Universidad de las Ciencias Informáticas
Facultad V

Gestión de Configuración para el proyecto Simulador Quirúrgico

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores:

Yaines Fernández Concepción

Yani Lis Valle Pérez

Tutor:

Ing. Yusleidy Guelmes León

Ciudad de la Habana, junio 2008.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 11 días del mes de junio del año 2008.

Yaines Fernández Concepción

_____

(Autora)

Yani Lis Valle Pérez

_____

(Autora)

Ing.Yusleidy Guelmes León

_____

(Tutora)

DATOS DE CONTACTO

Tutora:

Ing. Yusleidy Guelmes León

yguelmes@uci.cu

Agradecimientos Compartidos

Deseamos al concluir este trabajo de tesis agradecer a todas las personas que han contribuido de una forma u otra en nuestra formación profesional.

A la Revolución Cubana, por darnos la posibilidad de estudiar en la Universidad de las Ciencias Informáticas y contribuir en nuestro futuro desarrollo.

A la UJC y la FEU por hacer de nosotras jóvenes cada vez más comprometidas con el momento histórico que vivimos.

A Fidel por ser nuestro eterno Comandante.

A nuestra Tutora que nos brindó apoyo durante todo el desarrollo de la tesis

A profesores como Dania que nos enseñaron que no existe el fracaso, salvo cuando dejamos de esforzarnos.

A nuestro querido grupo docente que se ha mantenido a nuestro lado desde los primeros días en la universidad convirtiéndose en parte de nuestra familia.

Al equipo de desarrollo del proyecto SCADA, en especial a Raúl, René, Yasmany, Alejandro y Ariel por brindarnos documentación referente al tema de la tesis y dedicarnos parte de su preciado tiempo.

Agradecimientos

Yaines

A mis padres por ser mis ojos, por guiarme y orientarme en todo momento, por formarme como una mujer de bien y ayudarme a hacer realidad este gran sueño.

A mis hermanitas que más que eso son mis grandes amigas, siempre dispuestas a escucharme y aconsejarme.

A mis abuelos por estar siempre conmigo.

A mis tíos y primos por mantenerse a mi lado y darme fuerzas para seguir adelante.

A mi familia en general por apoyarme siempre que los he necesitado.

A (canelo, mimupo, trinki, yosi, josé) en fin a mi novio, por aguantar mis malcriadeces estos 5 años y brindarme su amor.

A mis suegros por todo el cariño que me han dado y acogerme como si fuera parte de su familia.

A Aiditi la P, Lily y Yanier por brindarme su amistad con tanto cariño y dedicación

A mi compañera de tesis, que más que eso, es mi amiga y sin su ayuda tampoco hubiese llegado a este momento.

A todos MUCHAS GRACIAS!!

Agradecimientos

Yani lis

A mis padres por ser lo mejores y enseñarme que el camino de nuestros miedos a veces es también el de nuestros triunfos.
A mis tíos que los adoro.
A mi abuela que es la luz de mis ojos.
A Kenia, Yorji, Yanier, Heidy y Aiditi por estar siempre dispuestos.

A los chicos del grupo que tanto me ayudaron en mi primer año cuando mi vida era un desastre
A Lily por contagiarme con esa alegría que la caracteriza.
A Ramón que me enseñó a ser fuerte y a decir: Si puedo.
A Yoanny que lo quiero mucho aunque peleemos todo el tiempo.
A Yaines porque más que mi mejor amiga es mi conciencia.
A todos, porque gran parte de lo que soy ahora se lo debo a lo que he vivido estos años en la UCI, a los que de una forma u otra me han apoyado incondicionalmente en los momentos de alegrías y de tristezas, a ustedes a los amigos de siempre: Gracias!!

Dedicatoria

A nuestros padres por ser la fuerza inspiradora
que nos ha guiado estos años
y que han dado todo por ver realizado
este gran sueño.

RESUMEN

En la Universidad de las Ciencias Informáticas (UCI) se realiza gran cantidad de proyectos productivos, pero muchos de estos presentan dificultades que afectan la eficiencia, calidad, y tiempo de desarrollo del producto, además de la productividad y la efectividad del equipo de desarrollo. Según estudios realizados se ha demostrado que muchas de estas dificultades se deben fundamentalmente a las malas prácticas que hay en cuanto a la Gestión de Configuración de Software, disciplina que encierra un conjunto de actividades de seguimiento y control de los cambios que comienzan cuando empieza a desarrollarse el proyecto y termina sólo una vez que el software queda fuera de circulación. El proyecto Simulador Quirúrgico no se encuentra ajeno a este problema, en tal sentido el objetivo del presente trabajo es realizar la propuesta y aplicación del proceso de GCS en dicho proyecto, además de proponer un curso de capacitación para los integrantes del equipo de desarrollo sobre GCS y el funcionamiento de las herramientas escogidas para facilitar el proceso.

PALABRAS CLAVE

Gestión de Configuración de Software, Simulador Quirúrgico.

ÍNDICE

INTRODUCCIÓN

El desarrollo de software constituye un sector de vital importancia a nivel mundial, se encuentra en el centro de todas las transformaciones; sobre todo si se considera que los grandes temas del momento, como lo son: la economía digital, la evolución de las empresas y el conocimiento, se resuelven con software. La demanda que actualmente existe de su producción asciende a un ritmo vertiginoso, desarrollar productos que cumplan con los requisitos y que respondan a la planificación y el presupuesto establecido son factores trascendentales para el éxito de una empresa. El software se ha convertido en el elemento clave de la evolución de los sistemas y productos informáticos, pasando de ser una resolución especializada de problemas y una herramienta de análisis de información, a una industria por sí misma.

La gran competencia entre las organizaciones que se desenvuelven en esta rama a nivel mundial exige que se deba asegurar que el proceso de desarrollo de software esté dirigido a perfeccionar el nivel de satisfacción del cliente como resultado de las actividades, lo que implica mejorar la calidad y la eficiencia del producto final.

Cuba no esta ajena a ese universo de invención y perfeccionamiento a pesar de ser un país donde el desarrollo de software es aún incipiente. Es por ello que una de las principales tareas del Gobierno es el progreso de la Industria del Software. Desgraciadamente en las industrias alejadas de grandes centros mundiales, se hace muy difícil obtener rápidamente adelantos tecnológicos y aun dentro del territorio nacional hay empresas con mayores o menores recursos. Pero algo impresionante es que se ha podido, aunque con esfuerzo, suplir gran parte de estos adelantos y finalmente se llega a competir en algún grado, reemplazando la alta tecnología con mayor rendimiento, eficiencia y superando cada vez más la calidad de fabricación.

La situación actual y las perspectivas de la industria de software cubana están caracterizadas por el trabajo que se viene realizando en materia de capacitación y en la inserción tanto en el mercado nacional como internacional. A esto se le adiciona la puesta en marcha de un nuevo paradigma productivo: Fábrica de Software; cuyo objetivo es responder productivamente a demandas crecientes del mercado Europeo y de Latinoamérica; organizados productivamente según los estándares internacionales [5].

En la Cumbre Mundial para la Sociedad de la Información en el 2004 el Ministerio de Relaciones Exteriores de Cuba presentó un informe en el que plantea:

"La Industria Cubana del Software (InCuSoft) está llamada a convertirse en una significativa fuente de ingresos para el país, como resultado del correcto aprovechamiento de las ventajas del alto capital humano disponible. La promoción de la industria cubana del software en el ámbito internacional ha tenido como línea estratégica aprovechar la enorme credibilidad que tiene Cuba en sectores tales como la salud, la educación y el deporte. El continuar la producción sostenida de software de alta calidad en prestaciones, imagen y soporte, para satisfacer las necesidades nacionales en estos sectores, tendrá una positiva repercusión en el incremento de la exportación" [9].

Estos años han sido testigo de que se han invertido los esfuerzos en la mejora de la calidad de los procesos y productos en aras de lograr esa ambiciosa meta, sin embargo se hace necesaria la aplicación de un nuevo enfoque en la producción del software, más cercano a una disciplina de ingeniería que a los hábitos y modos artesanales que, desafortunadamente, se han venido aplicando desde hace algún tiempo. Por tanto, si se quieren logros relevantes se debe comenzar la lucha contra este tipo de problema, pues aunque la demanda de software crece exponencialmente, en el territorio nacional su producción actualmente es baja y costosa. Esto indica que para elevar las oportunidades de la naciente Industria Cubana del Software se debe tener en cuenta aspectos importantes que son esenciales para lograr el éxito de un proyecto y por consiguiente de manera sostenida el de una empresa de software, estos son [1]:

- La Gestión de proyecto.
- El desarrollo técnico.
- El Sistema de Calidad, que incluyen las actividades de validación.
- El Sistema de Gestión de Configuración de Software (GCS).

La Universidad de las Ciencias Informáticas (UCI) es un ejemplo de cómo Cuba comienza a recorrer el camino hacia el universo del software. Fue creada en el 2002 con la idea de potenciar el desarrollo de software y el objetivo de formar profesionales íntegros, que fusionando la producción y la docencia podrían lograr mejorar la elaboración de software en Cuba. En esta se realizan un gran número de proyectos de desarrollo de software, y paulatinamente se va ganando en cantidad y complejidad, sin embargo, según estudios realizados se comprobó que muchos de los problemas en la

producción de software de la universidad surgen como resultado de que existen proyectos que no tienen definido que elementos desean controlar, pues no se realiza correctamente un control de las versiones del software ni el de los cambios que ocurren en el mismo.

Esto es a consecuencia de la poca experiencia que hay en cuanto a la gestión de configuración en los procesos de desarrollo de software de la universidad y trae como resultado una mayor utilización del recurso humano calificado, provocando atrasos en actividades de desarrollo y mantenimiento de los productos y un inadecuado control de los proyectos, sobre todo en aquellos que comienzan a desarrollarse. En este caso se encuentra el proyecto Simulador Quirúrgico que se crea en la facultad 5 (facultad que inclina su trabajo fundamentalmente al perfil de Realidad Virtual y Automatización) el mismo emerge de la necesidad que hay de contribuir a la solución del problema del entrenamiento de la cirugía de mínimo acceso en el país y la formación actualizada de estudiantes de pregrado y postgrado, debido a la importancia que traería para el país su desarrollo, se hace imprescindible controlar la evolución de los cambios para así lograr la futura integridad y consistencia del producto. Actualmente el proyecto consta de 36 integrantes, entre ellos profesores y estudiantes que no se encuentran cursando el mismo año de la carrera, además se trabaja en horarios diferentes, esto trae consigo que se afecte la coordinación entre ellos fundamentalmente a la hora de hacer modificaciones importantes.

Por lo antes expuesto y teniendo en cuenta que la gestión de configuración es el fundamento de un proyecto de software, se plantea la siguiente interrogante como **Problema de Investigación:** ¿Cómo aplicar una Gestión de Configuración que facilite el trabajo y control de los cambios al proyecto Simulador Quirúrgico?

El **objeto de estudio** se enmarca en el Proceso de desarrollo de Software, teniendo como **campo de acción:** El proceso de Gestión de Configuración de Software en el proyecto Simulador Quirúrgico.

Para dar respuesta al problema de investigación, se definió como **objetivo general:** Planificar la Gestión de Configuración al proyecto Simulador Quirúrgico**.**

Para cumplir con lo planteado anteriormente se realizaron las siguientes **tareas:**

- Estudiar la teoría referente al proceso de Gestión de Configuración de Software.

- Analizar de qué forma las diferentes metodologías desarrollan el proceso de Gestión de Configuración de Software.
- Investigar cómo otros proyectos de la universidad llevan a cabo el Control de los cambios.
- Definir las actividades a realizar para implantar el proceso de Gestión de Configuración en el proyecto Simulador Quirúrgico.
- Definir las herramientas de software a usar.
- Definir los elementos de configuración, líneas base y actividades de la Gestión de Configuración para el proyecto Simulador Quirúrgico.
- Instruir al equipo de desarrollo en el uso de las herramientas y procedimientos.
- Poner en práctica el proceso (responsabilidad conjunta con el Líder del proyecto)
- Evaluar el nivel de aceptación en el equipo de desarrollo del proceso de Gestión de Configuración aplicado en el proyecto.

Al finalizar de este trabajo se esperan como **Posibles resultados**:

- Definición del proceso de GCS en el proyecto Simulador Quirúrgico.

- Instalación y configuración de las herramientas de trabajo para el control de los cambios en el proyecto Simulador Quirúrgico, seleccionadas sobre bases fundamentadas en un estudio.

- Mayor capacitación de los miembros del equipo de trabajo del Simulador Quirúrgico, en cuanto al manejo de las herramientas.
- Creación del Plan de Gestión de Configuración al proyecto Simulador Quirúrgico.

Se utilizaron en la elaboración de esta investigación los siguientes métodos científicos:

Métodos teóricos

Dentro de este método se encuentran:

- **Histórico-lógico:** se utilizó para hacer un estudio detallado de los trabajos anteriores referentes a la Gestión de Configuración del Software (GCS), como fue la consulta de tesis, planteamientos

hechos por instituciones importantes como por ejemplo el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) y la Organización Internacional para la Normalización (ISO) que hacen referencia al tema de la investigación.

- **Inductivo - Deductivo:** Se realizó un estudio del proceso de la Gestión de Configuración del Software (GCS) en diferentes proyectos de la universidad obteniéndose elementos que pudieran aplicarse en nuestro proyecto,
- **Análisis-síntesis:** se realizó el análisis de documentos y teorías referentes a este tema, seleccionando los elementos concretos que se ajustan al desarrollo de la investigación.

Métodos empíricos

Dentro de este método se utilizó**:**

- **Entrevista:** se entrevistaron expertos en el tema para obtener información acerca del procedimiento en cuestión, además de diferentes profesores, líderes de proyectos y estudiantes.

- **Encuesta:** se utilizó para saber el nivel de conocimiento y aceptación del proceso de Gestión de Configuración entre los integrantes del equipo, para ponerlo en marcha.

Este trabajo consta de una introducción, tres capítulos, conclusiones y recomendaciones. En el **capítulo I** se hace un recorrido por el estado del arte sobre los temas de Gestión de Configuración de Software. En el **capítulo II** se caracteriza y diseña el proceso a aplicar en el proyecto, lo que constituye el objetivo principal de este trabajo, además se hace el estudio de las herramientas que se utilizarán. En el **capítulo III** se exponen los resultados obtenidos y se realiza la evaluación de aceptación del proceso aplicado.

CAPÍTULO I

Fundamentación Teórica

Introducción

En este capítulo se hace referencia al estado del arte a nivel internacional, nacional y de universidad, respecto a lo investigado y desarrollado sobre la Administración de la Configuración, teniendo en cuenta definiciones y planteamientos hechos sobre el tema por personalidades e instituciones importantes. Además se trata la importancia que tiene el proceso de desarrollo de software y todos los aspectos de la Gestión de Configuración haciendo alusión al proceso de GCS en algunas metodologías y modelos de calidad.

1.1 Proceso de Desarrollo del Software

La Ingeniería de Software es una tecnología que indica como construir técnicamente un software: económico, fiable y que funcione eficientemente, es decir, es la rama de la ingeniería que crea y mantiene las aplicaciones de software aplicando tecnologías y prácticas de las ciencias computacionales, manejo de proyectos, ingeniería, el ámbito de la aplicación, y otros campos.

Uno de los objetivos de cualquier proyecto o mantenimiento de software, independientemente de su envergadura, es entregar un producto de calidad, o sea que el producto construido cumpla con las especificaciones establecidas, dentro de los tiempos, costos y recursos planificados o acordados. Para esto, se hace necesario que el equipo de trabajo adopte un proceso de desarrollo de software, es decir, un marco que defina las actividades necesarias para garantizar, técnica y administrativamente, que un software pueda ser construido o mantenido de manera organizada, disciplinada y previsible.

El proceso de desarrollo de software “es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo”. Concretamente “define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo” [8].

