Universidad de las Ciencias Informáticas

FACULTAD 5

Propuesta de estrategia para la Gestión de la Configuración de Software haciendo uso de Metodologías Ágiles de Desarrollo.

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autor: Danae Camara González

Tutores: Ing. Yurian Díaz Capote

Ing. Felipe Pérez Hernández

La Habana, Junio de 2011

"Año 53 de la Revolución"

Declaración de Autoría

Declaro ser la única autora de este trabajo y concedo a la Universidad de la Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los días de junio de 2011.

_____

Autor: Danae Camara González

_____ _____

Tutor (es): Ing. Yurian Díaz Capote Ing. Felipe Pérez Hernández

Tutora:

Nombre: Yurian Díaz Capote.

Título: Ingeniera en Ciencias Informáticas.

Categoría docente:

Email: ydcapote@uci.cu.

Graduada como Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Actualmente se encuentra desarrollando la labor de Profesor Instructor y vinculada al Centro de Desarrollo de Informática Industrial (CEDIN) como Administradora de Configuración.

Tutor:

Nombre: Felipe Pérez Hernández.

Título: Ingeniero en Ciencias Informáticas.

Categoría docente: Profesor adiestrado.

Email: fphernandez@uci.cu.

Graduado como Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2009. Se encuentra desarrollando la labor de profesor de Práctica Profesional y vinculado al departamento de Integración y despliegue del Centro de Desarrollo de Informática Industrial (CEDIN).

Un verdadero héroe es aquel que con sus actos desinteresados, logra ganarse un lugar en el corazón de las personas. Estos agradecimientos van para mis héroes:

Principalmente a mi mami por apoyarme, quererme y comprenderme como nadie; la idolatro y la admiro.

A mi papi, que ha sabido educarme y darme su amor y comprensión, lo amo.

A mis abuelos porque me han convertido en su razón de ser.

A mi hermano David y mi hermana de alma Laura, por su preocupación, apoyo, todo su cariño y hasta por las discusiones tontas que hemos tenido.

A Fidel y a la Revolución por darme la oportunidad de estudiar en la Universidad del Futuro, en la que me he convertido en una buena estudiante.

A Yeniset le debo todo la ayuda incondicional brindada en la elaboración del trabajo.

A mis amigos incondicionales, a los que les adeudo la paciencia de tolerarme las espinas más agudas; los arrebatos de humor, los temores, las duda: Adriano, Adriel y Juli, les agradezco cada minuto, cada segundo que hemos compartido juntos desde el momento en que nos conocimos. Eileen, Sule, Yani, Titi, a pesar de ser muy diferentes, nos une un lazo de amistad tan grande como a 3 metros sobre el cielo. Ley (cacharro), el primer compañero de aula con el que conversé y se quedó prendido en mi alma. Nelis, Barzi, Elena, Daylen, Lulú, Luis Enrique, Oscar, Arlan, Osmar, Andy, El belo, nani, Enier, Ronni, Beatriz, Rosaura, Héctor, por ser amigos y compañeros de parrandas y de estudios, por compartir conmigo las alegrías y penas de su vida.

Agradezco a Alain por haberme brindado todo su apoyo, amor, amistad y confianza.

A mis tutores por su preocupación y apoyo.

A la decana Mayra porque es una excelente persona.

Al profesor Millet por ser excepcional y ayudarme como lo ha hecho.

A todos aquellos que de una forma u otra contribuyeron en el desarrollo de esta investigación y que han influido en mi educación.

Dedico este trabajo y todo el esfuerzo realizado en él, principalmente a mi mami, que es mi razón de ser y a quien le debo todo el amor del mundo.

A mi papá, por hacer de mí la persona que soy hoy día, y a mi hermano que a pesar de estar lejos es mi gran amor.

A mis abuelos, para que sepan que sus sacrificios no han sido en vano y se sientan orgullosos de su única nietecita.

Quiero dedicar también esta investigación a mis familiares, a los cuales no les ha faltado una palabra de aliento y cariño hacia mi persona.

A Laura, la hermana que no me dio la naturaleza pero sí la sociedad y el convivir diario.

A mi decana Mayra, a Fidel, a los profesores de la Facultad 5 que han influido en mi educación universitaria y me han apoyado en actividades importantes.

A todos los amigos que he encontrado en el camino de la vida y me han demostrado su cariño y el valor de su amistad.

Los Quiero!

La Industria Cubana de Software está encaminada, según las tendencias actuales, a la construcción de sistemas cada día más eficientes y complejos. De modo que se ha propuesto alcanzar un lugar relevante en el mercado mundial, en la rama de la industria del software, donde la Universidad de las Ciencias Informáticas contribuye a lograr este objetivo. En esta universidad se han creado centros de desarrollo de software, como el CEDIN (Centro de Desarrollo de Informática Industrial), en el cual se necesita definir una guía de procesos para aplicar la Gestión de la Configuración de Software en proyectos que utilicen Metodologías Ágiles de Desarrollo, para asegurar la calidad en la entrega de los productos realizados.

En el presente trabajo se efectúa una propuesta de estrategia para el departamento de Integración y Despliegue. De manera introductoria se exponen algunos conceptos clave para entender el proceso, se resalta la importancia que tiene la Gestión de Configuración en el desarrollo de productos y servicios de software y los riesgos de no realizarla. Además en la propuesta se definen procedimientos a aplicar con el objetivo de mantener un control sobre los elementos que se generan durante el desarrollo de un producto. También se mencionan los principales roles que intervienen en el proceso y cuáles son sus responsabilidades, se describen las distintas actividades que conforman el proceso de gestión de configuración, indicando en cada una de ellas tareas, entradas, salidas, etc.

Palabras clave: Elementos de Configuración de Software, Gestión de la Configuración de Software, Línea base.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Introducción

La producción de software crece a un ritmo acelerado, los ingresos por exportación de software a nivel mundial han alcanzado cifras astronómicas, lo que demuestra la existencia de un mercado nada fácil de conquistar y el hecho de que los software requieran de una mayor calidad, ya que en ocasiones los resultados no llegan a ser fructíferos; los costos pueden ser muy elevados y la producción muy baja. Esto generalmente sucede cuando no se aplican debidamente las técnicas de Ingeniería y Gestión de Software, o porque se trabaja con métodos asistemáticos, sin definición de procesos ni Gestión de la Configuración.

La Gestión de la Configuración de Software (GCS) es un tema muy importante en la rama de la Informática y las Telecomunicaciones; el no administrar correctamente el proceso de desarrollo de un software puede llevar al fracaso cualquier proyecto, al no llevar un control adecuado de los elementos de software y las acciones que se realizan sobre ellos. Por esto es necesaria la Gestión de la Configuración de Software, ya que es el proceso que controla la evolución e integridad de un producto mediante la identificación de sus elementos, el control de los cambios sobre ellos, y la verificación, registro e informe de la información de configuración.

La industria cubana del software está llamada a constituir un renglón de enorme importancia para la economía del país, donde la Universidad de las Ciencias Informáticas juega un papel fundamental, contando dentro de ella con diversos centros de desarrollo, como el Centro de Desarrollo de Informática Industrial (CEDIN), el cual tiene entre sus metas desarrollar la producción de software. En este centro existen proyectos que utilizan Metodologías Ágiles de Desarrollo de Software (MADS), las cuales, rigiéndose por el Manifiesto Ágil, no definen para sus marcos de trabajo la Gestión de la Configuración de Software entre otros grupos de procesos, provocando que surjan inconvenientes como:

- ✓ Los elementos de configuración de los productos no son identificados correctamente, y por tanto, no se establece un adecuado esquema de identificación para estos elementos y sus versiones.
- ✓ No se controlan los artefactos generados por los desarrolladores, de esta manera no se puede controlar el trabajo de los mismos.
- ✓ Las versiones que se generan durante el desarrollo del producto no son revisadas como es debido, provocando así conflictos durante la integración de componentes.

- ✓ En determinados momentos del proceso de desarrollo de un producto, se deben agrupar todos los elementos previamente revisados para mantener un control de los mismos en un futuro, esto no se realiza si no se aplica un proceso que guie la actividad y oriente en la creación de líneas bases que contengan estos elementos.
- ✓ Si no se realizan revisiones esquemáticas, no se pueden detectar con antelación los problemas en el software afectando esto finalmente la calidad en los productos.
- ✓ Es muy difícil reutilizar funcionalidades desarrolladas previamente sin el uso de una herramienta que facilite la actividad.

Esta situación trae como consecuencia que se invierta un mayor esfuerzo en la corrección de un error que aparezca en el software, los especialistas se enfrentan a escenarios en los que los esfuerzos y conocimientos humanos no bastan, es necesario encontrar apoyo en algo más, por lo que surge el **problema a resolver:** ¿Cómo definir la Gestión de la Configuración de Software en proyectos guiados por Metodologías Ágiles de Desarrollo en el CEDIN para administrar eficientemente los elementos de configuración de un producto de software?

Se toma como **objeto de estudio**: el proceso de Gestión de la Configuración de Software, centrado específicamente en el **campo de acción:** el proceso de Gestión de la Configuración de Software enfocado en Metodologías Ágiles de Desarrollo.

Para darle solución al problema planteado se establece como **objetivo general**: Definir una estrategia que garantice cómo aplicar la Gestión de la Configuración de Software en proyectos guiados por Metodologías Ágiles de Desarrollo de Software.

La **idea a defender** en este trabajo es que si se define una estrategia que garantice cómo aplicar la Gestión de la Configuración de Software en proyectos guiados por Metodologías Ágiles de Desarrollo de Software se podrá administrar eficientemente los elementos de configuración de un producto de software.

Para cumplir con el objetivo general establecido se han definido las siguientes **tareas de investigación:**

- ➢ Investigación de las tendencias actuales sobre aplicación de la Gestión de Configuración en Metodologías Ágiles de Desarrollo de Software, para la elaboración del estado del arte de la investigación.
- ➢ Caracterización de las diferentes Metodologías Ágiles de Desarrollo de Software.

- Descripción de las diferentes herramientas utilizadas en la gestión de proyectos donde se utilicen Metodologías Ágiles de Desarrollo de Software, para seleccionar las más adecuadas.
- Definición de políticas, herramientas, roles y responsabilidades que apoyan el proceso de Gestión de la Configuración de Software.
- Definición de subprocesos que garanticen la Gestión de la Configuración de Software en Metodologías Ágiles de Desarrollo.
- Validación de la propuesta a partir de la valoración de expertos en Ingeniería y Gestión de Software en el CEDIN para determinar si la propuesta realizada es efectiva.

En la realización de este trabajo se utilizarán algunos métodos investigativo que ayudarán en el desarrollo de la investigación como son:

- **Analítico – sintético:** Permite realizar un estudio profundo acerca de la Gestión de la Configuración y las Metodologías Ágiles de Desarrollo de Software para conocer sus características principales, ventajas, herramientas, y luego sintetizar estos elementos en la solución de la propuesta.
- **Inductivo-Deductivo:** Permite desde el estudio de las herramientas utilizadas en la Gestión de Configuración, sus características, las ventajas que ofrecen, deducir cual es la más apropiada para aplicarla en la estrategia propuesta.
- **Histórico – lógico:** Permite realizar una investigación sobre cómo ha evolucionado el proceso de Gestión de la Configuración, cómo han evolucionado los resultados obtenidos por estos procesos en los proyectos productivos para lograr un nivel de madurez avanzado en estos.
- **Encuesta:** Apoya la actividad de validación de la estrategia propuesta, se usa con el objetivo de que varios especialistas en Gestión de Configuración de Software en la UCI evalúen con sus criterios la estrategia.

La estructura del trabajo constará de la Introducción y tres capítulos; en el Capítulo 1 se aborda el tema de la Gestión de la Configuración de Software desde el punto de vista conceptual, así como de las herramientas más utilizadas en la actualidad que implementan la Gestión de la Configuración de Software, tanto en el mundo como en la UCI. Se realiza una valoración del estado del arte de las Metodologías Ágiles de Desarrollo de Software más utilizadas en la actualidad en la universidad.

En el Capítulo 2 se describe detalladamente la estrategia de Gestión de la Configuración de Software propuesta. Se establecen las actividades a desarrollar en el proceso de Gestión de la Configuración. Se definen las fases, así como los roles y artefactos a desarrollar.

En el Capítulo 3 se realiza una evaluación a la estrategia propuesta, recopilando los resultados de evaluaciones emitidas por especialistas en el tema de la Gestión de la Configuración, a través de la aplicación de una encuesta a los mismos.

Capítulo 1: Fundamentación Teórica.

Introducción

A continuación se describe un estudio sobre el estado del arte de la Gestión de la Configuración a nivel mundial y el rol que desempeña en el proceso de desarrollo de software, tomando como referencia las definiciones emitidas por organizaciones y autores prestigiosos en el tema. Además, se presentan las características de diferentes herramientas usadas para automatizar el proceso de GCS, así como un estudio del estado del arte de las MADS y de las perspectivas ágiles para los procesos y herramientas de configuración de software.

1.1 La Gestión de la Configuración de Software. Definiciones.

La necesidad de la Gestión de la Configuración de Software está dada por el incremento de la complejidad de los sistemas de software y la integración entre diversas tecnologías debido a la dinámica del mercado. Teniendo en cuenta también la naturaleza cambiante del software, el nivel de personalización y los cambios en las reglas del negocio. Actualmente la dependencia de los sistemas de software en las empresas incrementa la presión para actualizar los mismos, lo cual conlleva a un incremento en la demanda, es por ello que cada día se aboga más por la optimización en cuanto a tiempo y organización en el desarrollo de productos de software, donde entra a jugar un papel fundamental la Gestión de la Configuración de Software.

A lo largo del ciclo de vida del proceso de software, los productos evolucionan, desde la concepción del producto y 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. La Gestión de la Configuración de Software es una disciplina encargada del control de la evolución de los productos de software.

“La gestión es el arte de identificar, organizar y controlar las modificaciones que sufre el software, la meta es maximizar la productividad minimizando errores.” (Babich, 1986)

Por su parte, Roger S. Pressman en su libro *“Ingeniería del Software. Un enfoque práctico”* plantea que: la Gestión de la Configuración de Software es una actividad de autoprotección que se aplica durante el proceso del software y que constituye “una parte esencial de una buena gestión de proyecto y una práctica formal de la Industria del Software”. Como el cambio se puede producir en cualquier momento, las actividades de Gestión de la Configuración de Software sirven para: identificar el cambio, controlar el cambio, garantizar que el cambio se

implemente adecuadamente e informar el cambio a todos aquellos que puedan estar interesados (Pressman, 2002).

