos, esfuerzo, y tiempo

Turver y Munro definen el análisis de impacto de cambio como “la valoración de un cambio, al código de fuente de un módulo, en los otros módulos del sistema. Determina el alcance de un cambio y proporciona una medida de su complejidad.”

Arnold y Bohner definen el análisis de impacto de cambio como identificar las consecuencias potenciales de un cambio, o estimar lo que necesita ser modificado para lograr un cambio dando énfasis a la estimación de los impactos. La definición de Pfleeger extiende su definición a la evaluación de impactos. Se define el efecto de la onda de un cambio al código de la fuente de un sistema del software como los efectos consiguientes en otras partes del sistema que es el resultado de ese cambio.

Se considera que la definición mas completa es la dada por Pfleeger y Bohner ya que incluye todos los elementos a tomar en cuenta para la evaluación del cambio.

2.4.1 Ventajas

El análisis de impacto puede ofrecer facilidades útiles ya que puede ayudar a determinar el alcance y la magnitud de un cambio propuesto y de esta forma desarrollar las estimaciones con respecto a los recursos requeridos y el costo potencial de un cambio propuesto.

Además de esto puede ayudar también en la comunicación de la complejidad de un cambio pedido a los clientes o usuarios terminales que lo propusieron, esto es útil al intentar demostrar que el cambio puede no merecer el costo. Finalmente este tipo de análisis reduce la cantidad de

mantenimiento correctivo que resulta de introducir el cambio, previniendo los impactos grandes y las consecuencias de llevarlo a cabo.

Uno de los beneficios más importantes es poder asegurar que, para los cambios que se llevan a cabo, todos los impactos se identifican y se asegura la decisión apropiada para mantener exactitud y consistencia dentro del sistema.

2.4.2 Clasificación de los métodos de análisis de impacto

Se han propuesto varios métodos qué intentan analizar el impacto de cambio en las varias fases de desarrollo del sistema. Estos pueden ser clasificados en los tipos principales siguientes (Lock, 1998):

Los métodos semánticos: El análisis de gráficos, tabla, el código etc. para extraer la información con respecto a las dependencias entre los componentes del sistema.

Los métodos heurísticos: El uso de reglas basadas en métodos de Inteligencia Artificial para predecir el impacto potencial de un cambio propuesto.

Los métodos estocásticos: El uso de probabilidades de propagación de impacto para determinar el probable efecto de un cambio.

Los métodos exhaustivos: Técnicas de prerregistro de trazabilidad basadas en la fuerza bruta.

Los métodos híbridos: Métodos que consisten en una combinación de los cuatro métodos anteriores.

El objetivo de todos estos métodos es intentar extraer, guardar y analizar las relaciones de trazabilidad de la representación de un sistema. Las relaciones que existen entre los componentes de un sistema son a menudo implícitas en la representación del sistema y es el trabajo del análisis de impacto ayudar al diseñador a hacer estas relaciones explícitas para facilitar el análisis.

La mayoría del trabajo hecho en el análisis de impacto profundiza en los artefactos de código fuente, esto es porque el código fuente proporciona la cantidad máxima de información de trazabilidad.

2.5 Análisis de impacto como proceso.

Las fases individuales que se llevan a cabo en el proceso de análisis de impacto son las siguientes:

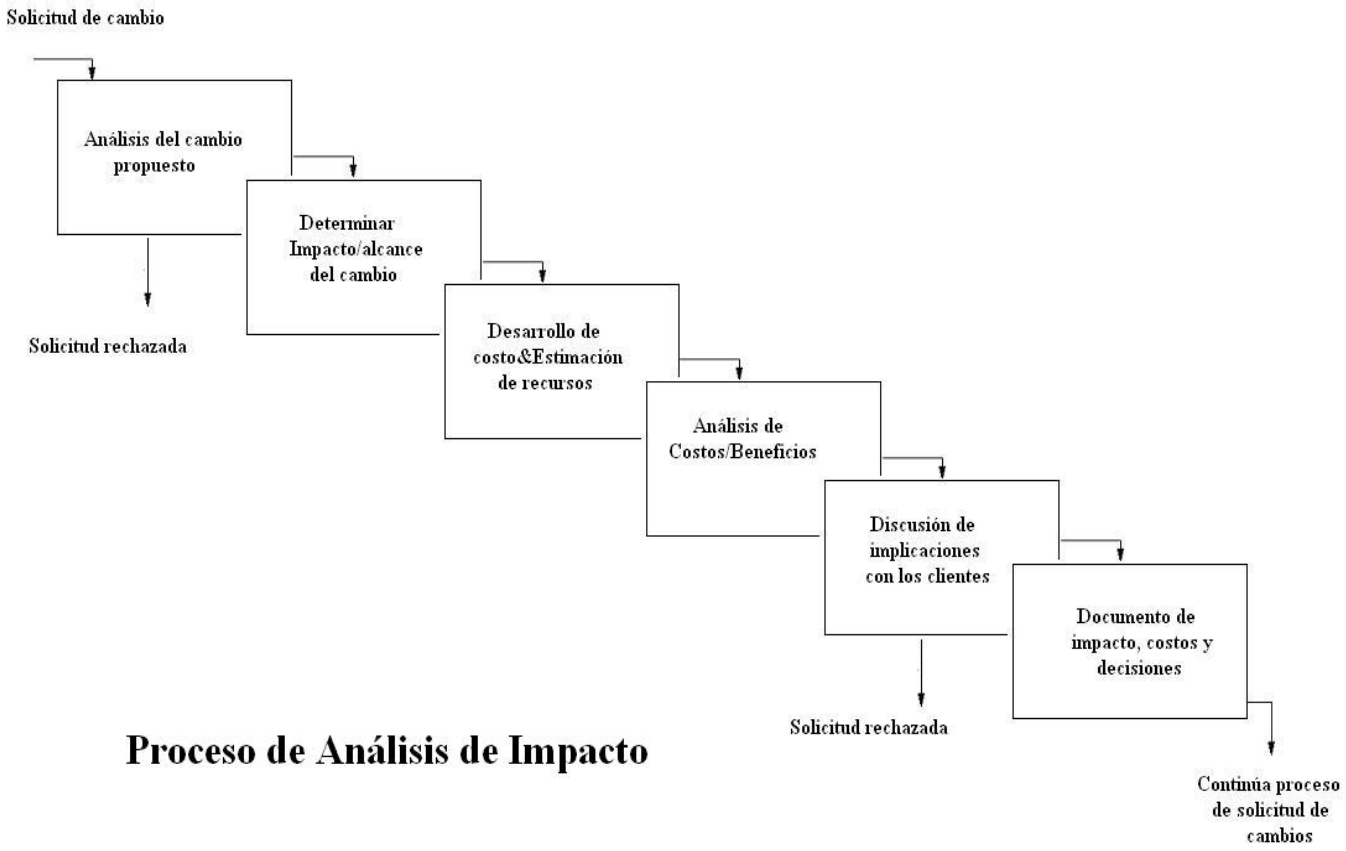


Figura 8. Fases del proceso de análisis de impacto**.**

Análisis del cambio propuesto: El desarrollador debe asegurarse de que se entiende el cambio propuesto y los componentes del sistema afectados. Se debe verificar el cambio para su validez, con vistas a entender de forma profunda los objetivos del cambio propuesto.

Determinar el impacto/alcance del cambio: Esta fase involucra el uso de una técnica del análisis para identificar el impacto en los artefactos de todas las fases del ciclo de vida de desarrollo.

Desarrollo del costo y estimaciones de los recursos: Durante esta fase se aplican métodos de estimación como modelos de costo algorítmico o el análisis estático que se usan para intentar calcular el costo potencial del cambio.

Análisis del costo/beneficio: Durante esta fase se hace una comparación entre el peso de los costos estimados del cambio contra los beneficios esperados para determinar si el cambio sería eficaz. El dato usado para este análisis se extrae de la información en la forma de petición de cambio.

Discutir las implicaciones con clientes: Una vez que se han identificado los costos relativos y beneficios involucrados, la decisión para rechazar o proceder con el cambio puede tomarse. Esto o debe hacerse después de la consultación con la organización del cliente por medio de una discusión informal o vía una tabla de la revisión formal.

El documento impacto, costo y decisiones: El diseñador debe registrar el efecto potencial y el costo probable del cambio en todos los artefactos de desarrollo calculado en las fases anteriores. Esto puede incluir artefactos que relacionan al software, hardware, datos, documentación y factores humanos (la ayuda en línea, los dispositivos de la entrada etc.). Tal documentación debe hacerse aún cuando se ha tomado la decisión de rechazar el cambio debido a que esta información puede ser útil para evaluar los cambios futuros.

2.6 Información de trazabilidad.

Antes de estudiar las diferentes técnicas de análisis de impacto se hará un pequeño estudio de la **Información de trazabilidad.** Esta es la información clave que se utilizará para determinar la magnitud del cambio ya que indica las relaciones potenciales entre los ECS y coleccionándolas es posible identificar los caminos de propagación e impacto dentro de un sistema, información en la cual están basadas las diferentes técnicas que se presentan a continuación.

Un buen análisis de impacto y por tanto administración de la Gestión de Configuración depende del mantenimiento y accesibilidad a esta información.

2.7 Técnicas de análisis de impacto.

Una vez definidos los tipos de relaciones que se pueden establecer entre los ECS, la estructura del proceso de control de cambios y del proceso de análisis de impacto quedaría por definir el método a utilizar para la realización del mismo. A continuación se hará un análisis crítico de las técnicas de análisis más conocidas (Lock, 1998).

2.7.1 Pre-registro de información de Trazabilidad.

Esta técnica consiste en la acumulación de los detalles de las relaciones que se establecen entre los diferentes artefactos que componen el sistema. Las relaciones pueden estar entre los artefactos en la misma fase de desarrollo, en diferentes fases de desarrollo y entre versiones diferentes del mismo artefacto.

Esta información es denominada 'Pre-registrada' porque se identifica manualmente, se colecciona durante el ciclo de vida de desarrollo entero del sistema y es documentada por los desarrolladores del proyecto.

Esta técnica es considerada poco formal y práctica debido a que muchas relaciones que son difíciles de distinguir son obviadas en este proceso. Puede ser muy difícil identificar las relaciones

entre los requisitos funcionales y no funcionales usando este tipo de trazabilidad o determinar el impacto de la alteración en la funcionalidad de un componente observando solamente los registros de trazabilidad. Además la política que se sigue para recoger la información puede ser costosa e innecesaria.

A pesar de su simplicidad esta técnica es poco usado debido a que el algoritmo que se sigue para determinar los elementos afectados es poco realista e ineficaz para la mayoría de los sistemas no triviales, sin embargo es posible usarlo combinado con otras técnicas para producir un método híbrido que supere algunas de las deficiencias del otro método utilizado.

Estructura del pre-registro de información de trazabilidad

Una estructura común usada para el mantenimiento de la trazabilidad usando esta técnica es la "matriz de trazabilidad". Esto consiste en una tabla que contenga una fila y una columna para cada EC de un sistema en particular. Si existe una relación entre un componente X y otro Y entonces se marca (puede ser con una cruz (X), o un (1) numérico. Este tipo de matrices no tienen que ser simétricas ya que un si un cambio en un ECS (X) implica otro en un ECS (Y) no necesariamente un cambio en Y implica cambio en X.