Si se sigue un proceso de desarrollo que se ocupa de plantear cómo se realiza el análisis y el diseño, y cómo se relacionan los productos de ambos, entonces la construcción de sistemas software va a poder ser planificable y repetible, y la probabilidad de obtener un sistema de mejor calidad al final del proceso aumentará considerablemente, especialmente cuando se trata de un equipo de desarrollo formado por varias personas

1.2 Configuración del Software

En el Proceso de Desarrollo de Software se obtienen elementos que pueden ser catalogados como ejecutables, códigos fuentes, documentación, ayudas, datos y otros; estos constituyen el centro del desarrollo de un software, es todo lo que se produce, utiliza, emplea, se da mantenimiento y repara. Al conjunto de esta información creada como parte del proceso de ingeniería durante el ciclo de vida del proyecto se le llama *Configuración de Software*.

Cada uno de estos componentes por separados es denominado Elemento de Configuración de Software (ECS). Un ECS debe ser un elemento que se pueda definir y controlar de forma separada. Es decir, debe ser una unidad en sí mismo [1].

Se pueden considerar como ECSs:

- Especificaciones del Sistema.
- Estimaciones y Planes.
- Especificación de requisitos software.
- Diseño arquitectónico.
- Diseño detallado.
- Prototipos generados.
- Código fuente.
- Documentación relacionada con la determinación de los factores de riesgo y su gestión a efectos de minimizar sus consecuencias.
- Programas ejecutables y librerías asociadas.
- Manuales del usuario, de operación e instalación.
- Documentación relacionada con cursos de formación en el uso del producto.

- Plan de pruebas.
- Casos de Prueba y resultados obtenidos.
- Estándares y procedimientos de Ingeniería de Software utilizados.
- Informes de incidencia.
- Pedidos de mantenimiento.
- Órdenes de cambio.
- Documentación del Software y Hardware utilizados como herramientas de Desarrollo.
- Diseño de bases de datos.
- Bases de Datos.
- Información del entorno de desarrollo y de implantación.
- Contenidos iniciales de las bases de datos.

Estos se pueden organizar como objetos de configuración y deben ser archivados en el repositorio del proyecto con un nombre único. Para cada proyecto concreto se debe determinar qué se va a considerar como ECS y debe quedar claro que es función del Jefe de Proyecto tomar esta decisión en dependencia de lo que tenga que implementar y la profundidad con la que se querrá hacerlo.

1.3 Gestión de Configuración de Software

En el desarrollo de software, los cambios, debidos principalmente a modificaciones de requisitos y fallos, son inevitables. Normalmente se trabaja en equipo, por lo que es preciso llevar un control y registro de los cambios en el afán de reducir errores que podrían provocar confusión e incertidumbre, sobre todo cuando no se han analizado o pronosticado correctamente, esto influye, por supuesto, en la calidad y la productividad del producto final, por tanto se hace necesario considerar ciertas modificaciones en el transcurso de su ciclo de vida, con el fin de evitar problemas que puedan acarrear una incorrecta sincronización y afectar a otros elementos del sistema o a las tareas realizadas por otros miembros del equipo del proyecto.

Una definición en términos sencillos de Gestión de Configuración es brindada por Angélica de Antonio [1]: "… disciplina, cuya misión es controlar la evolución de un sistema software" En realidad esta definición de Gestión de Configuración (GC) es muy simple aunque bastante abarcadora.

Para Babich [2] "El arte de coordinar el desarrollo de software para minimizar la confusión se denomina Gestión de Configuración. … es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de programación. El objetivo es maximizar la productividad minimizando los errores."

El diccionario de términos IEEE [3] define la GCS como el proceso de identificar y definir los elementos de configuración en un sistema, controlando la entrega y el cambio de estos elementos a través del ciclo de vida del sistema, almacenando el estado de los elementos de configuración y de las solicitudes de cambio, y verificando la completitud con respecto a los requerimientos especificados.

En otras palabras se puede decir que la gestión de configuración es la encargada de mantener la integridad de los productos que se obtienen a lo largo del desarrollo del software, garantizando que no se realicen cambios incontrolados y que todos los participantes en el desarrollo del sistema dispongan de la versión adecuada de los productos que manejan.

La GCS se realiza durante todas las actividades asociadas al desarrollo del sistema, y continúa registrando los cambios hasta que éste deja de utilizarse, facilitando el desenvolvimiento del mismo y aportando información precisa para valorar el impacto de los cambios solicitados y reduciendo el tiempo de implementación de estos. Asimismo, permite controlar el sistema como producto global a lo largo de su desarrollo, obteniendo informes sobre el estado en que se encuentra y reduciendo el número de errores de adaptación del sistema, lo que se traduce en un aumento de calidad del producto, y como resultado la satisfacción del cliente.

Algunos aspectos fundamentales y a menudo muy tradicionales dentro del desarrollo del software que puede traer consigo cambios al sistema son:

- Nuevos requisitos del negocio o condiciones que dictan los cambios en las condiciones del producto o en las normas comerciales.
- Nuevas necesidades de los clientes que demandan la modificación de los datos producidos por un sistema basado en computadora.
- Reorganización y/o reducción del volumen comercial que provoca cambios en las prioridades del proyecto o en la estructura del equipo de ingeniería del software.

- Restricciones presupuestarias o de planificaciones que provocan una redefinición del sistema o del producto.

La norma ISO 10007 [7] por su parte define como el objetivo de las actividades de la Gestión de Configuración, establecer y mantener la integridad y coherencia del producto software a fin de facilitar el seguimiento de los cambios que sobre él se implementan, asegurando la posibilidad de realizar auditorías de control sobre la evolución de las diferentes configuraciones.

De forma más específica:

- Dar soporte a los métodos de desarrollo.
- Mantener la integridad del producto.
- Asegurar que los productos desarrollados sean completados y corregidos debidamente.
- Proveer un ambiente estable durante el desarrollo del producto.
- Restringir los cambios a los productos según las políticas del proyecto.
- Proveer una brecha para auditorías que permita identificar el por qué, cuándo y por quién ha sido modificado un proyecto.
- Mantener la consistencia de los productos durante la evolución de los mismos.
- Proveer datos para medir el progreso.
- Proveer los medios eficientes para adaptarse apropiadamente a los cambios y a los replanteamientos de planes de trabajo.

1.3.1 Línea Base

El requerimiento de un cambio durante el desarrollo del software generalmente es justificado, "…los clientes quieren modificar los requisitos. El equipo de desarrollo quiere modificar el enfoque técnico. Los gerentes desean modificar el enfoque del proyecto… A medida que pasa el tiempo, todo el mundo sabe más (sobre lo que necesita, sobre el mejor enfoque posible al problema y sobre cómo hacerlo ganando dinero)..." [11]. Así surgen entonces, con el propósito de controlar los cambios: las *Líneas Base* que se constituyen en función de la aprobación de uno o varios ECSs mediante la ejecución de revisiones técnicas formales.

La IEEE (Estándar IEEE 610.12-1990) define una línea base como una especificación o producto que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios. [12]

Una línea base es un concepto de gestión de configuración del software que ayuda a controlar los cambios sin impedir seriamente los cambios justificados.

En el contexto de la ingeniería del software se define una línea base como un punto de referencia en el desarrollo del software y que queda marcado por el envío de uno o más elementos de configuración del software (ECSs) y la aprobación de estos obtenida mediante una revisión técnica formal.

1.3.2 Actividades de la Gestión de Configuración

La gestión de configuración del software realiza un conjunto de actividades desarrolladas para gestionar y registrar los cambios a lo largo del ciclo de vida del software de computadora.

Estas actividades según RUP son [6]***:***

1. Identificar (a los ECSs)

2. Controlar (los cambios)

3. Auditar (la forma en que se implementa el proceso)

4. Informar el estado (reportes) (de los cambios y el proceso)

Estas actividades han sido definidas por diferentes autores, instituciones y metodologías de forma bastante homogénea, sin lugar a dudas el éxito está en tener claros cuales son los elementos qué componen el software, controlar los cambios que sufren estos y las versiones que se generan a partir de esos cambios, manteniendo en todo momento la comunicación entre los involucrados en el proceso de desarrollo. Usualmente la forma más popular de definir estas actividades son las que especifican Pressman [12] y la IEEE [13]:

- ***Identificación de la configuración*:** Consiste en identificar la estructura del producto, sus componentes y tipo, haciéndolos únicos y accesibles mediante algún procedimiento.
- ***Control de Versiones:*** Consiste en combinar procedimientos y herramientas para gestionar las versiones de los objetos de configuración creados durante el proceso del software.
- ***Control de cambios en la configuración:*** Consiste en controlar las versiones de un producto y los cambios que sobre él se producen a lo largo de su ciclo de vida.
- ***Auditorías de configuración***: Consisten en la validación de la integridad de un producto, manteniendo la consistencia entre sus componentes.
- ***Generación de informes de estado:*** Consisten en la producción de informes sobre el estado de los ECSs de un producto y de las solicitudes de cambio realizadas sobre ellos.

Identificación de la configuración

En esta actividad se realizan las siguientes actividades:

1-Identificación de los elementos de configuración (ECSs): La misma consiste en la selección de los elementos de configuración para un sistema y almacenar tanto sus características funcionales como físicas en un documento técnico. Debe crearse una nomenclatura única de forma tal que el ECS quede etiquetado de forma unívoca. Las etiquetas identifican las versiones de los ECSs.

Sea cual fuere el esquema de identificación utilizado debe proporcionar, al menos, la siguiente información:

- Identificación del proyecto.
- Fase a la que pertenece el ECS.
- Tipo de ECS.
- Individualización del ECS.
- Versión.
- Fecha de creación del ECS.
- Descripción.
- Identificación de los ECSs con los que está relacionado.

Esta información puede codificarse y agruparse en un único código de identificación o puede referenciarse por separado.

Para controlar y gestionar los ECSs se puede seguir un enfoque orientado a objetos. Se pueden identificar dos tipos de objetos: [10]

- Objetos básicos.
- Objetos compuestos.

Un objeto básico es una unidad de texto creada por un miembro del equipo durante el proceso de IS, por ejemplo:

- Una parte del diseño.
- Un listado de código.
- Un conjunto de casos de prueba.

Un *objeto compuesto* es una colección de objetos básicos y de otros objetos compuestos. Conceptualmente se pueden ver como una lista de referencias que identifican los objetos básicos que los componen. Cada objeto tiene un conjunto de características que lo identifican:

- Nombre.
- Descripción.
- Lista de recursos.
- Lista de realización.

El nombre del objeto es una cadena de caracteres que identifican al objeto sin ambigüedad. La descripción es una lista que identifica el tipo de ECS, un identificador de proyecto, la información de la versión y/o cambio. Los recursos son entidades que proporciona, procesa, referencia o son, de alguna otra forma requeridos por el objeto. La realización es una referencia a la unidad de texto para objetos básicos y nulos para objetos compuestos. [10]

2-*Identificación del esquema de almacenamiento de los ECSs en la base de datos o repositorio del proyecto:* El propósito es el de asegurar que el proyecto de software y sus documentos son salvados, catalogados y transferidos a un sitio definido (repositorio o base de datos del proyecto).

3-Identificación de las relaciones que poseen los ECSs: Se pueden organizar como objeto de configuración con su respectivo nombre y atributos conectándolo a otros objetos mediante relaciones. *Estas* relaciones son básicamente de agregación y dependencia. Además se puede especificar en el cuerpo del elemento las posibles relaciones que puede tener con otros ECSs.

4-*Identificación de los momentos en que vamos a definir las diferentes líneas base*:

Existen tres razones por las que se establecen las líneas base:

Reproducibilidad: Es la habilidad de retroceder en el tiempo y reproducir una determinada liberación de un sistema de software o reproducir un determinado ambiente de desarrollo en cualquier momento anterior del proyecto.

Trazabilidad: Establece la relación de antecesor y sucesor entre los diferentes artefactos del proyecto (ECS).

Reportes: Los reportes están basados en la comparación del contenido de una línea base con relación a otra. La comparación de las líneas base ayudan a generar las notas de liberación.

Todos los elementos que constituyan una línea base necesitan ser etiquetados de forma tal que puedan ser identificados unívocamente.

Control de Versiones

Una versión es un número que indica el nivel de desarrollo de un programa. Es habitual que una aplicación sufra modificaciones, mejoras o correcciones durante su ciclo de vida, todos estos cambios sobre un objeto al que ya se le trazó su línea base conducen a la creación de una nueva versión del objeto. Como no se puede definir cuantas versiones podría tener un producto es necesario realizar un control de versiones.

Según Pressman [11] “La gestión de configuración permite a un usuario especificar configuraciones alternativas del sistema software mediante la selección de versiones adecuadas. Esto se puede gestionar asociando atributos a cada versión del software y permitiendo luego especificar y construir una configuración describiendo el conjunto de atributos deseado.”

Se puede observar una representación de diferentes versiones de un producto en el gráfico de evolución de la figura 1. Cada nodo del gráfico representa a una versión completa del producto software. Cada versión del software es una colección de líneas base que contienen todos los ECSs y cada versión puede estar compuesta por diferentes variantes. Se define variante como una versión que es una alternativa a otra versión. Las variantes pueden permitir a un elemento de configuración satisfacer requerimientos en conflicto, constituyen una nueva versión de un elemento que será añadida a la configuración sin reemplazar a la versión anterior.

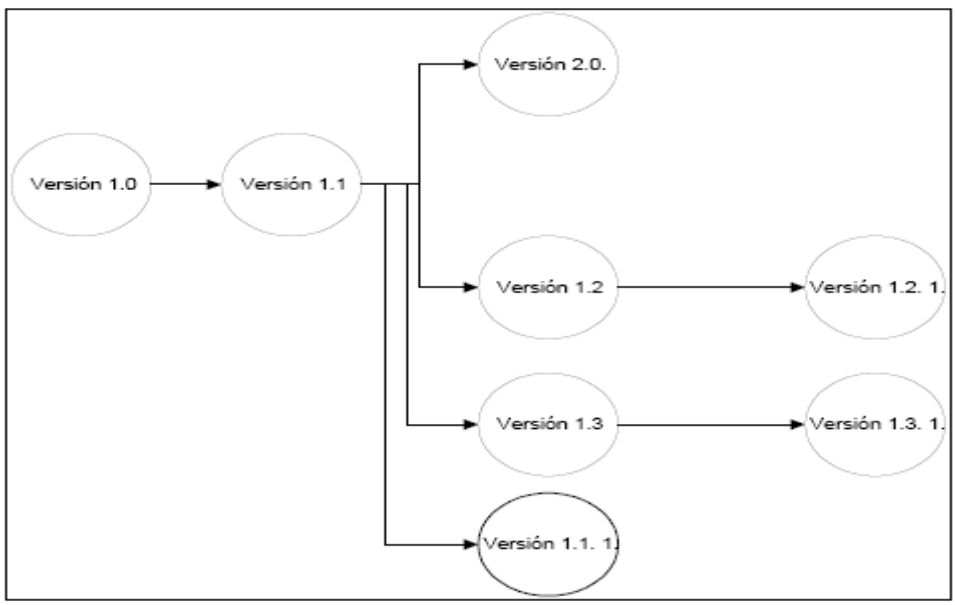

Fig. 1 Grafo de evolución

Control de cambios en la configuración

El proceso de control de cambios puede ser considerado como la actividad más trascendente de la Gestión de Configuración, ya que asegura que los impactos que se puedan generar producto de

cambios en el proyecto, sean considerados y busca establecer los mecanismos para gestionar adecuadamente estos cambios.

Esta actividad debe ofrecer instrumentos que permitan:

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

Para darle solución a cada uno de estos elementos primeramente es necesario crear un *Comité de Control de Cambios* encargado de tomar las decisiones finales acerca del estado y la prioridad de las peticiones de cambio y estará integrado por miembros del equipo de desarrollo que poseen una autoridad real sobre el resto de los integrantes así como deberán tener suficiente experiencia para impedir cambios cuyo propósito sea imprudente o costoso. Luego se precisa describir e implantar un proceso mediante el cual sea analizado cada solicitud de cambio para con ello lograr un canal único y evitar cambios no aprobados quedando definido de esta forma un *Proceso de Solicitud de Cambio*.

