Universidad de las Ciencias Informáticas

Facultad 8

Título: PROPUESTA DE HERRAMIENTAS PARA LA GESTIÓN DE LA CONFIGURACIÓN EN EL PROYECTO CICPC

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias Informáticas

Autor(es): Merly Cabrera Pérez

Dayneri Idelisa Corps Agüero

Tutor: Lic. Javier Martínez Silva

Junio, 2007

Primera Ley de la Ingeniería del Sistema:

"Sin importar en que momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida".

"Those who cannot remember the past are condemned to repeat it"

George Santayana

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Merly Cabrera Pérez	**Dayneri Idelisa Corps Agüero**	**Lic. Javier Martínez Silva**
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

DATOS DE CONTACTO

Tutor: Lic.: Javier Martinez Silva. Graduado en Ciencias de la Computación por la Universidad de Oriente en Julio de 2005. Se desempeña como profesor de la asignatura de Programación en la Factura 8. Actualmente es el Jefe de la disciplina de Programación de dicha facultad de la Universidad de las Ciencias Informáticas.
Correo Electrónico: javierm@uci.cu

AGRADECIMIENTOS

Cada día vivido en la Universidad de las Ciencias Informáticos nos ha convertido en una gran familia a la cual hoy queremos darle las gracias, al igual que a nuestra Revolución y a su figura insigne nuestro Comandante en Jefe, a quien queremos agradecer por haber confiado en cada uno de nosotros su sueño, por habernos convertido en sus soldados del futuro, por iluminarnos el camino, por habernos regalado la paz de crecer y vivir en un país donde el hombre se miden por lo que es, no por los bienes con los que cuenta.

De forma muy especial a nuestro tutor el Lic. Javier Martínez Silva que con su esmero y dedicación nos enseño que la virtud, se consagra constantemente a lo que es difícil de hacer, y cuanto más dura es la tarea, más brillante es el éxito,

A nuestro decano: Alcides Cabrera Campos, que con su ejemplo, su persistencia y empeño nos demostró que la verdadera recompensa se encuentra más en el esfuerzo que en el resultado. Un esfuerzo total es una victoria completa.

A nuestros profesores, que han sabido ser guías en nuestro cursar por esta Universidad, y muy especialmente a aquellos que han colaborado de manera activa en la realización de Nuestra Tesis (Doctor Osvaldo Ruiz (Cujae), Ramsés Delgado Martínez). Así como al estudiante Carlos Vegas del proyecto CICPC por ayudarnos en todo el proceso de Investigación y a Raisel Millian.

.

A todos nuestros compañeros que siempre nos han tenido palabras de fuerzas para seguir en los días que el trabajo se nos hizo más difícil y que estuvieron disponibles en cualquier momento.

A nuestros amigos, los que han estado siempre llorando nuestras tristezas y celebrando nuestros triunfos, por confiar en nosotros y llenarnos la vida con esos pequeños momentos que nos harán recordarlos por siempre (Yudisbel, Greisy, Aurorita, Yanetzy, Edita, Ismel).

A nuestros padres por estar siempre preocupados por el avance de la tesis, y por el apoyo que nos han dado para seguir adelante y hacernos mejores personas, hijos, hermanos y amigos.

DEDICATORIA

De Dayneri:

A ti que hoy ya no estas conmigo, que has sido siempre y serás mi fuerza para seguir adelante y para hacer lo correcto, a ti que tanto te he querido y que tanto te extraño…y que me he arrepentido por no haber estado contigo en esos últimos momentos…a mi abuela (Idelisa Fontaine)

A mis padres que han sido siempre la meta por la que he querido vivir, por lo que he caminado hacia delante, y de los cuales estoy tan sumamente orgullosa: Adael y Carmen...

A mi abuelito Segundo, que tanto quiero…

A mis hermanos Niyima y Misael que siempre han estado en mi vida, y que los llevo en mi corazón…

A mis sobrinitos Heyser y Adael que aún con su poca edad siempre me preguntan como están mis estudios y que los quiero como si fueran míos…

A mis amigos Yudisbel, Merly amigos de batalla, a Mindrey, Yaidelin y a todos con los que he compartido mi vida en la UCI y que ya forman parte de mi vida…

Y a ti que te he dejado para ultimo pero que hoy ocupas un lugar primordial en mi vida, que sé que aún lejos me has querido, me has amado, me has cambiado y convertido en la mujer que siempre he querido ser, que has sido mi compañero, mi amigo… a mi esposo Alain

DEDICATORIA

De Merly:

A ti que ya no estas a mi lado, pero que vives en mi corazón, y que tanto extraño, porque no importa donde estés, siempre serás mi ángel… (Mi abuelita cuca)

A ti que luchas incansablemente aún cuando la vida exige de ti grandes sacrificios, por ser el único de mis abuelitos que me ha acompañado todo este tiempo…tú me has enseñado que la vida vale cuando tienes el valor de enfrentarla. (Mi abuelito Baldo)

A mis padres, por no haberme permitido dudar ni un momento, por haberme permitido elegir, por siempre estar a mi lado apoyándome y por no haberme dejado sola jamás …por ser siempre mas que parte de mi vida, mi mayor razón de vivir….Gracias (Mami y papi).!!!

A mis hermanitos que tanto quiero, por ser las dos personitas que desde que llegaron a este mundo, llenaron mi vida de bendiciones y de alegrías… (Maide y Oscarito)

Para ustedes, la familia que la vida me ha permitido elegir y que me han enseñado que el mejor triunfo que se puede tener, es tener el derecho de llamar a alguien "Amigo"... (Yudisbel, China, Greysi, Aurora, Marbe, Grisel,Ismel, a mi tropa de Fomento).

A ti, que tan lejos estas de mí, a ti que iluminas mi vida con tu cariño, con tu ternura, por hacerme sentir feliz, por ser mi compañero, mi amigo, por enseñarme que el amor es una filosofía de la vida y que los sueños son solamente para hacerse realidad...a mi esposo Migue.

RESUMEN

El desarrollo de Software, o la habilidad para producirlo, se reconocen como la clave para obtener una ventaja competitiva en los negocios actuales que cuentan con altas tecnologías. En los últimos años se han obtenido grandes avances en disciplinas como la Gestión de la Configuración, dado el desarrollo tecnológico. En el siguiente trabajo se presentan los resultados de una investigación dirigida a las herramientas para esta disciplina. Se siguen como objetivos: proponer herramientas tanto para el control del progreso del proyecto como para el control de versiones y automatizar los procesos dentro del proyecto: Cuerpo de Investigaciones Científicas, Penales y Criminalística (CICPC). Los resultados más importantes de la investigación proponen como las herramientas a utilizar: el Subversion para el Control de Versiones, el TortoiseSVN como cliente y el Trac para la Gestión de la Configuración, todas validadas por un caso de estudio. Entre las principales conclusiones a las que se arribaron estuvieron que constituye una necesidad para la UCI, al igual que para la Industria Cubana de Software, conferirle una mayor importancia al proceso de Gestión de Configuración, ya que este juega un papel primordial dentro el ciclo de vida de cualquier software; usar herramientas que se ajusten a las necesidades y características de cada proyecto; lograr con el uso de las herramientas, que el grupo de trabajo se mantenga en constante comunicación y un control sobre los Elementos de la Configuración del Software, para ganar en eficiencia dentro del proceso de desarrollo del proyecto CICPC.

ÍNDICE

TABLA DE CONTENIDOS

INTRODUCCIÓN

Hoy en día la informática, vocablo proveniente del francés *informatique*, acuñado por el ingeniero Philippe Dreyfus en 1962, formado por la conjunción de las palabras *information* y *automatique*, es la disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales. En lo que hoy se conoce como informática confluyen muchas de las técnicas y de las máquinas que el hombre ha desarrollado a lo largo de la historia para apoyar y potenciar sus capacidades de memoria, de pensamiento y de comunicación que tuvo sus primeros orígenes como respuesta a una de las más viejas aspiraciones del hombre: simplificar sus tareas.

La ingeniería del Software nace como una disciplina para aplicar los principios, técnicas y herramientas de desarrollo de software. En la década de los 80's, se realizaba software de forma artística, es decir utilizando métodos y técnicas ad-hoc donde la experiencia era el camino a seguir. Este enfoque produjo grandes y exitosos productos de programación, pero conforme los proyectos se volvieron más complejos debido al avance del hardware y software y la penetración cada vez mayor de la informática en todos los ámbitos de la sociedad, se comenzó a producir software sin calidad, se incumplieron los presupuestos y se incrementaron dramáticamente los costos de mantenimiento.(VALDÉS 2006)

En Cuba no existe una producción elevada de software, debido en gran medida al bloqueo económico impuesto por los EEUU y la hostilidad incesante, sin embargo en los últimos años se ha evidenciado un avance científico técnico en la industria del software, esto ha sido posible debido a que se ha incrementado la cantidad de técnicos medios y de universitarios que se preparan para la producción de software en Cuba gracias a la batalla de ideas, existe una prioridad hacia el desarrollo de esta industria por lo que se ha construido una Universidad de las Ciencias Informáticas (UCI) con deseos de llegar a la excelencia, con 10 facultades cada una con perfiles informáticos distintos. Esto juntamente con las anteriores universidades del país donde se cursaban estudios de este perfil y con los nuevos tecnológicos de informática, se revoluciona la educación hacia esta nueva rama de la economía.

Actualmente existen diversos programas que se están llevando a cabo en Cuba entre los cuales se pueden mencionar, el IC – CULT: Fomento de la Cultura Digital, que tiene como objetivo fundamental

garantizar la necesaria preparación de los recursos humanos, instrumentar un proceso de educación continuo y ampliar la cultura general de la población sobre las tecnologías. Este ha tenido como logros unos 300 Joven Club en construcción, crecimiento y continuo desarrollo de la red de los centros de educación superior, incluyendo las sedes municipales, surge paulatinamente pero con firmeza la red del Ministerio de Educación, crece a niveles extraordinarios la especialidad de la Informática en las universidades y los politécnicos entre otros programas.

En los proyectos de producción, de la Universidad de las Ciencias Informáticas, se han presentado problemas con la Gestión de la Configuración, en específico con los procesos de Control de Cambios, entre los cuales se encuentran: desconocimiento sobre la Gestión de la Configuración, restándole importancia al efecto que tiene el cambio incontrolado sobre el software; la pérdida de información relevante ante algún cambio realizado sobre cualquier Elemento de la Configuración del Software(ECS); sobrescripción de los ECS con sus respectiva pérdida de información al trabajar varios desarrolladores sobre un mismo archivo; falta de documentación sobre quién realiza los cambios, cuándo y qué es lo que se modifica; cambios que se realizan sobre un archivo, en el que se introduce un error, sobre el cual se sigue trabajando y posteriormente al ser identificado el error, no se conoce a que versión corresponde; problemas con la identificación de los procesos de Control de Cambio; falta de conocimiento sobre las herramientas que automatizan los procesos; no se tiene una idea de cuales son los ECS que van a formar parte del software, no se puede realizar un seguimiento del proyecto, no se entregan los productos en las fechas acordadas, muy difícil reutilizar las funcionalidades desarrolladas previamente, dificultad para realizar el mantenimiento del software, no siempre se cuenta con la última versión de un ECS, restando calidad al producto, provocando que se invierta un mayor esfuerzo en la corrección de un error que aparezca en el software, que no se puedan obtener reportes del estado actual del proyecto, ni del esfuerzo de los trabajadores. Actualmente en la UCI, existen alrededor de 166 proyectos, de los cuales en sus inicios alrededor del 60 %, no llevaban acabo la Gestión de la Configuración del Software (GCS), ya que se creía innecesario, y algo burocrático, pero el desempeño diario de los mismo trajo consigo la necesidad de aplicar la GCS. Entre estos proyectos se pueden mencionar CICPC, Prisiones, Correos de Cuba, Identidad entre otros.

Ante tal situación se definió como problema científico la interrogante de: ¿Cuáles de las herramientas informáticas existentes permitirá llevar a cabo con mayor eficiencia los procesos de Gestión de la Configuración, y de Control de Cambios en un proyecto informático de producción?, basados en los problemas definidos anteriormente en los proyectos que se desarrollan en la UCI, como CICPC, donde existen problemas con la Gestión de la Configuración , el Control de Cambios y dentro de este el Control de Versiones, actividades importantes dentro de la disciplina, estos problemas se deben a la no existencia de conocimientos de herramientas, que permitan controlar dichas acciones.

Con la investigación se persigue: Proponer herramientas automáticas para los procesos de la Gestión de la Configuración en el proyecto CICPC.

De este objetivo se desprenden los siguientes objetivos específicos:

- Brindar al equipo de trabajo del proyecto CICPC, un grupo de herramientas que permitan automatizar los procesos de Control de Cambios y Control de Versiones.