Este tipo de estructura de almacenaje llega a ser poco manejable como se había dicho anteriormente cuando el sistema es muy complejo, en tal caso existe una alternativa aunque puede resultar menos eficiente: La lista de trazabilidad. En esta estructura a cada ECS se le asocia una lista que contiene referencias a todos los elementos que este está relacionado.

Esta estructura se encuentra más lejos de ser eficiente que la matriz ya que se desperdicia mucho tiempo identificando las relaciones inversas.

2.7.2 Técnica basada en el conocimiento.

El principal objetivo de esta técnica es extraer la información de trazabilidad entre ECS del sistema basándose en los efectos del impacto de cambios anteriores. Para hacer este tipo de evaluación los datos referentes a los cambios y sus efectos deben ser registrados.

Se registra cada cambio y sus consecuencias como un solo caso o escenario. Cuando se han identificado un número significativo de casos es entonces posible analizar las relaciones de trazabilidad usando los datos almacenados. Durante la primera fase de esta técnica, cuando se está recuperando la información necesaria, es posible realizar el análisis usando uno de las técnicas anteriores. Cuando se hayan recogido los datos suficientes el uso de esta base basada en el razonamiento debe mejorar la eficacia y exactitud del análisis realizado.

Esta técnica emplea los elementos siguientes:

- Una estructura de la clasificación del concepto para permitir un nivel de entendimiento sobre las entidades referidas por los datos del cambio.
- Un método de representar relaciones de trazabilidad, y un mecanismo para navegar a través de los vínculos.
- Un mecanismo de aprendizaje, que incluye la ayuda para la extracción de la regla (de los datos registrados) y la regla integración (en la base de la regla existente).
- Un sistema analógico para permitir identificar similitud entre escenarios diferentes.

Estructura de clasificación de Concepto

Estructurar la información que se posee sobre un sistema para que puedan hacerse las comparaciones e integrar las nuevas informaciones fácilmente puede ser un problema difícil. Este problema se alivia clasificando todas las entidades extensivamente en una representación del sistema (por ejemplo todos los requisitos, los cambios, y entidades del dominio, el modelo del sistema etc.) en jerarquías de conceptos. Semejante jerarquía se ilustra debajo en la *figura. 9*:

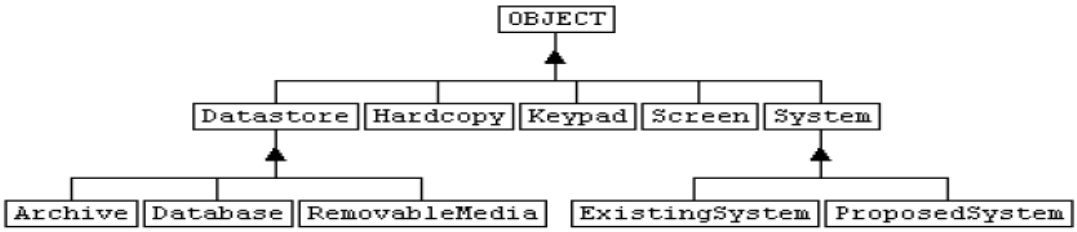


Figura 9. Un ejemplo Concepto Clasificación Jerarquía

Los beneficios de mantener toda la información en las jerarquías pueden resumirse como sigue:

- Todos los conceptos son bien ordenados, bien estructurados y fáciles de manejar.
- La estructura conceptual puede extenderse fácilmente vía herencia, permitiendo ser agregados tipos adicionales de información.
- Es posible identificar y guardar un alto nivel de relaciones entre las clases abstractas.
- El análisis puede generalizarse (es decir clases abstractas, preferibles a las específicas, clases concretas que pueden usarse en el análisis). Esto puede encontrar impactos fantasmas, pero permite análisis de clases vagas y descubrirá todos los impactos reales. Considerando las superclases en lugar de subclases individuales también es posible reducir la cantidad de análisis requerido.
- Debido al estado actual de las tecnologías del software, tal estructura de clasificación se apoya en muchos de los idiomas de la programación de hoy y sistemas de base de datos.

Representación de las relaciones

Pueden usarse diferentes propuestas para guardar las relaciones identificadas entre los componentes del sistema. Sin embargo, la mayoría de los métodos generalmente son similares y están basados en los mismos principios básicos.

Un método para recoger el conocimiento sobre las relaciones es el empleo de ¨ acciones justificadas ¨. Para cada entidad en un sistema se puede guardar cualquier acción (los tipos de

cambio) realizado en él, y la justificación para cada acción. La justificación está compuesta de uno o más acciones que han tenido lugar en las entidades separadas, o la propia entidad, y qué ha llevado a o ha requerido que tenga lugar una nueva acción.

Este tipo de regla tiene la siguiente forma:

< Lista de acciones (justificación) > => < acción >

Una acción no se limita simplemente a la alteración de una entidad, también puede involucrar la eliminación o creación de entidades Por ejemplo:

Creación de una instancia de tipo X = Creación de una instancia de tipo Y

La regla anterior puede interpretarse como "dado la creación de una instancia de X se requiere o podemos predecir la necesidad de la creación de una instancia Y ¨.

La regla debe ser repetida cada vez que la salida se convierta en entrada de otra regla asegurando así que se cumplan todas las relaciones de transitividad.

No importa qué método se use para expresar el conocimiento, una clasificación jerárquica de las entidades del sistema puede simplificar la acción de guardar tal conocimiento. Esto es porque una relación puede asociarse con una superclase cuyas hijas pueden participar en el tipo de relación que se guarda. Esto tiene los beneficios siguientes:

- Reduce la cantidad de datos que se necesita poseer para representar los datos de las relaciones.
- Simplifica la acción de agregar subclases adicionales a la jerarquía de entidades. Las Nuevas entidades heredarán automáticamente las relaciones de los padres cuando se agregan a la jerarquía.
- Puede facilitar el análisis generalizado del tipo abordado en una sección anterior.

2.7.3 Métodos basados en Probabilidad.

En lugar de identificar las relaciones reales de trazabilidad, la técnica basada en la probabilidad proporciona información adicional sobre las relaciones de los componentes previamente identificados. Estas técnicas dependen de valores estimados de las probabilidades asignados a relaciones del trazabilidad dentro de un sistema. En este caso se asigna a la relación de trazabilidad un valor de 'conductividad' (o fuerza de impacto) que represente la probabilidad de que el objetivo de la relación sea traceable desde la fuente.

La escala que se utiliza para los valores de la conductividad puede ser:

- Simple: la probabilidad de la propagación puede ser uno de tres valores: definido, posible o imposible.
- Curso: la probabilidad es representada por un valor en una escala del sistema (ej. 1 a 5).
- Aceptable: probabilidades decimales continuas desde cero (imposible) hasta uno (definido).

La conductividad de una relación puede ser condicional de modo que la probabilidad de la propagación sea diferente, dependiendo del tipo de cambio que es propuesto.

La conductividad de una trayectoria transitiva compuesta (es decir una trayectoria entre dos entidades compuestas de un número de relaciones consecutivas) es igual al valor mínimo de la conducción que aparece en la trayectoria. Un ejemplo abajo en *figura 10*.

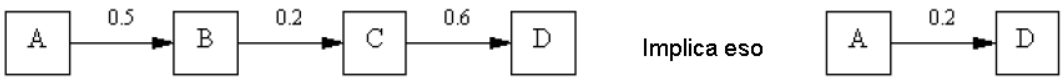


Figura 10. Ejemplo de una trayectoria transitiva cerrada.

En muchos casos más de una trayectoria pueden existir en medio de dos nodos en el diagrama de relación. Para calcular el valor de la conductividad para una trayectoria transitiva la

conductividad de cada trayectoria debe ser calculada primero usando el método de 'reducción al mínimo' abordado arriba. Una vez que se haya hecho esto la conductividad máxima de varias trayectorias se debe utilizar como el valor final.

El valor de la conductividad dado a una relación se puede calcular de la información almacenada sobre las entidades respectivas de la fuente y del objetivo. Usando tal información es posible producir automáticamente una valoración del valor para una relación particular. Alternativamente, los usuarios pueden requerir incorporar un valor que consideren sea apropiado.

Para mejorar la exactitud y el realismo de los valores asignados a cada relación, deben ser ajustados teniendo en cuenta la nueva información. Sobre la duración del ciclo vital del sistema es posible recoger la información relevante que permitirá una valoración o un cálculo más entendido de los valores producidos.

2.7.4 Problemas con técnicas de extracción.

Muchas de las estrategias para la extracción de trazabilidad tienen fallas significativas. La siguiente lista esboza algunas de las más significativas:

Los métodos basados en la dependencia proporcionan un detallado análisis para la información formalizada tal como código fuente y modelos formales (Ej. diagramas de estado finitos), pero tiene poco apoyo para la información no-formal, información abstracta tal como documentos de la lengua natural (Ej. definiciones de requisito). Además de esto, la ayuda que estos métodos proveen para el análisis inter-niveladas es también limitada.

Aunque el análisis previo de la trazabilidad proporciona ayuda para todos los niveles de formalismo, no proporciona un análisis profundizado debido a la naturaleza generalmente vaga de las relaciones previas. Esto es porque la definición informal y vaga de muchos de estos datos observados de trazabilidad hace algunas relaciones importantes difíciles de distinguir. Además, el algoritmo transitivo de cierre usado para identificar los posibles componentes impactados en métodos previos de trazabilidad es ineficaz para la mayoría de los sistemas no triviales. Esto es

debido al número potencialmente enorme de componentes y del número extenso de trazabilidad vinculados entre ellos.

Las estructuras usadas para almacenar la información de trazabilidad pueden llegar a ser rápidamente muy grandes y son así imprácticas para la mayoría de los sistemas no triviales. Además, tales estructuras no proveen la ingeniería de requisitos con un mecanismo obvio para determinar el impacto de cambios propuestos, sino solamente de la propagación de los impactos existentes.

Durante la etapa aprendizaje del método basado en conocimiento se puede hacer poco análisis de trazabilidad. Esto es porque una base de conocimiento de las integraciones anteriores del cambio debe ser construida antes de que se pueda hacer un análisis basado en el conocimiento completo.

Los valores de la propagación usados por probabilidad métodos basados en probabilidad son intrínsecamente inexactos debido al hecho de que deben ser estimados o se calculan de métrica con confiabilidad incierta. Debido al hecho que las técnicas de la probabilidad proporcionan solamente información adicional sobre impactos identificados, no pueden ser utilizadas en ella misma para el análisis de impacto.

No se garantiza ningún método individual para identificar todo vínculo de trazabilidad único dentro de un sistema. Además de esto, todos los métodos identificarán vínculos de trazabilidad que no implican necesariamente una trayectoria de propagación de impacto.

2.8 Métricas para medir el tamaño del impacto.

Aunque es útil visualizar el impacto de un cambio usando una representación gráfica, a veces es deseable enumerar el efecto del impacto en un solo valor numérico. Este valor puede ser particularmente útil durante fase de análisis del costo y de beneficio en el proceso de

administración de cambio. Un valor numérico proporciona una idea más clara de la magnitud del cambio propuesto y ayuda a una estimación más exacta del cambio.
Calculando el valor del impacto de cada uno de los cambios propuestos es también posible comparar varias opciones de aplicación para la inclusión de nuevas funcionalidades en un sistema.