En el ciclo de vida de un proyecto los integrantes del grupo de desarrollo podrán realizar: el llamado *Cambio Informal* de forma justificada a los ECSs que ellos mismos hayan desarrollados antes de que este pase a formar parte de una línea base; el *Cambio a nivel de proyecto:* aparece una vez que un ECS pasa la revisión técnica formal y se convierte en una línea base, en este caso para poder realizar el cambio, el encargado de realizar este cambio debe recibir la aprobación del Líder del proyecto, en el caso de que el cambio sea local, o del Comité de Control de Cambio (CCC) si el cambio influye en otros ECSs y por último cuando el producto es distribuido a los clientes y se hace necesario realizar cierta modificación se está en presencia de un *Cambio Formal*, para ser efectuado debe ser aprobado por el CCC.

"El arte del progreso está en mantener el orden en medio del cambio y mantener el cambio en medio del orden". Alfred North Whitehead. [11]

Auditoría de la configuración

Para una mayor consistencia de todas las tareas de la Gestión de Configuración mencionadas, resulta necesario asegurarse que los cambios han sido implementados correctamente. El proceso de gestión de configuración con este propósito prevé la ejecución de diferentes actividades de control de calidad nombradas Auditorías de Configuración.

Generalmente las auditorías que se efectúan son:

- ***Auditoría funcional:*** Cuyo objetivo es comprobar que se han completado todas las pruebas necesarias para el / los ECSs auditados, y que, teniendo en cuenta los resultados de los tests, se puede afirmar que el / los ECSs satisfacen los requisitos que se impusieron sobre él.
- ***Auditoría física:*** Cuyo objetivo es verificar la adecuación, integridad y precisión del producto final.
- ***Revisión formal de certificación:*** Cuyo objetivo es certificar que el / los ECSs se comportan correctamente en su entorno operativo.

Generación de informes de estado

Cuando en el ciclo de vida de un software, se crea o se actualiza un ECS, se emite una orden de cambio o se lleva a cabo una actividad de auditoría, se hace inevitable la generación de informes de estado de la configuración (IEC). Esta actividad persigue como objetivo proporcionar información sobre cada cambio y su dinámica a gestores y desarrolladores. Es una tarea de la Gestión de Configuración de Software que responde a las siguientes preguntas:

- ¿Qué pasó?
- ¿Quién lo hizo?
- ¿Cuándo pasó?
- ¿Qué más se vio afectado?

En esta actividad para gestionar los cambios y generar los informes correspondientes se producen dos tipos de documentación:

- Registros: Documentos o bases de datos donde se acumula información relacionada con la Gestión de Configuración de un determinado producto.
- Informes: Formularios, generalmente prefijados, donde se vuelca información o datos referidos a registros relacionados con la Gestión de Configuración (Líneas Base, Solicitudes de Cambio, ECS, Control de Versiones, etc.)

Para lograr una mejor comunicación sobre las modificaciones que invariablemente se dan al desarrollar el software, cuando se adopte una metodología de Gestión de Configuración, deben establecerse claramente cuales serán los registros e informes que se deberán implementar.

1.4 GCS en Modelos de calidad

La época actual conduce a un mundo competitivo donde las empresas de software dirigen su objetivo a desarrollar productos de alta calidad. Sin embargo en muchos casos no se tiene en cuenta que la Gestión de la Calidad de Software es una actividad de protección que se aplica a lo largo de todo el proceso del software y no una vez que se ha generado el código. Philip Crosby en relación a esta situación plantea: "El problema de la gestión de la calidad no es lo que la gente no sabe sobre ella. El problema es lo que creen que saben……" [4]

Para llevar a cabo la Gestión de Calidad de Software se han seguido principalmente dos tendencias: la primera a seguir son las reglas implantadas por las oficinas internacionales de estandarización para los productos y servicios a través de las normas ISO y la IEEE, y la segunda a seguir constituyen las creadas específicamente para el mundo del software como CMMI y SPICE.

1.4.1 Organización Internacional para la Normalización (ISO)

ISO es la Organización Internacional para la Normalización (International Organization for Standarization) que tiene como finalidad principal la de promover el desarrollo de estándares internacionales y actividades relacionadas incluyendo la conformidad de estos para facilitar el intercambio de bienes y servicios en todo el mundo.

En 1987 ISO creó la serie de estandarización ISO 9000 tomando como base los elementos de las normas británicas BS 5750 (British Standard Institute, BSI.). Estos modelos no pretenden establecer una uniformidad de los sistemas de calidad, al contrario esta reconocido en dichas normas la variedad para ajustarse a las cambiantes necesidades de cada industria.

ISO 9000 brinda al usuario una guía para la selección y uso de los modelos de sistemas de calidad ISO 9001, ISO 9002, ISO 9003 e ISO 9004. Un modelo relevante dentro de estos estándares para la Industria del Software lo constituye ISO 9003 pues fue creado como una guía para la aplicación de ISO 9001 en la producción y mantenimiento de un software. Dentro de los aspectos que tiene en cuenta se encuentran:

- Sistema de calidad.
- Especificación de los requisitos del comprador.
- Planificación del desarrollo.
- Planificación de la calidad.
- Pruebas y validaciones.
- Gestión de Configuración de Software.
- Mediciones

ISO 9003 consta de diversas secciones en las que desarrolla actividades que deben realizar y mantener las empresas que utilicen esta norma, dentro de estas actividades se encuentran: el control de documentos y datos, garantizando que todos los documentos relacionados con esta norma sean controlados, ubicados en lugares disponibles y removidos en caso de ser obsoletos; otra de las actividades son las auditorías con el objetivo de verificar que las actividades de calidad cumplan con lo planeado, estas auditorías son llevadas a cabo de acuerdo a los procedimientos documentados y el resultado de las mismas debe ser documentado y mostrado al encargado del área auditada el cual tomará acciones correctiva sobre las deficiencia encontradas. Las secciones anteriormente explicadas coinciden con dos de las actividades del proceso de GCS, uno de los temas que incluye ISO 9003 a pesar que no se encuentra dentro de las normas genéricas de ISO 9000 ya que pueden existir productos o servicios que para alcanzar un alto nivel de calidad sea necesario la utilización de otros procesos específicos que no contenga esta norma.

1.4.2 Integración del Modelo de Madurez de las Capacidades (CMMI)

La producción del software a inicios de la década de los 90 es afectada por una serie de problemas que dañan la calidad del producto. Teniendo en cuenta la necesidad de solucionar las deficiencias derivadas en este sector es creado en 1991 por el Instituto del Software (SEI) el modelo de calidad CMM. CMM para Software o CMM-SW no es más que una descripción de los escenarios por los cuales las organizaciones de software avanzan, es decir cómo estas definen, implementan, controlan y desarrollan los diferentes pasos en la elaboración del software.

Dado el éxito del modelo y al avance en las metodologías, CMM-SW es ampliado y en evolución del mismo surge en diciembre del 2001 la Integración del Modelo de Madurez de las Capacidades (CMMI, Capability Maturity Model Integration). CMMI dirige las organizaciones a la mejora de los procesos a lo largo de un proyecto; puede ser empleado para integrar funciones organizacionales y otorgar metas, además de brindar una guía para la obtención específica de la calidad.

Clasifica las empresas mediante los 5 niveles de madurez que define de forma escalonada:

- Nivel1 (Inicial).
- Nivel2 (Repetible).
- Nivel3 (Definido).
- Nivel4 (Gestionado).
- Nivel5 (Optimizado).

Una empresa para implantar de forma adecuada este modelo o simplemente mejorar su forma de trabajar para adquirir mejores resultados es de vital importancia que alcance el nivel 2. Este nivel tiene como principal objetivo el de lograr que el proyecto sea gestionado y controlado durante su desarrollo, llevando a cabo las siguientes tareas:

- Gestión de los requisitos del producto y del proyecto.
- Planificación de los proyectos.
- Seguimiento y control de los proyectos.
- Gestión de acuerdos con los proveedores de productos y servicios.

- Selección y supervisión de los proveedores.
- Medición y análisis.
- Aseguramiento de la calidad del producto y del proceso.
- Gestión de Configuración de Software.

Se puede deducir que el proceso de GCS en este nivel 2 o Repetido tiene un lugar básico pues en gran medida parte de sus objetivos son abarcados con una adecuada realización de las actividades de la Gestión de Configuración.

1.4.3 Instituto de Ingenieros Eléctricos y Electrónicos (IEEE)

Con el fin de lograr organización, control y supervisión de las tareas durante el desarrollo del software ha sido creado basado en estándares del Instituto de Ingenieros en Electricidad y Electrónica (Institute of Electrical and Electronics Engineers, IEEE) el Plan de Garantía de Calidad de Software. Dicho plan se basa en el estándar IEEE 730-1998 para controlar y documentar las etapas del ciclo de vida del software con el fin de obtener un producto íntegro y que satisfaga las necesidades del cliente. Uno de sus requerimientos lo constituye la Gestión de Configuración de Software adjudicándole gran importancia dentro de la documentación al Plan para la Gestión de Configuración de Software reflejado en el estándar IEEE 828-1998.

1.5 El proceso de GCS en algunas metodologías

Muchas veces cuando se desarrolla un software se realiza su diseño de forma rígida, sin tener en cuenta que a pesar de que el producto en su etapa de prueba cumpla con los requerimientos del cliente, el mismo puede sentir la necesidad de solicitar un cambio. En este momento el equipo se encuentra en una situación difícil pues si lleva a cabo este cambio se podrían alterar cosas que no se tenían previstos, ocasionando un atraso en el proyecto y en caso contrario provocaría la incomodidad por parte del cliente por no tomar en cuenta su pedido. La solución a este problema está en utilizar metodologías de desarrollo que guíen el equipo mediante un conjunto de procedimientos que permitan producir y mantener software de alta calidad. La Gestión de Configuración de Software es el proceso encargado de llevar el control de todos los cambios, a continuación se realizará un breve análisis de cómo algunas metodologías conciben este proceso.

Extreme Programming (XP) es una metodología de desarrollo que fue creada por Kent Beck cuando entró a formar parte en un proyecto de Daimler Chrysler, donde debido a las necesidades del propio proyecto, se vio obligado a poner en marcha una serie de prácticas aprendidas anteriormente, las que fue refinando hasta lo que se convertirían en las bases de XP.

XP es una de las metodologías más exitosas en la actualidad utilizada para proyectos de corto plazo y equipo pequeño. Dentro de las 12 reglas que puntualiza para la creación de un producto no tiene en cuenta la Gestión de Configuración, pero a pesar de esto de una forma u otra desarrolla actividades que están muy relacionadas con este proceso, pues la idea de XP es mantener versiones simples y manejables del sistema, desde un principio se abarcan las funcionalidades básicas requeridas por el cliente que constituyan un conjunto mínimo de tal forma que la aplicación funcione y sobre la cual se puedan introducir posteriormente cambios solicitados, obteniendo pequeñas versiones del producto y llevando a cabo un control de versiones; además todos los cambios que se presenten deberán ser integrados siempre al sistema continuamente, por lo menos, una vez al día pues aunque con esta metodología no se realicen todas las tareas de la GCS uno de sus principales objetivos es atender las necesidades del usuario con mayor exactitud llevando el control de todos los cambios producidos durante el ciclo de vida del software.

En aras de obtener un método que maximizara el éxito de las empresas en cuanto a organización y calidad en los productos fabricados, surge como resultado de las experiencias alcanzadas en diversas áreas de Microsoft con proyectos exitosos, la metodología: ***MICROSOFT SOLUTIONS FRAMEWORK (MSF)*** que se compone de varios modelos que se encargan de cada una de las fases del desarrollo de un proyecto: modelo de arquitectura del proyecto, modelo de equipo, modelo de procesos, modelo de gestión de riesgo, modelo de diseño de procesos y modelo de aplicación. Es una metodología conducida por metas, las metas son puntos en el proyecto que son deseables que se terminen y que pueden ser revisados, está diseñada para acomodar cambios de requerimientos en los proyectos utilizando mejoras incrementales.

En MSF se efectúan una serie de actividades que son comunes con el proceso de GCS tales como el control de los cambios y de las versiones y a pesar que no se llevan a cabo de la misma forma, resultan indispensables cuando se desea trabajar en equipo de forma organizada para obtener un producto final que cumpla con los requerimientos planteados por el usuario. En esta metodología

los cambios son considerados como riesgos inherentes y para llevar un mejor control de los mismos, a través de la Administración de Riesgos, una de las disciplinas que aborda MSF, primeramente son identificados y analizadas las probabilidades de ocurrencia de cada uno de los riesgos y luego que se hacen evidentes son registrados y reportados al resto del equipo con el fin de darle seguimiento y un adecuado manejo de los mismos. La codificación, la documentación, los diseños y los planes en MSF están elaborados en base a un proceso iterativo. Donde cada iteración es una nueva versión. Las versiones están en función de las funcionalidades del software y del tiempo de desarrollo. Las versiones se controlan mediante Team Foundation Server (TFS): un conjunto de tecnologías que constituyen una plataforma de colaboración que incluye Visual Studio Team System (VSTS) para administrar y dar seguimiento al avance y al estado del trabajo de un proyecto en base a una serie de servicios Web y repositorios integrados. TFS posee dos plantillas de proceso por defecto, *MSF for Agile Software Development* para estrategias más ligeras e iterativas y *MSF para CMMI Process Improvement* para aproximaciones más estructuradas y rigurosas, estas plantillas definen tanto la configuración como el contenido inicial del proyecto de equipo a través de seis complementos: Seguimiento de elementos de trabajo, Clasificación, Windows SharePoint Services, Control de versiones, Informes y Grupos y permisos. El componente control de versiones sustentado por SQL Server, ha sido construido desde cero para ofrecer fiabilidad, seguridad y alta velocidad de acceso a datos versionados, tiene mecanismos de control para el check-in, check-out y la gestión de versiones. El archivo XML del control de versiones se denomina VersionControl.xml y está ubicado en la carpeta Control de versiones de la jerarquía de carpetas de la plantilla de procesos.

***Rational Unified Process* (*RUP*)** constituye un proceso de ingeniería de software que al igual que las metodologías anteriormente analizadas tiene como propósito asegurar una producción de software de alta calidad que se ajuste a las exigencias del usuario con costos y calendarios predecibles. En conjunto con el lenguaje de UML, RUP es la metodología más utilizada para el análisis, desarrollo y documentación de sistemas orientados a objetos. En RUP se desarrolla un ciclo de vida iterativo dividido en 4 fases : Inicio, Elaboración, Construcción y Transición y se definen 9 flujos de trabajos, 6 conocidos como flujos de ingeniería y 3 como flujos de apoyo, dentro de los cuales, a diferencia de XP y de MSF, se encuentra la Configuración y gestión de cambios.

Este flujo con la finalidad de mantener la integridad de los artefactos que surgen durante la elaboración del producto final y de mantener información de la evolución de los mismos cubre las

actividades comprendidas dentro del proceso de GCS: Identificar, Controlar, Auditar e Informar explicadas anteriormente.

1.6 Aplicación de la GCS en proyectos de la universidad

El mercado mundial de software exige, a cada instante, productos de mayor calidad, y la única forma verdaderamente efectiva de mejorar los productos es mejorando el proceso de desarrollo de estos. La Universidad de las Ciencias Informáticas comienza a enfrentarse al mundo competitivo del software, esto requiere de una gran coordinación y esfuerzo por parte de todos los trabajadores y estudiantes que se encuentran trabajando en proyectos productivos. Mejorar paulatinamente la calidad de los productos de software se ha convertido en la meta inmediata de la institución, por supuesto, esto sólo se logra con gran sacrificio y organización en todo momento, desde que se decide quién va a cubrir cada rol y las tareas que realizará hasta la forma en que se van a definir los elementos que se desean controlar y cómo hacerlo. La Gestión de Configuración es un flujo imprescindible del proceso desarrollo de software y de la cual aún existe mucho desconocimiento y en no pocas ocasiones malas prácticas. Para el estudio del estado del arte de la misma, fue necesario hacer una investigación sobre el tema en proyectos que se están desarrollando actualmente en la universidad, algunos de estos con gran magnitud y resultados significativos dentro de la producción de software del país. Según encuestas realizadas a integrantes de estos proyectos (ver **ANEXO 1**) donde se tuvieron en cuenta aspectos como:

- Conocimiento general sobre Gestión de Configuración.
- Forma en que se desarrolla la Gestión de Configuración del proyecto.
    - Actividades que se desarrollan.
    - Herramientas utilizadas para el control de versiones.