Como la definición más completa y concisa, se considera la de Ivar Jacobson, en la que se plantea:

“Gestión de Configuración: Procesos de soporte cuyo propósito es identificar, definir y almacenar en una línea base los elementos de software; controla los cambios, reporta y registra el estado de los elementos y de las Solicitudes de Cambio; asegura la completitud, consistencia y corrección de los elementos; controla, almacena, maneja y libera los elementos asociados al producto de software” (Jacobson, 2000)**.**

De las definiciones referenciadas anteriormente se puede concluir que la Gestión de la Configuración de Software es una colección de tareas que se realizan para controlar la evolución de un producto, estas tareas incluyen la identificación de los elementos del software, la gestión de los cambios que ocurran en ellos y la verificación, registro e informe de la información de la configuración.

1.2 Modelos y estándares de calidad tenidos en cuenta para la investigación.

En la actualidad existen diferentes estándares y modelos internacionales de calidad, que proponen guías para el desarrollo de los productos de software y miden la calidad del proceso y/o producto, los cuales se han tenido en cuenta para la investigación y estudio acerca de la Gestión de la Configuración de Software.

1.2.1 Integración de Modelo de Madurez de Capacidades (CMMI).

La Integración de Modelo de Madurez de Capacidad (CMMI) ayuda a mejorar la calidad del proceso a través de un proyecto, de una división, o de una organización entera. Las ayudas de CMMI integran tradicionalmente funciones de organización separadas, fijan metas de las prioridades y mejoras del proceso, proporcionan la dirección para los procesos de la calidad, y proporcionan un punto de referencia para valorar procesos actuales (CMMI, 2010).

CMMI establece que para alcanzar el nivel 2 “Repetible” es necesario cumplir con algunos requisitos o áreas de procesos, entre las que se encuentra la GCS y a la cual se le atribuye gran importancia, planteando de esta lo siguiente:

El objetivo de la GCS es establecer y mantener la integridad de los elementos de trabajo identificando, controlando y auditando dichos elementos. Más concretamente mediante: (Gracia, 2005)

- La identificación de los elementos de trabajo que componen una línea base.
- Controlando los cambios de dichos elementos.
- Realizando planificaciones para gestionar la configuración.
- Proporcionando formas de construir los elementos de trabajo a partir del sistema de control de la configuración.
- Mantener la integridad de las líneas bases.

1.2.2 Organización Internacional para la Estandarización (ISO).

La Organización Internacional para la Estandarización (ISO) es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional (ISO, 2010).

Esta organización ha definido una serie de estándares que son generalmente aplicables a todos los procesos de producción. Específicamente ISO 9000, proporciona un conjunto de estándares para la gestión de la calidad en las más diversas actividades relacionadas con el proceso de desarrollo de software, y dentro de ellas la GCS, para la cual sugiere como subprocesos elementales (ISO, 2010):

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

1.2.3 Instituto de Ingenieros Eléctricos y Electrónicos (IEEE).

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) es una asociación técnico-profesional mundial dedicada a la estandarización, está formada por profesionales de las nuevas tecnologías como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación, ingenieros en informática, ingenieros en biomédica e ingenieros en telecomunicación; su trabajo es promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad y de los mismos profesionales (IEEE, 2010). Respecto al tema de la GCS establece que: “La Gestión de la Configuración de Software 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”.

La IEEE Std 828-1990, describe un estándar para los Planes de GCS, establece lo mínimo que debe contener un Plan de Configuración y define las actividades específicas a gestionar y sus requerimientos para cualquier parte del ciclo de vida de desarrollo de un producto software, donde coincide con la sugerencia de subprocesos que propone la ISO, a pesar de que no enfatiza con profundidad sobre el control de versiones de los elementos de configuración, siendo esta una debilidad.

1.3 Proceso de Gestión de la Configuración de Software.

Un proceso de software se puede caracterizar como un marco común que define un número de actividades que son aplicables a todos los proyectos, con independencia de su tamaño o complejidad, es decir, una colección de tareas de trabajo de ingeniería de software, hitos de proyectos, productos de trabajo y puntos de garantía de calidad, que permiten que las actividades dentro del marco de trabajo se adapten a las características del software y a los requisitos del usuario (Pressman, 2002).

Según lo establecido por los mayores conocedores del tema y tomando como apoyo lo que exponen cada uno de los distintos estándares y modelos de mejora citados en los acápites 1.1 y 1.2 respectivamente; para que se aplique un correcto proceso de GCS deben establecerse los siguientes subprocesos:

1. Identificar Elementos de Configuración de Software.

2. Establecer mecanismos para el control del cambio y de las versiones.
3. Realizar informes de estado de la configuración.
4. Auditar la configuración. (Para garantizar que se mantiene la calidad mientras se realizan los cambios).

1.3.1 Identificación de la configuración.

Una de las actividades más importantes a desarrollar durante el proceso de GCS es sin duda la definición de prácticas que permitan identificar y almacenar los artefactos en un repositorio seguro.

Según la IEEE “La Identificación de la configuración consiste en la identificación de los elementos de configuración de un sistema y registrar sus características funcionales y físicas en documentación técnica” (IEEE, 1990).

Definimos como un **Elemento de Configuración de Software** (ECS) 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 (Tema5, 2011).

Los ECS incluyen:

- Ejecutables.
- Código Fuente.
- Modelos de datos.
- Modelos de procesos.
- Especificaciones de requisitos.
- Pruebas.

La tarea de Identificación de la configuración comienza con la definición de los elementos representativos de los productos en cada línea base establecida, además se asignan identificadores apropiados a todos los programas, documentos y periféricos, usando un

esquema numerado que proporciona información sobre el ECS. Finalmente, la identificación debe facilitar el control de cambios, para acomodar actualizaciones y modificaciones.

La importancia que se le confiere a la identificación de los ECS radica en que esta actividad brinda la capacidad de encontrar e identificar la versión correcta de cualquier artefacto del proyecto de una manera rápida y sencilla.

1.3.1.1 Definición y establecimiento de líneas bases.

En la fase de identificación de configuración, se identifican también las líneas bases de un proyecto y su contenido. Una ***línea base*** es una especificación o producto revisado y aprobado formalmente, que sirve como base para el desarrollo posterior, y puede ser modificado solo a través de procedimientos formales de control de cambios (Tema5, 2011).

Las líneas bases más comunes en el desarrollo del software según Pressman (Pressman, 1997) son:

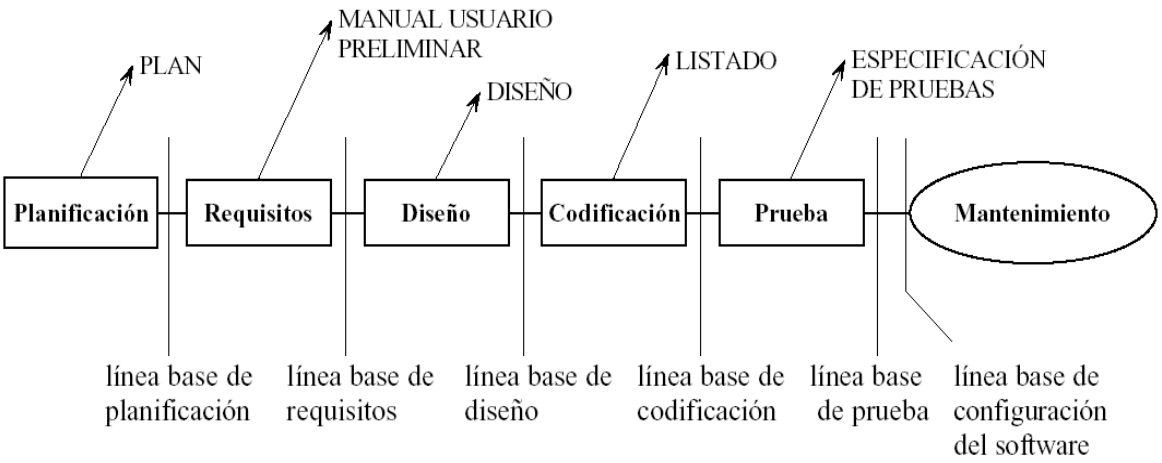


Figura 1 Líneas bases según Pressman.

1.3.2 Control de cambios.

El definir y controlar los cambios ocurridos durante el desarrollo de un software constituye la actividad más importante dentro de la GCS, su objetivo fundamental es establecer un mecanismo para controlar los cambios a los elementos de configuración.

Según León, las líneas bases juegan un papel importante en el control de los cambios, sólo pueden cambiarse formalmente a través de los procedimientos de control de cambio que

incluyen los pasos de evaluación, coordinación, aprobación o desaprobación, y aplicación de cambios a los elementos de configuración (Leon, 2000).

Pressman, refiriéndose a este tema de los cambios en el desarrollo del software expresa: “Cuando se construye software de computadora, los cambios son inevitables. Además, los cambios aumentan el grado de confusión entre los ingenieros del software que están trabajando en el proyecto. La confusión surge cuando no se han analizado los cambios antes de realizarlos, no se han registrados antes de implementarlos, no se les han comunicado a aquellas personas que necesitan saberlo o no se han controlado de manera que mejoren la calidad y reduzcan los errores” (Pressman, 1997).

1.3.2.1 Control de versiones.

Una versión es una instancia de un elemento, se usa para señalar a un ECS que tiene un conjunto definido de características funcionales.

Una importante actividad dentro del subproceso del Control de cambios es *Mantener un control de versiones*. En el desarrollo de software se hace muy común la existencia de varias versiones de un ECS. A medida que se va desarrollando el producto pueden surgir cambios que conlleven a que se establezca una nueva variante de ese ECS que ahora posee características diferentes, es por ello que surge la necesidad del control de las versiones, para tener la certeza de que se trabaja sobre la versión correcta.

Retomando lo planteado por Pressman, se puede decir que el seguimiento de los cambios ayuda a que se determine el impacto que puede ocasionar la realización de los mismos a un proyecto, además se les informa a los interesados acerca del cambio y se actualizan las versiones de los elementos que fueron cambiados. Este subproceso garantiza que el sistema que se entregue no presente errores y cumpla con los requerimientos del cliente.

1.3.3 Informes de estado de la configuración.

La generación de informes de estado de la configuración es una tarea que permite tener un conocimiento dentro del proyecto acerca de qué ocurre con los ECS, quién ha trabajado sobre los mismos, en qué momento, y quiénes se afectan con un cambio al elemento.

“El objetivo es mantener a los usuarios, a los gestores y a los desarrolladores, al tanto del estado de la configuración y su evolución. En resumen, pretende dar respuesta a las preguntas

¿Qué ocurrió?, ¿Cuándo ocurrió? Ayuda también a mejorar los problemas de comunicación entre los participantes en un proyecto" (de Antonio, 2001).

En esta actividad se diseña y opera un sistema para la captura y reporte de la información necesaria a medida que avanza el ciclo de vida, esta información que se quiere gestionar debe ser identificada, recogida y mantenida.

1.3.4 Auditorías a la Configuración.

Una auditoría de software es una actividad llevada a cabo para evaluar de forma independiente y objetiva la conformidad de los productos y procesos de software con respecto a las regulaciones, estándares, guías, planes y procedimientos aplicables (Pressman, 2002).

La actividad de Auditoría a la Configuración de Software determina en qué medida un ECS satisface sus características funcionales y físicas requeridas. Se pueden realizar auditorías de este tipo en puntos clave del ciclo de vida. El resultado satisfactorio de una auditoría se puede utilizar como prerrequisito para establecer una línea base del producto, y de esta manera se asegura y mantiene la integridad de dicha línea base.

La verificación no se realiza sobre los propios productos sino que consiste en comprobar que los productos que conforman una línea base están gestionados correctamente bajo el control de la configuración, que todos los cambios realizados sobre estos productos han sido registrados y, por tanto, se puede establecer una trazabilidad entre cambios y productos afectados.

El objetivo de las auditorías a la configuración es asegurarse de que:

- Los elementos de configuración se encuentran en el directorio apropiado.
- El estado actual de los elementos de configuración es consistente.
- La información de las líneas bases se mantiene de forma correcta.

1.4 Herramientas utilizadas para la Gestión de la Configuración de Software.

Como todo proceso, la GCS también puede ser sistematizada y automatizada. Actualmente existen en el mercado diversas herramientas que permiten apoyar una o más actividades de este proceso.

"Cada herramienta tiene sus propias fortalezas y debilidades. Por ejemplo, algunas son mejores en la gestión de cambios, mientras que otras tienen una excelente capacidad para la gestión y construcción del control de versiones" (Delgado Martínez, et al., 2006).

No existe aún una herramienta que haga conjuntamente el control de versiones y el control de cambios, por el momento se deben seleccionar herramientas separadas para llevar a cabo estas actividades.

1.4.1 Herramientas para el Control de Versiones.

Un sistema de control de versiones es el responsable de mantener las trazas de varias revisiones de la misma unidad de información. Se utiliza normalmente en proyectos de desarrollo de software para gestionar el código, pero también puede usarse para gestionar versiones de documentación de cualquier tipo de proyecto (IEEE, 2010).
A continuación se presentan ejemplos de algunas herramientas más difundidas y utilizadas hoy en día, pertenecientes tanto a software privativo como software libre, y se agrega una pequeña descripción de las mismas.

Software privativo

- ***IBM Rational ClearCase***

Es una solución líder en la industria y que proporciona un sofisticado Control de Versiones. Ofrece una gestión fiable, ampliable y flexible de los activos de software para equipos de desarrollo de gran tamaño y tamaño medio. IBM Rational ClearCase proporciona una gestión del ciclo de vida y control de los activos de desarrollo de software. Con un control integrado de versiones, una gestión del espacio de trabajo automatizado, un soporte de desarrollo en paralelo, una gestión de línea base, así como gestión de construcciones y liberaciones.

Permite efectuar seguimiento de defectos y cambios en toda la empresa (IBM, 2011). 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.

- ***Visual SourceSafe***

Sistema de Control de Versiones en el nivel de archivos. Para los desarrolladores Visual SourceSafe (VSS) aloja código reutilizable u orientado a objetos, asimismo, facilita el seguimiento de las aplicaciones que utilizan módulos de código concretos. Permite a los equipos de desarrollo proteger y llevar un registro de manera automática de su código fuente, documentación, archivos binarios, y todos los demás tipos de archivo a medida que cambian a través del ciclo de vida de software.

También existe otro conjunto de herramientas que satisfacen los mismos objetivos pero en este caso pertenecen a la comunidad de Software Libre como por ejemplo:

Software Libre

- **Sistema de Control de Versiones**

El Sistema de Control de Versiones (CVS por sus siglas en inglés) fue liberado en 1986 y si bien el CVS puede ser una tecnología más antigua, es todavía muy útil para cualquier diseñador o programador para hacer copias de seguridad y compartir archivos. CVS es un software de código abierto que ejecuta en la mayoría de las plataformas UNIX existentes: Linux, Mac OS X, así como Microsoft Windows. Combina una arquitectura cliente-servidor lo que permite a los desarrolladores funcionar como un solo equipo aunque estén dispersos en la geografía. Funciona en base a un conjunto de comandos para administrar repositorios, módulos y archivos. CVS se ha hecho popular en el mundo del software libre. Sus desarrolladores difunden el sistema bajo la licencia GPL.