Este valor no es sólo una cuenta del número de componentes afectados por un cambio, debe tenerse en cuenta el hecho de que algunos componentes requieran alteraciones más complicadas que otras, debemos dar un valor que refleje el tamaño y la dificultad del cambio.

2.8.1 Métricas de tamaño físico.

Estos métodos usan métricas de tamaño físicos como líneas de código, líneas de especificación, número de objetos de diseño etc. usando un número de técnicas para estimar la suma física del área de cambio entre las que se encuentran:
La estimación por la analogía: El cambio actual se compara con los cambios similares, previamente documentados.
La estimación por expertos de la aplicación: los llamados 'gurús ' de los sistemas usan su conocimiento para desarrollar una estimación.
La estimación por comparación: El cambio actual es ubicado en una escala de complejidad usando el tamaño de cambios previos cuyo tamaño ya se conoce y que son similares al propuesto.
La estimación vía cálculo: Una característica intrínseca del cambio se evalúa y se usa una fórmula para calcular el tamaño físico inferido.
Existen un conjunto de técnicas para estimar el tamaño físico del cambio pero hay inconvenientes que hacen que el resultado final pueda ser inexacto. Este tipo de métricas se usan mayormente cuando se está trabajando con metodologías ágiles para el proceso de desarrollo de software.

2.8.2 Métrica de puntos de función.

La métrica de puntos de función es mucho más conveniente que las métricas del tamaño físico para estimar el grado de cambio ya que permite estimar el tamaño y la complejidad de un proyecto desde etapas más tempranas del ciclo de vida desarrollo.
Las técnicas basadas en puntos de función se enfocan en calcular la cantidad de funcionalidad implícita en la representación del sistema en lugar del tamaño. Evaluando la cantidad de funcionalidad que debe alterarse y la complejidad de tales alteraciones, se puede hacer una estimación más exacta de la magnitud de cambio.
. Esta métrica propone:

Obtener los puntos de caso de uso sin ajustar

Teniendo la especificación de los requerimientos mediante casos de uso y los actores del sistema se puede hacer una primera estimación del tamaño.

Paso 1:

Identificar las características o funciones del sistema.

- Entradas externas. EI.
- Salidas externas. EO.
- Ficheros lógicos internos. ILF.
- Ficheros de interfaz externa. ELF.
- Consultas (peticiones) externas. EQ.

Clasificar los archivos y transacciones de acuerdo al nivel de complejidad.

La complejidad de las transacciones y los Archivos en el Análisis de Puntos de función, se puede clasificar y cuantificar de acuerdo con los criterios que se muestran a continuación:

ILF, ELF			
	Elementos de Datos		
Archivos	1-19	20-50	51 +
1	Bajo	Bajo	Media
2-5	Bajo	Medio	Alto
6+	Medio	Alto	Alto

Tabla 1. Ficheros lógicos internos. ILF, Ficheros de interfaz externa. ELF

EO, EQ			
	Elementos de Datos		
Ficheros	1-5	6-19	20+
0, 1	Bajo	Bajo	Media
2-3	Bajo	Medio	Alto
4+	Medio	Alto	Alto

Tabla 2. Salidas externas. EO. Consultas (peticiones) externas. EQ

EO, EQ			
	Elementos de Datos		
Ficheros	1-4	5-15	16+
0, 1	Bajo	Bajo	Media
2-3	Medio	Alto	Alto
4+	Medio	Alto	Alto

Tabla 3. Entradas externas. EI

Aplicar los pesos

Se cuenta la cantidad de transacciones por cada Nivel de complejidad (bajo, medio, alto) y se multiplica por el peso asociado en la tabla 4. Todos estos productos se suman y se obtienen los puntos de casos de uso desajustados (UUCP).

	Niveles de Complejidad		
Características	Bajo	Medio	Alto
ILF	7	10	15
ELF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

Tabla 4. Pesos según nivel de complejidad.

Ej.

Características	**Complejidad**			
	Baja	**Media**	**Alta**	**Aporte**
Entradas externas	3*3			9
Salidas Externas				
Consultas externas	1*3			3
Archivos Lógicos Internos	1*7			7
Archivos de Interfaz externos				
Total				19

Tabla 5. Puntos de casos de uso sin ajustar

Por tanto la cantidad de puntos de casos de uso sin ajustar es UUCP = 19

2.8.3 Métrica de puntos por casos de uso.

Existe una relación natural entre los puntos de función y los casos de uso, los Puntos de Función permiten estimar el tamaño del software a partir de sus requerimientos, mientras que los Casos de Uso permiten documentar los requerimientos del software y la especificación de los requerimientos a través de los casos de uso es un método sumamente efectivo para capturar la funcionalidad del sistema que tiene el mismo enfoque que la métrica de puntos de función:

1. **Calculo de casos de puntos de casos de uso sin ajustar.**

Se calcula a partir de:

UUCP = UAW + UUCW donde:

UUCP: Puntos de Casos de Uso sin ajustar
UAW: Factor de Peso de los Actores sin ajustar
UUCW: Factor de Peso de los Casos de Uso sin ajustar

Factor de peso de los actores sin ajustar (UAW):

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema.

Los criterios se muestran en la siguiente tabla:

Tipo de Actor	Descripción	Factor de peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación	1

	(API, Application Programming Interface)	
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

Tabla 6. Criterios para calcular el factor de peso según complejidad de los actores

UAW = ∑ cantidad de actores por complejidad *factor de peso asociado.

Factor de Peso de los Casos de Uso sin ajustar (UUCW):

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia. Los criterios se muestran en la siguiente tabla:

Tipo de caso de uso	Descripción	Factor de peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10
Complejo	El Caso de Uso contiene más de 8 transacciones	15

Tabla 7. Criterio para determinar el factor de peso de los casos de uso según tipo.

UUCW= ∑ cantidad de casos de uso por complejidad * complejidad asociada.

2. Cálculo de Puntos de Casos de Uso ajustados

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar éste valor mediante la siguiente ecuación:

UCP = UUCP x TCF x EF donde:

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

Factor de complejidad técnica (TCF)

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante.

Factor	Descripción	Peso
T1	Sistema distribuido	2
T2	Objetivos de performance o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	El código debe ser reutilizable	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Incluye objetivos especiales de seguridad	1

T12	Provee acceso directo a terceras partes	1
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1

Tabla 8. Significado y el peso de los factores que determinan la complejidad técnica del sistema.

El Factor de complejidad técnica se calcula mediante la siguiente ecuación:

$TCF = 0.6 + 0.01 \times \Sigma (Peso_i \times Valor\ asignado_i)$.

Factor de ambiente (EF)

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del Factor de ambiente. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5. En la siguiente tabla se muestra el significado y el peso de cada uno de éstos factores:

Factor	Descripción	Peso
E1	Familiaridad con el modelo de proyecto utilizado	1.5
E2	Experiencia en la aplicación	0.5
E3	Experiencia en orientación a objetos	1
E4	Capacidad del analista líder	0.5
E5	Motivación	1
E6	Estabilidad de los requerimientos	2
E7	Personal part-time	-1
E8	Dificultad del lenguaje de programación	-1

Tabla 9. Factores y peso a tener en cuenta para el Factor ambiente

- Para los factores E1 al E4, un valor asignado de 0 significa sin experiencia, 3 experiencia media y 5 amplia experiencia (experto).
- Para el factor E5, 0 significa sin motivación para el proyecto, 3 motivación media y 5 alta motivación.
- Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media y 5 requerimientos estables sin posibilidad de cambios.
- Para el factor E7, 0 significa que no hay personal part-time (es decir todos son full-time), 3 significa mitad y mitad, y 5 significa que todo el personal es part-time (nadie es full-time).
- Para el factor E8, 0 significa que el lenguaje de programación es fácil de usar, 3 medio y 5 que el lenguaje es extremadamente difícil.

El Factor de ambiente se calcula mediante la siguiente ecuación:

EF =1.4 - 0.03 x Σ (Peso i x Valor asignado i).

3. De los Puntos de Casos de Uso a la estimación del esfuerzo

Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.

Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

- ✓ Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.
- ✓ Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre.
- ✓ Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

El esfuerzo en horas-hombre viene dado por:

E = UCP x CF

E: esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: factor de conversión

Este método proporciona una estimación del esfuerzo en horas-hombre contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso.

Para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software. Para ello se puede tener en cuenta el siguiente criterio, que estadísticamente se considera aceptable. El criterio plantea la distribución del esfuerzo entre las diferentes actividades de un proyecto, según la siguiente aproximación:

Actividad	Análisis
Análisis	10.00%
Diseño	20.00%
Programación	40.00%
Pruebas	15.00%
Sobrecarga (otras actividades)	15.00%

Tabla 10. Distribución del esfuerzo entre actividades del proyecto

2.9 Propuesta del procedimiento.

La creciente complejidad de las aplicaciones y posteriores cambios hace virtualmente imposible que los responsables de la toma de decisiones y los comités de evaluación puedan controlar el impacto que los cambios pueden tener, el propósito de este procedimiento es proporcionar ayuda a los encargados de realizar el análisis de impacto.

La propuesta de procedimiento para el análisis de impacto de los cambios en el proceso de desarrollo de software consta de seis actividades fundamentales ordenadas de la siguiente forma:

1. Análisis del cambio propuesto.
2. Análisis de la dependencia entre los elementos de configuración de software.
3. Análisis del alcance del cambio.
4. Estimación de Costos y recursos implicados en el cambio.
5. Análisis de costos – Beneficios.
6. Negociación con el cliente y/o solicitante del cambio.
7. Generación del documento de factibilidad de cambio.