Se comprobó que de un total de 60 encuestados, el 97% desconoce el término “Gestión de Configuración”, sin embargo, un 99.9% tiende a desarrollar 2 actividades fundamentalmente dentro de este proceso: Control de versiones y Control de los cambios en la configuración, en el desarrollo de la primera existe un punto en común entre los proyectos investigados y es que todos utilizan la herramienta Subversion aunque un 2% en conjunto con el Trac; en cuanto a la segunda solo un 1% la

realiza de forma adecuada, pues en la mayoría de los casos no existe un comité de cambio y el líder del proyecto es el único responsable de aceptar o no alguna petición de cambio. La cuestión es que sólo un 1% de los proyectos que se encuestaron desarrollaban de forma completa la Gestión de la Configuración y todas las actividades que son imprescindibles para realizarla de manera correcta.

Lo antes expuesto da idea del nivel de desconocimiento que hay en los proyectos de la universidad en cuanto a la GCS, lo que trae consigo malas prácticas de la misma pues se tiende a dejar a un lado el resto de las actividades que se deben realizar para que se logre un proceso de GCS eficiente y completo.

CAPÍTULO II

Caracterización y diseño del proceso a aplicar

Introducción

En este capítulo se abordará la evolución del proyecto a partir de la aplicación de la estrategia específica de gestión de configuración, especificando como se llevan a cabo las actividades de este proceso, se ofrece una propuesta de la herramienta de control de versiones a utilizar en el proyecto Simulador Quirúrgico, luego de comparar y analizar de forma detallada alguna de las principales características de cada una de los sistemas de Control de Versiones que pudieran ser utilizadas. Además se propone una herramienta para gestionar la administración de configuración y se explica brevemente de que forma se configuraron las mismas.

2.1 Identificación de la Configuración del Proyecto Simulador Quirúrgico.

Si se habla de Gestión de Configuración de Software entonces como primer paso se tiene la identificación inequívoca de esta configuración. La clave para desarrollar esta actividad está en identificar de un software tanto sus atributos funcionales como físicos de forma técnica, es decir el mismo código del software, datos utilizados, documentación sobre las dependencias con otros componentes de software, manuales, casos de prueba, en fin, todos los datos y documentación sobre los elementos que permiten que el software funcione y pueda ser utilizado.

La identificación de la configuración del proyecto Simulador Quirúrgico (SMQ) se realizó basándose en las cuatro tareas definidas en cuanto a este tema, explicadas anteriormente en el capítulo 1 del documento actual: identificación de los EC, identificación del esquema de almacenamiento de estos EC, las relaciones existentes entre ellos e identificar las diferentes líneas bases del proyecto.

2.1.1 Elementos de Configuración

Para la identificación de los elementos de configuración del proyecto se asignó la responsabilidad de definir los principales ECSs de cada módulo al Líder, el resto de los integrantes del

equipo en dependencia del rol que desempeñen seleccionan los ECSs específicos. Cada uno de estos elementos tendrá un identificador que es regido por una nomenclatura definida por el Administrador de la Configuración.

Todos los ECSs deben ser archivados con el fin de llevar un control de los mismos y de esta forma darles seguimiento, al inicio de la ingeniería del software eran documentos de papel que se almacenaban físicamente por lo tanto se hacia muy difícil conocer que elemento había sufrido un cambio, cuando y por qué, la construcción de nuevas versiones consumía mucho tiempo y era propenso al error; con el objetivo de corregir estos problemas en la actualidad los ECSs se conservan en base de datos o en un depósito del proyecto. El uso de las herramientas que se escojan para complementar la gestión de configuración en el proyecto SMQ permitirá almacenar los elementos de configuración obteniendo un contenido histórico de cada uno de estos archivos.

Los ECSs que hasta la fecha han sido definidos son los siguientes:

Plan de gestión de requisitos.

Requisitos funcionales

Requisitos no funcionales

Modelo de CU del sistema.

Documento de arquitectura de software

Modelo de diseño

Especificación de casos de prueba

Plan de pruebas

Modelo de despliegue

Documento visión

Plan del proyecto

Lista de riesgos

Plan de mitigación de riesgos

Roles y responsabilidades

Ambiente de desarrollo

Plan de capacitación

Plan de aseguramiento de la calidad

Normas y estándares

Registro de no conformidades

Glosario de términos

Plan de mediciones

Estilos de códigos

Código fuente

La relación entre los ECSs se especifica dentro del contenido de estos, escribiendo una sección al inicio de cada uno en la que se enumeran los archivos con los que el ECS se relaciona o mantiene referencias.

2.1.2 Líneas bases

La configuración del software en un punto dado en el tiempo es conocida como línea base. Una línea base (LB) puede ser creada por un EC o un conjunto de estos, una vez que son revisados y aprobados formalmente. La siguiente figura muestra una idea lógica de lo que constituye una LB.

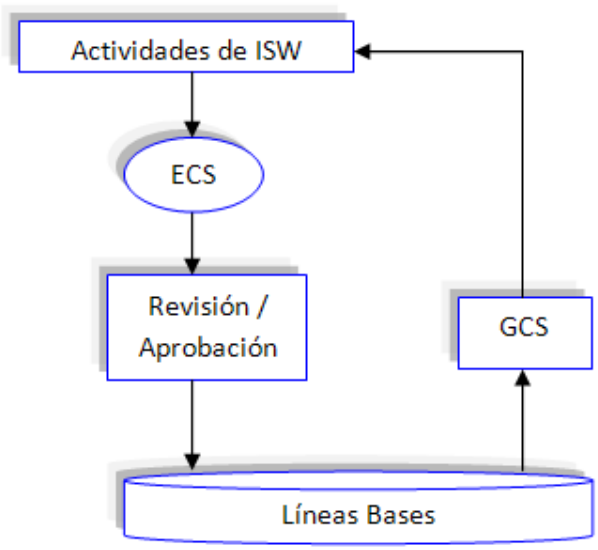

Fig. 2 Representación de líneas bases

El desarrollo del software procede de una línea base a otra, acumulando nuevos elementos de configuración revisados y aprobados al proceso. En el proyecto SMQ las líneas bases serán creadas por el líder del proyecto con el propósito de establecer un estándar de trabajo para las verificaciones, procesos, reportes y demás actividades dentro del desarrollo del proyecto en la fase en que éstas se creen.

Cada una de estas líneas base con la utilización de las herramientas que se seleccionen se almacenarán y se conservará el historial de todas las tareas realizadas sobre estas con el fin de solucionarlas.

Durante el ciclo de vida del software pueden aparecer nuevas líneas bases, la creación de una nueva línea base es ordenada por el líder o por el organismo superior y será creada una vez que el Comité de Control de Cambios revise y apruebe los aspectos que la conformarán.

Antes de que un elemento de configuración se convierta en LB, es posible realizar cambios rápidos e informales sobre el mismo. Sin embargo una vez establecida si surge la necesidad de modificarla se requerirá una revisión formal y una justificación de todas las modificaciones que se

desean realizar; para esto se ha creado un procedimiento para cambiar las líneas bases, en epígrafes posteriores se explicarán los pasos para llevarlo a cabo.

2.2 Control de Versiones

Cuando se trabaja en grupo, surge la necesidad de coordinar el trabajo, no sólo por una cuestión natural sino también como mecanismo de optimizar los recursos. Por eso, es imprescindible la buena comunicación y entendimiento entre los integrantes. Sin embargo aunque las relaciones entre el personal involucrado en un proyecto sean las mejores, siempre surgen ciertas incomodidades técnicas pues cuando el proyecto se divide en varios subgrupos cada integrante tiene una forma y un flujo particular de trabajar, lo que trae consigo que haya más de una persona modificando el código fuente de forma simultánea, y entonces resulta mucho más complejo el hecho de sincronizarlo y mantenerlo coherente entre todos los miembros del grupo. Entonces surge una nueva necesidad, que va a ser la de la integración de múltiples trabajos individuales, la distribución del mismo en distintas máquinas y la coordinación para que todos puedan trabajar sobre la misma base de código. El control de versiones es simplemente la administración y mantenimiento de todas estas versiones en un archivo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas, por tal razones para controlar las versiones dentro de un proyecto se hace más factible la utilización de estos sistemas.

La principal clasificación que se puede establecer entre los sistemas de control de versiones está basada en el almacenamiento del código:

- *Centralizados*: Existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). Se facilitan las tareas administrativas a cambio de reducir la potencia y flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable.
- *Distribuidos*: No existe un punto central de desarrollo sino que los repositorios están distribuidos y descentralizados en diversas máquinas que pueden o no ser independientes entre sí, y técnicamente no hay ninguno mas importante que otro. Es decir el modo de trabajo suele ser que cada desarrollador tenga su repositorio propio sobre el cual trabaje de forma independiente, y

periódicamente se pongan en común los trabajos de todos en algún repositorio convenido a tal efecto. Esto da más flexibilidad pero puede dificultar bastante la sincronización.

Todos los sistemas de control de versiones se basan en disponer de un repositorio que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados.

Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial. Este proceso se suele conocer como *check out* o *desproteger*. Para modificar la copia local existen dos semánticas básicas:

- *Exclusivos*: Para poder realizar un cambio es necesario marcar en el repositorio el elemento que se desea modificar y el sistema se encargará de impedir que otro usuario pueda modificar dicho elemento.
- *Colaborativos*: En el que cada usuario descarga la copia la modifica y el sistema automáticamente mezcla las diversas modificaciones. El principal problema es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas. Además, esta semántica no es apropiada para ficheros binarios.

Tras realizar la modificación es necesario actualizar el repositorio con los cambios realizados. Habitualmente este proceso se denomina *commit*, *check in* o *proteger*.

2.2.1 ¿Por qué herramientas de software libre?

El uso de la informática en Cuba ha estado soportado en sistemas Windows; en el contexto actual LINUX, ha ganado espacios como sistema libre y de código abierto lo cual le confiere numerosas ventajas principalmente para aquellos países que como Cuba se encuentran en vías de desarrollo, donde se hace necesario establecer una política que deje a un lado el consumismo y vaya más encaminada a estimular la creación de las nuevas tecnologías, poniéndolas al servicio de la Sociedad. Cuba se prepara para realizar la migración hacia este mundo del software libre cuya filosofía se puede enunciar a partir de las 4 libertades siguientes:

1. Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito y para siempre.

2. Libertad para estudiarlo y adaptarlo a las necesidades de cada persona. Esto exige el acceso al código fuente.

3. Libertad de redistribución, de modo que se permita colaborar con vecinos y amigos.

4. Libertad para mejorar el programa y publicar las mejoras. También exige el código fuente.

Estas posibilidades adquieren un relevante significado para el desarrollo Científico-Tecnológico en el campo de la Informática para Cuba, facilitando la colaboración con otros gobiernos y organizaciones sin impedimentos de monopolios norteamericanos para estos propósitos.

La Universidad de las Ciencias Informáticas (UCI), en conjunto con el resto de las universidades cubanas refleja el mismo estado del país en este tema y comienza a tomar como política la utilización de software libre. Por estas razones a continuación se hace un estudio de un conjunto de sistemas de control de versiones para el proyecto Simulador Quirúrgico teniendo como requisito fundamental que sean de software libre.

2.2.2 Estudio de herramientas para el control de versiones

Monotone

Monotone es un sistema de control de versiones libre distribuido. Ofrece un almacén simple de versión transaccional de un solo fichero, con una operación completa de desconexión y un protocolo de sincronización eficiente llamado netsync. Comprende el mezclado susceptible al historial, ramas ligeras, revisión de código integrado y pruebas de terceros. Al igual que Git y Mercurial, Monótona utiliza SHA-1 para identificar archivos específicos o grupos de archivos en lugar de los números de versión, los clientes usan llaves RSA para autenticarse. Es una herramienta fácil de aprender debido a un conjunto de comandos similares a los de CVS. Posee un buen soporte de internacionalización, se ejecuta en Linux, Solaris, OSX y Windows y está licenciado bajo la GNU GPL.

En la actualidad los posibles inconvenientes de Monotone incluyen:

- Menos popular que algunos sistemas competitivos como Git, Mercurial o GNU arch.
- Los nombres de archivos, certificados y claves tienen que ser ASCII.
- No puede eliminar directorios del repositorio, hay que borrar los archivos individualmente.
- No tiene cliente gráfico.
- No tiene cliente Web.
- El soporte para “cherry picking” no es el mejor.

GNU Arch

GNU Arch es una herramienta libre para el control de versiones que fue creada y escrita por Tom Lord y actualmente mantenida por Andy Tai. GNU Arch tiene algunas características que lo hacen especialmente útil para el público en proyectos de software libre: es fácil de aprender; es barata y fácil de administrar, es un sistema distribuido por lo que no hay necesidad de dar permisos de escritura a cada participante en el proyecto aunque también soporta desarrollo centralizado. Los desarrolladores suben sus cambios a un "archivo", que puede ser local, y los cambios pueden ser metidos y quitados de otros archivos tal y como los administradores de esos archivos vean conveniente. Como toda metodología implica, Arch posee más soporte de mezclado que CVS y permite crear fácilmente ramas de archivos a las cuales uno no tiene acceso para subir cambios.

Existen una serie de componentes adicionales para el sistema GNU Arch:

- GUI Front-ends : existen varias versiones en función de la plataforma en la que operan: Octopy, xtla (emacs), tlator (GTK2), ArchWay, ArchLog
- Browsers de revisión: Navegadores para archivos (viewARCH, ArchZoom) o de bibliotecas (perspective, PAB)
- Módulos adicionales: existen módulos python para migración de repositorios CVS, representación gráfica, simplificación de interfaces, etc.

GNU Arch a pesar de ser una innovadora solución para mantener una gestión de versiones distribuida, mediante relaciones 1 a 1 hasta el momento no se ha encontrado nada que facilite su integración en una web, ni ningún plugin que permita integrar esta herramienta en el IDE Eclipse, no posee suficiente documentación disponible e incluso algunas no están vigentes, también es poco

extendida pues no aporta capacidades adicionales frente a otros gestores de versiones como cvs y svn y carece de una versión para trabajo en plataformas Microsoft, lo cual cierra su empleo en entornos de desarrollo.

Darcs

Darcs es una herramienta de control de versiones distribuida y libre. Este sistema fue originalmente desarrollado por David Roundy. Una arquitectura descentralizada como la de Darcs tiene importantes ventajas: Las distintas ramas de desarrollo son más fáciles de administrar, y no es necesario crear cuentas de usuario en distintas máquinas para dar acceso a los desarrolladores. Si se desea trabajar solo, la administración es mínima, por lo que es muy bueno para quienes nunca han usado control de versiones. Algunas características interesantes de Darcs son: Interfaz bastante sencilla e interactiva, aspecto de gran relevancia pues muy pocos sistemas de control de versiones tienen esta característica. Por ejemplo, el commit generalmente en los sistemas de control de versiones, lo único que tiene de interactivo es que pregunta por la descripción del cambio, pues bien, en Darcs el commit (que se llama «record») va preguntando por todos los cambios que se han hecho para que se vayan validando uno a uno y sólo manda los que acepten; no tiene servidor, con sólo el cliente se puede trabajar, no se hace necesario configurar nada. Funciona en Linux y Windows y tiene bastantes herramientas disponibles. Entre ellas está una integración de Darcs en Eclipse, una interfaz web y un experimento para añadir Darcs al Trac. Su manual contiene un breve tutorial y una sólida referencia.