- Brindar la posibilidad de controlar el progreso del proyecto, de forma tal que finalmente se pueda cumplir con los objetivos que han sido fijados en los planes.

- Proveer una herramienta que posibilite la comunicación entre todo el equipo de trabajo del proyecto.

- Ayudar a cada integrante del grupo de trabajo, a mejorar su productividad, a través del uso de herramientas, brindándoles la posibilidad de autoevaluarse, determinando cuánto tiempo le dedican a un tipo de tarea en específico, y que puedan determinar cuán bien avanzan, según los planes asignados.

Se sigue como Hipótesis que si se realiza un estudio de las herramientas automatizadas, para la Gestión de la Configuración, siguiéndose criterios especializados de selección, se podrán aplicar las más adecuadas, tributándose a una mayor eficiencia en los procesos de desarrollo de software del proyecto CICPC.

INTRODUCCIÓN

Con el desarrollo de esta investigación se han identificado los siguientes beneficios:

- El estudio de las herramientas automáticas para la GCS y Control de Cambios (CC).
- La descripción de las principales características de las herramientas automáticas para la GCS y CC.
- La propuesta de Herramientas automáticas para la GCS y CC en el proyecto CICPC.
- La aplicación de dichas herramientas a un caso de estudio del proyecto de CICPC de la facultad 8.
- La propuesta de Políticas de Integración y Uso de las herramientas para el proyecto CICPC.

Para el desarrollo completo de la investigación se realizarán las siguientes Tareas:

- ✓ Hacer búsqueda de la información en los diferentes medios tanto de las herramientas como de los procesos de Gestión de la Configuración.
- ✓ Estudiar la información que se encontró en la actividad de investigación, acerca de la Gestión de la Configuración, y los procesos de Control de Cambio y de Versiones así como de las herramientas.
- ✓ Presentar información acerca de las herramientas que actualmente responden tanto a las necesidades del proyecto y que se ajusten a las características del mismo.
- ✓ Definir Criterios de Selección según estudio realizado para la comparación de las herramientas.
- ✓ Seleccionar a partir de los resultados las herramientas más eficientes, teniendo en cuenta los criterios de selección establecidos.
- ✓ Aplicar las herramientas a un determinado caso de estudio desarrollado en el proyecto de producción CICPC.
- ✓ Optimizar los resultados de los proceso de Gestión del Software a partir de los resultados obtenidos de la investigación y la aplicación al Caso de Estudio, para el proyecto CICPC.

La tesis quedó estructurada en tres capítulos. El Capítulo 1, referido al marco teórico, relacionado con la GCS, la comparación y selección de las herramientas, para la definición de la propuesta en el proyecto CICPC. En el Capítulo 2, se justifican las herramientas seleccionadas y la utilización de estas en los procesos de Control de Cambio y de Versiones. En el Capítulo 3, se aplican dichas herramientas a un caso de estudio que se desarrolló en el proyecto CICPC, y se proponen políticas de Integración y uso de las herramientas seleccionadas.

CAPÍTULO 1: INTRODUCCIÓN A LA GESTIÓN DE LA CONFIGURACIÓN. HERRAMIENTAS

Los ambientes cambiantes en el área de desarrollo de software y la competencia globalizada han conllevado a que se vayan desarrollando paralelamente, disciplinas como la Gestión de la Configuración dentro del desarrollo de software, tomando auge e importancia en el desarrollo de sus productos.

En este Capítulo se han desarrollado diferentes temas como los antecedentes a partir de estudios realizados por organizaciones internacionales desde hace varios años, se hace un estudio de la Gestión de la Configuración, definiéndose cada una de las tareas fundamentales, tales como la identificación de los Elementos de la Configuración del Software, el Control de Cambios, entre otras, y se presentan un análisis y estudio de las herramientas tanto para la Gestión de la Configuración, como para el caso específico del Control de Versiones, estableciéndose los parámetros a partir de los cuales se realiza la comparación de las herramientas y también aparecen los resultados de la comparación, que son las herramientas que se proponen y que son descritas con mayor profundidad en el Capítulo 2.

1.1 Antecedentes

Hacia el final del segundo milenio de la era cristiana, varios acontecimientos de trascendencia histórica han transformado el paisaje social de la vida humana. Una revolución tecnológica, centrada en torno a las tecnologías de la información, está modificando la base material de la sociedad a un ritmo acelerado de la que Cuba, y la Universidad de las Ciencias Informáticas es una muestra. Con el desarrollo de esta, comenzaron también los proyectos informáticos que fueron parte de los primeros pasos que se dieron dentro de la informática.

La Industria del Software en el mundo se desarrolla a un ritmo vertiginoso, aunque la producción sigue siendo aún baja y los costos muy elevados. Esta situación se debe, en la mayoría de los casos, a la no aplicación de técnicas de Ingeniería y Gestión de Software y la no definición de roles y procesos adecuados en el desarrollo de software. (FEBLES 2004)

Como ejemplo ilustrativo podrían mencionarse los alrededor de 5 mil millones de dólares que se calculan como ingresos de la India anualmente(MARTÍNEZ 2006), incluso países en vías de desarrollo como Venezuela en el año 2003 reportaron ingresos cercanos a los $237 millones de dólares por el concepto de producción de software (MARTÍNEZ 2006). Estos números son indicadores de la existencia de un mercado que, aunque nada fácil de conquistar, puede constituirse en pilar del crecimiento económico de no pocos países. Como derivación lógica de la misma, se encuentra el hecho de que el software requiera de una mayor calidad, y sean más competitivas las empresas que intentan ser exitosas, pues la calidad en el servicio marca el prestigio, y puede determinar el establecimiento de negocios significativos.

En ese sentido, y observando las enormes ventajas que puede representar para un país el desarrollo de los servicios informáticos y la producción de software, el Ministro cubano de Informática y Comunicaciones, Ignacio González, ha expresado que Cuba debe aspirar a hacerse un espacio en el mercado mundial de software. Un negocio que mueve más de 440.000 millones de euros al año y que podría suponer una importante fuente de recursos para Cuba (AMÉRICA 2002).

En el informe presentado por el Ministerio de Relaciones Exteriores de Cuba en la Cumbre Mundial para la Sociedad de la Información en el 2004 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" (MINREX 2004).

En resumen, el país observa como una posibilidad alcanzable la incorporación en el mercado mundial de software, y serios pasos se muestran encaminados a cumplir ese objetivo: la creación del Ministerio de la Informática y las Comunicaciones, la creación de la Universidad de las Ciencias Informáticas, la revitalización y potenciación de los tecnológicos de informática y la Universalización de la Carrera de Ingeniería Informática, son ejemplos ilustrativos (MARTÍNEZ 2006).

CAPÍTULO 1

El desarrollo de modelos de negocio basados en la gestión de proyectos genera nuevos valores para los clientes y más riqueza para los inversores, mejores resultados dentro de los procesos de un proyecto, y también la calidad de los resultados y del producto informático.

A lo largo de los años varias compañías han realizado investigaciones que han obtenido resultados cuantitativos de los proyectos más importantes. Un estudio realizado por Standish Group (grupo de investigadores formado en 1985, Massachussets, se dedica a la determinación de riesgo en proyectos grandes) analizó el desarrollo de 8000 proyectos de software, realizados por 350 empresas diferentes y concluyó que sólo el 16% de los proyectos de software se realizan con éxito.

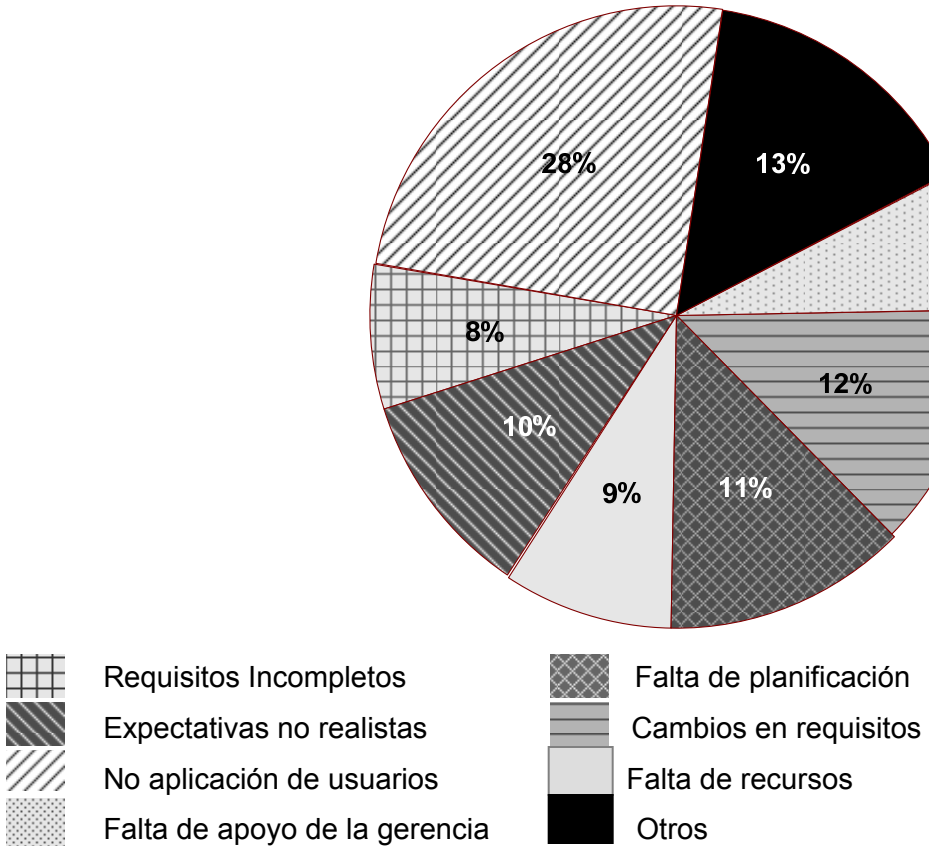


Fig 1. Resultados de Investigaciones realizadas

➢Desde 1994 la firma de investigaciones *The Standish Group* (www.standishgroup.com) realiza bianualmente el llamado *CHAOS Research Study*, sobre el número de proyectos que culminan exitosamente y los que se quedan en el camino.

➢El estudio más reciente, de marzo de 2003, mostró los siguientes resultados: 34% de los proyectos fueron culminados exitosamente. 51% estaban operando, pero excedieron el tiempo de entrega y el presupuesto previsto, además de no cumplir con las características y funciones especificadas inicialmente. El 15% restante fueron proyectos cancelados antes de culminar.

➢Las causas que provocan el fracaso de un proyecto son diversas. *Standish* logró identificar, en 1998, los 10 componentes más influyentes que causan el fracaso en un proyecto. Entre ellos, los factores de "la no integración de los usuarios", "la falta de apoyo ejecutivo" y "el cambio incontrolado en los requisitos".

1.2 Actualidad y necesidad del trabajo

En la Universidad de las Ciencias Informáticas, los proyectos de producción se encuentran en un estado de formación y desarrollo, vinculada intensamente a la producción de un alto volumen de productos, siendo necesaria la aplicación de excelentes procesos tanto de la Gestión de la Configuración así como de los demás roles de los proyectos.

Los proyectos productivos son uno de los aspectos importantes que se realizan en la Universidad de las Ciencias Informáticas, y que datan desde el primer año(2001-2002), proyectos como Zafre, Registro de Notaria, Identidad, Multimedia, CICPC (existe alrededor de 166 actualmente), son algunos de los proyectos que se han desarrollado y se desarrollan en la UCI, siempre teniendo en cuenta que es muy importante el proceso de investigación, y no aislar conceptos y partes del mismo, a elaboraciones secundarias dando la prioridad a otras, sino que el cuerpo del proyecto debe ser secuencial, usando los procesos correspondiente, y que exista una buena comunicación, para lograr el éxito del proyecto.

Varias son las necesidades identificadas en los procesos de Gestión de la Configuración, entre las cuales se puede mencionar la importancia de controlar el cambio, y con este, opciones importantes, que permitan

registrar quienes realizan los cambios, y sobre los mismos tener un grado de accesibilidad, desde el punto de vista de cada una de las modificaciones y de las posibles acciones a realizar. Esto siempre justificado con la necesidad de controlar los cambios, pues en un proyecto informático donde participan alrededor de 100 estudiantes, el uso de herramientas para el Control de Cambio permitirá controlar y distribuir el trabajo, y llevar un seguimiento de la evolución del producto.

1.3 Metodología utilizada: Rational Unified Process

Para lograr el éxito de un proyecto, en este caso CICPC, fue necesario definir la metodología que ha conducido al mismo por el camino más óptimo. Con la gran variedad de modelos y metodologías, es difícil reconocer cuál escoger, cuál se adaptará mejor a las necesidades que se requieren, por lo cual es necesario primeramente definir el alcance y las características que posee el proyecto CICPC.