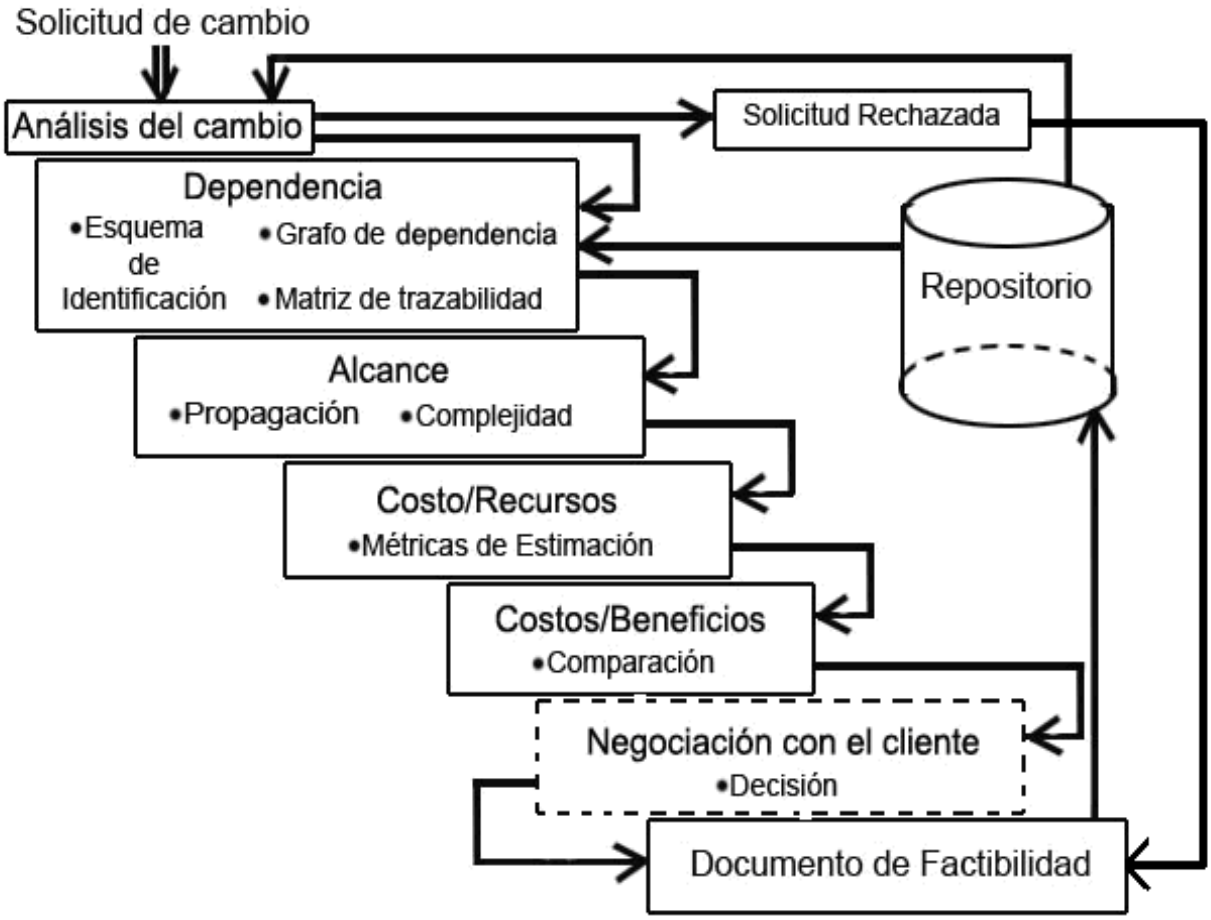


Figura 11. Procedimiento para el análisis de Impacto.

En el análisis y cumplimento de estas actividades intervienen las siguientes personas:

Comité de Control de Cambios:

- Presidente del Comité de Control de Cambios.
- Planificador del proyecto.
- Cliente y/o representante de la Empresa cliente.
- Cualquier Rol del proyecto

2.9.1 Análisis del cambio propuesto.

Esta actividad se concentra en analizar los fundamentos y objetivos del cambio para ello se sugiere:

- Se reúne el comité de control de cambios para hacer un estudio de la información de la planilla de solicitud con el fin de lograr un entendimiento del cambio propuesto.
- Se identifica el elemento de configuración afectado directamente por el cambio.
- Se identifica el flujo de trabajo (Modelado del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba, Instalación, Administración de proyectos, Gestión de Configuración, Ambiente) en el que se desea introducir el cambio y se hace un análisis preliminar del cambio.

Durante la realización de esta actividad puede ser rechazada la solicitud de cambio (Solicitud existente, solicitud rechaza anteriormente etc.)

Responsable	**Tareas a Realizar**
Presidente del comité de control de cambios	Encargado de revisar la solicitud de cambio y chequear que no esté duplicada o haya sido rechazada anteriormente.
Administrador de la Configuración.	Especifica el elemento de configuración afectado directamente por el cambio así como el flujo de trabajo en el que se encuentra dicho elemento.

Tabla 11. Roles y Tareas dentro de la actividad.

2.9.2 Dependencia entre los elementos de configuración de software.

Esta actividad está centrada en un análisis de las relaciones establecidas entre los elementos de configuración para ello se indica:

1. **Definición de un esquema de identificación de las relaciones**

En el estudio realizado se llegó a la conclusión de que las relaciones que mayormente se establecen entre los elementos de configuración serán las de dependencia, composición, sucesión y derivación lo que no significa que no puedan estar presentes las demás relaciones (Ej. equivalencia). Un de esquema de identificación podría ser el propuesto por (Franco, 2003) que consiste en una representación visual compatible con el lenguaje UML en el que los ECS serán representados como componentes UML y las relaciones como dependencias de UML, etiquetadas con el valor “xxxx”. Que define el tipo de relación.

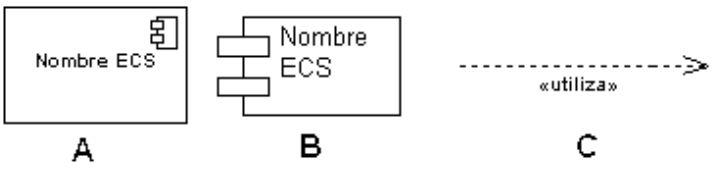


Figura 12. Elementos empleados en el Grafo de Relaciones.

Esta tiene algunos inconvenientes, esta es una propuesta hecha para la herramienta Configcase, el grafo resultante para esta herramienta sólo muestra un tipo de relación: la dependencia, en este caso ¨ utiliza ¨ representa las dependencias de importación. Este tipo de notación resulta confusa si se representan varios tipos de relaciones o si el grafo es muy extenso.

Para evitar estos problemas se sugiere un esquema donde la relación sea representada de forma tal que pueda ser identificada fácilmente:

Figura 13. Relación de dependencia

Figura 14. Relación de derivación

Figura 15. Relación de composición

Figura 16. Relación de sucesión

2. Realizar el grafo de dependencia entre los elementos de configuración de software.

El grafo de dependencia es un grafo cuyos nodos representan los elementos de configuración del sistema. Los elementos de configuración de software están relacionados entre sí y muchos son artefactos generados durante el proceso de desarrollo de software. A medida que progresa el proceso de IS el número de Elementos de Configuración Software (ECS) crece rápidamente, los ECS producen otros ECS para crear una jerarquía de información. Se pueden distinguir las dependencias de rastro inter-niveladas y las dependencias de rastro intra-niveladas. Un nivel

puede estar dado por un flujo de trabajo determinado (Ej. Requerimiento, análisis y diseño, implementación etc.). Haciendo uso de la metodología RUP son originados los siguientes artefactos (ECS):

Figura 17. Relación intra-nivelada en Modelo del Negocio.

En la Fig. 11 el modelo del negocio está compuesto por otros ECS (DCUN, Modelo de Dominio etc.); como muestra la flecha, igualmente, el modelo de dominio está compuesto por otros elementos (Documento Visión, Lista de riesgos y otros). Se muestra además como el DACUN y las especificaciones de los casos de uso tienen una relación de dependencia del DCUN.

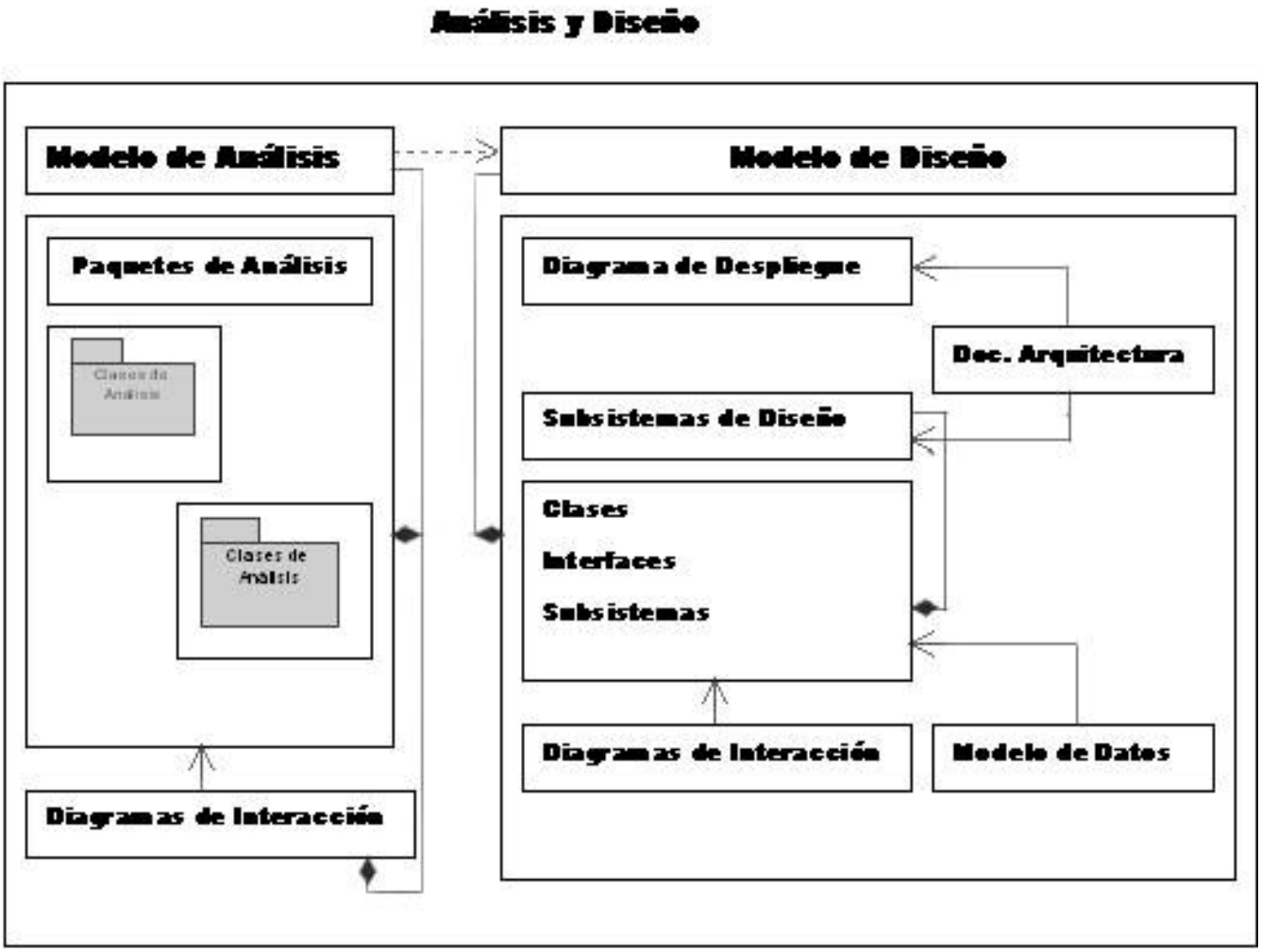


Figura 18. Relaciones entre artefactos del flujo de trabajo de Análisis y Diseño.

A pesar de que en el modelo de análisis hay un refinamiento de los requisitos su objetivo es comprender los requisitos del software y no precisar cómo se implementará la solución, por lo que el resultado de sus actividades sirven de entrada al diseño garantizando este último el cubrimiento de los requisitos funcionales y no funcionales considerando además el entorno de

implementación (Lenguaje de programación, plataforma, Sistemas heredados con los cuales interactuar).

Una clase de análisis podría convertirse en una clase de diseño con el mismo fin, pero podría también para garantizar sus requisitos, tener que dar aparición a clases especificas del entorno de programación, lo cual terminaría en varias clases, y también podría darse el caso en que varias clases simples de análisis, son conjugados en una sola clase de diseño, los mismo puede suceder con Paquetes y Subsistemas.