- **Subversion**

Subversion (SVN) es una herramienta libre “open – source” para el control de versiones. Permite las funcionalidades de Recuperar viejas versiones de los ficheros, ver su historial, etiquetar versiones, ramificar, unir y efectuar retorno a versiones anteriores. El servidor de Subversion gestiona un repositorio centralizado con la política Copiar-Modificar-Unir. Fue diseñado para reemplazar a CVS, no sólo para sustituirlo sino con la intención de mejorarlo ya que SVN mantiene las ideas fundamentales de CVS pero suple sus carencias y evita sus

errores. Puede conectarse al servidor Apache mediante el protocolo HTTP, como un módulo de extensión, lo que proporciona a esta herramienta una gran ventaja en estabilidad e interoperabilidad, y acceso instantáneo a las características existentes que ofrece este servidor como autenticación, autorización y compresión de la conexión. Constituye un excelente sistema de control de versiones por lo que es rápidamente acogido por la comunidad de desarrolladores de software libre.

- ***Git***

Es un sistema de control de versiones de código abierto, totalmente gratuito. Fue diseñado con el objetivo de posibilitar la gestión eficiente y veloz de cualquier tipo de proyecto de software. Con GIT se puede ir llevando el control de los cambios en el código así no hay que preocuparse si se arruina alguna funcionalidad, si llega a pasar algo malo solo se revierte a una versión anterior.

Diseñado por Linus Torvalds. Es empleado por el grupo de programación del núcleo del Sistema Operativo Linux. Git abre un mundo de posibilidades y ventajas, aumentando la velocidad al trabajar casi siempre en local, o sea que cada directorio de trabajo local es un repositorio completo independiente de acceso a un servidor o a la red por lo que se reduce la necesidad de estar siempre conectado.

1.4.2 Herramientas para el Control de Cambios.

Existen también alternativas de software de herramientas que permiten controlar los cambios que surjan en un producto durante su desarrollo, ejemplos de estas herramientas son:

- **Trac**

El Trac es un sistema web libre para la gestión de proyectos de software y seguimiento de errores, centrado en un wiki con sistemas montados alrededor de él: tags, blog, sistema de tickets/incidente y gran cantidad de plugins, extensiones y macros. Está pensado para la organización de proyectos de desarrollo software de forma principalmente colaborativa. Soporte configurable para workflow en sistema de ticket. Mejoras en sistema de ayuda y documentación, en sistema de usuarios y sesiones. Trac permite enlazar información entre una base de datos de errores de software, un sistema de control de versiones y el contenido de una wiki. También sirve como interfaz web de un sistema de control de versiones como Subversion, Git y otros.

- **Mantis**

Excelente herramienta en la categoría de Código Abierto. Es una de las soluciones más completas hoy en día para la gestión de tareas en un equipo de trabajo. Mantis es un sistema de control de fallos de programación que funciona a través del navegador, está programado en php y puede funcionar con varias bases de datos: MySQL, PostgreSQL, MS SQL, etc. La interfaz de Mantis muestra los errores detectados, a quién se le ha asignado la tarea de corregirlos y el estado en que se encuentran. Este sistema de seguimiento funciona a través de cuentas de usuario desde las que se informa de los fallos detectados mediante formularios, una vez aceptada la tarea, los informadores reciben correos con cada variación en el estado de ésta.

- ***Redmine***

Moderna herramienta para la gestión de proyectos de software con interface web. Es un sistema multi-plataforma, programado con Ruby on Rails, es una herramienta Open Source con licencia GPL. Posee una interfaz sencilla y muy configurable. Brinda unas funcionalidades asombrosas como foros, envío automático de e-mail a los desarrolladores cada vez que se les asigna una tarea o ante cualquier evento relacionado con el proyecto.

Posibilita subir ficheros y documentos, bien al proyecto, como adjuntos a las tareas y errores. También brinda la posibilidad de definir nuevos tipos de tareas y errores, con campos personalizados, todo ello fácilmente a través de la interface web. Se puede ver a través de Redmine los cambios en el repositorio. Entiende CVS, Subversion y algunos de los sistemas de Control de Versiones más conocidos. Contiene control de acceso por roles, Multi-proyectos, gráficos Gantt, gestión de avisos, documentos y ficheros, wiki, registro de tiempos, campos personalizables, multi-lenguaje.

Si bien es cierto que todas estas herramientas dan soporte a muchas de las actividades incluidas en la GCS, dejan a un lado temas muy importantes como la identificación de los elementos de configuración y la administración de líneas base, solo se evidencian en actividades de control de cambio y de versiones.

1.5 Metodologías de Desarrollo de Software.

1.5.1 ¿Qué es una metodología?

Una metodología puede ser una guía que nos va indicando qué hacer y cómo actuar cuando se quiere obtener un objetivo, o en términos informáticos, cuando se quiere desarrollar un

producto de software. Es posible definir una metodología como aquel enfoque que permite observar un problema de una forma total, sistemática y disciplinada.

Una **Metodología de Desarrollo de Software** es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información (Boehm, 1988).

Las metodologías surgen debido a la existencia de problemas serios de costos asociados al desarrollo de software por conceptos de retrasos, presupuestos excedidos y sobre todo, por la no implementación de toda la funcionalidad originalmente especificada, lo que demostraba la falta de sistematización que existía para construir un producto de software. Primero surgieron las Metodologías Tradicionales, o como se les llama también, las metodologías pesadas, con la idea de tener un modelo que ordenara el proceso de desarrollo. Estas metodologías fueron evolucionando, desde el modelo de cascada (en este modelo se requería de la finalización de la etapa anterior para empezar la siguiente), hasta los procesos iterativos (subsanaba muchas de las carencias del modelo en cascada). Luego surgen las Metodologías Ágiles de Desarrollo de Software, como una extensión a las metodologías tradicionales para mejorar el desarrollo de un sistema, según el tipo de proyecto y empresa, optimizando las prácticas de desarrollo de software.

A continuación se muestran los principales aspectos en los que difieren estas dos clasificaciones de metodologías de desarrollo:

Metodologías Tradicionales	**Metodologías Ágiles**
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.	Basadas en heurísticas provenientes de prácticas de producción de código.
Cierta resistencia a los cambios.	Especialmente preparados para cambios durante el proyecto.
Impuestas externamente.	Impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas.	Proceso menos controlado, con pocos principios.
Existe un contrato prefijado.	No existe contrato tradicional o al menos es bastante flexible.
El cliente interactúa con el equipo de desarrollo mediante reuniones.	El cliente es parte del equipo de desarrollo.

Grupos grandes y posiblemente distribuidos.	Grupos pequeños (menos de diez integrantes) y trabajando en el mismo sitio.
Más artefactos.	Pocos artefactos.
Más roles.	Pocos roles.
La arquitectura del software es esencial y se expresa mediante modelos.	Menos énfasis en la arquitectura del software.

Tabla 1 Metodologías Tradicionales vs. Metodologías Ágiles de desarrollo

1.5.2 Metodologías Ágiles de Desarrollo. Características.

Las MADS se rigen por el Manifiesto Ágil, el cual fue y es el punto de partida de estas metodologías, es el documento que resume la filosofía ágil, y que marca las diferencias entre las metodologías pesadas y las livianas al valorar:

- *Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.* La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno.
- *Desarrollar software funciona más que conseguir una buena documentación*. La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata, para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.
- *La colaboración con el cliente más que la negociación de un contrato*. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- *Responder a los cambios más que seguir estrictamente un plan*. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

A continuación se muestran algunos ejemplos de MADS con sus principales características.

1.5.2.1 Programación extrema (XP).

Según las bibliografías consultadas, es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

XP se basa en realizar pruebas unitarias, las que se utilizan como un medio para determinar si el código cumple con las necesidades requeridas; están basadas en la funcionalidad de los módulos del programa como funciones, procedimientos, módulos de clase, etc. Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento, ya sea diseñar, probar, implementar o integrar el código de la tarea. Además se caracteriza por realizar un proceso de integración continua, el cual se basa en integrar el proyecto de software automáticamente lo más frecuentemente posible para detectar cuanto antes fallos en el código o en la integración.

Los roles de XP de acuerdo con la propuesta original de Beck son (Beck, 1999):

- *Programador*: El programador escribe las pruebas unitarias, es el responsable de tomar decisiones técnicas y de construir el sistema.
- *Cliente:* Forma parte del equipo, escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- *Encargado de pruebas* (Tester): Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- *Encargado de seguimiento* (Tracker): Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- *Entrenador* (Coach): Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

- *Gestor* (Big boss): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

El ciclo de vida de XP es el siguiente (Beck, 1999):

- *Exploración*: En esta fase los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.
- *Planificación de la Entrega*: En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.
- *Iteraciones:* Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas y tareas no terminadas en la iteración anterior.
- *Producción*: Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.
- *Mantenimiento*: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
- *Muerte del proyecto*: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos

como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Artefactos de XP:

A continuación se describen los artefactos generados durante el proceso de desarrollo en la metodología Programación Extrema.

- *Historias de usuarios:* técnica utilizada para especificar los requisitos del Software. Su formato son tres sentencias de texto escritas por el cliente, en su terminología sin sintaxis técnica, cuando llega el momento de implementarla, los desarrolladores van con el cliente y reciben una descripción detallada de los requerimientos, cara a cara. No son casos de uso pero describen escenarios, además conducen las pruebas de aceptación (funcionales) y reemplazan un gran documento de requisitos.

Historia de Usuario	
Número:	Nombre Historia de Usuario:
Modificación (o extensión) de Historia de Usuario (Nro. y Nombre):	
Usuario:	Iteración Asignada:
Prioridad en Negocio: (Alta / Media / Baja)	 Puntos Estimados:
Riesgo en Desarrollo: (Alto / Medio / Bajo)	 Puntos Reales:
Descripción: *Se introducen los datos del artículo (título, fichero adjunto, resumen, tópicos) y de los autores (nombre, e-mail, afiliación). Uno de los autores debe indicarse como autor de contacto. El sistema confirma la correcta recepción del artículo enviando un e-mail al autor de contacto con un usuario y contraseña para que el autor pueda posteriormente acceder al artículo.*	
Observaciones:	

Tabla 2 Modelo de Historia de Usuarios.

Las Historias de Usuario tienen tres aspectos:

- Conversación: cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación)
- Pruebas de Aceptación: permite confirmar que la historia ha sido implementada correctamente.

Caso de Prueba de Aceptación	
Código:	Historia de Usuario (No. y Nombre):
Nombre:	
Descripción:	
Condiciones de ejecución:	
Entrada/Pasos de ejecución:	
Resultado esperado:	
Evaluación de la prueba:	

Tabla 3 Modelo de Pruebas de Aceptación.

- Las *Tareas de Ingeniería*: Sse usan para guardar los datos de las tareas definidas en las Historias de Usuario.

1.5.2.2 Scrum.

Scrum es una metodología para la gestión y desarrollo de software basada en un proceso iterativo e incremental, utilizado comúnmente en entornos basados en el desarrollo ágil de software. Permite la creación de equipos auto-organizados, impulsando la co-localización de todos los miembros del equipo y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.

Los roles principales de Scrum se describen a continuación:

- *El Scrum Master*: Interactúa con el cliente y el equipo. Es responsable de asegurarse que el proyecto se lleve a cabo de acuerdo con las prácticas, valores y reglas de Scrum, y que progrese según lo previsto. Coordina los encuentros diarios, debe ser miembro del equipo y trabajar a la par.

- *Propietario del Proyecto*: Es el responsable oficial del proyecto, gestión, control y visibilidad de la lista de acumulación o lista de retraso del producto (producto Backlog). Toma las decisiones finales de las tareas asignadas al registro y convierte sus elementos en rasgos a desarrollar.
- *Equipo de Scrum:* Tiene autoridad para reorganizarse y definir las acciones necesarias o sugerir remoción de impedimentos. Responsable de transformar el Backlog de la iteración en un incremento de la funcionalidad del software.
- *Cliente:* Participa en las tareas relacionadas con el registro de requisitos, están presentes en la planeación del proceso.
- *Administrador*: Está a cargo de las decisiones fundamentales y participa en la definición de los objetivos y requerimientos. Por ejemplo, selecciona al Dueño del Producto, evalúa el progreso y reduce el registro de acumulación junto con el Scrum Master.

Ciclo de vida de Scrum.

Esta metodología está estructurada por cuatro fases fundamentales:

- *Planificación*: El propósito de esta fase es establecer la visión, definir expectativas y asegurar la financiación. Las actividades son la escritura de la visión, el presupuesto, el registro de acumulación o retraso (Backlog) del producto inicial y los elementos estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos.

- *Montaje*: El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.

- *Desarrollo*: El propósito es implementar un sistema listo para entrega en una serie de iteraciones de treinta días llamadas “corridas” (sprints). Las actividades son un encuentro de planeamiento de corridas en cada iteración y la definición del registro de acumulación de corridas, los estimados y los encuentros diarios de Scrum.

- *Liberación*: El propósito es el despliegue operacional; las actividades: documentación, entrenamiento, mercadeo y venta.

Los artefactos *que se generan en Scrum* se describen a continuación:

- *Product Backlog*: Los requisitos del sistema o del producto desarrollado se enumeran en el Product Backlog. El propietario es responsable del contenido, priorización, y

disponibilidad del este artefacto. El Product Backlog usado en la planificación del proyecto, es simplemente una estimación inicial de los requisitos, se desarrolla paralelamente a medida que el producto y el ambiente en el cual se trabaja evoluciona.

Id	Módulo	Descripción	Est.	Por
Crítico				
1		Plataforma tecnológica	30	AR
2	Cliente	Interfaz de usuario	40	LR
3	Cliente	Un usuario se registra en el sistema	40	LR
4	Trastienda	El operador define el flujo y textos de un expediente	60	AR
5	Trastienda	Etc...	999.	XX
Necesario				
6	Cliente	El usuario modifica su ficha personal	30	AR
7	Cliente	El usuario consulta los expedientes asignados	15	LR
8	Cliente	El usuario tramita un expediente	35	LR

Figura 2 Modelo del artefacto: Product Backlog.

- *Carta burndown*: demuestra la cantidad de trabajo restante a través de tiempo. La carta burndown es una manera excelente de visualizar la correlación entre la cantidad de trabajo restante en cualquier punto, y el progreso de los equipos de proyecto en la reducción de este trabajo.

1.5.2.3 Proceso Unificado Ágil.

El Proceso Unificado Ágil (AUP por sus siglas en inglés) es una versión simplificada de Proceso Unificado de Rational (RUP por sus siglas en inglés) desarrollada por Scott Amber. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas.

Roles de AUP:

Dentro de los roles de AUP se encuentran:

- *Administrador de la configuración*: se encarga de proporcionar la infraestructura y crear el medio ambiente para el equipo de desarrollo.
- *Implementador:* es el responsable de poner el sistema en los ambientes de pre-producción y producción.
- *Desarrollador:* Es quien escribe el código, realiza las pruebas y construye el software.