Se partió analizando que el trabajo a realizar sería efectuado entre los países de Venezuela y Cuba, lo cual indica que son dos naciones que se encuentra a una distancia de gran consideración; lo cual indica que no se tendrán contactos sistemáticos con los clientes, por lo que los mismos serán de forma periódica, permitiendo el intercambio de información por ambas partes. El proyecto debe ser entregado en un plazo fijo, y para su realización se dispone de un presupuesto especifico, esto hace imprescindible de una planificación general de todos los desarrolladores; lo cual permitirá tener un control de la calidad y el avance con que se desarrolla el proyecto. Teniendo en cuenta estos elementos los Analistas del proyecto CICPC seleccionaron como la metodología a utilizar a *Rational Unified Process*, cuyas características descritas anteriormente hacen que se asemejen a las necesidades que describen a este proyecto. El proyecto CICPC se basa en esta metodología para realizar todas las actividades de Gestión de la Configuración del Software pues con la aparición de la metodología RUP, se ha logrado centrar los ciclos de vida de una aplicación y su desarrollo sobre mecanismos estándares que han ayudado a organizar mejor los proyectos de Sistemas.

El **Proceso Racional Unificado (**RUP), es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis,

implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado.

Sus principales características son:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende Implementa las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

El RUP es un producto de Rational (IBM). Se caracteriza por ser **iterativo e incremental** pues se basa en la evolución de prototipos ejecutables que se muestran a los usuarios y clientes. En el ciclo de vida iterativo a cada iteración se reproduce el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Está **centrado en la arquitectura**, pues la arquitectura de un sistema es la organización o estructura de sus partes más relevantes. Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades. RUP establece refinamientos sucesivos de una arquitectura ejecutable, construida como un prototipo evolutivo.

Guiado por los casos de uso.

Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- Inicio: Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos
- Elaboración: Se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos
- Construcción: Se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario
- Transición: Se implementa el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

1.4 La Gestión de la Configuración como disciplina

La integridad de un producto software depende de la acción combinada de tres tipos de disciplinas:

- Desarrollo
- Gestión
- Control

Dentro de las disciplinas de control se encuentra la Gestión de la Configuración del Software(GCS), cuyo objetivo es mantener la integridad de los componentes del producto software, evaluar y controlar los cambios sobre ellos, y facilitar la visibilidad del producto. Esta disciplina está presente desde el comienzo del proyecto, en cada una de las etapas y aún cuando ya se comience a comercializar, mientras tenga vida útil.

Los horrores que la Gestión de Configuración de un Software resuelve:

El software perdido: "sé que lo escribí, pero no dónde lo puse..."
Enlaces desaparecidos: "solía funcionar, pero usa componentes que ya no están..."
Pisar el código de otro: desarrolladores que hacen distintos cambios en el mismo código sobrescribiendo su trabajo mutuamente
No hay botón deshacer: los nuevos cambios son peores, pero no se puede volver atrás...
¿Qué versión tiene el cliente? ¿A cuál corresponde el bug?

1.4.1 Qué es la Gestión de Configuración del Software

Según Babich "El arte de coordinar el desarrollo de software para minimizar…la confusión, se denomina gestión de la configuración. La gestión es el arte de identificar, organizar y controlar las modificaciones que sufre el software…la meta es maximizar la productividad minimizando errores." (BABICH 1996)

Otra definición y bastante sencilla es la brindada por Angélica Antonio "… disciplina, cuya misión es controlar la evolución de un sistema software" (ANTONIO 2001).

Para el Proceso Unificado del Rational (RUP) la GCS "describe la estructura del producto e identifica los elementos que lo constituyen y que son tratados como entidades que pueden ser puestas bajo control de versiones en el proceso de GCS. La GCS tiene que ver con la definición de la configuración así como la construcción, el etiquetado y recolección de versiones de los artefactos" (IBM 2001).

De todas las definiciones revisadas para esta investigación, (PAULK and WEBER 1999), (ANTONIO 2001), (Appleton, 2000), (PRESSMAN, ROGER S. 2000), (GERVÁS 2002) se ha considerado la más completa y a su vez concisa la de Ivar Jacobson, Martín Griss y Patrick Jonson en el libro "Software Reuse: Architecture, Process, and Organizartions for Busisness Success" en la que plantean :

"Gestión de Configuración: Proceso de soporte cuyo propósito es identificar, definir y almacenar en una línea base los elementos de software; controla los cambios, reporta y registra el estado de los elementos y de las solicitudes de cambio; asegura la completitud, consistencia y corrección de los elementos; controla, almacena, maneja y libera los elementos asociados al producto de software". (JACOBSON *et al.* 1997).

La Ley Fundamental de la Gestión de Configuración define el papel de la Gestión de Configuración de Software en el proceso de desarrollo de la siguiente manera:

"La Gestión de Configuración es el fundamento de un proyecto software. Sin ella, no importa cuán talentoso sea el equipo, cuán grande sea el presupuesto, cuán robusto sean los procesos de desarrollo y prueba o cuán superior sean las herramientas de desarrollo técnicamente, la disciplina del proyecto colapsará y se perderá la posibilidad de triunfo. Haz bien la Gestión de Configuración, u olvídate de avanzar en el proceso de desarrollo de Software". (BROWN and MICHAEL 1998)

El objetivo fundamental de la GCS es intentar controlar el mundo irracional de los cambios, para lograrlo se debe:

1. Establecer y mantener la integridad de los productos generados durante un proyecto de desarrollo de software y a lo largo de todo el ciclo de vida del producto.
2. Evaluar y controlar los cambios sobre ellos, es decir, controlar la evolución del sistema software.
3. Hacer el producto visible a todo el equipo.

Son varios los autores que están de acuerdo con respecto a que los procesos asociados a la GCS que aparecen en el estándar de la IEEE son suficientes para conseguir sus objetivos principales, (GERVÁS 2002), (ANTONIO 2001), (IEEE 1990), (IEEE 1987). Para conseguir dichos objetivos se debe cumplir con un grupo de actividades (también llamadas actividades de autoprotección):

1- **Identificación de la Configuración**: En esta actividad se identificará la estructura del producto, sus componentes y el tipo de estos, y hacerlos únicos y accesibles de alguna forma.
2- **Control de Cambios en la Configuración**: Consiste en controlar las versiones y entregas de un producto y los cambios que se producen en él a lo largo del ciclo de vida.
3- **Generación de Informes de Estado**: Consiste en informar acerca del estado de los componentes de un producto y de las solicitudes de cambio, recogiendo estadísticas acerca de la evolución del producto.
4- **Auditoria de la Configuración**: Consiste en validar la completitud de un producto y la consistencia entre sus componentes, asegurando que el producto es lo que el usuario quiere.

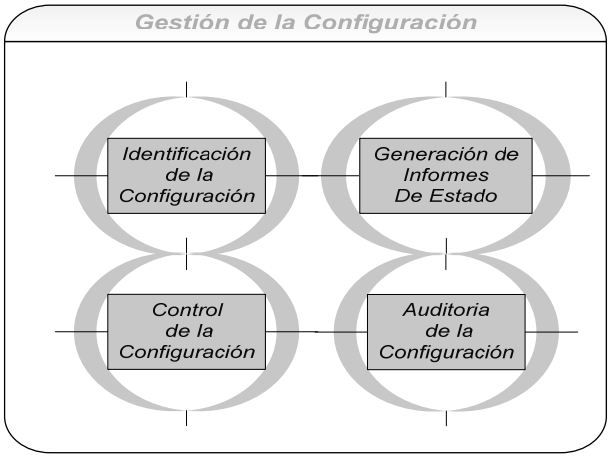


Fig. 2. Actividades de autoprotección.

1.4.2 Papel que juega el Cambio en el proceso de Ingeniería de Software

Una variable súper importante, que desarrolla un papel fundamental en el proceso de la Ingeniería de Software es el **Cambio** y por ende, dentro de la realización de un producto.

La primera Ley de la ingeniería de sistemas establece:
"Sin importar en que momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida". (BERSOFF *et al.* 1980).

Bersoff, expresa con gran claridad como el factor: "cambio", está presente a lo largo de todo el ciclo de vida del sistema. Pressman añade además que "los cambios son un hecho inevitable durante el desarrollo

de un software" (PRESSMAN 1998). "Es un hecho que los cambios durante el transcurso de un proyecto son inevitables. Ocurren, fundamentalmente, porque los involucrados en los proyectos con el tiempo incrementan sus conocimientos y aumentan o varían las necesidades o requisitos".(FEBLES 2000a), (FEBLES 2002), (FEBLES *et al.* 2000c), (FEBLES 2002).

El cambio debe ser asumido, no evitado. La mayoría de los cambios están justificados, ya que a medida que pasa el tiempo se sabe más acerca del problema y de cómo resolverlo.

¿Qué produce nuevos cambios en el sistema?:

- 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.

1.4.3 Elementos de Configuración del Software (ECS)

Como resultado de la aplicación del proceso de desarrollo del software, se obtienen y emplean a lo largo del ciclo de vida del mismo, un conjunto de elementos que pueden ser clasificados en ejecutables, fuentes, ficheros de datos, documentación, ayudas etc., los cuales vistos como conjunto son denominados como Configuración del Software (PRESSMAN 1998). "El ECS es la unidad de trabajo para la GCS" (ANTONIO 2001).

Cada uno de los componentes que designan el conjunto de toda la información y productos utilizados o producidos en un proyecto como resultado del proceso de Ingeniería de Software se le denomina: Elementos de Configuración del Software (ECS).

Como fuera anteriormente expuesto, varios autores concuerdan que una de las tareas básicas que forma parte de la GCS es la Identificación de la Configuración de Software (IBM 2003), (ANTONIO 2001), (NASA 1995), (IEEE 1990), (FEBLES 2004), (GERVÁS 2002), (IEEE 1987), llegando a catalogarla como la actividad más importante para el proceso de GCS (FEBLES 2004), (PRESSMAN 1997).

Una de las definiciones más clara sobre la Identificación de la Configuración de Software es dada por Angélica de Antonio (ANTONIO 2001), donde enuncia que "la Identificación de la Configuración de Software consiste en identificar la estructura del producto de software, sus componentes y el tipo de estos, y en hacerlos únicos y accesibles, de alguna forma".

En el Plan de la GCS del proyecto CICPC se identificaron un grupo significativo de ECS. **(Ver Anexo2).**

La selección de un número adecuado de ECS es muy importante. El tener demasiados puede provocar un número excesivamente elevado de especificaciones y documentos que al final resulta inmanejable. Sin embargo, el tener pocos ECS puede hacer que no se tenga la suficiente visibilidad sobre el producto. La Especificación del Sistema da lugar al Plan del Proyecto y a la Especificación de Requisitos Software. A su vez estos dan lugar a otros documentos. Si simplemente cada ECS produjera otros ECS, no habría prácticamente confusión. El problema aparece cuando entra en juego la variable ***CAMBIO***.

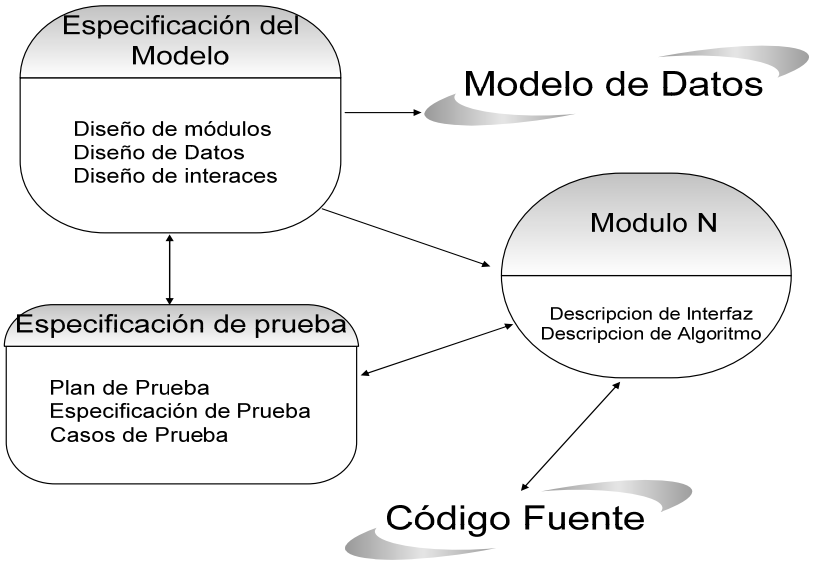


Fig 3. Objetos configuración

"Durante el proceso de desarrollo, el número de los ECS al igual que las relaciones que se establecen entre ellos crece a medida que va transitando el proyecto por el ciclo de vida, provocando confusión entre los involucrados en la realización del proyecto" (PRESSMAN 1998).