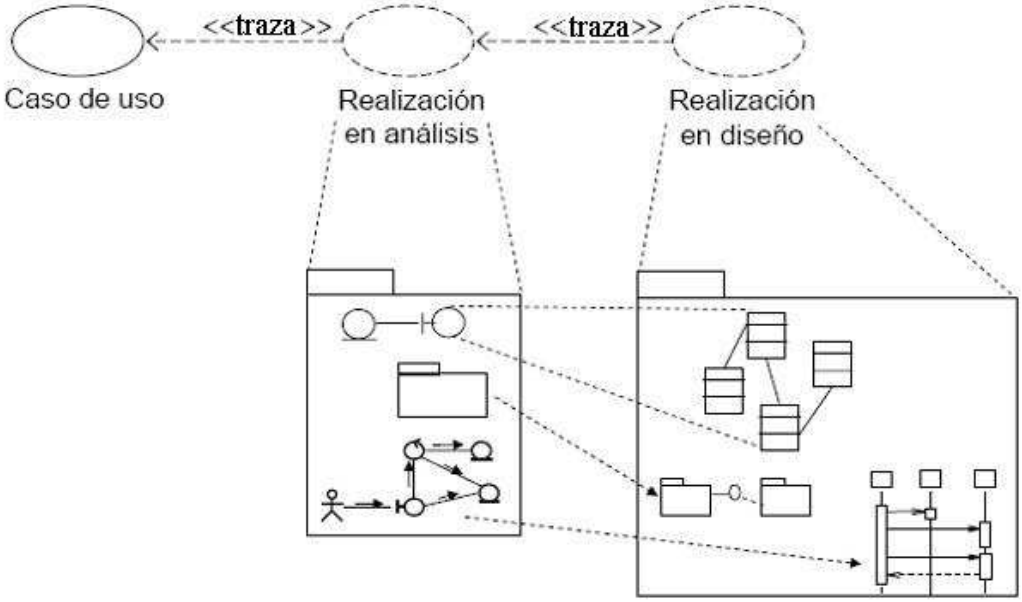


Figura 19. Trazabilidad entre la realización del análisis y la realización del diseño.

De forma similar las clases y subsistemas encontrados durante el diseño son implementadas en términos de componentes: ficheros de código fuente, scripts, ficheros de código binarios, ejecutables y similares durante la implementación.

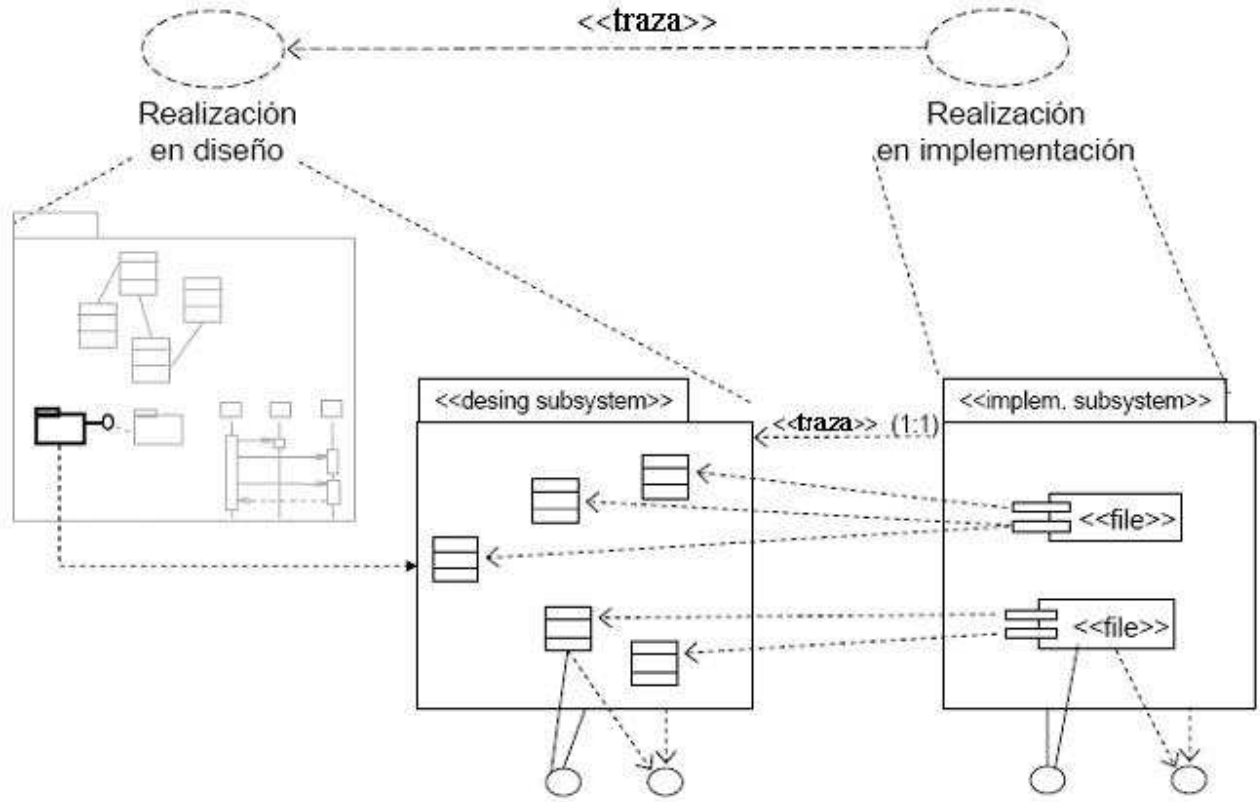


Figura 20. Trazabilidad entre la realización del diseño y la implementación.

Un grafo de dependencia quedaría de la siguiente forma:

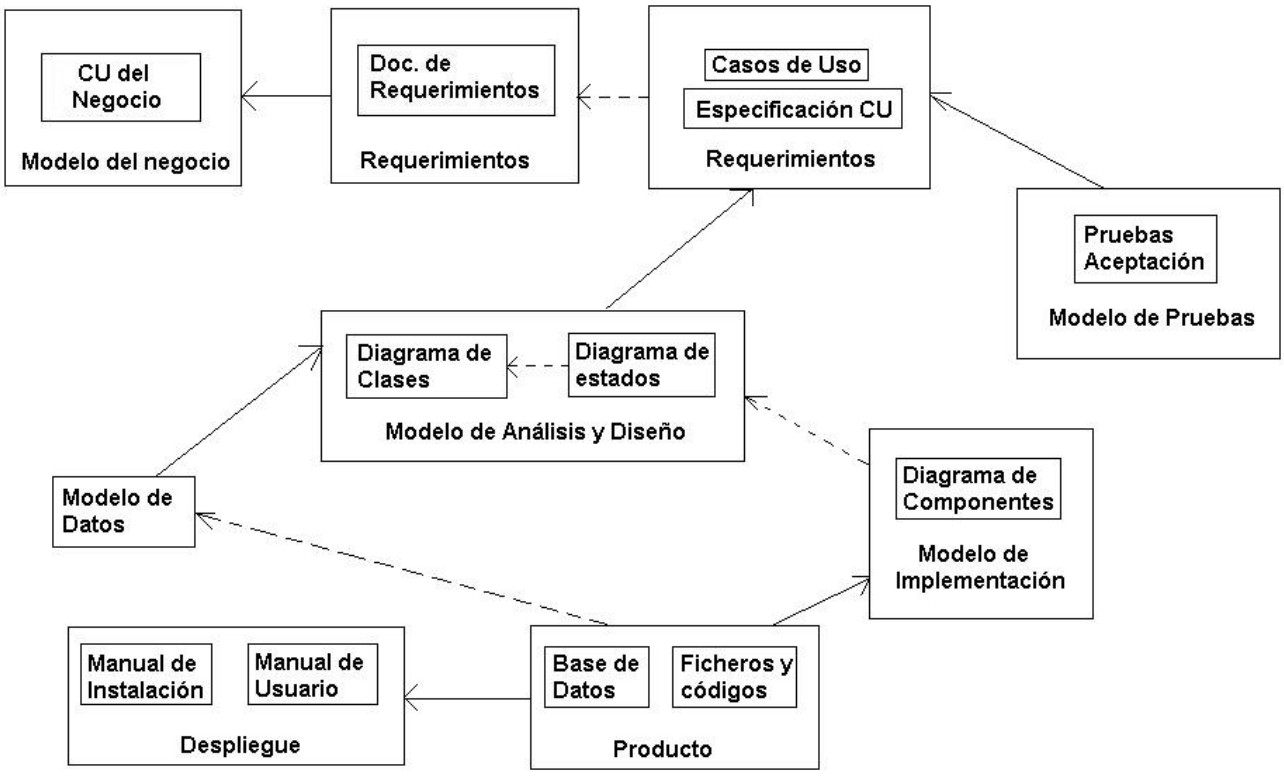


Figura 21. Trazabilidad entre artefactos.

Las relaciones establecidas en este grafo sirven de base para futuros análisis de impacto en proyectos que siguen la metodología RUP.

Del grafo de dependencia se deriva la matriz de dependencia de los elementos de configuración o viceversa, como su nombre lo indica esta matriz muestra la dependencia que existe entre los ECS:

ECS / ECS	ECS1	ECS2	……..	ECS n
ECS1		**X**		**X**
ECS2				**X**
……				
ECS n	**x**			

Cambios a nivel de requisitos

Los cambios a nivel de requisitos son los cambios más comunes en un proyecto ya que los clientes muchas veces no saben lo que quieren o no están muy seguros, los clientes no son expertos en informática pero tienen una clara idea de su negocio y a medida que va avanzando el proceso de desarrollo van teniendo una visión más clara de lo que necesitan y por tanto de las nuevas funcionalidades que requiere, los cambios a nivel de requerimientos generalmente son los cambios pedidos por el usuario. Cambios a los requisitos involucra modificar el tiempo en el que se va a implementar una característica en particular, modificación que a la vez puede tener impacto en otros requerimientos.

Los requisitos deben ser "rastreables": por su origen (¿quién lo propuso?); necesidad (¿por qué existe?); por su relación con otros requisitos; por su relación con elementos del diseño y/o la implementación. Esta información es útil para saber qué afecta un cambio en un requisito. Para resolver esta necesidad se usan las matrices de trazabilidad de los requisitos, entre ellas:

- Matriz de trazabilidad de dependencias: indica cómo se relacionan los requisitos entre sí.
- Matriz de trazabilidad de subsistemas: vincula a los requisitos con los subsistemas (o módulos) que los manejan.
- Matriz de trazabilidad de interfases: muestra cómo los requisitos están vinculados a las interfases internas y externas del sistema.

- Matriz de trazabilidad de requisitos y casos de uso.

Entonces si queremos ver que repercusión ocasiona la introducción de cambios a nivel de requerimientos en los casos de uso:

Casos de uso / Requisitos	CU1	CU2	CU3	CU4
Requisito 1	**x**			**X**
Requisito 1		**x**		
…….				
Requisito n	**x**			

Tabla 12. Ej. Matriz de trazabilidad de requisitos y casos de uso

Si el proyecto está dividido en varios módulos entonces también podríamos ver la trazabilidad de los requisitos hacia los distintos módulos de este proyecto

Módulos / Requisitos	M1	M2	M..	M N
Requisito1	**x**			**X**
Requisito1		**x**	**x**	
…….				
Requisito n				

Tabla 13. Ej. Matriz de trazabilidad de Módulos.

Este tipo de matrices garantizan la trazabilidad de los requisitos a lo largo de todo el ciclo de vida, hacia atrás, hacia delante y entre ellos, además permite proveer una relación entre requisitos, diseño e implementación del sistema.