- *Especialista del proceso*: Desarrolla, adapta y apoya el material de los procesos de la organización (descripción de procesos, plantillas, guías, ejemplos, entre otros).
- *Administrador de bases de datos (DBA):* trabaja de manera colaborativa con los integrantes del equipo del proyecto para diseñar, probar y brindar soporte a los diferentes esquemas de datos.
- *Modelador Ágil:* Alguien que cree y desarrolle modelos, ya sean dibujos, tarjetas, o archivos complejos realizados con herramientas CASE, de manera colaborativa y evolutiva.
- *Administrador del proyecto:* Administra los miembros de los equipos de trabajo, crea relaciones con los involucrados, coordina las interacciones con los involucrados, planea, administra y dispone recursos, enmarca prioridades y mantiene el equipo enfocado.
- *Involucrado:* cualquiera que sea usuario directo, usuario indirecto, administrador de usuarios, administrador, miembro de equipo de operación o soporte, desarrolladores que trabajan en otros sistemas que se integran o interactúan con el sistema implementado, en fin todo aquel que se vea afectado de una u otra forma con el proyecto.
- *Especialista en herramientas:* Es responsable de seleccionar, adquirir, configurar y brindar mantenimiento al equipo requerido.

Ciclo de vida de AUP:

Divide el ciclo de desarrollo en 4 fases:

- Inicio: El objetivo es identificar el alcance inicial del proyecto, una arquitectura potencial de su sistema, y para obtener la financiación inicial del proyecto y la aceptación de las partes interesadas. Típicamente es una fase breve que puede durar unos pocos días o unas pocas semanas.
- Elaboración: Se analiza el dominio del problema y se define el plan del proyecto. Al acabar esta fase, deben estar identificados la mayoría de los casos de uso y los actores, debe quedar descripta la arquitectura de software. El objetivo es probar la arquitectura del sistema.
- Construcción: Se desarrollan, integran y verifican todos los componentes y rasgos de la aplicación. Los resultados de esta fase (las versiones alfa, beta y otras versiones de prueba) se crean tan rápido como sea posible. Se debe compilar también una versión

de entrega. La meta es construir un software que responda a las necesidades de los clientes.

- Transición: Comienza cuando el producto está suficientemente maduro para ser entregado. Se corrigen los últimos errores y se agregan los rasgos pospuestos. La fase consiste en prueba beta, piloto, entrenamiento a usuarios y despacho del producto a mercadeo, distribución y ventas. El objetivo es validar y desplegar el sistema en su entorno de producción.

Los artefactos *que se generan en las distintas fases del método AUP son*:

- *Sistema:* El software de trabajo, el hardware y la documentación para ser liberada a producción.
- *Modelo de Diseño:* Describe el diseño de su sistema. Consta de una variedad de productos de trabajo, incluye potencialmente un modelo de despliegue y un modelo de objetos.
- *Pruebas de Aceptación:* describen los requerimientos de caja-negra, identificados por sus usuarios del proyecto, a los cuales su sistema se debe ajustar.
- *Reporte de Defectos:* Un tipo de solicitudes de cambio que definen problemas relacionados al sistema; es decir, el sistema no está funcionando de la forma en que tiene que hacerlo.
- *Glosario del Proyecto:* Describe los términos críticos y técnicos del negocio en su proyecto.
- *Cronograma del Proyecto:* Indica las actividades, las dependencias entre ellos, y los hitos del proyecto.

1.6 Perspectivas ágiles para la Gestión de la Configuración de Software.

Las Metodologías de Desarrollo de Software tradicionalmente incluyen procesos como Análisis, Arquitectura, Implementación, Gestión de Proyecto, Gestión de la Configuración de Software, Gestión de Riesgos, Gestión de Cambios y una larga lista de procesos dependientes de Áreas de Procesos, sin embargo las MADS no promueven estos grupos de procesos y de hecho, no los definen para sus marcos de trabajo, esto puede verse justificado por el hecho de que las MADS tienen como insignia que lo que demuestra y garantiza el proceso es el producto entregado y no la documentación que se pueda presentar al cliente. Esto trae consigo que el equipo de desarrollo pueda presentarse con diferentes inconvenientes durante el desarrollo de

un producto, por ello existen perspectivas ágiles para la configuración de software que demuestran cómo se tienen en cuenta los subprocesos de la GCS para los equipos de desarrollo ágil.

1.6.1 Perspectiva Ágil para la Identificación de la configuración.

Muchos de los métodos ágiles estipulan que cada artefacto de desarrollo debe estar bajo gestión de la configuración, estos artefactos pueden ser modelos de análisis, diseños, datos, software, pruebas, resultados de las pruebas, etc. Además se propone la creación y liberación de líneas bases que contendrán dichos elementos de configuración y los cambios que se realicen sobre ellos. Parte de la documentación de nivel detallado que se definen en las especificaciones de las líneas bases en las metodologías tradicionales, pueden ser vistas como innecesarias en un entorno de desarrollo ágil, en el que algunos de estos detalles se recogen en un debate cara a cara con un mínimo de documentación.

1.6.2 Perspectiva Ágil para el Control de Versiones y Control de Cambios.

Las MADS consideran de gran importancia la tecnología de control de versiones para la gestión de los frecuentes cambios que se producen dentro de cada iteración.

Por otra parte el procedimiento de control de cambios obtiene poco valor en los métodos ágiles, al menos de la forma en que es percibido con el uso de métodos tradicionales. El control de cambios no es necesario en sí mismo, a medida que se van realizando las iteraciones en el proceso de desarrollo y durante la planificación de la misma, los cambios pueden ser introducidos. En general, la noción de control de cambios se hace mucho menos formal y se transforma en la actividad de *Planificación de iteraciones* en el comienzo de cada iteración.

1.6.3 Perspectiva Ágil para la generación de informes.

El subproceso de Generar informes en la GCS obtiene poco valor para los equipos de desarrollo ágil, siguiendo el principio ágil de que desarrollar software funciona más que conseguir una buena documentación.

1.6.4 Perspectiva Ágil para la realización de Auditorías a la configuración.

Como se había explicado en la definición de los subprocesos de GCS, la Auditoría a la configuración es una actividad centrada en la revisión de líneas bases, verificando que cada

línea base incluye lo acordado, puede explicar cómo y cuándo se realizaron cambios en las líneas bases.

Al considerar las auditorías como un subproceso de Gestión de la Configuración de Software para métodos ágiles, este puede utilizarse para reducir el riesgo de accidente por aplicar el subproceso de forma incorrecta y se centra en la mejora del mismo (es decir, con menos énfasis en las personas y más en la realización de las auditorías).

1.6.5 Perspectiva Ágil para las herramientas de Gestión de la Configuración de Software.

Las herramientas de gestión de la configuración no solo juegan un papel fundamental en el proceso de desarrollo de software, sino que también pueden protagonizar la transición hacia el desarrollo ágil, todo depende de que cuente con las capacidades necesarias. En un entorno ágil las herramientas deben caracterizarse por alto rendimiento, es importante que la herramienta tenga capacidades orientadas al desempeño, esto incluye la creación de configuración rápida, soporte nativo de Internet, integración con otras herramientas de gestión de proyectos con el fin de tener todos los trabajos unificados.

Consideraciones parciales.

A raíz del estudio realizado de la GCS y las aplicaciones de esta en proyectos que utilizan MADS, se considera que ninguna de las bibliografías consultadas se adapta del todo a las necesidades, por lo que se hace imprescindible tomar de cada una de ellas los elementos necesarios para conformar la estrategia que dé solución a la problemática existente, profundizando en las MADS más difundidas a nivel mundial como XP, Scrum y AUP, teniendo en cuenta las fases en las que está divido cada proceso de desarrollo, y los roles y artefactos que se generan. Además, tomando como apoyo el estudio realizado de las herramientas que ayudan en el proceso de GCS y cada una de sus características, se podrá proponer aquella que se ajuste a la solución del problema de la no existencia de una guía de procesos para la Gestión de Configuración mediante el uso de MADS.

Capítulo 2: Propuesta de estrategia para la Gestión de la Configuración de Software en proyectos que utilizan Metodologías Ágiles de Desarrollo.

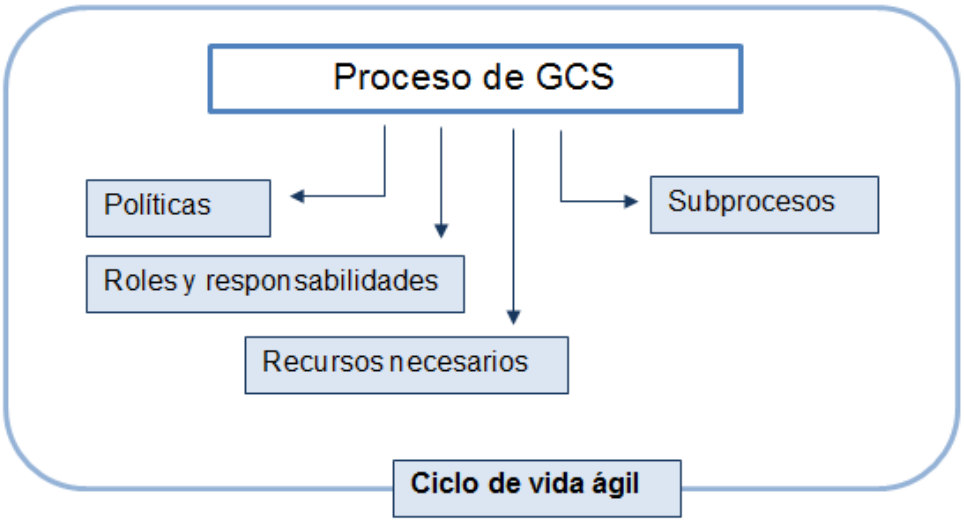
Introducción

El presente capítulo se centra en cómo la Gestión de la Configuración de Software, con sus prácticas e infraestructura, puede ser adaptada y automatizada con el fin de beneficiar directamente a aquellos proyectos que trabajan con MADS. Se describe detalladamente una estrategia para la GCS aplicada a métodos ágiles, la cual consiste en definir una serie de actividades que deben realizarse para aplicar correctamente el proceso.

2.1 Proceso de Gestión de la Configuración de Software mediante el uso de Metodologías Ágiles de Desarrollo de Software.

La GCS se debe ver como un motor de cambio para los métodos ágiles, su objetivo es apoyar el desarrollo de software garantizando la integridad en el seguimiento de los cambios, para que el equipo de desarrollo pueda centrarse en crear sus funcionalidades.

Para llevar a cabo la implementación del proceso de GCS es necesario establecer las políticas por las que se regirá el área de proceso, definir los roles y responsabilidades, el ciclo de vida del desarrollo de software y los subprocesos que guiarán las actividades y garantizarán que se cumpla con el objetivo propuesto. Además, se deben describir las herramientas que apoyarán la realización de estas actividades. El siguiente gráfico muestra cuáles son los principales elementos que integran el proceso de GCS de acuerdo a la propuesta de solución elaborada.



¿Qué intención tiene cada uno de los elementos contenidos en el gráfico?

Políticas: Las políticas de GCS son reglas que se establecen para guiar a los implicados en el proceso de qué se debe hacer para llevar a cabo el mismo.

Roles y responsabilidades: Los roles son funciones que se le asignan a los integrantes de un proyecto en dependencia de las actividades que desarrollen dentro del mismo para intervenir en un proceso.

Recursos necesarios: Se deben seleccionar herramientas que apoyen una o más actividades del proceso de GCS.

Ciclo de vida ágil: El ciclo de vida de un producto está conformado por fases que conectan el inicio de un producto hasta el momento en que es creado y liberado al cliente. Las fases se realizan en un momento determinado, las cuales incluyen actividades propias del momento en que se establecen.

2.2 Políticas de Gestión de la Configuración de Software.

A. Identificación de la configuración.

Se deben identificar los elementos de configuración que serán controlados y definir un esquema para la identificación de estos elementos y sus versiones.

B. Control de cambios.

Se deben controlar los cambios que se le hacen al software a través de su ciclo de vida, asegurando que el producto sea consistente y cumpla con las necesidades del cliente.

C. Creación y liberación de líneas bases.

Crear o liberar líneas bases para mantener un control de los elementos de configuración identificados por etapas.

D. Auditorías y revisión.

Se debe validar que el producto este completo y mantener la consistencia entre los elementos de configuración, asegurando que estén en un estado apropiado a través de todo el ciclo de vida del producto considerándose una colección bien definida de elementos.

2.3 Recursos necesarios para la Gestión de la Configuración de Software.

Luego del profundo estudio realizado de las distintas herramientas de configuración que existen actualmente en el mercado del software y las ventajas que brinda cada una de ellas, se realiza la propuesta de las herramientas adecuadas para el proceso de Gestión de la Configuración de Software que sirven de apoyo a las actividades de Control de cambio y Control de versiones que se realizan durante la construcción de un producto de software.

- Para el Control de Versiones:

Se propone el uso de la herramienta **SVN (Subversion)** por las funciones que es capaz de realizar y por las ventajas que brinda, entre ellas se incluye la de ser una herramienta Open Source, lo que permite acceder a ella fácilmente, además puede conectarse al servidor HTTP Apache como un módulo de extensión, esto proporciona a la herramienta el acceso instantáneo a las características existentes que ofrece este servidor como autenticación, autorización y compresión de la conexión. El servidor de Subversion gestiona un repositorio centralizado con la política Copiar-Modificar-Unir.

- Para el Control de Cambios:

En este caso se propone como herramienta el **Redmine**, brinda un servicio fácil de manejar y tiene características ventajosas para los proyectos de producción de software. Su principal ventaja es que esta herramienta es la que está establecida para utilizar en el CEDIN y cuenta con un equipo de trabajo que se encarga de crear y agregar funcionalidades nuevas a la misma, en dependencia de las necesidades de los proyectos. Se puede ver a través de Redmine los cambios en el repositorio, posibilita subir ficheros y documentos al proyecto, como

adjuntos a las tareas, contiene control de acceso por roles. Entiende CVS, Subversion y algunos de los sistemas de Control de Versiones más conocidos.

- Servidor:

Se necesita una computadora ubicada como servidor para montar las herramientas antes definidas.

2.4 Roles y responsabilidades de la Gestión de la Configuración de Software.

Como parte del proceso fue necesario definir los roles y las responsabilidades para realizar las actividades del área de Gestión de la Configuración de Software, así como los involucrados relevantes en el proceso.