Subversion

Subversion es un software para el control de versiones que actualmente se ha popularizado bastante dentro de la comunidad de desarrolladores de software libre. Es un sistema que se distribuye bajo la licencia de tipo Apache/BSD y surge con la intención de sustituir y mejorar al conocido Concurrent Versions System, mantiene las ideas fundamentales del CVS con el mismo modelo de desarrollo pero suple sus carencias y evita sus errores.

Al discutir acerca de las características que Subversion aporta al mundo del control de versiones, resulta más fácil comprenderlas explicándolas en términos de cómo han mejorado sobre el diseño de CVS:

- Versionado de directorios: CVS solamente lleva el historial de ficheros individuales, pero Subversion implementa un sistema de ficheros versionado "virtual" que sigue los cambios sobre árboles de directorios completos a través del tiempo. Ambos, ficheros y directorios, se encuentran bajo el control de versiones.
- Verdadero historial de versiones: Dado que CVS está limitado al versionado de ficheros, operaciones como copiar y renombrar, con Subversion, se puede añadir, borrar, copiar, y renombrar ficheros y directorios. Y cada fichero nuevo añadido comienza con un historial nuevo, limpio y completamente suyo.
- Envíos atómicos: Para una colección cualquiera de modificaciones o bien entra por completo al repositorio, o bien no lo hace en absoluto. Esto permite a los desarrolladores construir y enviar los cambios como fragmentos lógicos e impide que ocurran problemas cuando sólo una parte de los cambios enviados lo hace con éxito aportando seguridad a la integridad de los datos.
- Elección de las capas de red: Subversion tiene una noción abstracta del acceso al repositorio, facilitando a las personas implementar nuevos mecanismos de red. Subversion puede conectarse al servidor HTTP Apache como un módulo de extensión. Esto proporciona a Subversion una gran ventaja en estabilidad e interoperabilidad, y acceso instantáneo a las características existentes que ofrece este servidor: autenticación, autorización, compresión de la conexión, etcétera. También tiene disponible un servidor de Subversion independiente, y más ligero. Este servidor habla un protocolo propio, el cual puede ser encaminado fácilmente a través de un túnel SSH.
- Manipulación consistente de datos: Subversion expresa las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de texto (legibles para humanos) y ficheros binarios (ilegibles para humanos). Ambos tipos de ficheros son almacenados igualmente comprimidos en el repositorio, y las diferencias son transmitidas en ambas direcciones a través de la red.
- Ramificación y etiquetado eficientes: El coste de ramificación y etiquetado no necesita ser proporcional al tamaño del proyecto. Subversion crea ramas y etiquetas simplemente copiando el proyecto, usando un mecanismo similar al enlace duro ("acceso directo"). De este modo estas operaciones toman solamente una cantidad de tiempo pequeña y constante.
- Totalmente reutilizable: Subversion no tiene un equipaje histórico; está implementado como una colección de bibliotecas compartidas en C. Esto hace a Subversion extremadamente fácil de mantener y reutilizable por otras aplicaciones y lenguajes.

Subversion también trabaja con una arquitectura cliente/servidor y tanto los clientes como servidores pueden trabajar sobre UNIX, Windows, Mac OSX y Linux. En cuanto a la documentación disponible para el conocimiento de esta herramienta existe un libro en línea gratis y algunos tutoriales y recursos. El libro está escrito en DocBook / XML y así es convertible a muchos formatos diferentes. El command-line-client también es un buen sistema de ayuda en línea que puede ser utilizado como referencia.

Subversion posee deficiencias que constituyen detalles técnicos que -en principio- no afectan el seguimiento de cambios tales como:

- El cambio en el nombre de un archivo es interpretado como dos operaciones: copia y borrado de la primera copia. Esto ocasiona que los cambios en el código sean perdidos de manera inadvertida tras múltiples operaciones de combinación de copias.
- No implementa algunas operaciones administrativas, como la eliminación de registros de algunos datos almacenados en el repositorio, la cual puede hacerse de forma manual y engorrosa.
- Requiere que cada carpeta en el lado del cliente incluya una carpeta oculta ".svn". Estas carpetas impiden la generación de copias distribuidas de una versión individual para esto se requiere realizar una operación adicional de exportación y son las causantes de la mayoría de problemas que enfrentan los usuarios de esta herramienta.

Git

Git es un popular sistema de control de versiones distribuido, diseñado para manejar grandes proyectos con rapidez y eficiencia. Fue creado por Linus Torvalds para manejar el código fuente del núcleo de Linux e inspirado en BitKeeper y Monotone. Es actualmente mantenido por C Junio Hamano y posee una gran comunidad de usuarios y desarrolladores. Es muy versátil, con varias funcionalidades novedosas, y entre los numerosos proyectos abiertos que lo usan se incluyen el kernel de Linux, Xorg, y Wine. Linus describe a GIT como una infraestructura fundamental (“plomería”), a la que se conectan artefactos más estéticos que ofrecen una interfaz amigable. Sobre los comandos básicos de GIT se construyen sistemas más estéticos, sencillos de usar, de variadas funcionalidades (Porcelana). En Git se trabaja con ramas, estas son distintos caminos del desarrollo; así que se crean

muchas ramas a lo largo de la vida del proyecto. Una rama no es más que una referencia al commit. Entre las características más relevantes se encuentran:

- Fuerte incidencia en la no linealidad de los cambios, por ende rapidez en la gestión de ramificaciones y mezclado de diferentes versiones.
- Gestión distribuida. Los cambios se importan como ramificaciones, y pueden ser mezcladas en la manera en que lo hace una ramificación del almacenamiento en local.
- Los almacenes de información pueden publicarse por HTTP, FTP, SSH o mediante un protocolo nativo, aparte de ser posible emular CVS.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica de historial). Esto existía en Monotone.
- Resulta algo más caro trabajar con ficheros concretos frente a proyectos, eso diferencia el trabajo frente a CVS, que trabaja en base a cambios de fichero, pero mejora el trabajo con afectaciones de código que concurren en operaciones similares en varios archivos.
- Los renombrados se trabajan en base a similitudes entre ficheros, aparte de nombres de ficheros, pero no se hacen marcas explícitas de cambios de nombre en base a supuestos nombres únicos de nodos de sistema de ficheros, lo que evita posibles coincidencias de ficheros diferentes en un único nombre.
- Re-almacenamiento periódico en paquetes (ficheros). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el re-empaquetado (en base a diferencias) no ocurre cada cierto tiempo.

Hasta ahora era posible ejecutarlo bajo cygwin (entorno de Linux para Windows) pero esto suponía una perdida de rendimiento y de comodidad. Actualmente queda trabajo por hacer para que

Git sea totalmente funcional en Windows. También se está desarrollando un interfaz al estilo TortoiseCVS llamado Git-cheetah para hacer más cómodo su uso.

2.2.3 Comparación entre las herramientas de Control de Versiones

Los sistemas se compararon según 8 criterios, estos se escogieron teniendo en cuenta las necesidades del proyecto:

- *Mecanismo de almacenaje:* Para permitir guardar de forma segura todos los elementos que se deban gestionar como archivos de texto, imágenes, documentación, entre otros.
- *Posibilidad de realizar cambios:* Para conseguir realizar cambios sobre los elementos almacenados como modificaciones parciales, añadir, borrar, renombrar o mover elementos.
- *Registro histórico de las acciones realizadas*: Normalmente se llevan a cabo acciones donde se involucra cada elemento o conjunto de elementos, el registro histórico de las acciones va a permitir volver a cada una de estas o extraer un estado anterior del producto.
- *Sistema centralizado*: Tiende a ser muy efectivo para proyectos de pocos integrantes y poca experiencia, aunque si aumentara el personal no traería problemas. Es mucho más cómodo para el proyecto ir actualizando en un repositorio único toda la información, esto facilitaría la organización a la hora de hacer algún cambio pues se necesitaría la aprobación del responsable.
- *Disponibilidad de documentación*: El equipo de desarrollo del proyecto es joven y con poca experiencia en el tema, lo que trae consigo una gran desinformación, por lo que se hace imprescindible que la herramienta a usar tenga una amplia documentación, fácil de encontrar y de comprender.
- *Buen soporte*: El proyecto no puede usar un sistema que no tenga buen respaldo técnico, la herramienta debe tener un equipo dedicado a su mantenimiento, actualización constante y continuidad, para evitar que en algún momento surjan contradicciones.
- *Multiplataforma:* El proyecto se encuentra trabajando en Windows pero en un futuro se piensa migrar a software libre, ya que esta es la política que se está implementando actualmente en la universidad, resulta de gran relevancia que el sistema escogido sea multiplataforma.
- *Integración con el Trac:* En el proyecto se piensa utilizar el Trac como herramienta para completar la realización de las actividades de la GCS debido a la experiencia que existe en

cuanto al uso de la misma, tanto en la universidad como en la facultad, por lo tanto se precisa escoger un sistema de control de versiones que se integre al Trac.

Sobre la tabla:

0: No se realiza.

1: Se realiza pero no de forma eficiente.

2: Se realiza eficiente.

Criterios / SCV	Integración con el Trac	Mecanismo de almacenaje	Posibilidad de realizar los cambios	Registro histórico de acciones	Disponibilidad de documentación	Buen soporte	Multi_ plataforma	Sistema centralizado
Monotone	2	1	1[1]	2	1	2	2	0
GNU Arch	0	1[4]	1[5]	1[6]	1	2	1[3]	1[2]
Darcs	2	2	2	2[7]	2	1	2	0
Subversion	2	2	2	2	2	2	2	2
Git	2	2	2	2	2	2	0	0

Tabla.1 Comparación de herramientas

[1]: No se puede eliminar eficientemente.

[2]: A pesar de ser un sistema distribuido soporta desarrollo centralizado.

[3]: No funciona en Windows.

[4]: Si se utiliza un archivo único durante mucho tiempo, al acumularse eventualmente una gran cantidad de datos se convierte en un inconveniente trabajar con ellos.

[5]: Muchos comandos innecesarios provocando más confusión, por ejemplo con el comando “mv” se mueve tanto la identificación y como el archivo, mientras que con el comando "mover" sólo se mueve el id.

[6]: Arch tiene buen acceso a las ramas esto puede también acelerar el acceso a versiones específicas, sin embargo, estos escondites a menudo tienen que ser creados a mano cuando por defecto, el instrumento debería crear automáticamente escondites.

[7]: Registro de cambios interactivo, permite, si se configura, navegar por las modificaciones previo a su registro en el repositorio.

Propuesta de Herramienta de control de versiones:

Se han analizado y comparado las principales características y funcionalidades de algunas herramientas para el control de versiones. Se tuvo en cuenta a la hora de tomar la herramienta la que tuviera la puntuación más alta en cada uno de los criterios seleccionados, descartando a las que en alguno de los casos tuviera cero. Luego con el estudio realizado se llegó a la conclusión, que teniendo en cuenta las necesidades del proyecto Simulador Quirúrgico, Subversion es la herramienta más indicada para ser utilizada, no solo porque sus características le permiten llevar ventaja sobre los demás Sistemas de Control de Versiones, sino porque luego del estudio se investigó su repercusión en la universidad y se evidenció que actualmente es de las más utilizadas, aspecto de gran relevancia para el proyecto pues resulta más sencillo tener apoyo de estudiantes y profesores que tengan experiencia y conocimiento en el tema, además de tener acceso a documentación e información especializada dentro de la escuela. Además la facultad 5 lleva algún tiempo trabajando con esta herramienta en no pocos proyectos productivos obteniendo muy buenos resultados, tanto es así que se piensa instalar en un futuro un único servidor de Subversion en el "Polo de Realidad Virtual", para controlar y almacenar allí todos los cambios que se realicen en los proyectos de la facultad que lo conforman.

2.3 Control de Cambios

El control de cambios es la columna vertebral del proceso de la GCS. En términos generales, es un proceso sistemático para evaluar, coordinar y decidir sobre los cambios propuestos, así como también, para monitorear la implantación e incorporación de aquellas modificaciones aprobadas a las líneas base y la documentación asociada. Con esta actividad se asegura que los cambios sean propuestos, justificados, evaluados, coordinados, aprobados o rechazados y por último documentados. Un cambio puede apuntar tanto a una modificación como a una corrección que se realice sobre un EC o un conjunto de estos una vez que son revisados y aprobados y pasan a formar una línea base, estos pueden ser:

- Cambios en los requerimientos.
- Cambios en el diseño.
- Cambios en la arquitectura.
- Cambios en las herramientas de desarrollo.
- Cambios en la documentación del proyecto. (agregar nuevos documentos o modificar la estructura de los existentes)

De acuerdo a esto es necesario contar con un proceso que garantice que sólo los cambios aprobados sean implementados en una línea base. Para una mejor organización y seguridad en el proyecto se deben crear dos procedimientos: el primero para realizar la solicitud de cambio sobre un elemento de configuración dentro de una línea base y el segundo para cambiar o crear una nueva línea base.

2.3.1 Procedimiento para la solicitud de cambio

Una solicitud de cambio de forma general se divide en cuatro fases: proponer el cambio, evaluar el cambio propuesto, aprobarlo/rechazarlo, e implementar el cambio en caso de ser aprobado. Dentro de cada una de estas etapas se realizan un conjunto de actividades que al darles cumplimientos confeccionan un procedimiento que deberá ser conocido y cumplido por el equipo de desarrollo.

La solicitud de un cambio comienza cuando es identificado un problema en un elemento de configuración y por lo tanto se hace necesario informarlo, objetivo principal dentro de la etapa de proponer el cambio. Inicialmente el problema es analizado por uno o varios desarrolladores informalmente con el fin de implantar las causas que pueda traer consigo la posible corrección, luego la persona que detectó el problema debe dirigir la petición del cambio al encargado dentro del proceso de GSC de esta función, dicha petición de cambio debe satisfacer como mínimo las siguientes preguntas:

- ¿La solución es clara, es decir, la solución propuesta puede ser implementada por alguien ajeno al sistema?
- ¿La solución es consistente, no introduce conflictos con otros ECSs?
- ¿La solución asegura resolver los problemas detectados?
- ¿La solución es completa?
- ¿Se identifican la calendarización y costos requerida para establecer las acciones correctivas?

Entrando a la etapa de evaluación del cambio propuesto, este debe ser analizado y estimado por el CCC en términos de su impacto sobre los requerimientos, la funcionalidad, interfaz, utilidad, costos y planificación del sistema, teniendo en cuenta la confiabilidad, y eficiencia del software. Como resultado de este análisis se decide si el cambio es aprobado o rechazado, si es denegado la petición se devuelve a su origen junto con las razones expuestas por el CCC y en caso de ser autorizada pues se pasa a implementar el cambio, autorizando el acceso a los ECSs solicitados. Al finalizar, los cambios realizados son probados y comunicados al equipo de desarrollo.

En el proyecto SMQ se hará uso de un procedimiento que cumpla con las características generales anteriormente explicadas para efectuar una solicitud de cambio.

2.4 Auditorías y Reportes a la configuración

La identificación y el control de los cambios ayudan a incorporar orden al desarrollo y la mantención del software. Más el éxito de la Gestión de Configuración de Software depende de asegurarse que los cambios hayan sido implementados correctamente. Para ello son de vital

importancia las auditorías, pues establecen que el producto haya sido construido de acuerdo a los requerimientos y que el software está realmente representado por la documentación que le acompaña.