Esta práctica puede ser costosa y tediosa para sistemas de gran tamaño o complejos. Algunas herramientas CASE posibilitan la generación automática de estas matrices o de forma automática generan elementos de un modelo a otro en diferentes niveles de abstracción (p. ej. del modelo de análisis al modelo de diseño), para obtener así una correlación (mapping) de trazado; sin embargo, no todos los elementos pueden ser correlacionados de forma automática o simplemente no tienen elementos definidos en los modelos destino (Martha, 2006).

Responsable	Tareas a Realizar
Cualquier Rol	Auxiliar en la identificación de los elementos de configuración afectados por el cambio.
Administrador de la configuración	Definir el esquema de identificación de las relaciones entre los elementos de configuración.

2.9.3 Análisis del alcance del cambio.

Esta actividad se concentra en la evaluación del impacto de los cambios y debe incluir:

- Impacto en los requerimientos.
- Impacto en arquitectura.
- Impacto en la implementación

El cálculo del porciento de ECS afectados nos dará una medida global de la propagación del cambio. Aunque esta métrica no es del todo determinante debido a que no tiene en cuenta la complejidad de implementar el cambio da una medida del volumen de ECS implicados en el cambio.

P(ECSA) = (∑ECSA / ∑ECS) * 100

ECS= Elementos de Configuración de Software

ECSA= Elementos de Configuración de Software Afectados

P(ECSA)= Porcentaje de Elementos de Configuración de Software Afectados

Para una mejor evaluación del alcance del cambio de ECS se tendrá en cuenta también la complejidad del cambio en que se ven involucrados cada uno de los ECS.

Para determinar la complejidad se debe tener en cuenta la complejidad de los flujos de trabajo del proceso de desarrollo y la cantidad de ECS afectados por fase de desarrollo.

Una distribución típica de esfuerzo de un proyecto muestra lo siguiente:

	Inicio	**Elaboración**	**Construcción**	**Transición**
Esfuerzo	**5 %**	**20 %**	**65 %**	**10%**
Tiempo Dedicado	**10 %**	**30 %**	**50 %**	**10%**

Tabla 14. Distribución de tiempo y esfuerzo en un proyecto

Un indicador como este no dice que los Elementos de Configuración implicados en un cambio que se encuentran en la fase de elaboración y de transición serían los más costosos en tiempo y esfuerzo por parte de los desarrolladores del proyecto.

Otra de los elementos a tener en cuenta a la hora de medir la complejidad es la distribución de recursos humanos por cada una de las fases de desarrollo de un proyecto

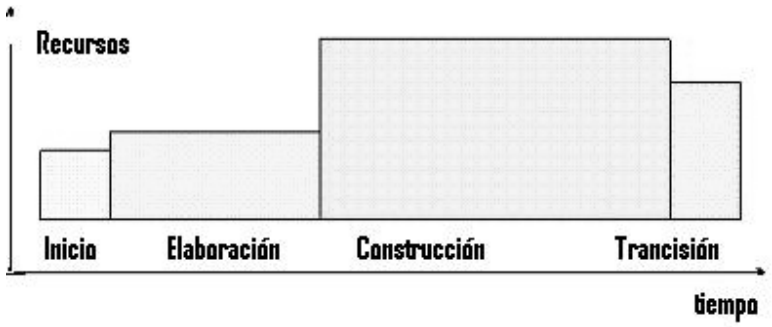


Figura 22. Distribución típica de recursos humanos

La construcción es la fase de desarrollo del software que requiere de una mayor cantidad de recursos humanos por lo que un cambio en el que se vean involucrados un porciento alto de ECS de esta fase podría ser muy costoso.

Otro elemento a tener en cuenta es la cantidad de ECS que afectan la arquitectura del software ya que históricamente los cambios más costosos son los que implican muchos cambios a este nivel.

También debemos determinar el número de ECS que afectan el plan de Riesgo del proyecto.

Consultando las relaciones de trazabilidad, se tiene un análisis de cobertura que sirve para asegurar la completitud y que el tiempo no se gasta inútilmente construyendo alguna parte del sistema que no se ajusta a las necesidades.

Responsable	Tareas a Realizar
Cualquier Rol	Determinar los impactos sobre los diferentes ECS
Jefe del proyecto	Afectaciones al plan de riesgo, determinación de los recursos humanos implicados en el cambio.
Presidente del comité de control de cambios.	Controla la información obtenida del análisis del cambio.

2.9.4 Estimación de Costos y recursos implicados en el cambio.

Esta actividad está dirigida a calcular el esfuerzo que conlleva realizar el cambio.

El costo más significativo asociado con implementar un cambio es el pago de costos de trabajo a los equipos de desarrollo. El costo de un cambio puede así ser considerado para ser una función del trabajo requerido para implementarlo, que alternadamente se relaciona con el grado y la complejidad del cambio. Por lo tanto, dado una valoración del tamaño potencial del impacto de introducir un cambio, (usando uno de los métodos previamente abordados) es entonces posible desarrollar una estimación de costo y del uso del recurso humano.

Para el desarrollo de sistemas enteros, un número de métodos para la estimación del costo se han desarrollado (lo más notablemente el CoCoMo de Boehm). Estos métodos han sido llevados a cabo usando datos históricos de varios y muchos esfuerzos de desarrollo para producir fórmulas generales para el cálculo del costo. El equipo individual de desarrollo puede entonces tomar estas fórmulas generales y con los datos de su propia experiencia individual, la concordancia correcta, entonces iguala su propia dinámica de la productividad del desarrollo.

Al contrario de la estimación de costo para el desarrollo entero del sistema, realizar un solo cambio no requiere el uso de los datos de los proyectos anteriores. Cada cambio de un sistema puede utilizar los datos anteriores de la productividad para todos los otros cambios realizados al mismo sistema. Esto significa que podemos fácilmente calcular el esfuerzo y el costo de poner un cambio en ejecución considerando la complejidad y el tamaño relativo de un nuevo cambio en comparación con la implementación de cambios anteriores.

Lo anterior significa que no necesitamos utilizar un método de costo (ej. cocomo) pero si únicamente un método de prevención (ej. análisis del punto de función).

El análisis de puntos de función se puede aplicar de manera sencilla con los casos de uso mejorando la calidad de los documentos de requerimientos y, a la vez, mejorando la estimación del proyecto de software. El aplicar la técnica de puntos de función permite verificar y validar el contenido de un documento de especificación de Requisitos de software.

Lo interesante de la aplicación de ambas técnicas es que se puede actualizar el conteo de los puntos de función cada vez que los casos de uso cambien y, de esta manera, determinar el impacto que un caso de uso específico puede producir en la estimación del desarrollo de todo el proyecto.

Por todo lo anteriormente planteado para la estimación de los costos y recursos implicados se sugiere la utilización de la métrica de puntos por casos de uso analizada en epígrafes anteriores. La cual fue automatizada en la Universidad de las Ciencias Informáticas, siendo usada ya en varios proyectos productivos. El nombre de esta herramienta es ESTIMAC.

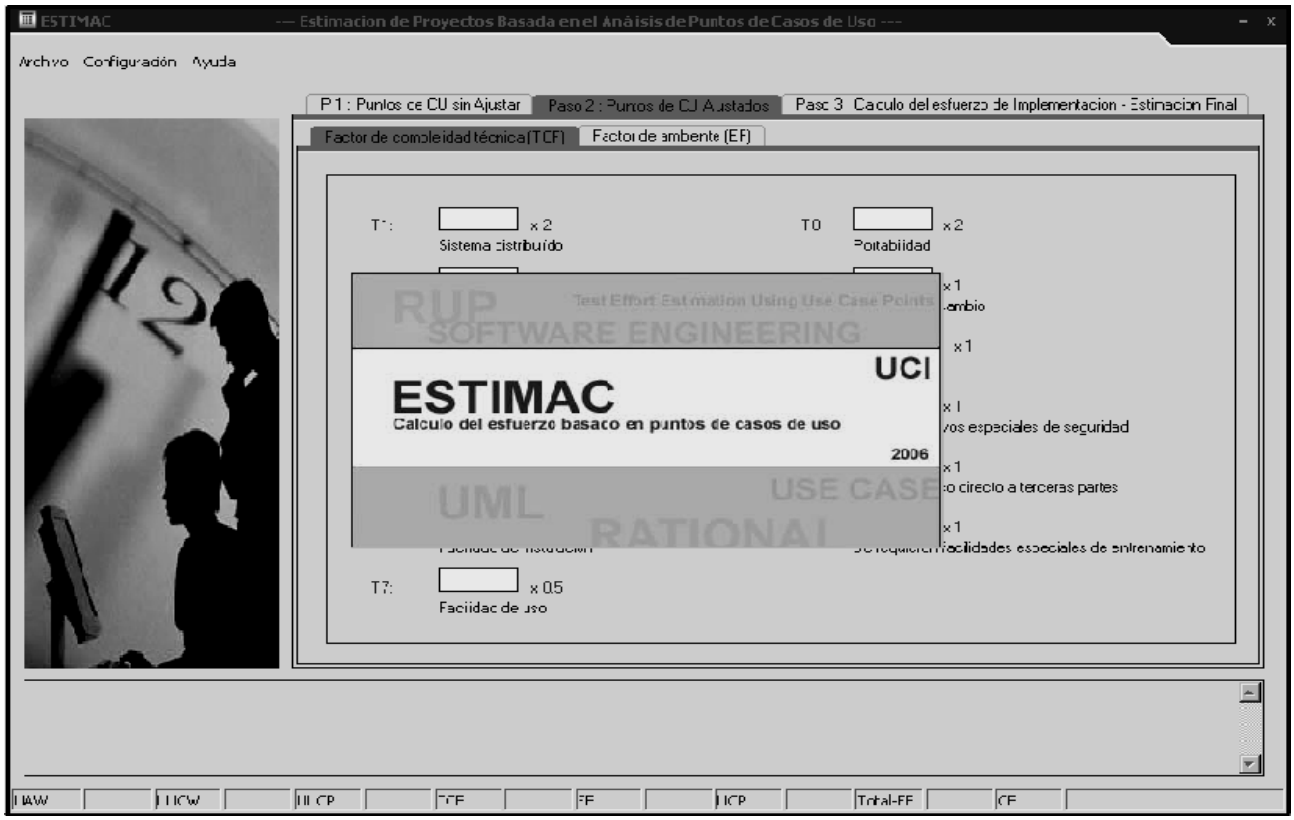


Figura 23. Herramienta para la estimación de esfuerzo ¨ESTIMAC¨

Responsable	Tareas a Realizar
Planificador del proyecto	Responsable de estimar los costos y beneficios implicados en el cambio.

2.9.5 Análisis de costos/beneficios.

Esta actividad proporciona una medida de los costos en que se incurre en la realización de un cambio y comparar dichos costos previstos con los beneficios esperados de la realización de dicho cambio, para determinar si el cambio propuesto es eficaz.
Se realizará una lista de todo lo que es necesario para implementar el cambio y una lista de los beneficios esperados.
En la realización de esta tarea es necesaria la siguiente información:

Para el análisis de los costos:

Personal asignado: Tener un registro de las personas que trabajan en el cambio y la cantidad de tiempo que cada desarrollador necesita para hacer el cambio.
Impactos: Una estimación de todos los componentes afectados por el cambio (determinado por el análisis de impacto).
Recursos estimados: Una estimación del costo, tiempo y esfuerzo original del proyecto.
Recursos reales: Los valores finales para el costo, tiempo y esfuerzo involucrados en la realización del cambio. (Se obtienen de la estimación de recursos implicados en el cambio)
Información de desviación: La diferencia entre los costos estimados y los costos reales de hacer el cambio.