Rol	Responsabilidades	Habilidades
Jefe de proyecto.	- Interactúa con el cliente y el equipo del proyecto. - Es responsable de asegurarse que el proyecto se lleve a cabo de acuerdo con las prácticas, valores, reglas, y que progrese según lo previsto. - Coordina los encuentros diarios. - Dirige la planificación del proyecto. - Toma las decisiones finales de las tareas asignadas al registro y convierte sus elementos en rasgos a desarrollar.	- Debe ser miembro del equipo del proyecto y trabajar a la par. - Capacitado y experimentado preferiblemente en gestión de proyectos ágiles. - Tomar decisiones rápidas.
Administrador de configuración.	- Establece y maneja la administración de configuración de los productos de software. - Establece las líneas base del proyecto. - Identifica los elementos de configuración. - Mantiene el control de las	- Capacidad de establecer un plan de CM y el programa para un equipo de la organización, garantizando el nivel adecuado de CM. - Habilidades de gestión de la calidad. - Dominar el ciclo de desarrollo de software. - Trabajar en grupo.

	versiones de los artefactos del proyecto. - Elabora Plan de Administración de la Configuración. - Audita la Gestión de la Configuración de Software dentro del proyecto. - Administra las herramientas que apoyan la Gestión de la Configuración de Software, y todas las tareas asociadas con la tecnología. - Gestiona todas las tareas asociadas con la construcción, las migraciones, y la liberación del software.	- Capacidad para diseñar un proceso de auditoría. - Capacidad de actualización, monitoreo y mantenimiento de la tecnología de la Gestión de la Configuración de Software. - Capacidad de automatizar la migración de paquetes de salida de un entorno a otro.
Cliente.	- Escribe las historias de usuario y las pruebas funcionales para validar su implementación. - Asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.	- Capacidad de extraer requisitos. - Capacidad de entender los procesos de mejora.
Equipo del proyecto. Dentro del equipo de trabajo se encuentran las funcionalidades: ○ Programador	- Comprometido con el proyecto a tiempo completo, se centran en completar el trabajo necesario. - Tiene autoridad para reorganizarse y definir las acciones necesarias. - Escribe las pruebas unitarias, es el	- Conocimientos en programación, análisis, pruebas y diseño. - Habilidades para desarrollar más de una función, en dependencia del rol que estén desarrollando. - Capacitados para tomar decisiones. - Habilidad para identificar riesgos.

	responsable de tomar decisiones técnicas y de construir el sistema.	
○ Probador	- Identificación, definición, implementación y realización de las pruebas necesarias, así como registrar los resultados de las pruebas y el análisis de los resultados.	- Capacidad para evaluar el producto actual, recoger las necesidades, y proponer una solución.

Tabla 4 Asignación de roles y responsabilidades.

Involucrados relevantes.

Hay roles involucrados en el proceso que pueden ser internos, en el caso de las personas que tienen que estar presente en las actividades del proceso de GCS, ya que intervienen directamente en el proceso dentro del proyecto y además involucrados externos, los cuales son aquellos que trabajan externos al proyecto pero en algunos momentos de éste intervienen en la realización de alguna actividad, como revisiones y auditorías.

Involucrados externos:

Calisoft:

Calisoft es la Dirección de Calidad de Software de la UCI, la cual garantiza el crecimiento continuo de una producción de software con calidad, brindando asesorías, entrenamiento, métodos de medición y servicios de verificación-validación.

Involucrados internos:

Jefe de proyecto.

Administrador de configuración.

2.5 Fases de desarrollo ágil.

Partiendo del análisis profundo realizado al ciclo de vida de las MADS, se definen tres fases de desarrollo para guiar el proceso de construcción de un software, garantizando que no se vean afectados los principios ágiles. Las fases propuestas son:

- **Planificación:** El propósito de esta fase es establecer la visión, definir expectativas y asegurar la financiación del producto a construir. Durante la planificación del producto los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto, al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Las actividades a realizar durante la fase son la escritura de la visión, el presupuesto, el registro de los requisitos del producto inicial y los elementos estimados, así como la arquitectura de alto nivel, el diseño exploratorio, los prototipos y la planificación de la configuración.

- **Construcción:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en revisión. Cada iteración del ciclo de vida incluye actividades de: planificación, análisis de requerimientos, diseño, codificación, pruebas unitarias, integración y documentación. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto. El objetivo es probar la arquitectura del sistema.

- **Liberación:** La fase comienza cuando el producto está suficientemente maduro para ser entregado. Se corrigen los últimos errores y se agregan los rasgos pospuestos. Consiste en prueba beta, piloto, entrenamiento a usuarios y despacho del producto a mercadeo, distribución y ventas. El objetivo es validar y desplegar el sistema en el entorno del cliente.

En la siguiente figura se muestra como ocurre la interacción entre las fases de desarrollo propuestas anteriormente para la construcción de productos guiados por métodos ágiles.

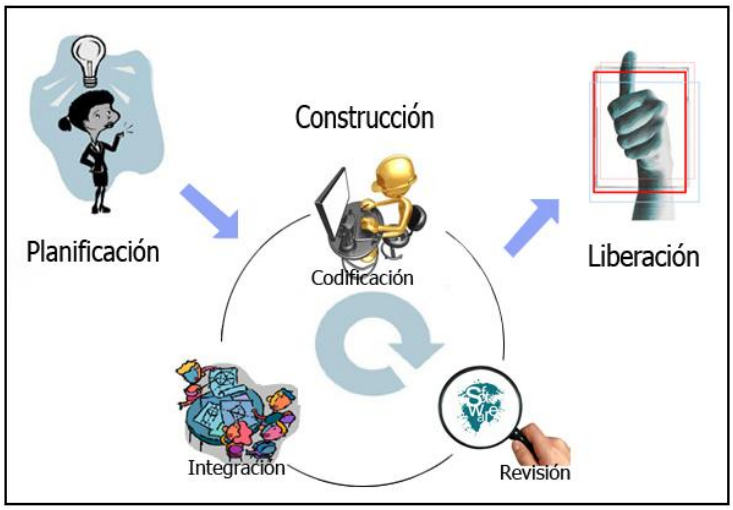


Figura 3 Estructura del desarrollo de software para MADS.

2.6 Subprocesos principales en el proceso de Gestión de la Configuración de Software.

Tras haber analizado los distintos planteamientos formulados por los conocedores más importantes del tema de Gestión de la Configuración en proyectos de desarrollo de software, se describen cuatro subprocesos principales propuestos para realizar el proceso de Gestión de la Configuración de Software en proyectos que utilicen MADS.

- **Identificación de la Configuración.**
- **Creación y Liberación de Líneas Bases.**
- **Control de cambios.**
- **Realización de Auditorías a la Configuración.**

La siguiente figura muestra la interacción entre los subprocesos propuestos dentro del proceso de GCS enfocado en MADS.

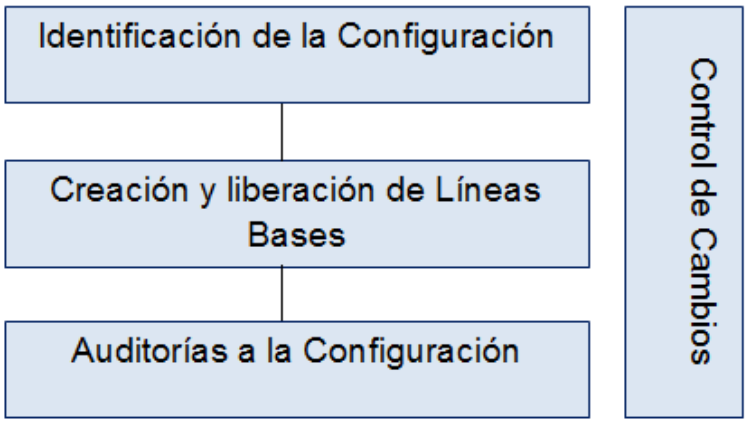


Figura 4 Diagrama de Interacción entre los subprocesos.

2.6.1 Relación de los Subprocesos con el ciclo de vida ágil.

En la siguiente figura se muestra la interacción entre los subprocesos y el ciclo de vida ágil, donde el subproceso Identificación de la Configuración es el primero que se ejecuta, preparando la base para la ejecución del resto de los subprocesos mediante la definición de políticas, técnicas y herramientas que guiarán y apoyarán el proceso de Gestión de la Configuración durante todo el ciclo de vida. El subproceso Control de Cambios se puede llevar a cabo en cualquier momento del desarrollo del software siempre que surja la necesidad de un cambio. Por su parte el subproceso Creación y liberación de Líneas Bases surge al finalizar cada iteración de las fases del ciclo de vida propuesto y luego las Auditorías a la Configuración a las líneas bases creadas en cada iteración.

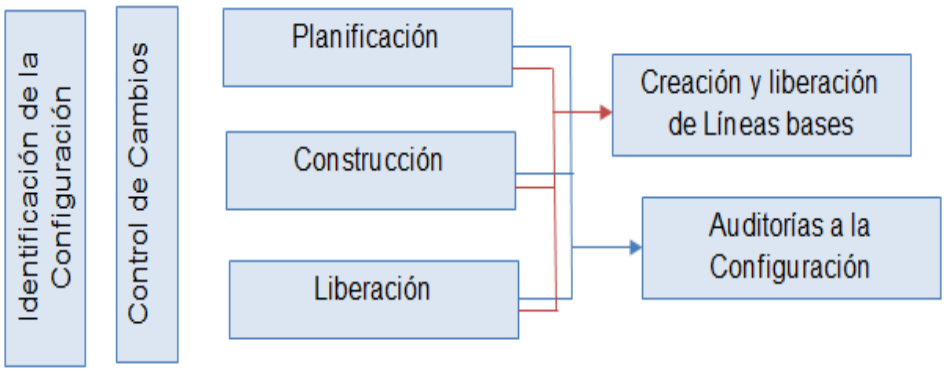


Figura 5Diagrama de relación de los subprocesos y el ciclo de vida.

2.6.2 Descripción del subproceso Identificación de la Configuración.

Este subproceso describe las actividades de Identificación de la configuración tales como: la planeación y el despliegue del proceso de Gestión de la Configuración de Software, la identificación de los elementos de configuración y las líneas bases que los contendrán. El subproceso se inicia en la fase de Planificación del ciclo de vida ágil establecido, preparando la base para el desarrollo de los otros subprocesos.

2.6.2.1 Descripción gráfica del subproceso Identificación de la Configuración.

<table>
<tr><td colspan="5">Identificación de la Configuración.</td></tr>
<tr><td>Criterios de entrada:</td><td colspan="4">Inicio del proyecto.</td></tr>
<tr><td>Criterios de salida:</td><td colspan="4">Establecimiento y mantenimiento de los recursos de trabajo. Identificación y control de los ECS y Líneas Bases. Auditorías a la Configuración.</td></tr>
<tr><td>Roles</td><td>Entrada</td><td>Control</td><td>Actividades</td><td>Salida</td></tr>
<tr><td></td><td></td><td></td><td> 
</td><td></td></tr>
<tr><td>Administrador de Configuración. Jefe de proyecto</td><td></td><td></td><td>Planeación del proyecto</td><td>Plan de Gestión de la configuración.</td></tr>
<tr><td>Administrador de Configuración y Jefe de Proyecto.</td><td>Plan de Gestión de la configuración.</td><td>Manual de Herramienta.</td><td>Creación y mantenimiento del repositorio</td><td>- Repositorio.
- Accesos al repositorio.</td></tr>
<tr><td>Administrador de Configuración</td><td>Plan de Gestión de la configuración.</td><td>Proceso de Gestión de la Configuración.</td><td>Identificación de los elementos de configuración y las líneas bases</td><td>Estándares de Configuración.</td></tr>
</table>

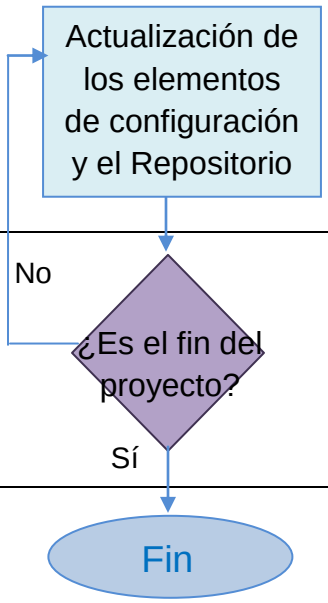
Equipo de Trabajo.	Estándares de Configuración.	Proceso de Gestión de la Configuración.		Repositorio (actualizado).
				

Tabla 5 Descripción de subproceso Identificación de la configuración.

2.6.2.2 Actividad: Planeación del proyecto.

El objetivo de esta actividad es identificar el alcance inicial de la administración de la configuración, evaluar la factibilidad de las diversas herramientas en el apoyo de las prácticas de Gestión de la Configuración de Software, y establecer una dirección razonable hacia donde debe dirigirse el desarrollo del proyecto. Es llevada a cabo por el Jefe del proyecto y el Administrador de configuración.

Tiene gran importancia porque ayuda a organizar las tareas en tiempo, establecer definiciones importantes acerca de la configuración, que deben ser de conocimiento de todos los integrantes del proyecto, destinar recursos, asignar responsabilidades y posteriormente poder darle seguimiento a lo planificado.

2.6.2.3 Actividad: Identificar Elementos de Configuración de Software (ECS).

Esta actividad propone la identificación de los ECS a fin de establecer un control sobre los artefactos que se crean, modifican y utilizan durante el desarrollo del producto. Para ello se tienen en cuenta los artefactos que se generan en los distintos métodos ágiles, y también aquellos que son creados por los diferentes roles de Gestión de la Configuración de Software, detectándose como ECS aquellos productos críticos para el cliente que se crean en un proyecto, ya sean documentos, ficheros, ejecutables, datos, etc.

Definición del esquema para la identificación de los Elementos de Configuración del Software.

Lo primero a tener en cuenta es que el esquema que identificará cada Elemento de Configuración de Software debe llevar implícito el número de versión del mismo.

Un elemento de configuración del software estará identificado por el siguiente esquema:

Área + Nombre del proyecto + Identificador del ECS + Nombre del ECS + Versión del ECS.

Dónde:

Áreas: representa el área de desarrollo a la que pertenece el ECS (CEDIN, CDAE).

Nombre del proyecto: Especificar el nombre del proyecto al que pertenece el ECS.

Nombre del ECS: Cadena de caracteres que permite identificar al Elemento de Configuración del Software sin imprecisiones.

Identificador del ECS: es el identificador del elemento de configuración, se establece una numeración única para cada elemento.

Versión del ECS: representa la versión a la que hace referencia el elemento de configuración.

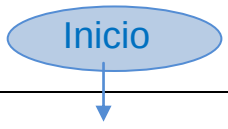
Un ejemplo de identificación de Elemento de Configuración de Software sería:

CEDIN_TETSCADA_2030PDSW.1.0.doc

2.6.3 Descripción del subproceso Creación y Liberación de Líneas Bases.

Consiste en la descripción de las líneas bases que se proponen para el desarrollo del software, y se describen además las actividades que se deben realizar para crear y liberar dichas líneas bases.