1.4.4 Línea Base

Son muchos los autores como Febles, Angélica Antonio, Pressman, así como en RUP que dentro de la Identificación de la Configuración de Software y en la GCS en general, abordan un concepto básico el cual es denominado: Línea Base.

En (PRESSMAN 1998) se plantea que " Línea Base es un concepto en la GCS que nos ayuda a controlar los cambios sin impedir que se realicen cambios que estén justificados".

El mismo autor define que en el contexto de la Ingeniería de Software, "una Línea Base es un hito, un hecho memorable en el desarrollo de un software que es señalado por la aprobación de uno o más elementos de configuración, donde la aprobación de los mismos es obtenida a través de una revisión técnica formal" (PRESSMAN 1997).

La IEEE (Instituto de Ingeniería Eléctrica y Electrónica**)**, (WIKIPEDIA 2005) 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."*

Por otra parte Angélica Antonio plantea que:" la idea de la creación de Líneas Base consiste en permitir cambios rápidos e informales sobre un ECS antes de que se pase a formar parte de una línea base, pero desde el momento en que se establece una línea base se debe aplicar un procedimiento formal para evaluar y verificar cada nuevo cambio propuesto". (ANTONIO 2001).

Analizando las diferentes definiciones antes expuestas se puede concluir en que existe un grupo de elementos comunes en las mismas, pues se toma la línea base como punto de partida para desarrollos

futuros dentro del proceso de desarrollo de software, que son establecidas al cumplir con un hito en el proyecto y que una vez creadas, los cambios sobre los ECS que la conforman se realizan bajo un proceso formal de control de cambio. Esta queda marcada por el envío de uno o más ECS y la aprobación de estos obtenidos mediante una revisión técnica formal (RTF).

En el Plan de la GCS del proyecto CICPC se determinaron las siguientes Líneas Base:

En la Etapa I del Sistema Policial, se establecerán las siguientes Líneas Bases:

- Al finalizar la Iteración I de Inicio. Línea Base(1)
- Al finalizar la Fase de Elaboración después de llevadas a cabo las 2 iteraciones concebidas para dicha fase, se establecerá una segunda Línea Base (2)
- Al finalizar la 1era y 2da iteración de la fase de Construcción de la primera versión Línea Base (3)
- Al concluir las pruebas de aceptación y pruebas piloto de la fase de Transición. Línea Base (5)

En la Etapa II se establecerán las Líneas Bases:

- Al finalizar la Iteración I de Inicio. (1)
- Al finalizar la iteración de Elaboración Línea Base (2)
- Al finalizar la 2da iteración de la fase de Construcción de la segunda versión Línea Base (3)
- Al finalizar la 4ta iteración de la fase de Construcción de la segunda versión Línea Base (4)
- Al concluir las pruebas de aceptación y pruebas piloto de la fase de Transición. Línea Base (5).

En el Portal Web, se establecerán las siguientes Líneas Bases:

En la Etapa I del Portal, se establecerán las siguientes Líneas Bases:

- Al finalizar la captura de requisitos del Portal. Línea Base (1).
- Al finalizar la primera versión del Portal Web. Línea Base (2).
- Al finalizar la segunda versión del Portal Web. Línea Base.

1.4.5 Auditoria de la Configuración

La Auditoria de la Configuración es una de las actividades de la GCS, que es desarrollada tanto por el equipo de administradores de la GCS, como por el personal de Calidad apoyando la actividad, y que da paso a las interrogantes de si el cambio se ha implementado correctamente.

Fig 4. Inquietud sobre las auditorias

Para asegurar que el cambio se ha implementado correctamente se debe tener en cuenta:

1) ***Revisiones técnicas formales:*** se centran en la corrección técnica del elemento de configuración que ha sido modificado. Los revisores evalúan el ECS para determinar la consistencia con otros ECS, las omisiones o los posibles efectos secundarios.

2) ***Auditorias de configuración del software****:* "Una auditoria de configuración del software complementa la revisión técnica formal al comprobar características que generalmente no tiene en cuenta la revisión"(ANTONIO 2001).

La Auditoria de la Configuración da respuesta a las siguientes preguntas:

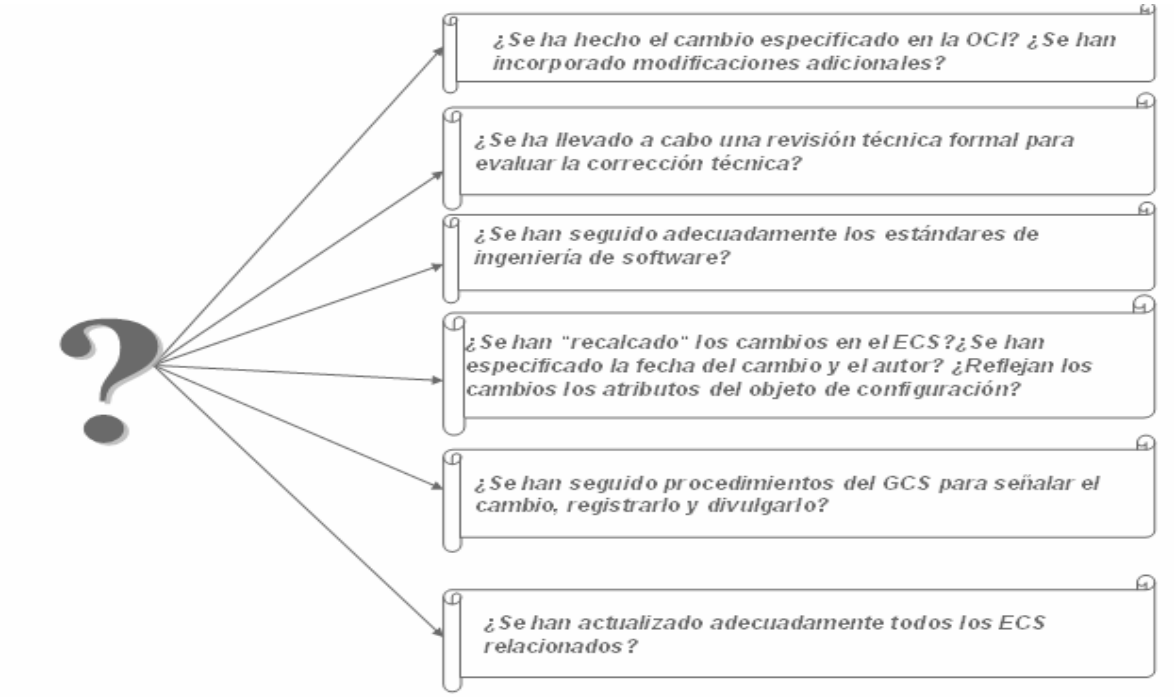


Fig 5. Interrogantes sobre la auditoria de la GCS

"Una auditoria es una verificación independiente de un trabajo o del resultado de un trabajo o grupo de trabajos para evaluar su conformidad respecto de especificaciones, estándares, acuerdos contractuales u otros criterios"(FEBLES 2004).

"La auditoria de la Configuración es la forma de comprobar que efectivamente el producto que se está construyendo es lo que pretende ser"(ANTONIO 2001).

Es por ellos que se concluye que es la actividad de GCS más costosa. Requiere de personal experimentado, y con un gran conocimiento del proceso de desarrollo. Sin embargo, debe ser realizada por personal ajeno al equipo de desarrollo técnico para mantener la objetividad.

1.4.6 Informes de Estado

La generación de informes de estado de la configuración es una tarea de GCS que responde a las siguientes preguntas:

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

"Consiste en un resumen del estado en que se encuentran todas las solicitudes de cambio registradas durante un determinado período de tiempo"(ANTONIO 2001). Por lo que ayuda también a mejorar los problemas de comunicación entre los participantes en un proyecto. Esto se consigue registrando toda la información necesaria acerca de lo que va ocurriendo y generando los informes necesarios.

1.5 El Control de Cambios y el Control de Versiones como casos particulares

Dentro de la GCS se pueden mencionar varios procesos, donde cada uno en su medida determinan la integridad y la evolución del software, y que son parte de los objetivos de la esta disciplina.

Control de Cambio: Según Angélica de Antonio:" es una de las actividades de la Gestión de Configuración más importante y su objetivo es proporcionar un mecanismo riguroso para controlar los cambios, partiendo de la base de que los cambios se van a producir. Normalmente combina procedimientos humanos y el uso de herramientas automáticas". (ANTONIO 2001).

Por lo general se establecen varios niveles de Control de Cambios: El cambios informal: Antes de que el Elemento de Configuración del Software pase a formar parte de una línea base, aquel que haya desarrollado el Elemento de Configuración del Software podrá realizar cualquier cambio justificado sobre él.

El Control de Cambio al nivel del proyecto o semi-formal: Una vez que el Elemento de Configuración del Software pasa la revisión técnica formal y se convierte en una línea base, para que el encargado del desarrollo pueda realizar un cambio debe recibir la aprobación de:

- El director del proyecto, si es un cambio local.

- El Comité de Control de Cambios, si el cambio tiene algún impacto sobre otros Elementos de Configuración del Software.

El Control de cambios formal: Se suele adoptar una vez que se empieza a comercializar el producto, cuando se transfieren los ECS a la Biblioteca Maestra. Todo cambio deberá ser aprobado por el Comité de Control de Cambios.

En un proceso formal o semi-formal, aparece una nueva figura en la Organización, el Comité de Control de Cambios (CCC). El Comité de Control de Cambios es una persona o grupo encargado de tomar las decisiones finales acerca del estado y la prioridad de las peticiones de cambio. Su obligación es tener una visión general del producto para poder evaluar el impacto de cada cambio en un determinado Elemento de Configuración sobre otros Elementos. En el proyecto CICPC el CCC está formado por todo el Equipo Técnico.

Control de versiones: “El control de versiones es la administración de múltiples revisiones de una misma unidad de información. Es más comúnmente utilizado en la ingeniería y desarrollo de software, para manejar el desarrollo en curso de documentos digitales tales como código fuente de aplicaciones, modelos y otro tipo de información clave que puede requerir del trabajo de múltiples personas o equipos de personas”. (WIKIPEDIA 2005).

Se llama control de versiones a los métodos y herramientas disponibles para controlar todo lo referente a los cambios en el tiempo de un archivo. Difícilmente un archivo de código o un documento de texto están terminados con la primera escritura; necesita cambios o reescrituras para corregir errores, modificar su contenido. A medida que el documento cambia existen dos opciones, mantener un historial de cambios o dejar que evolucione sin memoria. El control de versiones es un método estándar para mantener esta memoria haciendo además que sea útil para el desarrollo futuro. En documentos sencillos como un ensayo o un pequeño programa la memoria no es algo esencial, pero en la escritura de un libro o un programa con centenares de páginas y una docena de manos involucradas, no hay otra manera de trabajar. Esta es precisamente la palabra clave, mantener un control de las versiones de todos los archivos de un proyecto, es una manera de trabajar completamente de forma estandarizada; todas las prácticas tienen un nombre.

Esta actividad cobra importancia dentro de la GCS, pues es la que posibilita la gestión de versiones (revisiones) de todos los ECS que forman la línea base de un producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, junto a las posibles especializaciones realizadas para algún cliente específico. El control de versiones se realiza principalmente en la industria informática para controlar las distintas versiones del código fuente. Sin embargo, los mismos conceptos son aplicables a otros ámbitos y no sólo para código fuente sino para documentos, imágenes entre otros.

1.5.1 Características de los Sistemas de Control de Versiones

Un sistema de control de versiones debe proporcionar:

- Mecanismo de almacenaje de cada uno de los ítems que deba gestionarse (archivos de texto, imágenes, documentación...)
- Posibilidad de modificar, mover, borrar cada uno de los elementos.
- Histórico de las acciones realizadas con cada elemento pudiendo volver a un estado anterior dentro de ese historial.

Aunque no es estrictamente necesario, suele ser muy útil la generación de informes con los cambios introducidos entre dos versiones, informes de estado, marcado con nombre identificativo de la versión de un conjunto de ficheros, etcétera.

1.5.2 Terminología básica

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. 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. Este proceso se suele conocer como *check out* o *desproteger*.
- **Colaborativos**: en el que cada usuario se descarga la copia la modifica y el sistema automáticamente, mezcla las diversas modificaciones. El principal problema es la aparición de conflictos que deben 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*.

Cada uno de estos procesos, aún cuando dentro del Control de Cambio se encuentra el Control de Versiones, son procesos que al dirigir el estudio de las herramientas que actualmente los automatizan, se analizan por separado, aquellas que se pueden utilizar para el Control de Cambios que entre las cuales se encuentra el TRAC, Bugzilla,etc y las que su funcionalidad está basada en el Control de Versiones como el CVS “Concurrent Versions System“, SVN (Subversion), VSS (Visual SourceSafe) entre otros.