En el contexto del proyecto en cuestión las auditorías que se ejecutarán serán cuidadosamente planeadas y llevadas a cabo cumpliendo en forma estricta los pasos que contienen las fases del proceso de la auditoría: Planeación, Ejecución e Informe. Se establecerán auditorías físicas que se efectuarán con el objetivo de determinar si las especificaciones de diseño, producto y otros documentos referenciados representan el software que fue codificado y probado para uno o varios ECSs específicos, teniendo en cuenta las siguientes actividades:

1. Definición de Requerimientos: Determinar una lista de requerimientos funcionales, no funcionales, interfaz, desempeño, etc. que serán revisados durante el proceso de auditoría física.
2. Definición de Módulos: Determinar una lista de módulos o componentes que deben ser auditados. Los módulos se obtienen a través de la línea base entregada por el equipo de desarrollo y permite definir que módulos están relacionados durante el proceso de desarrollo del proyecto.
3. Elementos de configuración bajo auditoria: Se elegirán uno o más elementos de configuración de mayor prioridad.
4. Detectar y registrar las desviaciones detectadas.
5. Analizar las desviaciones con el objetivo de determinar las principales causas y factores que influyeron en obtener dichas desviaciones.
6. Como resultado del proceso de auditoría física se obtiene el Informe de Auditoría Física y el Registro de no Conformidades.

Además de estas auditorías físicas también se realizará una auditoría interna mensual con el fin de verificar si se cumple con lo reglamentado en el proyecto.

Para realizar un proceso de auditoría es importante que quienes participen en este proceso, tengan conocimiento relativo al desarrollo del proyecto, al contexto en el que este se realiza, a las actividades de aseguramiento de calidad, GCS y gestión de proyectos principalmente. Además deben entender y comprender el proceso de auditoría y los componentes a ser auditados.

Una vez que se hayan realizado las auditorías o que simplemente se efectúe cualquier actividad que afecte el estado actual de la Gestión de Configuración del proyecto, se procederá a

generar los informes de estados de la configuración conocidos como reportes, que respondiendo a un conjunto de preguntas, explicadas dentro del capítulo 1 en el epígrafe 1.4.2.5, brindarán información relacionada con cada cambio efectuado.

2.5 Trac, herramienta para administrar la configuración del proyecto

La gestión de configuración es un proceso que como se ha explicado está formado por un conjunto de actividades: identificación de la configuración, control de cambios, control de versiones, auditorías y generación de informes de estado; para el proyecto Simulador Quirúrgico según el estudio realizado se definió el Subversion para llevar a cabo el control de las versiones, sin embargo para gestionar una eficiente configuración de un proyecto es preciso no sólo controlar las versiones, por tal motivo se hace necesario encontrar una herramienta que integre a Subversion facilitando la realización de la mayor parte de las actividades de esta disciplina. En la universidad muchos proyectos utilizan el Trac como herramienta no sólo para la gestión de proyectos sino también como una solución para garantizar la administración de la configuración de software, actualmente existe una tendencia a incrementar el número de proyectos que la utilizan por ser un sistema web multiplataforma y libre que permite el manejo y gestión de hitos para el desarrollo de proyectos de software con una aproximación minimalista para el manejo de los mismos, ayudando a los desarrolladores a escribir un buen software manteniendo la gestión de los cambios controlada. Brinda una línea de tiempo que muestra todos los eventos del proyecto en orden, facilitando en forma simple la visualización, el control y monitoreo del mismo. Trac posee una interfaz con un repositorio Subversion mediante la cual se puede examinar el estado actual del código fuente que está bajo este, así como los cambios que se han ido produciendo.

Integra una Wiki mejorada que se puede emplear para documentar cualquier aspecto del proyecto de modo colaborativo, con herramientas y opciones para la creación de links y referencias independientes entre tareas, conjuntos de cambios, archivos y paginas wiki.

Aparte del conjunto de funcionalidades que incluye es un sistema que cuenta con una arquitectura que se puede mejorar pues está desarrollada en torno a la idea de un núcleo al que se le pueden añadir *plugins* que proporcionan nuevas funcionalidades. La gran mayoría de los componentes de Trac constituyen módulos que pueden ser activados, desactivados, modificados o reemplazados por otros.

Por tal razones en el proyecto Simulador Quirúrgico se utilizará esta herramienta en conjunto con Subversion pues además de la experiencia que existe sobre el trabajo con la misma, al ser una herramienta web es de fácil acceso por todos los miembros del proyecto sin tener que instalarla, solo es necesario tener el explorador y conexión al servidor donde se encuentra, además tiene una interfaz sencilla que facilita el trabajo a todos los miembros del proyecto. Dispone de una amplia documentación y cuenta con una gran comunidad que desarrolla actualizaciones.

2.6 Configuración del Trac y Subversion para el proyecto Simulador Quirúrgico

En el proyecto Simulador Quirúrgico se instala un repositorio Trac que posteriormente se integraría al Subversion, la gestión de las peticiones se realiza a través del Apache, lo cual puede dotarlo de una serie de funcionalidades añadidas, estas funcionalidades son proporcionadas por los distintos módulos del Apache existentes.

Para crear un proyecto en el Trac se utiliza el comando Trac-Admin desde la línea de órdenes, generando el directorio trac-ini con la configuración del proyecto. Primeramente se hizo necesario crear el repositorio svn. Una vez instalado se estructura la información del repositorio que se organiza en los siguientes directorios:

- Código Repositorio: Código del proyecto organizado por módulos.
- Gestión de Proyecto: Información del expediente del proyecto relacionada con la gestión de proyecto.
- Ingeniería: Información del expediente del proyecto relacionada con la ingeniería de software.
- Soporte: Información del expediente del proyecto relacionada con la calidad del mismo.
- Trunk: Directorio en el que se almacenarán las versiones estables del producto.

Para acceder a la información cada usuario debe estar autenticado, obteniendo permisos específicos en dependencia de la acción que desee realizar. Estos permisos se crearon y luego fueron asignados a la configuración del módulo del Apache encargado de levantar el servidor.

Seguidamente de tener el repositorio Subversion, este se conecta al Trac, especificando su ruta cuando el Trac solicita la dirección del repositorio que utilizará. Una vez creado, el proyecto se

configura como un sitio disponible del Apache. Después de esto la herramienta se adapta en dependencia de las necesidades del equipo de desarrollo:

- Los tickets se modifican teniendo en cuenta las políticas que se trazaron en el proyecto para su uso.
- Se agregan los hitos del proyecto dentro del Roadmap para darles seguimiento.
- En la wiki de la herramienta se expone documentación relacionada con el proyecto y que sirva de ayuda para el trabajo de los integrantes del equipo.

Para la autenticación se hizo uso de los mismos usuarios de Subversion pero para el acceso se crearon tres grupos: el de los usuarios no autenticados, con permiso solamente de lectura, el grupo de los autenticados, con los permisos heredados del grupo anterior además de poder crear y modificar los ticket y la wiki, y por último el grupo de los administradores con acceso a cualquier funcionalidad de la herramienta.

CAPÍTULO III

Resultado Final y evaluación del proceso aplicado

Introducción

En este capítulo se expondrán todos los temas incluidos en el plan de gestión de configuración de software, como resultado de la aplicación del proceso de GCS en el proyecto, además se diseña la confección de un curso de capacitación con el objetivo de adiestrar al equipo de desarrollo sobre las herramientas de configuración utilizadas y el resultado de este luego de su puesta en marcha. Por último se realiza una evaluación de la aceptación del proceso por parte de los integrantes del proyecto.

3.1 Plan de Gestión de Configuración de Software.

3.1.1 Introducción

En este documento se describe la gestión de configuración del proyecto Simulador Quirúrgico.

3.1.2 Propósito

El presente plan contempla las especificaciones a tener en cuenta por parte del equipo de desarrollo para llevar el control y gestión de los cambios del producto en cuestión, además de su organización en función de una adecuada Gestión de Configuración.

3.1.3 Alcance

Se contemplan en este plan las etapas desde febrero/2008 hasta básicamente finales de junio/2008, no se refleja el trabajo realizado anteriormente ya que no cumple objetivo pues esa etapa no fue documentada.

3.1.4 Definiciones, acrónimos y abreviaturas

SMQ: Simulador Quirúrgico,

KHEIPROS: Nombre de la primera versión estable.

ECS: Elemento de configuración de Software.

CCC: Comité de control de cambios.

GCS: Gestión de Configuración de Software.

3.1.5 Referencias

Código	*Nombre*
[1]	Elementos de Configuración de SW.doc
[2]	Plantilla DCS - Pedido de cambio v1.0.doc
[3]	Plan de desarrollo de software.
[4]	Plan de Capacitación.

Tabla 2. Referencia del Plan de GCS

3.1.6 Resumen

El plan describe los procedimientos para llevar a cabo las diversas actividades que abarcan el proceso de Gestión de Configuración de Software, así como las herramientas usadas para desarrollar algunas de estas actividades. Además incluye la forma en que se organizan las tareas y responsabilidades de todos los implicados en este proceso.

3.1.7 Gestión de Configuración de Software

- **Organización de la Gestión de Configuración de Software**

El proyecto cuenta con dos administradores de configuración que son los responsable de mantener el soporte tecnológico y metodológico de la configuración dentro del proyecto y la capacitación de los integrantes del equipo de desarrollo, en cuanto las herramientas utilizadas para el control de versiones

y gestión de cambios. Además de contar con un comité de control de cambio creado en conjunto con el líder del proyecto e integrado por representantes de los roles que definen de forma crucial el desarrollo del producto en cuestión, para así lograr de forma más eficiente el proceso de solicitud de cambio.

- **Responsabilidades**

Rol	Cantidad de personas	Responsabilidades
Administrador de Configuración	2	***1-Escoger las herramientas a usar para el control de los cambios y las versiones.*** •Asociar la arquitectura al repositorio. ***2- Escribir el Plan de Gestión de Configuración (GC).*** •Escribir el Plan de GCS. •Revisar y aprobar el Plan de GCS. •Mantener el Plan de GCS ***3- Establecer las políticas de la GCS.*** •Definir las actividades de identificación de la configuración. •Definir los requisitos de reporte •Definir la frecuencia de los reportes del estado de la configuración. ***4- Desarrollar auditorías de configuración.*** •Definir políticas para las auditorías. •Seleccionar asistentes de configuración para realizar auditorías ***5- Establecer el Proceso de Control de Cambios*** •Establecer el proceso de petición de cambio. •Seleccionar los miembros del Comité de Control de Cambios (CCC).

		•Mantener el Historial de cambios. ***6- Crear líneas base.*** ***7-Organizar y asegurar la capacitación en el uso de las herramientas que serán empleadas en el proyecto para la GCS.***
Cualquier Rol		***1- Actualizar las Peticiones de Cambios.*** ***2- Enviar una Petición de Cambio.*** ***3- Crear áreas de trabajo de desarrollo.*** ***4- Actualizar el estado de las órdenes de trabajo.*** ***5- Hacer cambios.*** ***6- Actualizar el área de trabajo.***
Líder de proyecto	1	***1- Programar y asignar el trabajo.*** •Asignar responsabilidades. •Establecer un cronograma. ***2-Identificar los elementos de configuración principales relacionados con la gestión del proyecto.*** ***3-Identificar los elementos de configuración principales de cada módulo del proyecto.***
Arquitecto Principal	1	***1-Identificar los elementos de configuración principales relacionados con la arquitectura de todo el proyecto.***
Analista Principal	1	***1-Identificar los elementos de configuración principales relacionados con el análisis y el diseño de todo el proyecto.***

Tabla 3. Roles del proyecto

- **Relación de la Gestión de Configuración con el ciclo de vida del proyecto**

- **Interfaces con otras organizaciones dentro del proyecto**

La comunicación se hará por medio de la herramienta TRAC, que muestra una interfaz web, donde se interactúa con cada integrante del equipo de desarrollo por medio de la generación de tickets. En algún caso en específico se tendrá en cuenta la comunicación a través del líder de proyecto.

- **Responsabilidades con otras organizaciones dentro del proyecto**

Las responsabilidades que tiene la gestión de configuración con las otras organizaciones del proyecto son las siguientes:

1. Verificar la creación del Plan de Seguridad en la Calidad del Software, el Plan de Prueba del Software, los reportes del Plan de Prueba del Software y la integración final de los módulos desarrollados (Grupo de prueba).
2. Inspeccionar el cumplimiento de los estándares de documentación, código fuente y el control de nombres y numeración única para nuevas versiones (Grupo de gestión de configuración).
3. Inspeccionar el progreso del proyecto controlando cada uno de los grupos indicados en los puntos anteriores (Administrador de proyecto).

3.1.8 Actividades de Gestión de Configuración de Software

1. Identificación de la configuración

Los elementos de configuración del proyecto son almacenados en el repositorio Subversion organizados en carpetas en dependencia al área que pertenezcan, gracias al sistema de versionado que permite esta herramienta unido a las facilidades que brinda integrada al Trac, se les dará seguimiento a todas las modificaciones realizadas sobre cada ECS.

- **Especificación de la identificación**

Las salvas se identifican con la fecha de realización de las mismas con el siguiente formato (para cualquier elemento de configuración, sean documentos, códigos, etc.):

NombreDelElemento_NumeroVersión_Fecha. Ejemplo: DEMOS_2.3_080215.

- El nombre del elemento debe brindar una breve información de dicho elemento, además se deben omitir los espacios y cada vez que comience una nueva palabra se escribe con mayúscula.
- El término número de versión se usa para definir una fase en la evolución de cada EC. Cada fase es marcada por un número de la versión. Cuando el ECS cambia, debe cambiar el número de su versión.
- El formato de la fecha definido es: aa-mm-dd.

Las versiones del producto se nombran según lo regulado en el documento:

"Nomenclatura de las versiones.odt, carpeta Otros documentos"

o **Identificación para el formulario de control de cambios**

Cada formulario se identifica con la fecha de realización del mismo en el siguiente formato:

Formulario_080215 (formato de fecha: año mes día)

o **Líneas base del proyecto**

Mediante el Trac las líneas bases del proyecto se conservan en la sección Roadmap de esta herramienta; a los integrantes del equipo de desarrollo se les asignan tareas asociadas a cada una de estas LBs con el fin de solucionarlas, el progreso de este cumplimiento es dado por un por ciento y una vez que llegue al 100 %, o sea que ha concluido la LB, entonces es revisada y almacenado su historial. Después de esto para realizar un cambio sobre la línea base se requiere de un procedimiento formal. Las líneas base seleccionadas con sus respectivos ECSs son las siguientes:

- *Línea base de Planificación del proyecto*

Documento Visión

Ambiente de desarrollo

Roles y responsabilidades

Plan de capacitación

Lista de riesgos

Plan de mitigación de riesgos.

Plan de desarrollo de software

- *Línea base de requerimientos del sistema*

Requisitos funcionales

Requisitos no funcionales

Plan de gestión de requisitos

Modelo de CU del sistema

- *Línea base de arquitectura y diseño*

Documento de arquitectura del software

Modelo de diseño

Modelo de despliegue

- *Línea base de implementación*

Códigos

Estilos de código

- *Línea base de pruebas*

Casos de prueba

Plan de pruebas

- *Línea base de aseguramiento de la calidad del producto*

Plan de aseguramiento de la calidad

Glosario de términos

Plan de GCS

- **Bibliotecas**

Se utiliza la STK, que se desarrolla en la propia facultad, cada vez que se obtiene una versión estable de esta, se salva la versión antigua y se adapta el código en caso de ser necesario. Para la detección de colisiones se va a utilizar en esta última versión el OPEN DYNAMICS ENGINE (ODE) un sistema que ya está acoplado a la STK, progresivamente se utilizará, en caso de ser necesario, un módulo que actualmente de esta desarrollando.

2. Control de la configuración

- **Procedimientos para cambiar una línea base**

Se necesitará establecer una nueva línea base del proyecto cada vez que ocurran cambios importantes en los documentos que estarán enmarcados dentro de esta(Plan de Desarrollo de Software, Plan de Aseguramiento de la Calidad, Plan de Gestión de la Configuración, Documento Visión). La necesidad de establecer una nueva línea base también podrá ser definida antes que ocurra la modificación de los documentos que la constituyen, ya sea por indicación del Líder de proyecto o el organismo superior, pero esta quedará creada solo después del establecimiento, revisión y aprobación de los cambios requeridos en la documentación. Las líneas base deberán ser verificadas por el Comité de Control de Cambios y será responsabilidad del mismo autorizarla.