2.6.3.1 Descripción gráfica del subproceso Creación y Liberación de Líneas Bases.

Creación y liberación de Líneas Bases.				
Criterios de entrada:	Elementos de configuración.			
Criterios de salida:	Creación y liberación de Líneas Bases.			
Roles	Entrada	Control	Actividades	Salida
				

Administrador de Calidad.	Estándares de Configuración.		Revisión de Elementos de Configuración.	No conformidades (NC).
Administrador de Calidad, Jefe de proyecto.	NC.		¿Se aprueba la creación de la LB? Sí No	
Jefe de proyecto.	NC.		Asignación de NC a responsables de resolverlas.	Historias de Usuarios (actualizada).
Equipo de proyecto.	Historias de Usuarios.		Solución a las NC.	
Administrador de configuración, Jefe de proyecto.	Estándares de Configuración. Repositorio.		Creación de la LB en el repositorio.	Repositorio (actualizado).
Administrador de Calidad. Probador	Línea Base	Proceso de Gestión de Configuración.	Revisión de la LB.	NC
Administrador de Configuración, Jefe de proyecto.	NC		Sí ¿Hay No conformidades? No	
			¿Contiene elementos del cliente? No Sí	

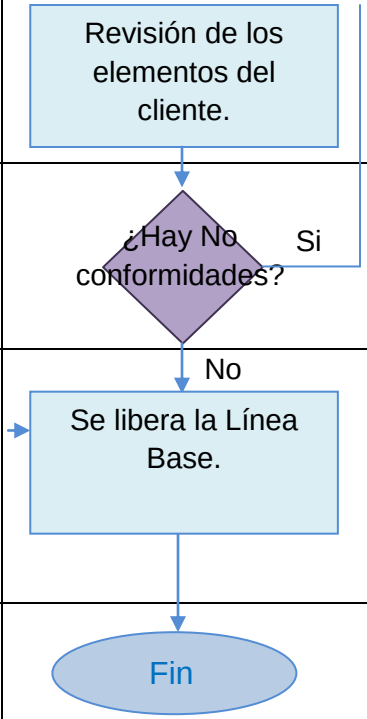
CALISOFT	Estándares de Configuración.			NC.
Jefe de Proyecto. Administrador de Configuración.	NC			

Tabla 6 Descripción del subproceso Creación y Liberación de las Líneas Bases.

2.6.3.2 Descripción de Líneas bases.

A continuación se proponen las líneas bases para un enfoque Ágil. Se propone la creación de tres líneas bases críticas, las cuales se establecerán de manera iterativa, lo que significa que se creará y liberará al finalizar cada iteración del desarrollo del software una línea base con los elementos identificados y revisados de la iteración. Para tener un control de en qué momento se crea y libera cada línea base, se define el siguiente esquema de identificación:

Línea_Base_*fase*_It#

Donde *fase* es el nombre de la fase en la que es creada la línea base y # sería el número de la iteración de la fase en la que es creada.

- **Línea Base de Planificación**. Esta línea base se crea en la fase de Planificación por primera vez, y contendrá aquellos elementos que surjan a partir de las planificaciones que se realicen sobre los requisitos y expectativas del producto a desarrollar, así como las actas de aceptación, inicio, el Plan de Desarrollo de Software.

Línea_Base_Planificación_It#

- **Línea Base de Construcción**. Se crea en la fase de Construcción e incluye elementos de codificación, revisión, documentación, elementos de pruebas y las versiones de entrega del producto.

Línea_Base_Construcción_It#

- **Línea Base de Liberación**. Esta línea base va a ser creada al finalizar la fase de Liberación, donde se obtiene el resultado de la obra (ejecutables, documentos de usuario, etc.) que se entrega listo al cliente al finalizar el trabajo del proyecto.

Línea_Base_Liberación_It#

Los ECS que forman parte de cada una de las líneas bases anteriormente descritas, se encuentran definidos explícitamente en el documento 03_EstandaresdeConfiguracion (artefacto generado a partir de la realización de la propuesta).

2.6.4 Descripción del subproceso Control de Cambios.

Según el estudio realizado a los procesos de Gestión de la Configuración existentes para proyectos que utilizan métodos tradicionales, el control de cambios suele consistir en llenar un formulario detallado de petición de cambios, que incluye atributos como detalles del cambio, impacto al proyecto, riesgos, mitigación, etc. Se puede determinar después de las investigaciones realizadas a las Metodologías Ágiles de Desarrollo de Software que el control de cambios tradicional queda fuera de lugar en estas, porque entra en conflicto con el principio de "Responder al cambio por sobre seguir el plan". Se torna difícil poder responder al cambio cuando deben completarse formularios extensos y se necesitan listas de aprobaciones. A continuación se describe el subproceso de Control de Cambios para proyecto que utilicen MADS.

Para realizar un seguimiento de cambios la clave es siempre la misma: mantener un proceso ágil y simple, y sólo preocuparse por los detalles que son necesarios.

Las recomendaciones principales son:

- ✓ Registrar los cambios a la Historia de usuarios o en alguna herramienta para seguir cambios.
- ✓ Eliminar la mayor cantidad de aprobaciones que sea posible.
- ✓ Agregar un formulario simple y pequeño para el control de cambios.

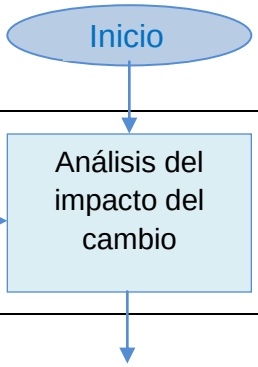
- ✓ Involucrar a los interesados.

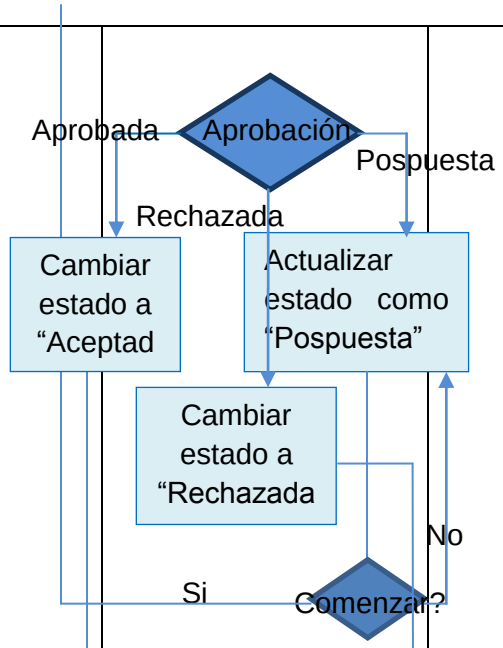
¿Cuándo se realizan cambios durante el proceso de desarrollo ágil?

Los cambios surgen cuando: El equipo de trabajo se da cuenta de que le falta alguna característica o requisito para el producto. Cuando se identifica un error en el software, se dan cuenta de que no entendían la necesidad real del software, o porque sencillamente un competidor lanzará un nuevo producto que implementa características que el software del cliente no cumple.

Para realizar el seguimiento de los cambios cada uno de ellos será tratado como un requisito más y debe ser agregado a la Historia de Usuarios, en la que se le puede agregar un atributo adicional a los elementos, que identifiquen el origen de la historia. Los valores para este atributo pueden ser "original", "nuevo" o "cambio".

2.6.4.1 Descripción gráfica del subproceso Control de Cambios.

Control de Cambio.				
Criterios de entrada:	Solicitud de cambio.			
Criterios de salida:	Cambio realizado.			
Roles	Entrada	Control	Actividades	Salida
				
Administrador de configuración, Jefe de proyecto, Cliente				Matriz de trazabilidad

Administrador de configuración, Jefe de proyecto				
Administrador de Configuración, Cliente.	Solicitud de cambio.	Herramienta de control de cambios.	Registro del cambio en la herramienta y en la Historia de usuarios	Historia de Usuarios (actualiza).
Jefe de proyecto y Administrador de Configuración	Historia de usuarios.		Asignar responsables de realizar el cambio	Historia de Usuarios (actualiza).
Jefe de proyecto, Administrador de configuración.	Historia de usuarios.		Revisar cumplimiento del cambio	

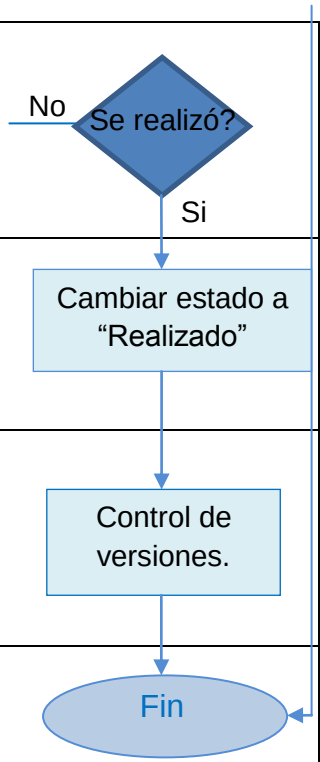
				
Administrador de configuración				Solicitud de cambios (actualizada)
Administrador de configuración.		Herramienta de control de versiones.		Repositorio (actualizado)

Tabla 7 Descripción del subproceso Control de Cambios.

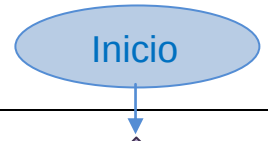
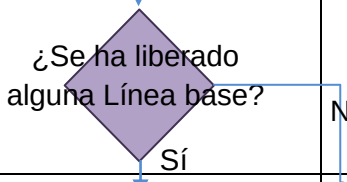
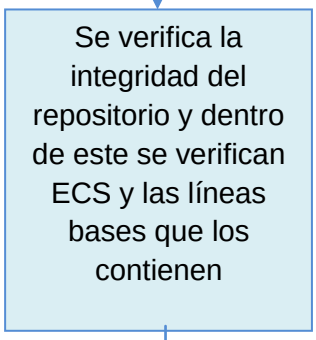
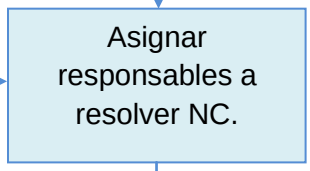
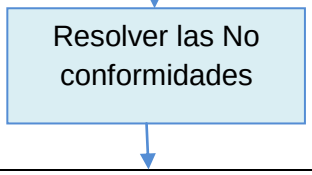
2.6.4.2 Actividad: Control de Versiones.

A partir de un cambio pudiese generarse una nueva versión o una nueva variante de un Elemento de Configuración de Software determinado. Durante la actividad de Control de Versiones el Administrador de configuración asume la responsabilidad del establecimiento de las nuevas versiones haciendo uso de la herramienta que apoya el proceso, asegurando de esta manera que luego de que ocurra el cambio se actualicen las versiones de los elementos que se vieron afectados por el mismo.

2.6.5 Descripción del subproceso Auditorías a la Configuración.

Las Auditorías a la Configuración se centran en la verificación del estado del repositorio y dentro de éste determinan la integridad de cada línea base y los elementos que las componen. Se realizarán auditorías al finalizar las iteraciones de cada fase del ciclo de vida, después de liberada una línea base.

2.6.5.1 Descripción gráfica del subproceso Auditorías a la Configuración.

<table>
<tr><td colspan="5">Auditorías a la Configuración.</td></tr>
<tr><td colspan="2">Criterios de entrada:</td><td colspan="3">Repositorio del proyecto.</td></tr>
<tr><td colspan="2">Criterios de salida:</td><td colspan="3">Establecer la integridad de las Líneas Bases, los ECS y el Repositorio.</td></tr>
<tr><td>Roles</td><td>Entrada</td><td>Control</td><td>Actividades</td><td>Salida</td></tr>
<tr><td></td><td></td><td></td><td> 
</td><td></td></tr>
<tr><td></td><td></td><td></td><td> 
</td><td></td></tr>
<tr><td>Administrador de Calidad.</td><td>Repositorio Estándares de Configuración.</td><td></td><td> 
</td><td>No conformidades (NC)</td></tr>
<tr><td></td><td></td><td></td><td> 
</td><td></td></tr>
<tr><td>Jefe de proyecto.</td><td>NC</td><td></td><td> 
</td><td>NC (asignadas)</td></tr>
<tr><td>Equipo de proyecto.</td><td>NC (asignadas)</td><td></td><td> 
</td><td>NC</td></tr>
</table>

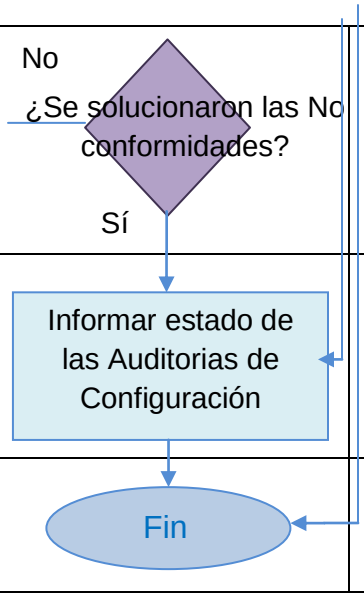
Administrador de Calidad.				
Administrador de Calidad. Jefe de Proyecto.	NC (Monitorizadas).			NC (Resueltas). Auditorías.

Tabla 8 Descripción del subproceso Auditorias a la Configuración.

2.7 Productos de trabajo del proceso de Gestión de la Configuración de Software.

Un producto de trabajo es una abstracción general que representa el resultado de un proceso. Hacen el trabajo más ameno, ya que marcan un esquema de organización para todo el que haga uso de ellos. En este trabajo surge la necesidad de crear una planilla, la cual se describe a continuación.

- **Planilla de Solicitud de cambio.**

En el momento exacto que se descubre la necesidad de un cambio, se genera una planilla de Solicitud de Cambios donde se describe con breves criterios la necesidad del cambio, los elementos de configuración que se verán afectados y alguna propuesta de respuesta al cambio (Ver documento: Planilla de Solicitud de Cambios.pdf).

2.8 Documentos de trabajo del proceso de Gestión de la Configuración de Software.

Los documentos de trabajo son aquellos informes finales de los proyectos que se proponen y son de interés para los objetivos del mismo. Con la realización de la propuesta surgen varios documentos de trabajo que son necesarios para guiar y describir el proceso de Gestión de la Configuración de Software, los cuales se describen a continuación.

- **Plan de Gestión de configuración.**

Se crea durante la Planeación del proceso de Gestión de la Configuración de Software, y consiste en un documento que define los pasos y actividades que describen como se desarrollan los subprocesos de control de la configuración en un proyecto durante el desarrollo del software. Este documento debe ser de conocimiento de cada integrante de los equipos de desarrollo (Ver documento: 01_Plan de Gestión de Configuración.pdf).

- **Estándares de configuración de software.**

Documento cuyo objetivo principal es describir los parámetros fundamentales a tenerse en cuenta para realizar la Gestión de la Configuración de Software. En él se definen los elementos de software que deben ser puesto bajo Gestión de la Configuración del Software, y aquellos que conformarán las líneas bases definidas para el proceso. Además de describir cómo debe realizarse el proceso de salvas y recuperación de datos en un proyecto que utilice MADS (Ver documento: 03_EstandaresdeConfiguracion.pdf).