1.6 La Gestión de la Configuración y las herramientas

Uno de los principales retos de todo desarrollo y una parte importante del éxito de todo proyecto reside en elaborar un cuidadoso planteamiento de la metodología de trabajo en equipo, así como en la labor de la documentación realizada. Para minimizar el impacto que tiene en los proyectos, el descontrol y la incertidumbre a la que conlleva el cambio, se debe contar con herramientas que simplifiquen y hagan más eficiente el control de versiones.

En los últimos años ha habido grandes avances en disciplinas de apoyo al desarrollo de software, tal es el caso de la Arquitectura de Software y la Administración de Configuraciones. Estas disciplinas hacen grandes aportes al control de los proyectos y de los productos de software, a lo largo de su ciclo de vida, pero cada uno abarca una dimensión diferente del producto. Para el control de la integridad y consistencia del código se utilizan herramientas de Administración de Configuraciones, las cuales obligan al usuario

(desarrollador) a manejar el producto en términos de la representación interna de la herramienta y dejar de lado el mundo del problema.

En un mundo en constante desarrollo como el del software, donde la aparición de herramientas no cesa, resulta complicado en ocasiones decidir qué herramienta es la adecuada para cumplir ciertas tareas, en el caso de la gestión de proyectos de software, resulta bastante complicado elegir una debido a la gran cantidad de herramientas que se pueden encontrar, por ejemplo: *CVS, Subversión (SVN), Git, AccuRev, Aldon, Bazaar, ClearCase, Code Co-op, Codeville, CVSNT, Darcs, GNU arch, Mercurial, Monote, Perforce, SourceHaven, Star Team, Subversion, SVK, Visual Source Safe(VSS),Team Foundation Server, VSS, Trac, BonsaiCVS, Plastic SCM, SCM (Software Configuration Management).*

1.7 Comparación y Selección de las herramientas existentes

Desde orígenes primitivos los hombres han buscado dispositivos artificiales que faciliten su trabajo ayudándole a resolver los problemas que resultan cotidianos. Esto ha evolucionado hasta el hombre actual, con sus nuevas necesidades a lo largo de los años y la creciente revolución tecnológica en la que se encuentra el mundo, no queda exenta del uso de herramientas para facilitar la realización de diferentes actividades.

Como ha mencionado **Pressman**, son muchos los cambios que se viven en el desarrollo de un Proyecto Informático, tanto aquellos que se realizan por parte de los desarrolladores, como por los clientes, y otras veces por necesidades identificadas en cada una de las Fases por las que va transitando el software, siempre buscando obtener un producto que responda a las necesidades tanto del cliente como del Equipo de Desarrollo.

¿Cómo se especifican las versiones?

El método más habitual de asignar una versión a un documento es mediante un número o un grupo de números. No existe un modo fijo de numerar una versión, se deja al criterio de cada desarrollador. Existen ciertas prácticas habituales en la numeración como el uso de tres cifras o la numeración decimal.

1.7.1 Qué son los repositorios y las copias de trabajo

Todas las herramientas de control de versiones se basan en la típica comunicación cliente-servidor. Desde el punto de vista de los documentos se puede decir que el repositorio es el código que está en el servidor y la copia de trabajo en el cliente. Siendo entonces la copia de trabajo una imagen del contenido en el repositorio de código y lo utilizado para el trabajo diario (modificar archivos existentes, crear documentos nuevos...) Todos los cambios que se hagan en dicha imagen no son definitivos hasta que se suban al repositorio. El control de versiones se ve influido en gran parte por cómo se realiza esta comunicación entre copia de trabajo-repositorio. (WIKIPEDIA 2006b)

El lector puede plantearse la necesidad de duplicar los datos del proyecto. Esta arquitectura tiene la ventaja de que pueden existir tantas copias de trabajo como sean necesarias, normalmente tantas como participantes tenga el proyecto. Cada participante tiene su copia de trabajo y es el servidor el encargado de que los cambios aportados por los desarrolladores no entren en conflicto.

1.8 Herramientas para el Control de Versiones

Actualmente en el mundo existen un gran número de herramientas para el Control de Versiones que se encargan de la identificación de las distintas versiones de cada elemento, de mantener información acerca de las relaciones que hay entre ellas y de su almacenamiento en un repositorio.
Las herramientas de control de versiones o gestión de revisiones nos ayudan a crear, identificar y almacenar nuevas versiones, al mismo tiempo que se mantienen las versiones anteriores.

En cuanto a la creación de nuevas versiones, un extremo consiste en que toda modificación, por pequeña que sea, conduzca a la creación de una nueva versión. El problema que tiene esta opción es que el número de revisiones crece desmesuradamente, y puede llegar a ser imposible mantenerlas todas. Otra opción es dejar que sea el usuario el que decida cuándo se debe crear una nueva versión. El desarrollador puede modificar un objeto tantas veces como desee, manteniéndose el mismo identificador de versión, hasta que decida que dicha versión debe ser CONGELADA. En este momento la versión se

almacena y ya no puede volver a ser modificada. Si se necesita realizar nuevas modificaciones se debe crear una nueva versión, sobre la que se trabajará hasta que sea congelada. (WIKIPEDIA 2007b)

El modelo de trabajo sobre el que se basan la mayoría de las herramientas de gestión de revisiones es el de la figura:

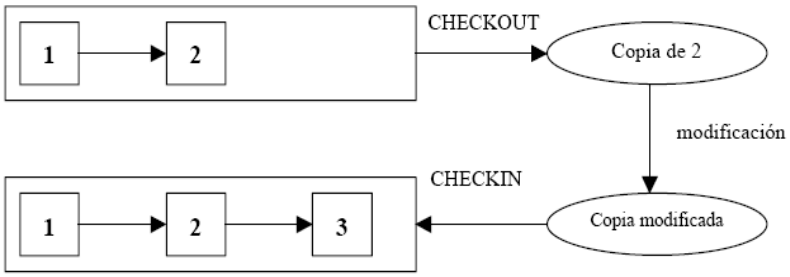


Fig 7. Modelo de Trabajo de los Repositorio

1.8.1 Análisis de las herramientas seleccionadas

En la Universidad de las Ciencias Informáticas en cada uno de los proyectos, existen algunas dificultades a la hora de la selección de las herramientas tanto para la Gestión del Software como para el Control de Versiones así como de los procesos que se desarrollan, teniendo en cuenta las principales tareas de los administradores de la GCS en los proyectos de producción.

Ante tal necesidad y dependiendo de las características identificadas del proyecto CICPC, se han definido algunas de las herramientas a partir de las cuales se realizaron los estudios que son: *CVS, Bazaar –ng, ClearCase, Visual SourceSafe, Subversión*, entre otras.

1.8.2 CVS

Concurrent Versions System (CVS), también conocido como *Concurrent Version System o Concurrent Versioning System* fue desarrollado por GNU, es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente

principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren. Sus desarrolladores difunden el sistema bajo la licencia GPL.

Esta aplicación tiene como limitaciones que los archivos en el repositorio sobre la plataforma CVS no pueden ser renombrados, estos deben ser eliminados y luego volver a agregarlos con el nuevo nombre. El protocolo CVS no provee una manera de que los directorios puedan ser eliminados o renombrados, cada archivo en cada subdirectorio debe ser eliminado y re-agregado con el nuevo nombre. Soporte limitado para archivos Unicode con nombres de archivo no ASCII.

CVS es capaz de producir versiones de archivo a cierto punto en el tiempo, es decir, si la versión mas reciente es 3.1.1, CVS puede producir la 1.4.1 o 2.4 del depósito, también es capaz de subjerarquizar archivos y otras funcionalidades utilizadas en proyectos que requieren control de versiones (WIKIPEDIA 2006a). (CHUIDIANG 2007)

1.8.3 Bazaar-ng

Es un sistema de control de versiones distribuido que, al igual que CVS o Subversion, nos permite guardar progresivamente los cambios que vayamos realizando sobre un conjunto de archivos de texto (habitualmente código fuente), recuperar versiones anteriores, mostrar diferencias, integrar el trabajo de diversos programadores, etc.

Sin embargo, a diferencia de CVS o Subversion, Bazaar-ng permite trabajar de forma mucho más flexible desde el típico esquema cliente-servidor hasta la descentralización de los repositorios, adaptándose al flujo de trabajo que queramos utilizar:

Bazaar-NG requiere que cada desarrollador especifique su nombre, e-mail para así registrar y vincular cada uno de los cambios que se realizan al código, lo que facilita que cualquier programador pueda ponerse en contacto con alguien que haya realizado un cambio determinado.

Este sistema tiene desventajas entre las que se pueden mencionar, que consume mas ancho de banda que los demás, y que para trabajar con el siempre se debe tener acceso al servidor remoto (STATION 2007).

1.8.4 ClearCase

"IBM Rational ClearCase es otra de las herramientas incluidas en la Suite Rational 2003. Ofrece una gestión de los activos de software para equipos de desarrollo de mediano y pequeño". (IBM 2005a).

Esta herramienta de control de versiones básica de gran fiabilidad, resulta ideal para equipos de proyectos medianos y pequeños y proporciona una gestión del ciclo de vida y control de los activos de desarrollo de software. Con un control integrado de versiones para archivos, directorios y otros activos de desarrollo. Configuración automática del espacio de trabajo con vistas rápidas, Soporte para desarrollo en paralelo, que incluye bifurcaciones automáticas, una gestión de línea base, además de las funciones necesarias para crear, actualizar, crear, ofrecer, reutilizar y mantener los activos más importantes del negocio. Los equipos con proyectos que no requieran de servidores distribuidos, replicación de base de datos, o acceso transparente a los archivos pueden comenzar con Rational ClearCase. A medida que el equipo crece y requiere de nuevas funciones, se puede migrar con facilidad hacia Rational ClearCase sin tener que volver a capacitar al equipo, cambiar los procesos, datos o herramientas. "Al igual que otras herramientas, Rational ClearCase se integra con el resto de la Suite Rational y también comparte un elevado costo, alrededor de unos $3319.38" (IBM 2005a).

1.8.5 Visual SourceSafe

Es un sistema de control de versiones que proporciona esta robusta funcionalidad de administración y configuración junto con capacidades de seguridad y funcionalidad para auditar. Es una herramienta muy flexible capaz de administrar virtualmente cualquier tipo de archivo creado con cualquier herramienta de desarrollo o productividad. Es un producto que, además de ser extremadamente sencillo de utilizar, está totalmente integrado en Visual Studio y trabaja sin problemas con documentos de cualquier fuente, incluidos los de MS Office.

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

Visual SourceSafe permite compartir archivos entre proyectos de forma rápida, eficaz y proporciona una serie de herramientas de mantenimiento de bases de datos, muy útiles que permiten que éstas funcionen de forma eficaz y segura. Admite el almacenamiento y la restauración mediante asistentes de uso sencillo, etc. Al combinar, se crea una versión en la base de datos que consiste en realizar un seguimiento de los historiales de los archivos y proyectos de la base de datos. Un historial incluye todas las versiones de un archivo o proyecto, desde la versión inicial a la actual, de la base de datos.

Debido al carácter propietario de la herramienta, y su procedencia, al ser un producto de Microsoft, corporación estadounidense, no puede ser comercializado con Cuba (MSDN 2007a), (MSDN 2007b).

1.8.6 Subversion

Subversion es un Sistema de Control de Versiones(SVN o svn) para el desarrollo y control del software de forma incremetal por el equipo de trabajo.
Subversion maneja archivos y directorios en el tiempo. Dichos archivos son puestos en un repositorio central. Este repositorio es como un servidor de archivos ordinario, con la salvedad de que recuerda todos los cambios realizados en los archivos y directorios que aloja. Esto permite mantener un historial de los cambios realizados en dichos archivos, y volver a versiones anteriores, en el caso de que sea necesario y tiene la posibilidad de ser accedido remotamente.

A partir del 2006, SVN ha sido una de las herramientas más usadas ampliamente, mucho mas que el CVS, y sigue incrementando y se plantea que es hoy quizás la mas popular alternativa. Los proyectos que usan el SVN incluyen el Apache Software Foundation, KDE, GNOME, GCC, Python, Samba, Mono y otros. Actualmente el SourceForge.net también proporciona SVN para proyectos de código abierto, y también el Código de Google y el sistema BountySource lo usan exclusivamente (COLLINS-SUSSMAN *et al.* 2004), (COLLINS-SUSSMAN *et al.* 2006), (FERRER and PASTOR).