Para el análisis de los beneficios:

Objetivos/ Intenciones de cambio - Razón / razones por la que se hace el cambio.
Beneficios anticipados - Una estimación del efecto positivo del cambio propuesto.
Beneficio real - El efecto positivo real, identificado después de la aplicación del cambio.
La información de desviación - La diferencia entre los anticipados y los beneficios reales de hacer el cambio.

Los costos suelen ser medibles y estimables en unidades económicas, no así los beneficios, los cuales pueden ser tangibles o no tangibles. En un análisis de costos y beneficios se deben considerar aquellos aspectos tangibles, es decir, medibles en valores como dinero, tiempo, etc., y no tangibles, es decir, no ponderables de una forma objetiva.

Aspectos tangibles	Aspectos no tangibles
• Dinero • Tiempo • Calidad	• El aumento de cuentas debido a un mejor servicio a los clientes. • La mejora en la toma de decisiones debido a una mejora del soporte informático. • La valoración de los beneficios no tangibles se debe valorar de una forma subjetiva y será realizada por las áreas correspondientes.

Figura 24. Aspectos tangibles y no tangibles

En la estimación de costos se considerarán, entre otros, los siguientes aspectos:

- Adquisición de hardware y software: El que sea preciso para el desarrollo, implantación y normal funcionamiento del proceso de desarrollo con el cambio introducido. Se debe considerar la saturación de máquinas o sistemas actuales como consecuencia de la entrada en vigor del nuevo cambio.
- Gastos de mantenimiento de hardware y software anteriores.
- Gastos de comunicaciones: Líneas, teléfono, correo, etc.
- Gastos de instalación: Cableado, acondicionamiento de sala, recursos humanos y materiales, gastos de viaje, etc.
- Costo de desarrollo del sistema.

- Gastos de consultoría: En caso de requerirse algún consultor externo en cualquier etapa del proyecto.
- Gastos de formación: De todo tipo (Desarrolladores, Operadores, Implantadores, Usuario Final, etc.).
- Gastos de material: Papel, tóner, etc.
- Costos derivados de la curva de aprendizaje: De todo el personal involucrado: Desarrolladores, Técnicos de Sistemas, Operadores, y desde luego, Usuarios.

En la estimación de beneficios se pueden considerar cuestiones como las siguientes:

- Incremento de la productividad: Ahorro o mejor utilización de recursos humanos.
- Ahorro de gastos de mantenimiento del sistema actual.
- Ahorros de adquisición y mantenimiento de hardware y software, o reutilización de plataformas sustituidas.
- Incremento de ventas o resultados, disminución de costos: Producidos por una mejora de la gestión.
- Ahorro de material de todo tipo: Sustituido por datos electrónicos que proporciona el sistema, como por ejemplo: papel, correo, etc.

Beneficios financieros.

- Otros beneficios tangibles: Ahorro de recursos externos, consultoría, formación, etc.
- Beneficios intangibles: Incremento de la calidad del producto o servicio, mejora de la imagen de la compañía, mejora en la atención al cliente, mejora en la explotación, etc.

Responsable	Tareas a Realizar
Planificador del proyecto	Responsable de estimar los costos y beneficios implicados en el cambio.
Presidente del Comité de control de cambios.	Define y registra la información como resultado de esta actividad.

2.9.6 Negociación con el cliente y/o solicitante del cambio. (Opcional)

Esta actividad consiste en informar y convenir con el cliente la implementación de cambios solicitados y llegar a un consenso o decisión de aceptar o rechazar el cambio. Es opcional, pues no todos los cambios deben ser informados o concertados con el cliente.

Para crear una aplicación de software, una organización de desarrollo necesita, para definir el problema, entrevistar los usuarios para descubrir sus necesidades específicas, y establecer el alcance del proyecto a desarrollar. Cuando se determina el alcance del proyecto, el líder del proyecto debe trazar pautas entre el que desea el sistema para ser utilizado y el diseñador que debe crear el sistema dentro de un marco de tiempo y un presupuesto fijo.
Naturalmente, los clientes buscan aumentar al máximo la funcionalidad entregada por el costo mínimo, así como las organizaciones de desarrollo buscan minimizar el riesgo y aumentar al máximo el retorno en la inversión. Lo que se necesita para resolver este dilema es la habilidad de negociar.

Aquí se muestran algunas pautas a seguir para una negociación eficaz:

- Establecer el beneficio mutuo como una fuerza motivadora.
- Negociar desde una posición de franqueza.
- No permanecer conforme y expresar el desacuerdo honestamente.
- Evitar discusiones sin sentido.
- Estar dispuesto a comprometerse.

Para negociar el alcance eficazmente, es necesario tener la información exacta y útil sobre los costos relativos y beneficios de un requisito. Durante el proyecto es más importante determinar el efecto de un cambio de requisito bajo su desarrollo en el sistema. Para determinar el impacto

probable de trabajo adicional, es necesario entender los riesgos, beneficios, costos, y esfuerzos asociados con el cambio(s).

La determinación de los riesgos está basada en el éxito global del proyecto. Es útil priorizar los cambios basándose primero en el riesgo más alto, seguido por el esfuerzo y prioridad del cliente.

Un riesgo es cualquier situación que puede impactar el éxito del proyecto negativamente siendo afectado por numerosos factores, incluyendo:

- La sensibilidad política.
- La dificultad técnica.
- El efecto en la moral del equipo.
- La percepción del cliente de progreso.
- El compromiso de administración.
- Impacto en la planificación.
- Impacto en el costo de entrega.
- Impacto en la calidad de producto.
- El efecto en la arquitectura del sistema.

El costo es determinado por la cantidad de dinero / tiempo que debe pasarse para producir la funcionalidad requerida. Este es por consiguiente un factor combinado de esfuerzo y horario.

La prioridad es normalmente determinada por el cliente, pero puede determinarse por el equipo de desarrollo de proyecto para los cambios del sistema internamente generados.

Los casos del uso (y los requisitos cambiados) puede clasificarse rápidamente en estas categorías que usan una escala simple (Alto, Medio, Bajo),

Caso de uso	**Esfuerzo**	**Riesgo**	**Costo**	**Prioridad**

				del cliente
UC1	H	H	H	1
UC2	H	L	L	5
UC3	M	M	M	4
UC4	M	H	L	3
UC5	L	L	H	2

Esta técnica le permite al líder del proyecto negociar con el cliente basándose en un sólido análisis cualitativo del beneficio relativo y qué se arriesga para un cambio de requisito dado.

El resultado es un rendimiento del proyecto que satisfará las necesidades más críticas del cliente, y permite la flexibilidad suficiente al equipo de desarrollo para producir el software de una manera oportuna.

Algunos criterios que se pueden tener en cuenta para tomar la decisión de aprobar o rechazar las solicitudes de cambio son los siguientes:

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

Responsable	Tareas a Realizar
Líder del proyecto	Informar y convenir la realización o no del cambio con el cliente.
Presidente del comité de control de cambios.	Detallar al cliente los argumentos como resultado del análisis realizado.
Cliente o solicitante del cambio.	Conciliar la realización o no del cambio.

2.9.7 Generación del documento factibilidad de cambio.

Una vez que se analizan todos los posibles impactos que un cambio puede producir en el proceso de desarrollo de software se procede a generar el documento de factibilidad de cambio que recoge todos los aspectos que deben ser guardados, Este documento debe llenarse independientemente de la decisión tomada ya que servirá para futuras evaluaciones. (Anexo 1)

El documento está conformado de la siguiente manera:

1. Proyecto : Corresponde al nombre del proyecto
2. Número de la Solicitud de cambio: Código o identificación de la solicitud.
3. Tipo de Solicitud: Se refiere a la clasificación del objetivo del cambio.(Problema o Mejora)
4. Fecha: Fecha en que se realiza la evaluación para la aceptación o rechazo del cambio.
5. Solicitante: Se refiere a la persona que solicita el cambio.
6. Descripción del cambio: Breve descripción del cambio a evaluar.
7. Flujo de trabajo actual: flujo de trabajo en el que se encuentra el proyecto en que es realizada la evaluación.
8. Porcentaje de elementos de Configuración afectados: Se refiere al porciento de elementos de Configuración que son afectados por la realización del cambio.

9. Elementos de configuración afectados por flujos de trabajo: Muestra los ECS afectados por flujos de trabajo.
10. Método utilizado para el análisis de dependencia: Se refiere a las estrategias o técnicas utilizadas para la evaluación del cambio.
11. Impacto en el plan de riesgo: Se refiere a efecto que causa el cambio sobre el plan de riesgos del proyecto.
12. Impacto en el rendimiento del producto: Se refiere a las consecuencias que trae el cambio sobre el uso de memoria y CPU etc.
13. Costos de implementación: Se refiere a la información de desviación estimada para el costo.
14. Beneficio de implementación del cambio: Se refiere a la información de desviación estimada para los beneficios.
15. Prioridad para el solicitante: Define cuan significativo es el cambio para la persona que lo solicita.
16. Recursos disponibles: Se determina los recursos con los que cuenta el proyecto (Hardware, software, recursos humanos, etc.)
17. Aceptación/Rechazo: se plasma la decisión de aceptación o rechazo del cambio.
18. Razón de rechazo: Causa fundamentales por la que el cambio fue rechazado.
19. Existencia de alternativas: Se enuncian las posibles soluciones alternativas que suplanten la realización del cambio.
20. Efectos reales del cambio: Breve descripción de los resultados reales teniendo en cuenta la decisión tomada.

Responsable	Tareas a Realizar
Presidente del Comité de control de cambio	Genera el documento factibilidad de cambios.

Plantilla de factibilidad de cambio.

Proyecto	Nombre del proyecto
Número de la solicitud de cambio	Código de la solicitud
Tipo de Solicitud	Problema o Mejora
Fecha	Fecha de evaluación
Solicitante	Quien solicita el cambio
Descripción del cambio	
Flujo de trabajo actual.	
Porcentaje de elementos de Configuración afectados	
Elementos de configuración afectados por flujos de trabajo.	
Negocio	
Requerimientos	
Análisis y Diseño	
Implementación	
Pruebas	
Despliegue	
Admin. Configuración	
Gestión de proyecto	
Ambiente	

Método utilizado para el análisis de factibilidad
Impacto en el plan de riesgo
Impacto en el rendimiento del producto(uso de memoria y CPU)
Costo de implementación del cambio
Beneficio de implementación del cambio
Prioridad para el solicitante
Recursos disponibles
Complejidad de implementación del cambio
Aceptación/Rechazo
Razón de rechazo

Existencia de alternativas
Efectos reales del cambio

2.10 Criterio de especialistas.

Se hizo una valoración por algunos especialistas de la propuesta de procedimiento para el análisis de factibilidad de cambios en el proceso de desarrollo de software. Para la obtención de esta se elaboraron las siguientes preguntas:

1. ¿Se realiza un análisis de impacto en la empresa o proyecto al que pertenece?
2. ¿Siguen algún tipo de procedimiento? En caso de ser afirmativa su respuesta argumente la misma.
3. ¿Cree necesaria la utilización de un procedimiento para una mejor evaluación del alcance del cambio?
4. ¿Considera que el procedimiento mostrado beneficia el proceso de GCS apoyando el aseguramiento de la calidad, ajuste de los cronogramas, la gestión de compromisos y costos de proyecto?