- **Libro de proceso para la Gestión de la Configuración del Software.**

El objetivo de este documento es describir detalladamente el proceso para la administración de la configuración de un software en el que se utilicen MADS. Apoya el proceso definiendo los roles y responsabilidades, las herramientas y las actividades que se deben realizar durante la Gestión de la Configuración de Software (Ver documento: 04_Libro de Proceso para la Gestión de la Configuración.pdf).

Consideraciones parciales.

En el capítulo se describió una estrategia que permitirá gestionar la configuración durante la construcción de un software en los proyectos que utilicen MADS. La GCS incluye identificar y controlar los elementos críticos generados y los cambios realizados a los mismos durante el desarrollo del software, para ello fueron identificados el ciclo de vida a seguir, los diferentes subprocesos que deben ser aplicados y las actividades a realizar en cada uno de los subprocesos con sus respectivos responsables.

A continuación la figura muestra un resumen gráfico de todo el proceso de GCS enfocado en MADS, donde se evidencian las fases del ciclo de vida, los subprocesos y su iteración con estas fases y los productos o documentos de trabajo creados durante la realización de los subprocesos.

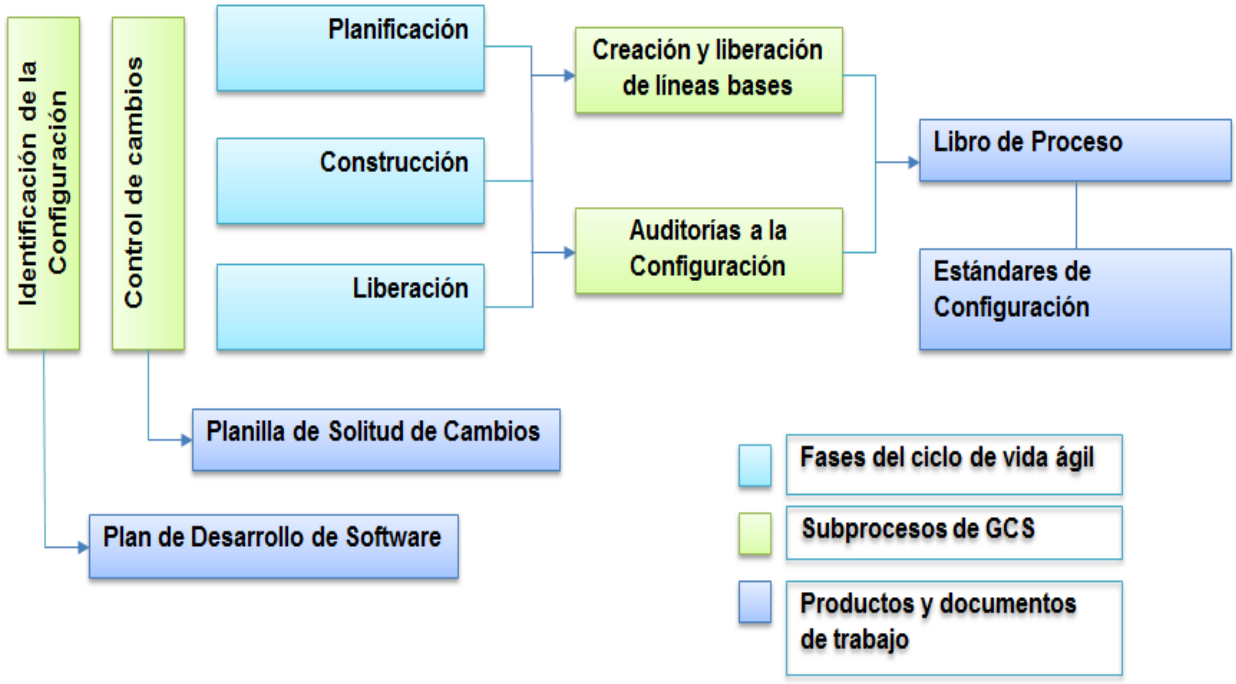


Figura 6 Resumen gráfico del proceso de GCS enfocado a MADS.

Capítulo 3: Validación de la estrategia propuesta.

Introducción

En el presente capítulo se realiza la validación de la propuesta a partir de la valoración de algunos expertos en el tema de Ingeniería y Gestión de Software, con el propósito de reflejar los resultados obtenidos de dicha evaluación y valorar el estado de la estrategia propuesta.

3.1 Método para la validación de la propuesta.

Para validar la estrategia se realiza una evaluación de los posibles resultados de la misma a partir de la valoración de varios profesionales del Centro de Informática Industrial (CEDIN), conocedores acerca de la Ingeniería y Gestión de Software. Para ello se seleccionó el método Delphi como uno de los métodos de estimación más confiables.

3.1.1 Método Delphi.

El método Delphi consiste en la utilización sistemática del juicio intuitivo de un grupo de expertos para obtener un consenso de opiniones informadas. Es un proceso repetitivo, su funcionamiento se basa en la elaboración de un cuestionario que ha de ser contestado por los expertos. Una vez recibida la información, se vuelve a realizar otro cuestionario basado en el anterior para ser contestado de nuevo. Finalmente el responsable del estudio elaborará sus conclusiones a partir de la explotación estadística de los datos obtenidos. Este método resulta más efectivo si se garantizan: el anonimato, la retroalimentación controlada y la respuesta estadística de grupo.

Brinda ventajas como:

- Permite la formación de un criterio con mayor grado de objetividad.
- El consenso logrado sobre la base de los criterios es muy confiable.
- Evita conflictos entre expertos al ser anónimo, (lo que constituye un requisito imprescindible para garantizar el éxito del método) y crea un clima favorable a la creatividad.
- El experto se siente involucrado plenamente en la solución del problema y facilita su implantación. De ello es importante el principio de voluntariedad del experto en participar en la investigación.

- Garantiza libertad de opiniones (por ser anónimo y confidencial). Ningún experto debe conocer que a su igual se le está solicitando opiniones.

3.1.2 Selección de expertos.

Para la selección de los expertos se tuvieron en cuento los siguientes criterios.

a) Graduado del Nivel Superior.

b) Vinculación al desarrollo de proyectos productivos.

c) Conocimientos y experiencia en liderazgo de proyectos y frente a la producción.

d) Conocimientos acerca del proceso de Gestión de la Configuración de Software.

e) Conocimientos acerca de Ingeniería de Software en general.

f) Conocimientos y experiencias en herramientas de gestión de proyectos.

Para poner en práctica el método se seleccionaron 10 posibles expertos para hacerles la encuesta, de la población seleccionada solo 8 respondieron afirmativamente para colaborar con la validación del trabajo. Se realizó una primera encuesta (Anexo 1) donde cada uno de los expertos seleccionados se autoevaluó con el objetivo de determinar su grado de conocimiento, para ello se determinó el Coeficiente de Competencia de cada uno. Este coeficiente se calcula mediante la fórmula: **k = ½ (kc + ka)**, donde kc (Coeficiente de Conocimientos) es el resultado de la primera pregunta de la encuesta de autovaloración multiplicado por "0.1", y ka (Coeficiente de Argumentación) se obtiene luego de analizar los resultados arrojados de la tabla que se encuentra en la segunda pregunta de la encuesta de autovaloración (Anexo 1). El análisis se hace de la siguiente forma: los expertos deben marcar con una "X", según su criterio, su grado de competencia sobre los aspectos sometidos a consideración, a estas marcas se le asignan valores de acuerdo a la siguiente escala:

Fuentes de los conocimientos	Calificación		
	alto	medio	bajo
Análisis teóricos realizados por usted	0.3	0.2	0.1
Su experiencia obtenida	0.5	0.4	0.2

Estudio de trabajos de autores nacionales	0.05	0.05	0.05
Estudio de trabajos de autores extranjeros	0.05	0.05	0.05
Su propio conocimiento del estado del problema en el extranjero	0.05	0.05	0.05
Su intuición	0.05	0.05	0.05

Tabla 9 Grados de influencia de la determinación del Coeficiente de Argumentación.

Ka será igual a la suma de los valores donde el posible experto haya marcado. Con esos elementos es suficiente para conocer el Coeficiente de Competencia de cada uno de los expertos, para determinar el nivel de cada experto son necesarios los siguientes intervalos:

Si $0,8 < k < 1,0$ el coeficiente de competencia es alto.

Si $0,5 < k < 0,8$ el coeficiente de competencia es medio.

Si $k < 0,5$ el coeficiente de competencia es bajo.

Después de haber realizado el análisis estadístico del Coeficiente de Competencia de cada experto autoevaluado, se decidió que formarían parte del grupo de validación de la propuesta, aquellos cuyo nivel resultara ser alto o medio. De los 8 expertos a los que se les aplico la encuesta de autoevaluación, solo 7 resultaron ser idóneos para continuar con la ejecución del método, a continuación se muestran los resultados obtenidos:

	Preg. 1	Preg. 2									
Expertos	Total	P1	P2	P3	P4	P5	P6	Kc	Ka	K	Nivel
Experto 1	8	0.2	0.4	0.05	0.05	0.05	0.05	0.8	0.8	0.8	Alto
Experto 2	4	0.3	0.4	0.05	0.05	0.05	0.05	0.4	0.9	0.65	Medio
Experto 3	6	0.2	0.4	0.05	0.05	0.05	0.05	0.6	0.8	0.7	Medio
Experto 4	2	0.1	0.2	0.05	0.05	0.05	0.05	0.2	0.5	0.35	Bajo

Experto 5	8	0.3	0.4	0.05	0.05	0.05	0.05	0.8	0.9	0.85	Alto
Experto 6	6	0.2	0.4	0.05	0.05	0.05	0.05	0.6	0.8	0.7	Medio
Experto 7	8	0.2	0.4	0.05	0.05	0.05	0.05	0.8	0.8	0.8	Alto
Experto 8	6	0.2	0.4	0.05	0.05	0.05	0.05	0.6	0.8	0.7	Medio

Tabla 10 Coeficiente de Competencia de los expertos autoevaluados.

El experto 4 fue el único cuyo Coeficiente de Competencia resulto Bajo, por lo que no se seleccionó para continuar la validación de la propuesta.

3.1.3 Elaboración del cuestionario para evaluar la propuesta de estrategia.

Una vez seleccionado los expertos a evaluar, se elabora la encuesta para validar la estrategia de GCS propuesta para proyectos que utilizan MADS. Para la confección de la encuesta se tuvo en cuenta varios aspectos como:

a) Demostrar que las actividades que se proponen en el proceso son necesarias y suficientes para cumplir con los objetivos establecidos.

b) Demostrar que los roles y artefactos propuestos son necesarios y suficientes.

c) Demostrar que las herramientas y tecnologías propuestas en el proceso son las adecuadas.

d) Conocer si los subprocesos propuestos son efectivos en proyectos que utilizan MADS.

La encuesta cuenta con una pregunta que incluye 7 incisos, las respuestas se etiquetaron con los indicadores: De acuerdo (Si) y No de acuerdo (No) (Anexo 2).

3.1.4 Desarrollo práctico y explotación de los resultados.

El cuestionario fue enviado acompañado de un resumen de la estrategia realizada para la Gestión de Configuración de Software para proyectos que utilicen MADS. A continuación se muestran los resultados recogidos de la encuesta:

Elementos	Si	No	Total
1 a)	7	0	7
1 b)	6	1	7

1 c)	7	0	7
1 d)	7	0	7
1 e)	7	0	7
1 f)	7	0	7
1 g)	6	1	7
Total de Aspectos a validar	47	2	

Tabla 11 Frecuencia Absoluta.

Luego de recopilar los datos, para obtener los resultados finales se realizaron los siguientes pasos:

- Obtener tabla de Frecuencias Acumuladas. En la tabla de Frecuencias Acumuladas cada número de la fila, excepto el primero, es resultado de la suma con el anterior.

Elementos	Si	No
1 a)	7	7
1 b)	6	7
1 c)	7	7
1 d)	7	7
1 e)	7	7
1 f)	7	7
1 g)	6	7

Tabla 12 Frecuencia absoluta.

- Determinación de la Frecuencia Relativa Acumulada. Sus datos se alcanzaron de la división de cada uno de los números de la tabla de Frecuencia Absoluta por el número total de expertos (en este caso 7).

Elementos	Si	No
1 a)	0.9999	0.9999

1 b)	0.8571	0.9999
1 c)	0.9999	0.9999
1 d)	0.9999	0.9999
1 e)	0.9999	0.9999
1 f)	0.9999	0.9999
1 g)	0,8571	0.9999

Tabla 13 Frecuencia Relativa Acumulada.

- Cálculo de los puntos de cortes. Se determinan los valores del intervalo en que van a estar comprendidas las categorías cualitativas, en este caso Adecuada y No adecuada.

Se obtuvieron las imágenes de los elementos de la tabla anterior mediante el uso de la función (Dist. Normal. Estándar Inv.) y el Microsoft Excel 2010 con el objetivo de recoger y visualizar los resultados. A la tabla se le adicionaron tres columnas colocar los resultados que se explican a continuación:

- Suma de las columnas.
- Suma de filas.
- Promedio de las columnas.
- Para hallar N, se divide la suma de las sumas entre el resultado de multiplicar el número de indicadores por el número de preguntas.
- El valor N-P da el valor promedio que otorgan los expertos para cada indicador propuesto.

				N = 3.83		
Elementos	**Si**	**No**	**Suma**	**P**	**N-P**	**Resultados**
1 a)	3.72	3.72	7.44	3.72	0.11	Muy Adecuada
1 b)	1.07	3.72	4.79	2.39	1.44	Muy Adecuada
1 c)	3.72	3.72	7.44	3.72	0.11	Muy Adecuada
1 d)	3.72	3.72	7.44	3.72	0.11	Muy Adecuada

1 e)	3.72	3.72	7.44	3.72	0.11	Muy Adecuada
1 f)	3.72	3.72	7.44	3.72	0.11	Muy Adecuada
1 g)	1.07	3.72	4.79	2.39	1.44	Muy Adecuada
Suma	20.74	26.04	46.78			
Puntos de corte	2.96	3.72				

Tabla 14 Puntos de corte.

Las sumas obtenidas de las columnas de los indicadores Si y No, divididas entre la cantidad de preguntas dan los puntos de cortes, los que se utilizan para determinar el grado de adecuación de los indicadores según los criterios de los expertos seleccionados, o sea, se escogen para convertir los valores cuantitativos en cualitativos, para tener una idea más exacta de la valoración de los expertos. Para ello se opera del modo siguiente.

Muy Adecuada	Bastante Adecuada	Adecuada	Poco adecuada	No adecuada
2.96	3.15	3.34	3.53	3.72

Tabla 15 Grados de adecuación.