1.9 Comparación entre las herramientas para el Control de Versiones.

Varios factores o características, miden aspectos importantes para cada una de las herramientas existentes, y que definen la selección. Además de las herramientas que se presentaron anteriormente existen otras tales como: Baazar, Aldo, AccuRev, Code Co-op, Codeville, entre otros tantos que se mostrarán en las tablas para tener un rango más amplio a la hora de comparar y llegar a una conclusión. (**Ver anexo 3**).

En la primera selección se descartan las herramientas que sean software propietario y se dejan solamente las libres. Las plataformas en la cual dichas herramientas deben tener soporte tienen que ser Linux y Windows, ya que en las PC servidores esta corriendo el Sistema Operativo (SO) Linux, sin embargo por la complejidad del mismo en las maquinas clientes aun el SO con el que se trabaja es Windows.

Las herramientas que cumplan con dichos requerimientos iniciales, seguirán un proceso de comparación según los parámetros siguientes:

- Información General y Técnica.
- Lenguaje de programación y protocolos de red.
- Interfaz de usuario.
- Otras características.

(***Ver anexos: 4, 5 y 6, respectivamente***).

Después de realizada la primera comparación siguiendo los requisitos iniciales (Softwares libre, con plataformas de soporte Windows y Linux), las herramientas que quedaron seleccionadas a investigar fueron: *Bazzar , CVS, Darcs, GNU arch, Subversion (SVN) .*

En un comienzo CVS era el sistema de control de versiones más utilizado que ha sido muy útil durante muchos años, pero tiene desventajas como por ejemplo: CVS no soporta mover o renombrar archivos o directorios conservando el historial; tampoco soporta copiar un archivo conservando este, las operaciones de escritura al repositorio (commits) no son atómicas, etc.) Esta herramienta como opción de trabajo en el

proyecto CICPC no seria muy conveniente tampoco. Para solucionar las deficiencias del CVS surge el SVN resolviendo cada limitante de la misma (OLEA and ARENAZA).

Sin embargo, el modelo de desarrollo centralizado de CVS y Subversion (que requiere un repositorio central al que los desarrolladores tengan acceso a través de la red) no es adecuado para muchos propósitos en dependencia del tipo de proyecto, es decir en el caso del proyecto CICPC no afecta, pues todas las PC están conectadas en red y lo que se busca es una herramienta con interfaz Web que permita centralizar el trabajo, pero si fuera el caso de un proyecto en el que se tengan varias PC que usen para desarrollo y que no tengan necesariamente una conexión permanente a la red, estas herramientas no serian factibles(WIKIPEDIA 2007e). Se han desarrollado por ello, sistemas de control de versiones distribuidos, que no requieren un repositorio central como por ejemplo: Arch, Mercurial, Monotone , Darcs, Bazaar etc. (Creado por el mismísimo Linus Torvalds para manejar el código fuente del núcleo de Linux); estas herramientas por lo general para Windows no son tan eficientes pues requieren de programas de instalación difíciles de obtener, de instalar, o porque para obtenerlos es necesario pagarlos, a continuación se explican algunas de las herramientas anteriormente mencionadas:

En el caso del **Bazzar** por ejemplo consume más ancho de banda que las demás herramientas, y para trabajar con él, siempre se debe tener acceso al servidor remoto esto es una desventaja enorme, por lo que esta herramienta no es la más óptima para el proyecto CICPC(STATION 2007).

Gnu ARch: Se destaca como ventaja su flexibilidad, que además se puede usar por http, ftp, ssh, sftp o WebDAV directamente. Es extremadamente pequeño, y no requiere ningún software de servidor. En su contra está que distingue entre un repositorio de servidor y uno de trabajo y, sobre todo, que para Unix/Linux es ideal pero aunque existen versiones para Windows dependen de Cygwin asÿ- los cuales son programas difíciles de instalar por el desconocimiento del trabajo con los mismos, la falta de documentación, además de requerir otros programas que son propietarios(WIKIPEDIA 2007c).

Darcs: Sigue el fundamento de que todo es un repositorio, así que resulta totalmente natural para él el trabajar off-line. Además sus comandos son extremadamente simples y fáciles de aprender y comprender. Al igual que Arch se puede dejar un repositorio disponible al público simplemente por http sin usar ningún

servidor extraño. En su contra tiene: que no es tan maduro como CVS o SVN, el ser el más lento de todos, a poca flexibilidad a la hora de enviar parches a otras personas. Pero el inconvenientes más connotado es el referente al uso de la memoria pues, al llegar a repositorios que cuenten con mas de doscientos cincuenta megas en ficheros y se desee hacer el Checkout inicial y quede todo el repositorio subido a la memoria, dicha herramienta se convierte en un gran problema, por otro lado la memoria física tampoco facilita mucho dicho proceso: por lo que se ha de tener una copia de todo el repositorio y de todo el historial (DARCSWIKI 2006b).

El análisis realizado arrojó como resultado de que cada proyecto, en dependencia de sus características, debe seleccionar las herramientas mas adecuadas, a través de un estudio minucioso y detallado de cada una. No se puede hablar de una mejor herramienta respecto a otra, todo depende de las necesidades de cada proyecto. Actualmente la herramienta que responde a cada una de las características propias del proyecto CICPC, y además a los aspectos tantos generales, como técnicos del mismo es el Subversion, por tanto es la herramienta que queda seleccionada para aplicarse ha dicho proyecto en el proceso de Control de Versiones(DARCSWIKI 2006a).

1.9.1 TortoiseSVN como cliente del Subversión

"*TortoiseSVN* es un inmejorable cliente gratuito de código abierto para el sistema de control de versiones Subversion" que maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un *repositorio* central. El repositorio es practicamente un servidor pero con la peculiaridad de que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios, lo que posibilita la recuperacion de versiones antiguas de ficheros, y examinar su historial, mucha gente piensa que Subversion, y los sistemas de control de versiones en general, son una especie de "máquinas del tiempo".

Lo que hace de TortoiseSVN tan buen cliente de Subversion son precisamente un grupo de características como por ejemplo :

- Sobreimpresión de iconos.
- El estado de cada carpeta y fichero versionado se indica por pequeños iconos sobreimpresionados.
- Fácil acceso a los comandos de Subversion:

- Todos los comandos de Subversion están disponibles desde el menú contextual del explorador.
- TortoiseSVN añade su propio submenú allí.

TortoiseSVN tiene varias opciones que vale la pena comentar, dada la necesidad que hay de control cada una de las acciones, más cuando el proyecto CICPC tiene más de 100 integrantes.

- Autentificación: Si el repositorio al que intenta acceder está protegido por contraseña, aparecerá un Diálogo de autentificación.

- Comprobar Modificaciones: A menudo es muy útil saber qué ficheros ha cambiado y también qué ficheros han cambiado y confirmado los demás (PROGRAMACIÓN.COM), (TIGRIS.ORG 2006).

1.9.2 RapidSVN como cliente del Subversion

RapidSVN es un cliente multiplataforma, ideal para Linux que se integra con el SVN. Este provee una interfaz fácil de usar para las características de SVN y es sencillo para principiantes y a la vez avanzado para usuarios experimentados. Esta interfaz gráfica es hasta el momento la mejor opción y catalogada como una “excelente alternativa” para utilizar SVN desde Linux.

En Windows el TortoiseSVN brinda una mayor cantidad de opciones, entre las cuales se puede mencionar: Obtener, Actualizar, Adicionar, Eliminar, Renombrar, Auditoria, Ver estados recurrentes entre otras opciones.

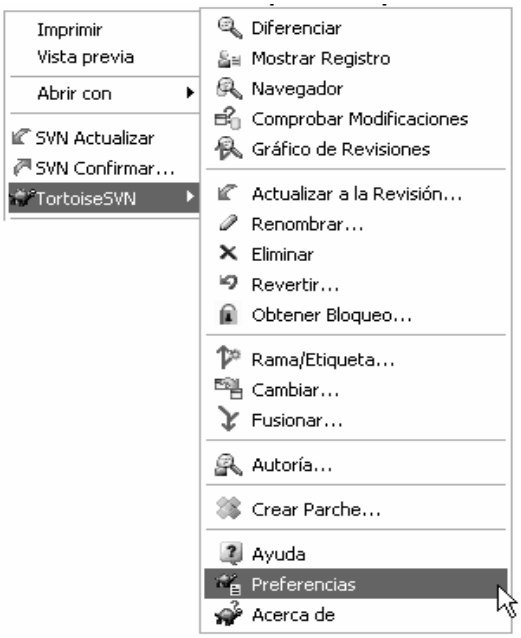


Fig 8. Opcines del ToritoiseSVN sobre Windows

A diferencia del RapidSVN que no brinda opciones como Ver estados recurrentes, Autoría, Preference y que tiene opciones como: Estado, Diff (muestra las diferencias entre dos versiones de un mismo archivo), Registro (Muestra las entradas de un registro para cada una de las revisiones del mismo), Confirmar, con las cuales cuenta el TortoiseSVN, entre otros opciones:

Fig 9. Opciones del RapidSVN

Otro ejemplo que demuestra la superioridad del TortoiseSVN al RapidSVN para Windows, es en el aspecto de la seguridad:

Para confiar cambios dentro de RapidSVN, se da clic derecho sobre el archivo o el directorio deseado y se selecciona el comando confiar, luego aparece una ventana de petición de entrada del registro, para el cambio en particular, al contrario del RapidSVN, el TortoiseSVN no asume automáticamente que se desea confiar todos los archivos pendientes (se debe Seleccionar un directorio para confiar en clientes, lo que implica que propone confiar los cambios realizados a todos los archivos bajo ese directorio), la ventana del mensaje del registro aparece para seguir los cambios realizados a los archivos seleccionados. Sin embargo, una lista de comprobación de archivos cambiantes también aparece, de modo que se pueda seleccionar (o de-select) archivos individuales para confiar. Además, con doble clic en un archivo, seleccionando el comando diff en el TortoiseSVN se realiza una lista de comprobación de archivos y demuestra los cambios realizados, lo que no ocurre con el RapidSVN (UNIVERSITY 2005), .

En el proyecto CICPC las máquinas clientes que se utilizan trabajan sobre una plataforma de Windows. Por lo anteriormente definido, la herramienta que mejor se integra con este Sistema Operativo es el TortoiseSVN, quedando así seleccionado como el cliente que se utilizará para que se integre con el SVN.

1.10 Herramientas para la Gestión de la Configuración

La Gestión de la Configuración como ha definido Jim Rickabaugh, es una combinación de iniciativas encaminadas a mejorar y desarrollar en un producto, los procesos, servicios, así como a la documentación de la misma, que juega un papel fundamental en la reeingeniería comercial: "los cuatros procesos básicos necesitados en el CM (*Management Configuration*) son la Identificación, el Control, la Contabilidad de estados y la comprobación". (RICKABAUGH 1994)

Cada una de las cuatro tareas que se han definido y que se llevan el mayor peso dentro de la GCS, son tareas que han aumentado su importancia con el desarrollo de las TIC y de la Ingeniería del Software en el gran mercado de los programadores y de las grandes industrias del software. Como los estudios han demostrado, es muy importante llevar tanto un control sobre los cambios que se realicen sobre los ECS,

como de las auditorias, los informes que tiene a cargo la GCS y por ende que se mantenga y se lleve un control sobre el proyecto tanto desde el punto de vista de estadísticas, como referencias de las actividades cumplidas, en el desarrollo por cada una de las tareas. Por lo que surge la necesidad de utilizar herramientas que sustituyeran la forma manual en que se realizaban y que conllevaba a que no se cubrieran las verdaderas necesidades de la administración del proyecto, ya que se hacía de forma tediosa y que no se llegaba a un control total, además que dependía hasta de la cantidad de personas que formaran parte de los equipos de desarrollo, de la complejidad del proyecto, cantidad de trabajo, teniendo en cuenta que la Gestión de la Configuración está presente en cada una de las fases por las que transita un producto, por lo que surgió la necesidad de desarrollar herramientas que supliera el trabajo manual del hombre, con idea de simplificar sus tareas y de aumentar la calidad de los procesos que se llevaban acabo.

Actualmente existe un conjunto de herramientas que cubren estas necesidades y que se pueden medir en dependencia de sus características y de las necesidades reales de los proyectos. Entre estas herramientas se pueden mencionar el Gforge, CVSTrac, el Trac entre otras.

El término software para la administración de la configuración es un término que cubre muchos tipos de software incluso de planificación, dirección del presupuesto, asignación de recursos, software de colaboración, comunicación, dirección de calidad y documentación o sistemas de administración que se usan para tratar con la complejidad de proyectos grandes.

- Tareas que realizan las herramientas:

Panificación:

Una de las tareas mas comunes es la de planificar una serie de eventos, y la complejidad de esta tarea puede variar en dependencia de como se usen las herramientas. Es por ello que algunos de los desafíos comunes son:

- Eventos que dependen entre si y que lo pueden hacer de maneras diferentes.
- Los desarrolladores que necesitan adelantar el trabajo y que requieren recursos y que van a depender tanto de otras tareas así como de la planificación de los recursos.