Estas preguntas fueron respondidas por los especialistas alcanzándose los siguientes resultados:

Especialista: Eugenia Genoveva Muiz Lodos

Correo: emlodos@uci.cu.

Años de experiencia: 34 años.

Pregunta 1

R/ Si se realiza un análisis de impacto.

Pregunta 2

R/ Si, se reúne el equipo de trabajo y se valora lo que representa en tiempo y recurso hacer el cambio, como repercute y quien o quienes deben realizarlo. Se analiza también la afectación al plan mensual.

Pregunta 3

R/ Si creo necesaria la utilización de un procedimiento ya que esto serviría de guía a los encargados de realizar el análisis de impacto.

Pregunta 4

R/ Si, creo que el procedimiento propuesto cumple con las actividades principales a considerar así como tiene en cuenta las personas que deben participar en ella asegurando que todos los aspectos a tener en cuenta a la hora de realizar el análisis de impacto sea valorados.

Especialista: Ing. Arnaldo Gandol Álvarez.

Correo: aga@uci.cu

Años de experiencia: 3 años

Pregunta 1

R/ Si se realiza un análisis de impacto.

Pregunta 2

R/ No existen procedimientos formales son todos empíricos o sea se organizan reuniones etc y se hablan los problemas o beneficios que traería consigo la implementación del cambio.

Pregunta 3

R/ Si sería bueno la utilización de un procedimiento estándar ya que todos los procesos llevan una metodología para proceder, una forma de llevarlos a cabo y no deben ser empíricos.

Pregunta 4

R/ El procedimiento es muy abarcador y adecuado permite visualizar el costo impacto y aceptación del cambio incluso a largo plazo ya que este incluye varias etapas de análisis en detalles de todo el trance del cambio e incluye análisis desde el punto de vista técnico, económico y de satisfacción por parte del cliente del producto.

Especialista: Ing. José Kadir Febrer Hernández.

Correo: kadirf@uci.cu

Años de experiencia: 10 años.

Pregunta 1

R/ Si se realiza un análisis de impacto.

Pregunta 2

R/ Se utilizan procedimientos en dependencia del área de trabajo.

Pregunta 3

R/ La utilización de un procedimiento sería beneficioso para una mejora en los resultados de los procesos de desarrollo de software.

Pregunta 4

El procedimiento realizado contempla aspectos significativos en el perfeccionamiento del producto. El uso de este podría introducir una mejora significativa en la planificación, influyendo también de forma positiva en la calidad de los productos que se desarrollen ya que incluye una serie de actividades y análisis de varios puntos de vista que son necesarios en la toma de decisiones a la hora de realizar los cambios.

De los criterios dados por parte de los especialistas se ha podido concluir que el procedimiento propuesto para el análisis de factibilidad de cambios cumple con los objetivos y expectativas concebidos. Existe un consenso por parte de estos en su importancia para la utilización del mismo en el proceso de desarrollo de software, considerándolo abarcador y apropiado ya que contempla una serie de actividades y análisis necesarios para una correcta toma de decisiones.

2.11 Conclusiones

En este capítulo se analizaron las actividades del proceso de Gestión de Configuración involucradas con el análisis del impacto del cambio, además de un estudio de las diferentes clasificaciones existentes para los métodos de análisis de impacto y las técnicas de estimación existentes para el análisis de costos y beneficios. El resultado de este análisis sirvió de base para hacer un procedimiento que basado en los ECS proporcione ayuda en la factibilidad de

cambios dentro del proceso de desarrollo de software catalogándose dentro de los métodos semánticos

CONCLUSIONES GENERALES

Con la culminación del presente trabajo se dan por cumplidos las tareas y por tanto el objetivo fundamental de este trabajo, arribando a las siguientes conclusiones:

- Se hizo un estudio de los conceptos fundamentales de la Gestión de Configuración que permitió obtener el máximo de información de los elementos a manejar en el desarrollo de este trabajo.
- Se hizo una valoración de las herramientas existentes que apoyan la Gestión de Configuración reafirmando la importancia de la existencia del procedimiento realizado.
- Se estudiaron las técnicas que existen para el análisis de impacto de los cambios en el proceso de desarrollo de software y se valoró la utilización de alguna de ellas.
- Se realizó un estudio de los métodos de estimación para el análisis de costos y recursos proponiendo la aplicación de unos de ellos en el procedimiento de análisis de impacto realizado.
- Se hizo una propuesta de un procedimiento para el análisis de factibilidad de los cambios en el proceso de desarrollo del software, contribuyendo al aseguramiento de la calidad, ajuste a los cronogramas expuestos, la gestión de compromisos y estimación de costos del proyecto.
- Evaluación de resultados a través del criterio de especialistas corroborando la trascendencia del procedimiento propuesto.

RECOMENDACIONES

- La utilización de este procedimiento en la automatización del proceso de análisis de factibilidad de cambios.
- Crear una base de conocimiento que facilite el análisis de dependencia entre ECS.
- Profundizar el estudio del análisis de impacto en los programas docentes.
- La creación de una herramienta que proporcione los mecanismos necesarios para la visualización gráfica de las relaciones existentes entre los ECS.
- Adicionar a la herramienta anterior soporte para el Escaneo y propuesta automática de otras relaciones entre ECS.
- La creación de una herramienta que proporcione un alto nivel de integración de las actividades de la Gestión de Configuración.

REFERENCIAS BIBLIOGRÁFICAS

A.BABICH, W. 1986. Software Configuration Management:: Coordination for Team Productivity. 1986.

Antonio, Angelica de. 2001. *LA GESTIÓN DE LA CONFIGURACION.* 2001.

BAMFORD, R. A. W. J. D. 1995. "Configuration Management and ISO 9001. 1995.

Brad, Appleton. 2000. SCM definitions. 2000.

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

Calidad de Software y la empresa, enseñanza de un tema imprescindible para el Ingeniero Informático.

Estrada, Aylin febles. 2003. Un modelo de Referencia para la Gestión de Configuración en la PYME de Software. 2003.

Estrada, MsC. Lic. Ailyn Febles. 2003. Un modelo de Referencia para la Gestión de Configuración en la PYME de Software. 2003.

innevo. http://www.innevo.com. *http://www.innevo.com.* [Online]

Institut, American National Standards. 1987. IEEE Guide to Software Configuration Management. 1987.

INSTITUTO-NACIONAL-ESTADISTICA-INFORMATICA. 1999. *Herramientas Case.* 1999.

ISO-10007. 1995. Quality Management - Guidelines For Configuration Management. 1995.

ISO-12207. ISO 12207.

ISO-9000. Selección y uso de la tercera edición de las normas ISO 9000.

kanav. http://www.kanav.com. *http://www.kanav.com.* [Online] http://www.kanav.com.

Lic. Ailyn Febles Estrada Dra. Sofía Alvarez Cárdenas, Ing. Humberto Fernández Yanes. Calidad de Software y la empresa, enseñanza de un tema imprescindible para el Ingeniero Informático.

Lock, Simon. 1998. Requirement Level Change Management and Impact Analysis. 1998.

Tabares, Martha Silvia.2006. Realizándose un análisis y valoración de las herramientas que brindan soporte al proceso de control de cambios a nivel mundial. Escuela de Ingeniería de Antioquia, Medellín (Colombia). Revista EIA, ISSN 1794-1237 Número 6, p. 33-42. Diciembre.

Martínez, Ramses Delgado. 2006. ConfigCASE 3.0 Herarmienta de apoyo a la Gestión de configuración. Propuesta arquitectónica.: Facultad de Ingeniería Industrial Centro de Estudios de Ingeniería y Sistemas. Habana : s.n., 2006.

MSDN. 2005. Microsoft Visual SourceSafe Roadmap, Microsoft Development Network. 2005.

Navarro., Ing. José Angel Franco. 2006. Entorno Unificado para la Gestión de Configuración de Software. 2006.

Peralta, Mario. Estimacion del esfuerzo basado en casos de uso. Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS) Escuela de Postgrado. Instituto Tecnológico de Buenos Aires.

Pressman, Roger S. 2002. Ingeniería de Software. 2002.

José Ramón Álvarez Sánchez, Manuel Arias Calleja.2002. Análisis, Diseño y Mantenimiento del software. Dpto. de Inteligencia Artificial - ETSI Informática – UNED

Rancán, Lic. Claudio Jorge. 2003. G*estión de Configuración de Productos Software en Etapa de Desarrollo.* 2003.

Rational. 2003. Rational. 2003.

Sergio R. Machuca, Gabriela A. Sasco, and Natalia Chiaro. 2002. Análisis de impacto en la gestión de cambios de servicios de Telecomunicaciones. 2002.

Shopper , 2007. www.shopper.cnet.com

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

Tuya, Javier 2004.Gestión de la Configuración del Software. Universidad de Oviedo. Departamento de Informática. TII

Wiegers, K. 1998. Molding the CMM to your organization. Software Development. 1998.

2007. http://teleformacion.uci.cu/. [Online] 2007.

GLOSARIO DE TÉRMINOS

A

Administración de requisitos: Se refiere al enfoque sistemático para obtener, organizar y documentar los requisitos del sistema, así como mantener un cuerdo entre el cliente y el equipo del proyecto en los cambios a los requisitos.

Artefactos: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

B

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.

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

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

Biblioteca de producción: Está compuesta por la Biblioteca de trabajo, la de integración y la Biblioteca de Soporte al Proyecto.

Biblioteca maestra: Es donde se almacenan 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.

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.

Biblioteca de backup: Su contenido no está sujeto a Gestión de Configuración (las copias contenidas en ella no están catalogadas en los registros de Gestión de Configuración).

C

Check out: Operación que desprotege un elemento en un control de versiones para permitir su modificación.
Check In: Protección de un elemento en el control de versiones. Sus datos pasan a estar de nuevo bajo el control del sistema.

Ciclo de vida: Se refiere al proceso que se sigue para construir, entregar y hacer evolucionar el software, desde la concepción de una idea hasta la entrega y el retiro del sistema.

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

Componentes: Un componente es una parte no trivial, casi independiente, y reemplazable de un sistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida. Un componente se conforma y provee la realización física por medio de un conjunto de interfaces.

CVS: Sistema de control de versiones (Control Version System)

E

ECS: Elementos de Configuración de Software

G

GCS: Gestión de Configuración de Software.

I

IDEs: Entornos de desarrollo integrado.

Integración: Operación en la que se mezclan los contenidos de varias líneas de desarrollo.

Iteración: Es una secuencia de actividades con un plan establecido y criterios de evaluación, cuyo resultado es una versión del software.

IS: Ingeniería de Software.

P

Procedimiento: Sucesión cronológica de operaciones concatenadas entre sí, que se constituyen en una unidad de función para la realización de una actividad o tarea específica dentro de un ámbito predeterminado de aplicación. Todo procedimiento involucra actividades y tareas del personal, determinación de tiempos de métodos de trabajo y de control para lograr el cabal, oportuno y eficiente desarrollo de las operaciones.

R

RCS: Revision Control System.

RTF: Revisión Técnica Formal.

RUP: Rational Unified Process.

S

SCM Software Configuration Management: Gestión de Configuración de Software

Stakeholder: cliente/usuario del sistema.