La tabla de los grados de adecuación se analiza de izquierda a derecha y siempre incluyendo solamente el valor de la izquierda en el rango. Finalmente a los resultados de las preguntas de cada inciso se les asignó el grado de adecuación, siendo el objetivo de la explotación de los resultados.

3.1.5 Resultados obtenidos de la encuesta para la selección de los expertos.

Participaron en la selección de criterios para medir competencias ocho expertos, todos Ingenieros Informáticos, los cuales se encuentran vinculados al desarrollo de software en el CEDIN, así como a temas relacionados con la investigación. En la selección solo aquellos expertos que tenían un coeficiente de competencia Alto o Medio fueron los escogidos para continuar con el proceso de validación.

En la siguiente figura se muestra el resumen de los resultados obtenidos en la encuesta de autoevaluación del nivel de competencia de los especialistas. En la misma se evidencia que el 37.5 % de los expertos encuestado tiene un nivel de conocimientos Alto, el 50 % posee un nivel

Medio y el 12.5 % un nivel Bajo. Estos datos demuestran que el 87.5 % de los expertos tienen el conocimiento necesario para validar la encuesta.

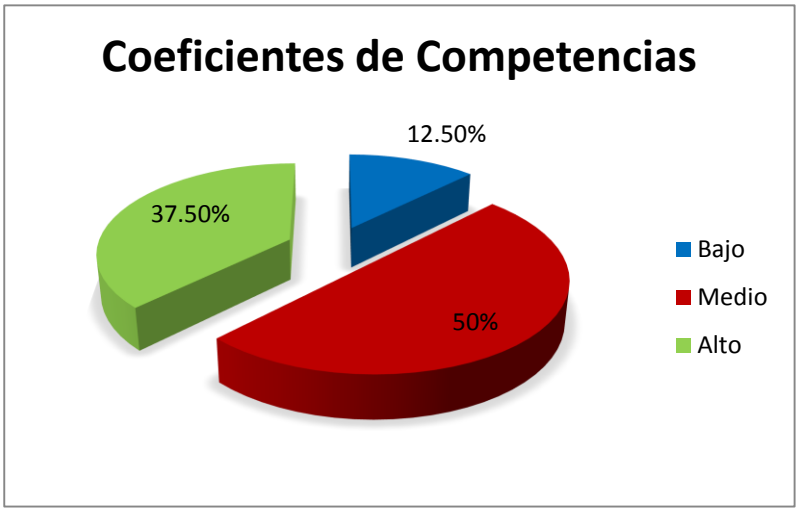


Figura 7 Coeficiente de competencia de los expertos.

3.1.6 Resultados obtenidos de la encuesta para validar la estrategia.

El resultado cuantitativo arrojado del análisis de las respuestas a la encuesta 2 se evidencia en la Figura 8, donde se muestra el nivel de adecuación de la pregunta 1 con sus respectivos incisos y el por ciento de los expertos que dieron su opinión, demostrando que según la evaluación de los expertos, la propuesta de estrategia para la Gestión de Configuración de Software haciendo uso de Metodologías Ágiles de Desarrollo es Muy Adecuada.

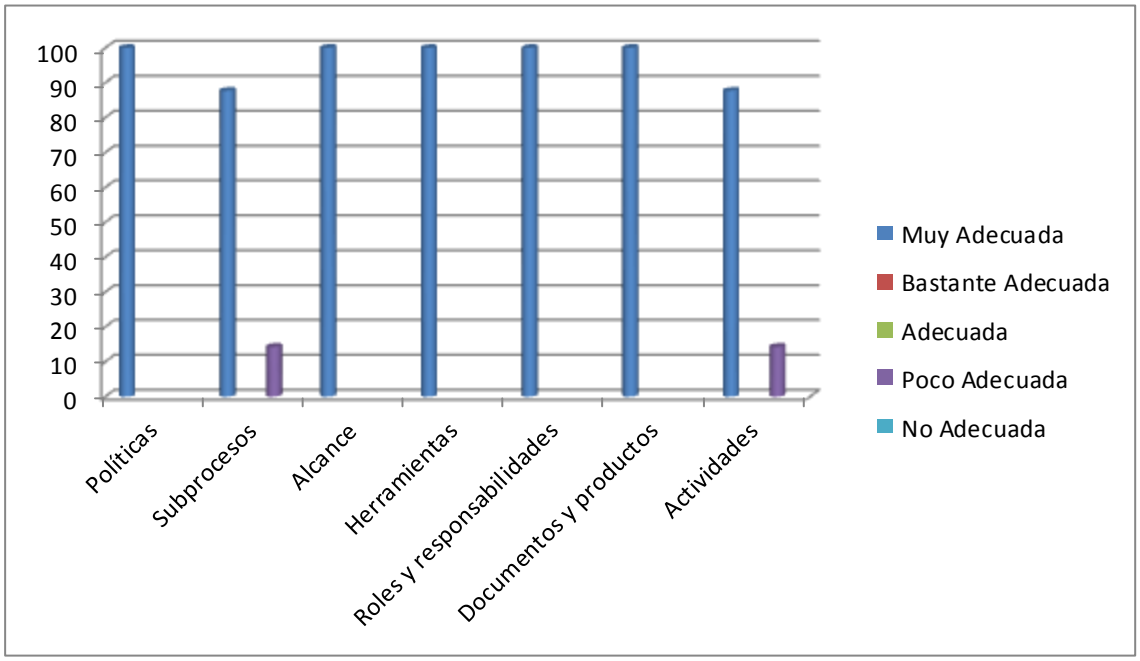


Figura 8 Nivel de adecuación de la pregunta de la Encuesta # 2.

En relación a los temas abordados en el cuestionario de validación los expertos emitieron algunas sugerencias como:

- La propuesta está bien definida, se recomendaría utilizar métricas para valorar la estabilidad de los subprocesos.
- El área de proceso está bien definida y presenta una secuencia lógica de actividades, lo más importante es aplicarla para que trascienda la investigación y ayude a mejorar el trabajo en los proyectos del CEDIN.

Consideraciones parciales.

A razón de la imposibilidad de aplicar la estrategia propuesta en un proyecto, debido al tiempo de ejecución de la metodología de la misma, se consideró aplicar un método de evaluación por expertos. Los resultados obtenidos después de la aplicación del método Delphi se pueden catalogar como satisfactorios. Todos los expertos encuestados consideraron que puede ser efectiva la aplicación de la estrategia propuesta.

Conclusiones Generales

- Se realizó un estudio referente a la Gestión de Configuración de Software que permitió demostrar que para desarrollar y mantener software efectivamente es imprescindible contar con los procedimientos, herramientas y recursos necesarios que permitan administrar los cambios y las configuraciones de software.
- Se realizó un estudio acerca del estado del arte de las Metodologías Ágiles de Desarrollo de Software, lo que permitió determinar sus principales características y vincularlas con el proceso de GCS.
- Se elaboró una estrategia para la Gestión de la Configuración de Software que ayuda a llevar un proceso de desarrollo en proyectos que utilicen Metodologías Ágiles de Desarrollo de Software.
- Se validó la estrategia utilizando el Método Delphi, el cual arrojó un alto nivel de satisfacción por parte de los expertos considerando la propuesta adecuada para aplicar.

Recomendaciones

Para darle continuidad a este trabajo en aras de perfeccionar el proceso de desarrollo de software se recomienda:

- Se recomienda aplicar la estrategia para que trascienda la investigación y apoye el proceso de desarrollo de proyectos ágiles en el centro y posteriormente en la universidad.
- Incluir aspectos sobre la GCS en cursos optativos para brindar una mayor capacitación en cuestiones propias de la Ingeniería y Gestión de Software.
- Potenciar las herramientas seleccionadas con mejores y nuevas funcionalidades para optimizar el proceso de GCS.
- Valorar el proceso propuesto con la utilización de métricas para la GCS.

Referencias bibliográficas.

Babich, Wayne A. 1986. *Software Configuration Management.* 1986.

Beck, Kent. 1999. *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999.

Boehm, Barry W. 1988. *A Spiral Model of Software Development an Eahancement.* s.l. : EEE Computer, 1988. Vol. 21, No. 15.

CMMI. 2010. Sitio oficial CMMI. [Online] 2010. [Cited: 11 10, 2010.] Disponible: http://www.sei.cmu.edu/cmmi/general.

de Antonio, Angélica. 2001. *Gestión de Configuración.* Chile : s.n., 2001.

Delgado Martínez, Ramses and González Soca, Emilio. 2006. *ConfigCASE 3.0 Herramienta de Apoyo a la Gestión de Configuración. Propuesta electrónica.* s.l. : Tesis, 2006.

Gracia, Joaquin. 2005. *CMMI - CMMI Nivel 2.* 2005.

IBM. 2011. Rational ClearCase. [Online] 2011. [Cited: 02 20, 2011.] Disponible: http://www.ibm.com/software/awdtools/clearcase.

IEEE. 2010. IEEE. [Online] 2010. [Cited: 11 20, 2010.] Disponible: http://www.ieee.org/web/aboutus/home/index.html.

ISO. 2010. ISO. [Online] 2010. [Cited: 11 13, 2010.] Disponible: http://www.iso.org/iso/home.html.

Jacobson, Ivar. 2000. *El Proceso Unificado de Desarrollo de Software.* Longman : s.n., 2000.

Leon, A. 2000. *A guide to Software Configuration Management.* Boston : Artech House, 2000.

Palacio, Juan. 2006. *Gestión de proyectos ágil: conceptos básicos.* 2006.

Pressman, Roger S. 1997. *Ingenieria de Software, un enfoque práctico.* Grawhill : 3, 1997.

—. 2002. *Ingeniería de Software, un enfoque práctico.* 2002. Vol. I.

Society, Computer. 1990. *IEEE Standar for Software Configuration Management.* 1990.

Tema5. 2011. [Online] 2011. [Cited: 01 13, 2011.] Disponible: http://www.ual.es/~rguirado/posi/Tema5-Apartado5.pdf.

Bibliografía

Canós, José H, Letelier, Patricio and Penadés, Carmen. *Metodologías ágiles para el desarrollo de software.* Valencia : Universidad Politécnica de Valencia.

Gracia, Joaquin. 2005. *CMMI - CMMI Nivel 2.* 2005.

IEEE, Computer Society. 1990. *IEEE Standar for Software Configuration Management.* 1990.

Moreira, Mario E. 2010. *Adapting Configuration Management for Agile Teams.* s.l. : John Wiley and Sons, 2010.

Glosario de términos.

Artefactos: Una parte de la información que es producida, modificada, o usada por un proceso. Define un área de responsabilidad y está sujeta al control de versiones.

Integración continua: Metodología informática que consiste en compilar y ejecutar pruebas automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes.

No conformidad: Manifestación de insatisfacción que surge al realizarse revisiones y auditorías a los Elementos de Configuración del Software, las distintas versiones liberadas del producto, o a la ejecución de algunos procesos en el proyecto.

Anexos

Anexo 1

Encuesta realizada para la selección de los expertos mediante el cálculo del coeficiente de conocimientos.

Encuesta # 1

(El siguiente cuestionario se aplicará de forma individual y sus resultados permanecerán bajo el absoluto anonimato, solo es de interés para el grupo de trabajo medir el nivel de dominio que tiene el encuestado sobre ciertos temas. El encuestado tiene como deber contestar todas las preguntas respetando siempre las reglas del cuestionario y apoyándose en los conocimientos que realmente posea)

Pregunta 1

Marque con una “x” la opción que dé respuesta a la pregunta que se le realiza en cada inciso.

A. Con la evolución del desarrollo de software se han creado e ido perfeccionando diferentes metodologías que guían este proceso. ¿Conoce las principales características de estas metodologías?

   __ SI
   __ NO

B. Existen varios tipos de metodologías, entre ellas las Metodologías Ágiles de Desarrollo de Software. ¿Conoce las principales diferencia de estas metodologías respecto a las metodologías tradicionales de desarrollo de software?

   __ SI
   __ NO

C. ¿Conoce las principales características del Modelo de Capacidad y Madurez Integrado (CMMI).?

   __ SI
   __ NO

D. Actualmente la UCI se encuentra inmersa en un programa de mejora de procesos, siendo CMMI el modelo que se utiliza. Para alcanzar el nivel 2 de este modelo se implementan 7 áreas de procesos siendo la Gestión de la Configuración de Software una de ellas. ¿Conoce cómo se desarrolla el trabajo en esta área?

__ SI
__ NO

Pregunta 2

La propuesta que se quiere evaluar consiste en una guía para aplicar el proceso de Gestión de la Configuración de Software a proyectos de desarrollo de software donde se empleen métodos ágiles como guía de desarrollo. Por este motivo es necesario poder definir de donde provienen los conocimientos de los encuestados en sentido general y de esta forma tener confiabilidad de las respuestas durante el proceso de validación. Marque con una “x”, según su criterio, su grado de competencia sobre los siguientes aspectos.

Fuentes de los conocimientos	**Calificación**		
	alto	**medio**	**bajo**
Análisis teóricos realizados por usted			
Su experiencia obtenida			
Estudio de trabajos de autores nacionales			
Estudio de trabajos de autores extranjeros			
Su propio conocimiento del estado del problema en el extranjero			
Su intuición			

Anexo 2

Encuesta realizada a los expertos para la validación de la estrategia propuesta.

Encuesta # 2

(El siguiente cuestionario se aplicará de forma individual y sus resultados permanecerán bajo el absoluto anonimato, es del interés del grupo de trabajo conocer las valoraciones del encuestado acerca de la propuesta creada. El encuestado tiene como deber contestar todas las preguntas respetando siempre las reglas del cuestionario, apoyándose en los conocimientos que realmente posea y partiendo del análisis del resumen de la propuesta puesto a su disposición)

Pregunta 1

Marque con una “x” la respuesta que considere correcta para el planteamiento realizado.

a) ¿Cree usted que las 4 políticas definidas garantizan las condiciones necesarias para alcanzar los objetivos del área de proceso Gestión de Configuración de CMMI?

Sí___

No___

b) ¿Considera que hay claridad en la definición de los 4 subprocesos principales propuestos?

Sí___

No___

c) ¿Cree usted que la estrategia tiene alcance a todo el proceso de desarrollo definido en el trabajo?

Sí___

No___

d) ¿Considera que las herramientas propuestas en la estrategia son las adecuadas?

Sí___

No___

e) ¿Cree que los roles y responsabilidades definidos para realizar las actividades del proceso son necesarios y suficientes para aplicar la estrategia en proyectos que usen Metodologías Ágiles de Desarrollo de Software?

Sí___

No___

f) ¿Considera que los documentos que se crearon para acompañar el proceso de Gestión de la Configuración de Software cumplen con el objetivo de servir de guía para los encargados de ejecutar el proceso?

Sí___

No___

g) ¿Considera que la descripción realizada de las actividades del proceso de configuración es lo suficientemente explicita?

Sí___

No___

h) Recomendaciones del encuestado a la estrategia.