- **Procedimiento para procesar pedidos de cambios y su aceptación**

En el proyecto para solicitar un cambio en un ECS que se encuentre dentro de una línea base se hará uso de la herramienta Trac, para esto se definieron dos tipos de ticket: SC para solicitar un cambio y OT para enviar una orden de trabajo.

Como entrada se tendrá:

Elementos de configuración que deben ser cambiados, misión y metas de la propuesta de cambios

Actividades en el procedimiento:

1. Mediante el Trac por un Ticket se notifica un problema por el que sea necesario la solicitud del cambio, este se le hace llegar al jefe del CCC y muestra:

- Quien la emite: Este dato no se especifica en ningún campo pues como el usuario se encuentra autenticado, automáticamente la herramienta toma el nombre de este usuario.
- Resumen de la propuesta: Consiste en una breve explicación del cambio que se solicita.
- Contenido de la propuesta: Se describe claramente la propuesta de cambio, los ECSs que podrían verse afectados con la misma y una posible solución al cambio.
- Prioridad: En este campo se especifica la prioridad de realizar el cambio.
- Versión: Versión a la que pertenece el cambio.
- Hito: Hito que se afecta.
- Keyword: En este punto se escoge la palabra clave que define el cambio, pues facilita la búsqueda del ticket en caso de ser necesaria.

2. Se almacena el problema con un código consecutivo y se clasifica la solicitud recibida en: error o mejora.
3. Se decide en el CCC si se acepta o no el cambio(Rechazado o Aprobado) teniendo en cuenta lo siguiente:

- Requisitos del sistema.
- Cálculo del impacto del cambio.
- Análisis de la carga laboral de los involucrados con el cambio.

4. Si la solicitud es rechazada se le comunica a la persona que solicita el cambio, los argumentos que condujeron a tomar tal decisión; en caso contrario se genera y se asigna una Orden de Trabajo asociada a la solicitud de cambio. Para asignar la orden de trabajo se envía un nuevo ticket de tipo OT donde se debe planificar el cambio, especificando que acciones serán realizadas para darle solución, además de estimar el tiempo de duración, complejidad y prioridad del mismo.
5. Se prueba el cambio realizado

- Si hay errores en los ECSs cambiados, entonces el comité vuelve a reunirse y define si la solicitud de cambio se aplaza, se rechaza o si se corrigen los errores, sino se pasa a la actividad 6
- Si se decide que se pueden corregir los errores se le envía un ticket con una nueva OT al miembro que solicitó el cambio. Una vez que el mismo concluya su trabajo se pasa nuevamente a la actividad 5.
- Si la solicitud de cambio por algún motivo se decide aplazar se le envía un ticket al que solicitó el cambio notificándole que la petición quedó aplazada.

6. Se notifica que es aceptada la solicitud del cambio al que la solicitó y se distribuye la nueva versión.

- **Comité de Control de Cambios**

El comité de control de cambios está formado por el Líder del proyecto, el Jefe del grupo de desarrollo y tres arquitectos, actualmente estas responsabilidades son de las siguientes personas:

Líder del proyecto: Juan Manuel Mederos Martínez

Jefe del grupo de desarrollo: Raissel Ramírez Orozco

Arquitectos: Yoisy Pérez

Mariela Nogueira

Osley Bretau Camejo

- **Revisión de documentos**

Se generará una copia de los documentos implicados y se procederá a su revisión señalando los posibles errores y marcándolos para la reparación. Las fallas serán corregidas con el líder, el encargado de aseguramiento de la calidad y de gestión de software. Si no existen problemas en el documento revisado, este se declara satisfactorio y se archiva en el expediente de proyecto. Si se realiza algún cambio es consultado por los miembros del comité de control de cambio y si se aprueba se procederá a generar tareas correspondientes para el arreglo de los documentos y su posterior actualización en el repositorio.

- **Herramientas automatizadas para el Control de Cambios**

En el proyecto para controlar los cambios se hace uso de dos herramientas integradas, el Subversion, para controlar las versiones tanto del código fuente como de la documentación que se va creando en el transcurso del proyecto, y el trac para reforzar el trabajo con las versiones, además de que esta última posee un sistema de ticket que posibilita automatizar el procedimiento de solicitud de cambios y controlar la asignación de tareas y todos los sucesos registrados en el proyecto hasta la fecha. El esquema a continuación muestra la organización de estas herramientas.

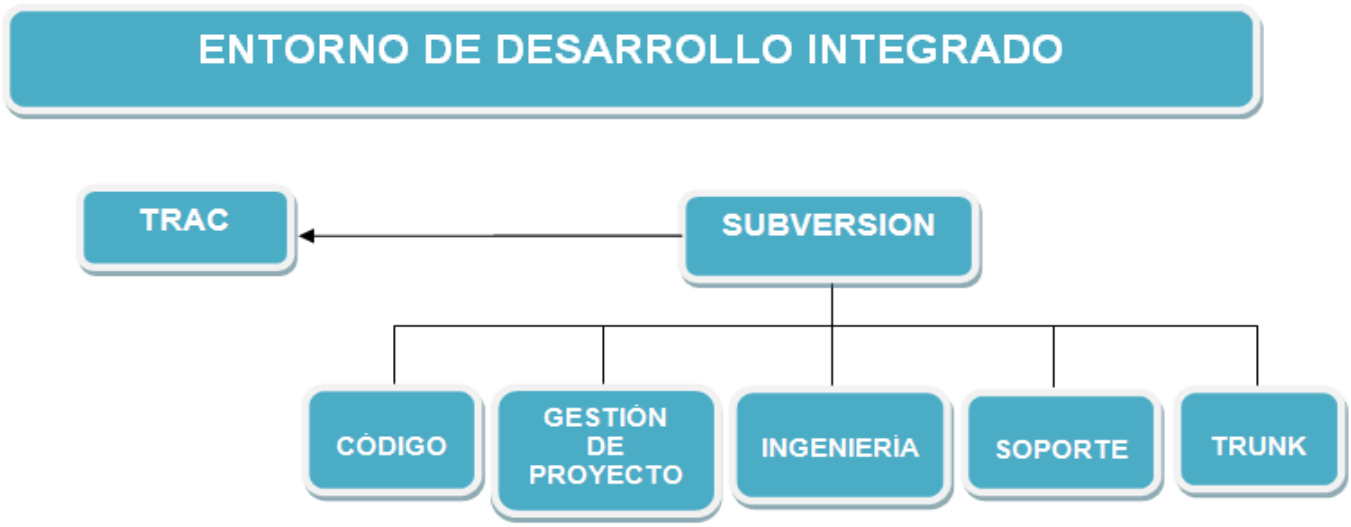

Fig.3 Organización de las herramientas

3. Estado de la configuración

- **Almacenamiento, manipulación y entregables del proyecto**

Se almacenan todos los entregables del proyecto:

- KHEIPROS 1.0.

- Medios necesarios para la utilización del producto (librerías, instaladores, drivers, etc.).

- Documentación y ayuda.

Se cuenta con una salva en el servidor y dos salvas alternativas, una en la PC del líder de proyecto y otra en la del arquitecto.

Entregables:

Entregable	***Fecha de Entrega***
Ejecutable del Simulador de habilidades básicas, a partir de modelos de imitación de procesos. (Esto integra los ejercicios de Coordinación, Selección y Cámara)	Febrero del 2008.
Documentación y ayuda necesaria para trabajar con el Simulador de habilidades básicas.	Febrero del 2008.
Ejecutable del Simulador de habilidades de avanzada a partir de la imitación de procesos, incluyendo endoscopía	Noviembre del 2008.
Documentación y ayuda necesaria para trabajar con el Simulador de habilidades de avanzada.	Noviembre del 2008.

Ejecutable de la primera versión del simulador a partir de la generación de órganos virtuales.	Junio 2008
Documentación y ayuda necesaria para trabajar con la primera versión del simulador a partir de la generación de órganos virtuales.	Junio 2008
Ejecutable del Simulador que incluye trabajo con órganos adquiridos a partir de scanner (TAC, RM) y retroalimentación al tacto.	Julio 2010
Documentación y ayuda necesaria para trabajar con el Simulador que incluye trabajo con órganos adquiridos a partir de scanner (TAC, RM) y retroalimentación al tacto.	Julio 2010

Tabla 4. Entregables del proyecto

- **Reportes**

Los reportes serán generados mensualmente en el que se señalarán los cambios si es que han habido, así como el estado actual de la configuración, los encargados de esta función son los administradores de la configuración y el encargado de almacenar cada reporte ya sea de un EC o de una auditoría es el líder del proyecto.

- **Proceso de entregas**

Hasta el momento no se ha definido como será el proceso de entregas a los clientes del sistema debido al nivel de madurez del proyecto pues este se encuentra en la primera fase y comienza a desarrollarse.

4. Auditorías a la configuración

- **Número de auditorías a realizar y cuándo serán llevadas a cabo**

En el proyecto se establecen auditorías físicas cada vez que se concluya con el desarrollo de una línea base y una vez concluido la construcción del producto final, en este caso habiéndose entregado el producto de software y documentación respectiva a los clientes/usuarios.

En estas auditorías en el caso de concluir con una línea base se chequea:

- Si una línea base constituye un ejecutable el cumplimiento de los requerimientos inicialmente seleccionados
- La documentación y actualización de los elementos de configuración que componen la línea base.

En el caso de finalizar el proyecto se chequea:

- El cumplimiento del producto final con los requerimientos.

- Consistencia del manual de usuario.
- La realización adecuada de las pruebas al producto final y la documentación de las mismas.

Además se efectuará una auditoría interna mensual, el último martes de cada mes, donde se prueba o verifica:

- El sistema de salvas
- La gestión de versiones
- La existencia y actualización de la documentación.
- Expediente del proyecto.
- El cumplimiento de las tareas en la fecha correspondiente
- Ejecución de los cambios archivados en las solicitudes de cambio.

Como responsables de realizar las auditorías se escogieron: un representante del grupo de calidad del proyecto y uno de gestión de configuración, además de cualquier personal adicional al que se le pueda asignar la responsabilidad para colaborar con la tarea.

En estos momentos los encargados de asumir dichos roles son:

Calidad: Yoisy Pérez.

Gestión de configuración: Osmany Valdés.

3.1.9 Hitos

Todos los hitos son almacenados en la herramienta Trac y en el repositorio subversión con el fin de dales seguimiento.

Fase	*Hito*
Inicio	Documentos de captura de Requisitos
Elaboración	Documentos de Diseño e Ingeniería de Software
Construcción	Versión del Simulador, documentación, ayuda.
Transición	No conformidades.

Tabla 5. Hitos del proyecto

3.1.10 Entrenamiento

Aparece en el Plan de Capacitación disponible en el repositorio del proyexto.

3.2 Propuesta de capacitación al personal

Como se explicaba en el capítulo 1 la Administración de Configuración de Software es la disciplina de la Ingeniería de Software que comprende las herramientas y técnicas (procesos o metodologías) que una organización utiliza para administrar las configuraciones de los componentes del software que desarrolla, por tanto es uno de los procesos clave para toda organización dedicada a la producción de SW, ya que posibilita una mejor organización del desarrollo y mantenimiento del producto, facilitando el resto de los procesos de producción en vistas de lograr una mejor calidad de los productos finales.

Con el objetivo de verificar el nivel de conocimiento en cuanto a GCS que tenía el personal involucrado en el proyecto, se hizo una pequeña en encuesta a 20 de los integrantes del equipo (ANEXO 2), algunas de las interrogantes presentadas en la misma fueron:

- ¿Sabe usted que es Gestión de Configuración de Software (GCS)?
- ¿Conoce usted las actividades que se llevan a cabo dentro de esta disciplina?
- ¿Considera usted que es importante llevar a cabo el proceso de GCS dentro del proyecto?
- ¿Considera usted que es importante la utilización de las herramientas para la GCS?
- ¿Conoce usted cómo funcionan las herramientas seleccionadas en el proyecto para facilitar el proceso de GCS?
- ¿Considera usted importante que se le bride al equipo de desarrollo del proyecto la oportunidad de participar en un curso de capacitación sobre el proceso de GCS y el uso de las herramientas que facilitan el trabajo que se lleva a cabo en este?

La encuesta arrojó los siguientes resultados:

- El 70% de los miembros del equipo desconoce qué es la GCS, lo que implica que solo el 30% tenga conocimiento sobre el tema , de este último porcentaje:
  - El 15% afirma que solo conoce que se realizan en la GCS 2 actividades: control de cambios en la configuración y control de las versiones.
  - El 1% conoce todas las actividades que se realizan.
  - El 14% desconoce las actividades que se realizan.
- El 60% desconoce cómo funciona la herramienta Subversion, lo que trae como consecuencia que solo un 40% tenga conocimientos básicos sobre el tema, de este último porcentaje:
  - Solo el 5% ha trabajado con la herramienta.
- El 100% desconoce como funciona la herramienta Trac.
- El 100% afirma que es importante que se le brinde al equipo de desarrollo del proyecto la oportunidad de participar en un curso de capacitación sobre el proceso de GCS y el uso de las herramientas que facilitan el trabajo que se lleva a cabo en este.

Todo lo antes expuesto demostró que había mucha desinformación en cuanto a la GCS en el proyecto y la preocupación del equipo por su preparación en cuanto al tema, por tanto se decidió en conjunto con el líder impartir un curso corto de capacitación sobre GCS y herramientas para facilitar el desarrollo de este proceso.

3.2.1 Duración.

La duración del curso es de 1 semana. El por ciento de horas clases asignado por temas se muestra en la siguiente figura:

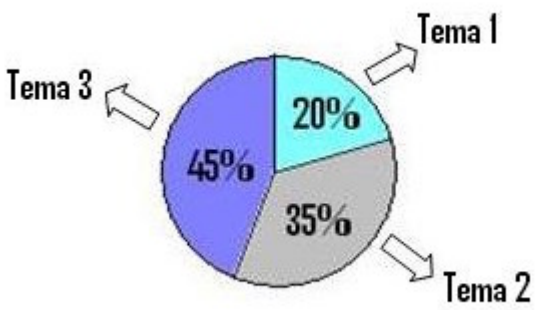

Fig. 4 Esquema de la duración del curso por temas.

Nota: se decidió otorgar más horas clases al tema 1 y 2 por la importancia que tienen en el proyecto, ya que son estos temas sobre los que radica la mayor cantidad de dificultades y son los que debe dominar de manera urgente el equipo de desarrollo.

3.2.2 Objetivo General

Reflexionar acerca de la importancia de la Gestión de Configuración, además de aprender el manejo de las herramientas escogidas para desarrollar esta disciplina en el proyecto Simulador Quirúrgico.

3.2.3 Objetivos específicos

- Explicar la necesidad de llevar un proceso de desarrollo de software con calidad.

- Describir el Proceso de Gestión de Configuración de Software.
- Identificar cada uno de los roles que intervienen en la Gestión de Configuración, y las responsabilidades que tienen asignadas.
- Caracterizar de forma general algunas herramientas que se puedan utilizar en el proceso.
- Describir el proceso de elección e implantación de herramientas para facilitar la gestión de configuración del proyecto simulador quirúrgico.
- Explicar el funcionamiento de las herramientas escogidas para controlar los cambios y las versiones

3.2.4 Metodología

Se combinan distintos tipos de estrategias metodológicas como intercambios con expertos en el tema, exposición de seminarios, talleres y conferencias.

3.2.5 Temas

➢ Tema 1 Gestión de Configuración.

- Introducción a la Gestión de Configuración. Importancia.
- Conceptos fundamentales.
- Actividades.
    - Identificación de la configuración
    - Control de Versiones
    - Control de cambios en la configuración
    - Auditorias de configuración
    - Generación de informes de estado

➢ Tema 2 Herramientas de control de versiones. Subversion.

- Introducción a las Herramientas de Control de versiones.
    - Conceptos generales.

- Subversion.
    - Características fundamentales.
    - Comparación de Subversion con otras Herramientas de Control de Versiones.
    - Instalación de cliente.
    - Uso práctico de la herramienta.

➢ Tema 3 Herramienta para complementar la Gestión de la configuración. Trac.