- A la hora de trabajar con la incertidumbre de si la estimación de tiempo es la correcta para el término de las tareas asignadas, o cuando tienen fechas topes entre otras.

- El camino crítico calculado:

En varias situaciones, habrá un camino crítico o series de eventos que dependen de alguien en específico, y de quien la duración del mismo determinará la longitud del proyecto.

- Proporcionando información:

Las personas encargadas de la planificación del proyecto, necesitan proporcionar información a la dirección del proyecto así como al resto del equipo, como justificar el tiempo gastado. Entre dicha información se encuentra:

- Las listas de tareas para las personas así como de recursos para estas.
- Información acerca del tiempo global en el que deben de ser terminadas las tareas.
- Advertencias ante cualquier riesgo del proyecto.
- La información histórica acerca de progreso del proyecto, la real y lo que se planea.

Entre otros.

- Acercamiento a las herramientas:

Existen varios criterios en los que se pueden analizar las herramientas tales como desktop, Web, personal, único usuario, licencias, entre otros.

Existen cuatro grandes grupos en base a los cuales se analizaron las herramientas, y son:

. Aplicaciones desktop open source.

. Aplicaciones Web open source.

. Aplicaciones desktop comerciales

. Aplicaciones Web Comerciales.

Algunos ejemplos de las herramientas son:

Tabla 1. Herramientas *Open Source*

Open Source	
Desktop	**Web**
Kplato	dotProject
Open Workbench	Gforge
TastJuggler	GNU Savannah
	LibreSource
	Project.net(comercial)
	Trac
	Wrike
	Bugzilla
	eGroupWare

Tabla 2. Herramientas Comerciales

Comerciales	
Desktop	**Web**
Artemis	24SevenOffice
Cando projects	AceProject
Merlin	JIRA
Otros	Microsoft Office project Server
	Otros

1.10.1 Análisis de las herramientas para la Configuración del Software

Ante la situación que vive Venezuela que está emigrando a software libre por seguridad, las herramientas que se analizaron son las que están dentro del rango de herramientas con aplicaciones Web Open Source, ya que cumplen con ser libres y además tienen interfaz Web, para interactuar con los usuarios, en

este caso los administradores de la configuración. Los aspectos fundamentales que se analizaron en cada uno de ellos son Desarrollador, Licencia, Sistemas Operativos, Plataforma, así como las funcionalidades, entre otros.

1.10.2 dotProject

DotProject fue creado por dotmarketing.org en el año 2000, con el fin de construir una herramienta para la Gestión de Proyectos. “dotProject” está construido por aplicaciones de código abierto y es mantenida por un pequeño pero dedicado grupo de voluntarios. Es una aplicación basada en Web, multiusuario, soporta varios lenguajes y es software libre. Fue programada en PHP, utilizando MySQL como base de datos (aunque otros motores como Postgres también pueden ser utilizados). La plataforma que se recomienda para utilizar dotProject se denomina LAMP (Linux + Apache + MySQL + PHP). De todas formas, existen binarios para instalar dotProject bajo otros sistemas operativos tales como Microsoft Windows (NT, 2000, XP) y Mac.(DOTPROJECT 2005)

El grupo que desarrolla dotProject se basa en los siguientes puntos:

• Proveer a los usuarios de funcionalidad orientada a la Gestión de Proyectos.
• Construir una herramienta con una interfaz de usuario simple, claro y consistente.
• Ser de código abierto, libre acceso y utilización.

Descripción del Producto: dotProject es una herramienta orientada a la Gestión de Proyectos. Orientada para el desarrollo de un producto, cuya producción requiera de un conjunto de actividades o tareas que se desarrollen entre ellas en forma paralela o independiente.

La aplicación consta de un conjunto de entidades ordenadas jerárquicamente las cuales permiten brindar la funcionalidad del producto.

A continuación se mencionan las entidades más importantes de dotProject:
• Compañías: Entidades que agrupan proyectos, actividades y usuarios.

• Departamentos: Áreas dentro de las compañías, que permiten agrupar usuarios en dicho nivel.
• Usuarios/Contactos: dotProject cuenta con usuarios los cuales se han logueado en dotProject y han trabajado dentro del esquema de permisos que posea el rol de dicho usuario. Los contactos son usuarios especiales que asignados a un determinado proyecto han podido recibir por ejemplo: correo, actualizaciones y noticias pero no necesariamente han debido tener acceso al sistema dotProject.
• Proyectos: Es la entidad que contiene el grupo de tareas necesarias para desarrollar un determinado producto.
• Actividades: son las tareas asignadas dentro de un proyecto. Son los componentes sobre los cuales se controla: la duración, dependencias, recursos asignados y progreso. Las actividades deben de pertenecer a un único proyecto.
• Diagramas de Gantt: Permite ver en forma grafica las actividades ordenadas jerárquicamente, mostrando las dependencias y solapamientos de las mismas.
• Tickets: para administrar todos los problemas relacionados a un proyecto.

• Archivos: Permite almacenar archivos dentro de un proyecto permitiendo un versionado básico de los mismos.
• Foros: Permite la creación de foros de discusión dentro de cada proyecto para distribuir información y discutir temas relativos al proyecto del foro.
• Administración del Sistema: Contiene la actividades relacionadas a la administración de usuarios, roles y configuración del sistema.
• Recursos: Permite asignar recursos no humanos (oficinas, equipamiento, etc) a un proyecto.

Esta herramienta no es considerada como una de las mas óptimas debido a que el código original de la misma se considera dependiente a otras versiones particulares del DotProject, las cuales varían en dependencia del tipo de proyecto en el que se este trabajando y aunque se puede modificar con el objetivo de ser adaptado e integrado al sistema, este aspecto constituye una gran limitación, pues una vez instalada dicha herramienta, en caso de que la misma no responda a las necesidades y exigencias del proyecto al que se le aplique, no logrando adaptarse a las características o entorno de trabajo del proyecto, se debe entonces comenzar a modificar los módulos de la misma (DOTPROJECT 2005), (SENTIDOWEB 2006).

1.10.3 GNU Savannah

Es un proyecto de la Fundación de Software Libre, que sirve como un software colaborativo para la dirección de proyectos. Este ofrece CVS, servicios de e-mail, listas de correos, archivos organizados, rastreo de servicios, entre otras opciones de trabajo. Este corre sobre Savane, que es basado en el mismo software que utilizó VA Software para ejecutar el portal de SourceForge.

A diferencia del SorceForge el enfoque de Savannah es para organizar el software libre y tiene políticas de organización muy estrictas, incluso una prohibición del uso de formato no libre como Macromedia Flash, es por ello que al registrar un proyecto hay que informar que licencia de software libre utiliza.

En el 2004 después de un compromiso de seguridad y resignaciones entre la Savannah y el equipo de voluntarios, FSF anunció que este (ahora Savane) iba a moverse a Gforge por problemas técnicos sobre el código base y el mantenimiento, pero el equipo de voluntarios protestaron y no hubo plan para emigrar(WIKIPEDIA 2007c), (GNU 2006).

1.10. 4 El TRAC

Es un gestor de SVN (esta definición no es oficial) que permite mediante el uso de la Web conocer el estado del proyecto, ver los archivos, los cambios producidos sobre estos en las diferentes revisiones. Además permite crear *líneas de tiempo* basadas en *ticket* para una previa planificación, comprobar de forma visual cuando se ha llegado a un objetivo o versión determinada.

De forma general se pueden mencionar aspectos de la definición tales como: Herramienta Web, apoyo a la gestión de la configuración dentro del desarrollo de software, apoyo a la administración y planificación de procesos, sistema de seguimiento (Tracking System), y GPL entre otros.

Esta herramienta que brinda una plena integración con SUBVERSION, contenido orientado a componentes, Bug-Tracking, Wiki integrada, Línea de tiempo (Time Line), Diagrama Gantt para la

planificación, e Interfaz de administración para el control de versiones (Subversion) (TRAC 2007), (PROGRAMMERS 2005).

1.10.5 Project.net

Es una herramienta open source, de norma o estándares libre, esta es una aplicación tanto para los sistemas operativos Microsoft Windows y para Unix-like, es de uso libre y se fundó en 1999 para desarrollar aplicaciones de colaboración de proyectos que utilizan las tecnologías de Internet. En un primer momento, el enfoque inicial de la compañía esta construyendo y desplegando un artefacto de colaboración para el uso público y privado los cambios que se basen en la WEB. La licencia del mismo es vía Mozilla Public License o privada si lo prefiere el usuario (SOURCE 2006), (WIKIPEDIA 2007d).

1.10.6 EGroupWare

EGroupware es una solución de trabajo en grupo vía Web, de código abierto. Está escrita en PHP utilizando bases de datos, tales como PostgreSQL o MySQL. Incluye un calendario, una libreta de direcciones, un gestor de contactos, un cliente de correo electrónico IMAP, un InfoLog, funciones de CRM, un gestor de proyectos, un gestor de recursos, un gestor de ficheros, una planilla de tiempos, un wiki, una base de conocimiento y un motor de flujos de trabajo.

El eGroupWare es una empresa libre el software del groupware listo para su red. Permite manejar los contactos, las citas, etc. El eGroupWare es un servidor del groupware. Tiene como característica que se puede acceder a sus datos desde cualquier plataforma. También tiene la opción para acceder el servidor del eGroupWare con su cliente del groupware favorito.

El eGroupWare es internacional. En el momento, apoya más de 25 idiomas. Su plataforma es independiente. El servidor corre en Linux, Mac, Windows y otros sistemas operativos. En el lado del cliente, todos lo que se necesita es un internetbrowser como Firefox, Konqueror, Internet Explorer u otros (EGROUPWARE 2007),(SOFTWAREKM 2005).

1.10.7 Gforge

Gforge es una herramienta para el desarrollo de software en forma comunitaria que permite organizar y administrar gran cantidades de proyectos, proporciona un conjunto integrado de herramientas que facilitan el trabajo en colaboración, y, en concreto, la gestión de proyectos de software libre que pueden acceder a diversos servicios:

1. Proporciona un entorno configurable con control de versiones.
2. Herramientas para comunicación entre desarrollos y servidor Web por proyecto.
3. Permite a los miembros del equipo desarrollar una base de conocimiento compartida del proyecto.

Características del Gforge:

1. Técnicas: web, abierto, libre, escalable, robusto.
2. Funcionales: gestión, comunicación, coordinación, centralización (agrupa fuentes de información y homogeneiza su modo de acceso), control.
3. Utilidad de gestión de proyectos: tareas, incidencias, listas de correo, discusiones, documentos y repositorio de código y software.

Ventajas del Gforge.

- Permite centralizar y homogeneizar la gestión de proyectos.
- Es una página única.
- Se consigue aumento de productividad.
- Se tienen herramientas comunes a toda la empresa o departamento.
- Centralización de los recursos técnicos en un servidor (en vez de tener que soportar múltiples máquinas por proyecto).

Esta herramienta integra un conjunto de herramientas apropiadas para el desarrollo de software y la gestión de proyectos de software de código abierto, de una manera sencilla y con una infraestructura básica, sobre la que se apoya la construcción de comunidades de software libre. Puede considerarse, por tanto, un entorno de desarrollo colaborativo o una herramienta de gestión del proceso de desarrollo en el entorno de las comunidades de desarrollo de software de código abierto, más no realiza un control de

cambio a nivel individual de proyecto (ALTAMIRANDA), (MORFEO-PROJECT 2006), (TELEFÓNICA 2005).

1.10.8 LibreSource

LibreSource es un servidor Web en el que los usuarios se pueden personalizar en línea combinando los siguientes recursos: páginas del wiki, foro, archivos, áreas de transferencia directa, etc. Por otra parte, LibreSource incluye un sincronizador llamado: herramienta de gestión de LibreSource de la configuración de los programas que sea de colaboración, simple y unido. LibreSource también integra al subversión SCM, se basa en Java y utiliza la mayor parte de los servicios avanzados proporcionados por el ObjectWeb Jonas llamado servidor de aplicaciones.

Algunas de las características del sincronizador de LibreSource:

- Es tan portable como Java.
- Mejora el proceso de combinación. El sincronizador de LibreSource combina los sistemas de ficheros, el archivo de texto y los archivos de XML.
- Permite las redes de la sincronización, también llamadas Dataflows. Un espacio de trabajo se puede sincronizar con más de un sincronizador. Por esta manera, un cambio se puede propagar a partir de un sincronizador a otro.
- Proporciona atómico confían y se ponen al día.
- Proporciona la operación de la retitulación en archivos y directorios.
- Cada subsistencia del espacio de trabajo una copia de la historia. Diffs, deshace, invierte se puede hacer fuera de línea.
- De acuerdo con el modelo determinado del cambio. Los sistemas del cambio se almacenan en archivos de XML.
- Permitir saber la historia para cada línea de archivo de texto, cada entrada de un sistema de ficheros, cada nodo del documento del XML.
- Permite hacer parcial para confiar solamente actualización no parcial.
- Permitir seguir el cambio sin compromiso. Puedes hacer este fuera de línea.