- Trac.
    - ¿Porque usar el Trac para complementar la Gestión de Configuración en el proyecto Simulador Quirúrgico?
    - Características fundamentales.
    - Uso práctico de la herramienta.

3.2.6 Bibliografía para el curso

de Antonio, Angélica. 2001. Gestión de Configuración. 2001.

Jacobson, Iver, Booch, Grady and Rumbwgh, James. 2002. El Proceso Unificado de Desarrollo de Software. 2002.

Pressman, Roger. 1997. Ingeniería del Software: Un Enfoque Práctico. Cuarta edición. 1997. —. 2002. Ingeniería del Software: Un Enfoque Práctico. Quinta Edición. 2002.

Subversion [Online] [Cited: 02 22, 2008.] http://subversion.tigris.org/.

Trac [Online] [Cited: 03 05, 2008] http://trac.edgewall.org/

Tortoisesvn [Online] [Cited: 03 05, 2008] http://tortoisesvn.tigris.com

3.3 Evaluación de la aceptación del proceso de Gestión de Configuración

El proyecto "Simulador Quirúrgico" actualmente se encuentra en su primera fase, pero empezar y mantener un esfuerzo de mejora en cualquier proyecto de software que comience a desarrollarse, requiere un compromiso enorme por parte de todos los que lo integran, ya que son muchos los factores que van a influir a la hora de obtener un producto final con la eficiencia requerida. Como se explicaba antes, la Gestión de Configuración juega un papel sumamente importante en el proceso de desarrollo de software, luego de la puesta en marcha de la misma en el proyecto se tomó la decisión de evaluar su nivel de aceptación en el equipo de desarrollo tomando como meta inmediata la conformidad y beneplácito de todos los involucrados, y de esta forma poder limar cualquier problema o dificultad a tiempo. Para facilitar este proceso de evaluación se realiza una encuesta (ver ANEXO 3) en la cual se hizo énfasis en interrogantes como:

- ¿Considera usted que el proceso de GCS ha sido de utilidad en el proyecto?
- ¿Considera usted que el proceso de Solicitud de Cambio se hace de la forma correcta?
- ¿Cómo evaluaría usted el funcionamiento de la herramienta Subversion?
- ¿Cómo evaluaría usted el funcionamiento de la herramienta Trac?
- ¿Puede usted mencionar 3 deficiencias que considere que tenga el proceso de GCS en el proyecto?

Se realizó un total de 20 encuestas. Los resultados obtenidos mostraron que un 100% de los integrantes del equipo de desarrollo afirman que el proceso de GCS ha sido de utilidad en el proyecto, ya que ha permitido agilizar y facilitar el trabajo con la ayuda de algunas herramientas, controlando por medio de las mismas los cambios que generan las diferentes versiones. De la misma forma un 99% de los integrantes del proyecto coinciden en que el proceso de Solicitud de Cambio se lleva a cabo de la forma correcta y un 1% alega que el comité de cambio demora mucho en dar respuesta cuando se solicita un cambio importante, provocando atrasos. Por otra parte un 95% y 99% del personal coincide en evaluar de "eficiente" el funcionamiento de la herramienta Subversión y Trac respectivamente, mientras que un 5% y 1% evalúan el funcionamiento de la herramienta Subversión y Trac respectivamente de "aceptable". Todo lo antes planteado demuestra que hay un elevado nivel de aceptación del proceso en el proyecto pues se han logrado erradicar deficiencias importantes entre

ellas: que no se tenía instalada ninguna herramienta para el control de los cambios y las versiones lo que traía como consecuencia el problema de los datos compartidos, esto surgía cuando más de una persona simultáneamente accedía y modifica los mismos datos, pues muchas veces los cambios introducidos por una persona eran una sorpresa inesperada para otra. Pero sin lugar a dudas la mayor dificultad aparecía cuando dos personas actualizaban el mismo módulo, entonces los cambios de uno de ellos podía ser sobrescrito por los cambios del otro.

CONCLUSIONES GENERALES

Después del estudio realizado referente a la Gestión de Configuración de Software se demostró el papel esencial que ocupa esta disciplina en el proceso de desarrollo de software, puesto que examina aspectos tan importantes como el control de versiones y control de cambio, el proceso de administración de los elementos de configuración y las líneas bases, además de las auditorías y los reportes.

Es caracterizado el proceso de GCS, haciendo énfasis en como se desarrollaba en la universidad, se realizó un estudio de un conjunto de herramientas para facilitar la realización de las actividades que incluye este proceso, se evidenció que existen muchas en el mercado internacional que tienen similitudes en cuanto a funcionalidades y a la estructura del modelo del repositorio en el caso de las controladoras de versiones.

Una vez planificado el proceso de GCS en el proyecto basándose en las necesidades y características del mismo, se logra establecer una disciplina de trabajo en las tareas asociadas al control de cambios teniendo en cuenta la descripción del resto de las actividades planteadas. Además se capacita a los integrantes del grupo de desarrollo de software en cuanto a la GCS y al funcionamiento de las herramientas seleccionadas.

Se obtuvo el Plan de GCS como resultado aportando una solución al gran problema de la administración de los cambios, este es muy útil para mantener todos los procesos documentados permitiendo saber dónde se está, qué se está haciendo y cuales son los pasos a seguir. Por parte del equipo de desarrollo se adquirió una evaluación positiva en cuanto a la aceptación del proceso.

El objetivo de la GCS es asegurar la integridad del producto haciendo más manejable su evolución y mantenimiento mediante el control de los cambios. De ahí la importancia de realizar eficientemente las actividades que en esta se desarrollan. No realizar una adecuada GCS puede acarrear grandes problemas y resultados poco eficientes.

Por lo anteriormente planteado se llega a la conclusión que se cumplieron con todas las tareas y el objetivo principal trazado al inicio de esta investigación.

RECOMENDACIONES

Se considera necesario al culminar el presente trabajo hacer las siguientes recomendaciones en aras de perfilar el camino de investigaciones futuras:

1. Aplicar el proceso de Gestión de Configuración de Software en los proyectos de la

   Universidad de las Ciencias Informáticas adaptándolo a las necesidades de cada uno.

2. Evaluar el proceso con la utilización de métricas para la GCS luego de haber transcurrido un tiempo de aplicado.
3. Enriquecer a las herramientas seleccionadas de nuevas funcionalidades.

REFERENCIAS BIBLIOGRÁFICAS

[1] Antonio, Angélica de. Gestión de Configuración. 200.

[2] Babich, W. Software Configuration Management. 1986.

[3] Calvo- Manzano, José A., et al. Análisis y Diseño detallado de aplicaciones informáticas de gestión. 1996.

[4] Crosby, Philip. Quality Is free. 1970.

[5] Febles, Ailyn. Medir el proceso de control de configuración ¿una utopía para la Industria Nacional de Software? 2002.

[6] IBM. Rational Unified Process. 2003.

[7] ISO. ISO 10007 Quality management – Guidelines for configuration management. 1995.

[8] Jacobson, Ivar. Applying UML in The Unified Process. 1998.

[9] MINREX. Informe del MINREX Cumbre Mundial de la Sociedad de la Información. 2004.[Disponible en: http://www.cubaminrex.cu/Cumbre-CMSI/La%20informatizaci%C3%B3n%20en%20Cuba.htm]

[10] Navarro, Antonio. Conferencia sobre Gestión de Configuración de Software. 2004.

[11] Pressman, Roger. Ingeniería del Software: Un Enfoque Práctico. Cuarta edición. 1997.

[12] —. Ingeniería del Software: Un Enfoque Práctico. Quinta Edición. 2002.

[13] Society, IEEE Computer. IEEE Standard for Software Configuration Management. 1990.

BIBLIOGRAFÍA

Brand, Horst H. 2005. git: Control de versiones estilo Linus. 2005.

Bulma. [Online] [Cited: 02 10, 2008.] http://bulma.net/body.phtml?nIdNoticia=2335.

Cabrera, Luis, Morales, Alberto and Arocas, Lorenzo. 2007. Una Introducción a Darcs. 2007.

Catrin, Franco M. 2003. Uso practico de CVS para control de versiones. 2003.

Control de versiones con Subversion. [Online] [Cited: 02 26, 2008.] http://svnbook.red-bean.com/index.es.html.

Cortizo, José Carlos, Expósito, Diego and lRuiz, Migue. 2007. eXtreme Programming. 2007.

de Antonio, Angélica. 2001. Gestión de Configuración. 2001.

Febles, Ailyn. 2002. Medir el proceso de control de configuración ¿una utopía para la Industria Nacional de Software? 2002.

Fast Version Control System. – Git [Online] [Cited: 02 18, 2008.] http://git.or.cz/.

GNU Arch. [Online] [Cited: 18 02, 2008.] http://www.gnu.org/software/gnu-arch/.

Instalacion de Subversion. [Online] [Cited: 02 26, 2008.] http://www.osmosislatina.com/subversion/instalacion.htm.

Jacobson, Iver, Booch, Grady and Rumbwgh, James. 2002. El Proceso Unificado de Desarrollo de Software. 2002.

LugFi. Introducción a sistemas de control. [Online] [Cited: 02 16, 2008.] http://www.lug.fi.uba.ar/documentos/scms/index.php.

Martínez, Alejandro and Martínez, Raúl. 2005. Guía a Rational Unified Process. 2005.

Mendoza, María A. 2004. Metodologías de Desarrollo de SofTware. 2004.

Microsoft Colombia. [Online] [Cited: 12 10, 2007.] http://www.microsoft.com/colombia/portafolio/msf.htm.

Monotone. [Online] [Cited: 02 21, 2008.] http://www.monotone.ca/.

Navarro, Antonio. 2004. Conferencia sobre Gestión de Configuración de Software. 2004.

página catálogo. [Online] [Cited: 02 15, 2008.] www.cdlibre.org/consultar/catalogo/Programacion_Sistemas-de-control-de-versiones.html.

Pressman, Roger. 1997. Ingeniería del Software: Un Enfoque Práctico. Cuarta edición. 1997.

—. 2002. Ingeniería del Software: Un Enfoque Práctico. Quinta Edición. 2002.

Rodríguez, Karina Laura. 2007. Gestión de configuraciones. 2007.

Saavedra, Esteban. 2007. Gestión de Proyectos de desarrollo de Software. 2007.

Sánchez, Emilio A., Letelier, Patricio and Canós, Jose H. 2007. Mejorando la gestión de historias de usuario en XP. 2007.

Subversion. [Online] [Cited: 02 22, 2008.] http://subversion.tigris.org/.

Trac. [Online] [Cited: 02 22, 2008.]

http://congreso.softwarelibre.org.bo/files/presentaciones/gestion_proyectos.pdf.

ANEXOS

Anexo I: Encuesta presentada a integrantes de algunos de los proyectos productivos de la Universidad de las Ciencias Informáticas.

Con motivo de conocer como se desarrolla el proceso de Gestión de Configuración de Software en algunos de los proyectos que se desarrollan actualmente en la universidad usted debe llenar el siguiente cuestionario.

1. ¿Sabe qué es Gestión de Configuración de Software?

   SI____ NO____

2. ¿Se realiza en su proyecto la GCS?

   SI____ NO____ NO ESTOY SEGURO_____

3. ¿Se controlan los cambios en su proyecto? En caso de que su respuesta sea "SI" argumente.

   SI____ NO____

   _____
   _____
   _____
   _____

4. ¿Se realiza el control de las versiones en su proyecto? En caso de que su respuesta sea "SI" diga que herramientas se utilizan para esta actividad.

   SI____ NO___

   _____
   _____
   _____

5. Mencionen las actividades dentro de la GCS que se realizan en su proyecto en exceptuando el control de los cambios en la configuración y el control de las versiones.

    1) _____
    2) _____
    3) _____
    4) _____
    5) _____
    6) _____
    7) _____

Se le garantiza la confidencialidad en cuanto a todo lo escrito en este documento.

Anexo II: Cuestionario presentado a los integrantes del proyecto Simulador Quirúrgico con vista al curso de capacitación.

Con motivo de evaluar el nivel de conocimiento sobre la disciplina de Gestión de Configuración (GSC) dentro del proyecto Simulador Quirúrgico usted debe llenar el siguiente cuestionario.

1. ¿Sabe usted que es Gestión de Configuración?

   SI____ NO___ MUY POCO____

2. ¿Conoce usted las actividades que se llevan a cabo dentro de esta disciplina? Escríbalas ordenadas según el nivel de importancia que crea que tiene cada una de ellas.

   1) _____
   2) _____
   3) _____
   4) _____
   5) _____
   6) _____
   7) _____

3. ¿Considera usted que es importante llevar a cabo el proceso de GCS dentro del proyecto?

   SI___ NO___

   ¿Por qué? _____

   _____

   _____

4. ¿Considera usted que es importante la utilización de las herramientas para la GCS? ¿Por qué?

   _____

_____

_____

5. Las herramientas escogidas para llevar a cabo la GCS del proyecto son:

   Subversion ___ y Trac____

   Marque con una "X" si tiene al menos un conocimiento básico y con "XX" si alguna vez la ha utilizado.

6. ¿Considera usted importante que se le bride al equipo de desarrollo del proyecto la oportunidad de participar en un curso de capacitación sobre el proceso de GSC y el uso de las herramientas que facilitan el trabajo que se lleva a cabo en este?

   SI___ NO___ ¿Por qué?_____

   _____

   _____

   Se le garantiza la confidencialidad en cuanto a todo lo escrito en este documento.

Anexo III: **Cuestionario presentado a los integrantes del proyecto Simulador Quirúrgico para evaluar el nivel de aceptación del proceso de Gestión de Configuración de Software.**

Con motivo de evaluar el nivel de aceptación del proceso de Gestión de Configuración de Software (GCS) luego de su puesta en marcha en el proyecto Simulador Quirúrgico usted debe llenar el siguiente cuestionario.

1. ¿Considera usted que el proceso de GCS ha sido de utilidad en el proyecto? ¿Por qué?

_____

_____

_____

2. Mencione 3 deficiencias que considere que tenga el proceso de GCS en el proyecto.

1) _____
2) _____
3) _____

3. ¿Considera usted que el proceso de Solicitud de Cambio se hace de la forma correcta? Explique

_____

_____

_____

4. ¿Cómo evaluaría usted el funcionamiento de la herramienta Subversion? Justifique su respuesta.

Aceptable ___

Eficiente ___

Ineficiente ___

_____

_____

_____

5. ¿Cómo evaluaría usted el funcionamiento de la herramienta Trac? Justifique su respuesta.

Aceptable ___

Eficiente ___

Ineficiente ___

_____

_____

_____

6. Realice un comentario sobre el proceso que está siendo evaluado por usted(qué le ve a favor o en contra)

_____

_____

_____

_____

Se le garantiza la confidencialidad en cuanto a todo lo escrito en este documento.

GLOSARIO DE TÉRMINOS

SMQ: Simulador Quirúrgico.

GCS: Gestión de Configuración de Software.

ECS: Elemento de Configuración de Software.

CCC: Comité de Control de Cambios.

Estándar: Es aquello que se considera modelo.

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos.

ISO: Organización Internacional de Normalización.

CMMI: Capability Maturity Model Integration: Integración del Modelo de Madurez de las Capacidades.

SPICE: Software Process Improvement and Capability Determination, también conocida como norma ISO 15504, constituye un estándar internacional para procesos de desarrollo de software que proporcione un marco de trabajo uniforme para gestión e ingeniería del software.

Repositorio: Lugar en el que se almacenan los datos actualizados e históricos, a menudo en un servidor.

SHA-1: Algoritmo de identificación de archivos.

Merge: Mezclar, fusionar. La función Merge realiza operaciones de combinación.

Releases: Liberaciones de un producto de software.

Cherry picking: Es un merge, mediante el cual, un cambio/s en el contenido de un elemento que ha sido introducido por una revisión, es aplicado en el espacio de trabajo. En este caso no se hace el

merge del contenido del elemento en el origen, sino de los cambios que este ha incorporado en esa revisión respecto a su revisión anterior.

1