- No permitir el por-archivo confían el mensaje. El sincronizador de LibreSource no permite parcial confía o actualización, así que confiar los mensajes están para los sistemas del cambio.
- Despliegue automático con comienzo del Web de Java
- Uso del protocolo del HTTP de conseguir a través de cortafuegos.
- Tener un eclipse enchufable.
- Fuente abierta con una licencia de QPL (Q Public License).

Una de las desventajas de esta herramienta es que todo software cubierto por esta licencia QPL es libre, e incluye en cierto modo todos los aspectos de la GPL, aunque son licencias incompatibles, pero tiene varias desventajas por las cuales casi nadie la utiliza, pues la QPL no permite correr software comercial sobre ella, además de que las modificaciones se deben distribuir separadas del software original, como parches, lo cual hace a esta licencia poco viable en la práctica. Por lo que esta herramienta no es prácticamente utilizada (LIBRESOURCE), (MURCIA).

1.10.9 Wrike

Wrike es una herramienta colaborativa para la creación y seguimiento de proyectos, que se integra con CVS. Esta aplicación no solo puede usarse mediante su interfaz Web, sino que también puede ser usada si es necesario o resulta más cómodo mediante un programa de correo electrónico, con solo tener en cuenta algunas reglas a la hora de introducir la información. Esta herramienta de administración de proyectos permite colaborar online con su equipo. No limita la cantidad de actividades de las cuales se puede llevar un registro. Se puede ingresar nuevas tareas vía e-mail simplemente agregando wrike@wrike.com a los destinatarios de su e-mail. No hay necesidad de descargar o instalar nada en su computadora. Wrike es de uso libre.

Tiene una interesante forma de gestionar las tareas compartidas entre un grupo. Se pueden asignar tareas a los participantes de un proyecto desde la propia Web o enviando un e-mail con write@wrike.com en la copia (CC). Los registros serán mostrados en los informes correspondientes con el estado y la fecha prevista de finalización de cada actividad (BITELINA) .

En fin, es un simple gestor de tareas online que permite la colaboración a otros usuarios, donde comparten las tareas, y a la vez se registran para poder participar, con lo que formarán parte de la lista de contactos para que en caso de crear otras tareas, ya se puedan seleccionar directamente (CHANG 2006).

1.10.10 Bugzilla

Bugzilla es una herramienta basada en Web de seguimiento de errores (bugs), (Bug Tracking System o BTS por sus siglas en inglés). Lanzado como software de código abierto por Netscape Communications en 1998, Bugzilla ha sido adoptado por una variedad de organizaciones para ser usado para el seguimiento de defectos (errores) tanto para software libre como para software propietario. Su licenciamiento es bajo la Licencia Pública de Mozilla.

Bugzilla permite organizar en múltiples formas los defectos de software, permitiendo el seguimiento de múltiples productos con diferentes versiones, a su vez compuestos de múltiples componentes. Permite además categorizar los defectos de software de acuerdo a su prioridad y severidad, así como asignarles versiones para su solución.

También permite anexar comentarios, propuestas de solución, responsables a quienes asignar la solución y el tipo de resolución que tuvo el defecto, todo ello llevando un seguimiento de fechas en las cuales sucede cada evento y, si se configura adecuadamente, enviando mensajes de correo a los interesados en el defecto.

Bugzilla utiliza un servidor HTTP (como puede ser Apache) y una base de datos (normalmente MySQL) para llevar a cabo su trabajo. Los errores pueden ser enviados por cualquiera, y pueden ser asignados a un desarrollador en particular. Cada error puede tener diferente prioridad y estar en diferentes estados, así como ir acompañado de notas del usuario o ejemplos de código que ayuden a corregir el error.

La noción de "error" en Bugzilla es muy general, por ejemplo, Mozilla.org lo utiliza también para registrar las peticiones de nuevas funcionalidades (BUGZILLA 2007), (WIKIPEDIA 2007a) .

1.10.11 Comparación de las herramientas

Luego de analizarse las herramientas más significativas para la GCS, se debe pasar a la selección de aquella cuyas funcionalidades y características se adecuen y respondan a las necesidades del proyecto CICPC.

Ya con anterioridad se había realizado la previa selección de las herramientas libres y que cumplieran con tener una interfaz Web, además de esto dicha herramienta debe cumplir con que corra sobre las plataformas de soporte de Linux y Windows, que además se integre a la herramienta seleccionada para realizar el Control de Versiones (SVN). Para realizar una primera decantación de aquellas herramientas que no cumplan con estos requerimientos, se utiliza la tabla:

Tabla 3. Comparación de las herramientas para la GCS

Herramienta	Licencia	Plataforma de Soporte(Linux & Windows)	Repositorio de Código fuente	Herramienta para la Administración de proyectos	Software Colaborativo de Varios Proyectos
DotProject	GPL(Libre)	Si	CVS, SVN	Si	Si
GNU Savannah	GPL(Libre)	Si	CVS-GNU	No	No
Project.net	Mozilla Public License(Libre) / privada	Si	CVS	Si	Si
EGroupWare	GPL(Libre)	Si	CVS, SVN	Si	Si
LibreSourse	QPL(Libre)	Si	SVN, LibreSource Synchronizet	No	No

TRAC	BSD(Libre)	Si	CVS, SVN	No	No
GForge	GPL(Libre)	Si	CVS, Subversion	Si	Si
Wrike	Creative Commons (Libre)	Si	CVS	No	No
Bugzilla	Mozilla Public License(Libre)	Si	CVS, Subversion, Perforce.	No	No
Clarity	Propietario	Si	VSS	No	No
FIT	Propietario	Si	Visual SourceSafe, Perforce, CVS	No	No

Las herramientas que cumplieron con los requerimientos definidos con anterioridad fueron:
DotProject, EGroupWare, LibreSourse, Trac, GForge, Bugzilla

A dichas herramientas obtenidas después de aplicar los criterios de selección iniciales, se aplicaron nuevos criterios comparativos con el objetivo de refinar el proceso de selección entre las mismas.

Tabla 4. Comparación de las herramientas seleccionadas para la GCS

Herramientas	**Evaluación y control de los cambios del producto.**	**Limitaciones Fundamentales para ser utilizada en el proyecto CICPC**
DotProject	No	-Código original se considera dependiente, varía en dependencia del tipo de proyecto.(los módulos se deben modificar cuando no responden a un proyecto)

		-Herramienta para la gestión de proyecto, no incluye un control de cambio a todo lo largo y ancho de un proyecto en específico.
EGroupWare	No	-No se integra al SVN -Poca flexibilidad
LibreSourse	No	-Licencia QPL -No permite llevar un control de usuarios. -No permite restringir el acceso público o las modificaciones a la documentación. -Herramienta para proyectos medianos y preferentemente pequeños.
TRAC	Si	-Herramienta creada específicamente para el control de cambios
GForge	No	Herramienta diseñada para la gestión de proyectos, por lo que no realiza un control de los cambios en los mismo,
Bugzilla	Si	.Concepto de "error" muy general

Como se ha analizado, la GCS es una disciplina de control dentro de un proyecto, por tanto uno de los requisitos fundamentales que no debe dejar de cumplir la herramienta seleccionada, es el de evaluar y controlar los cambios de los productos generados durante un proyecto de desarrollo de software y a lo largo de todo el ciclo de vida del producto, para de esta manera facilitar la visibilidad sobre el producto, independientemente de las demás opciones que brinde la herramienta.

Para hacerse una tercera decantación de herramientas, se debe tener en cuenta los aspectos comparativos de la tabla anterior. Del grupo de herramientas que inicialmente se tenía, se ha obtenido un número final de dos herramientas: TRAC y Bugzilla.

La herramienta Bugzilla, sistema de seguimiento de errores basado en web realizado en Perl y MySQL, que contiene una vulnerabilidad en la que puede permitir a usuarios remotos ejecutar comandos arbitrarios en el servidor web objetivo, tiene una limitante peligrosa que se debe tener en cuenta, ya que generaliza el concepto de error a tal punto de que esta herramienta considera el registro de nuevas peticiones y funcionalidades como “error”.

Por lo tanto la herramienta que queda seleccionada para que realice el Control de Cambio en el proyecto CICPC es el Trac, herramienta libre, que corre tanto en plataforma Linux como en Windows y se personaliza a cada proyecto, wiki integrada, que posibilita un seguimiento mucho más potente de los campos que se definen por el usuario, que ofrece un filtro de los mismos y se asocia e integra con la herramienta SVN; es capaz de manejar los cambios que en el proceso de desarrollo del sistema se presenta de una manera transparente y fiable, además de que provee una manera sencilla de capturar datos y crear logs para una más fácil auditoria y seguimiento del proyecto, a través de un exhaustivo control de los cambio, tributándose a una mayor eficiencia en la GCS, elevando así la calidad y rendimiento del producto.

Conclusiones

- Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable, por lo que la metodología a utilizar puede variar en dependencia de las características específicas de cada proyecto. Se logrará una mayor calidad y eficiencia siguiendo los procesos definidos en la metodología adecuada.
- Actualmente en la UCI se le resta importancia a la Gestión de la Configuración dentro del desarrollo de software, guiados por la falta de conocimiento sobre esta, y se debe conferir mayor importancia a los efectos que tienen los cambios incontrolados en las etapas por las que transita el producto.

- Las herramientas propuestas para la automatización de los procesos de la Gestión de la Configuración en el proyecto productivo CICPC son: el Trac para el GCS, Subversion para el Control de Versiones y el TortoiseSVN como cliente del SVN.

CAPÍTULO 2: ANÁLISIS DE LAS CARACTERÍSTICAS PRINCIPALES DE LAS HERRAMIENTAS SELECCIONADAS PARA EL PROYECTO CICPC

En la actualidad, los Sistemas de Control de Versiones son utilizados generalmente por personas y empresas que desarrollan software. Su funcionalidad permite llevar un orden de las versiones que van creando del producto, para así controlar quiénes pueden hacer modificaciones a los archivos, llevar un histórico de las versiones entregadas a sus clientes y trabajar en grupo. También, pueden usarlos personas que deseen llevar un control de los cambios realizados en sus archivos, ya sea de imágenes, documentos, hojas de cálculo.

En este capítulo se tratan temas tales como al trabajo de las herramientas en los procesos de la Gestión de la Configuración, de forma particular los procesos de Control de Cambios, además de abordarse aspectos del proyecto CICPC. También se presentan un estudio de las herramientas que se seleccionaron con anterioridad en el Capítulo 1, (Subversion para el Control de Versiones, TortoiseSVN como cliente del SVN, y el Trac para la GCS fundamentalmente para el proceso de Control de Cambios.

2.1 Proyecto CICPC

Ante la necesidad de tener las metodologías y herramientas más adecuadas, se planteó la necesidad de realizar un estudio y evaluar los resultados dependiendo de las características del proyecto CICPC el cual propone el desarrollo e implantación de un nuevo sistema policial para el **Cuerpo de Investigaciones Científicas, Penales y Criminalística de la República Bolivariana de Venezuela** (en lo adelante *CICPC*).

Como se definió en el Proyecto técnico del Proyecto “La solución de software será a partir del análisis detallado de las problemáticas existentes en la institución; así como la forma de desarrollo que se adoptará para llevar a cabo una respuesta personalizada” (SOTO 2006). Este es un proyecto con la República Bolivariana de Venezuela y que ha sido asignado por la dirección de la UCI a la Facultad 8.

Actualmente tiene una plantilla de alrededor 100 integrantes y está formado por 3 equipos de desarrollo. Los grupos que conforman la plantilla del proyecto CICPC son: de Calidad; de Gestión de proyecto; Base de Datos; Arquitectura; de Gestión de la Configuración (con cuatro administradores) y un grupo de Requerimiento.

En este proyecto se ha definido un Comité de Control de Cambios (CCC), compuesto por todo el Consejo Técnico, y dirigido por la Ing. Nilet Soto como líder del proyecto. Está concebido con una duración de un año y seis meses y junto a su solución se incluye un Portal Web, además de otros servicios. Está formado por estudiantes fundamentalmente de tercer y cuarto año en su mayoría y un grupo de profesores de la especialidad.

2.2 El uso de herramientas para el Control de Cambios

"El control de cambios es vital pero la fuerza que lo hacen necesario también lo hacen molesto. Nos preocupamos por el cambio porque una pequeña perturbación en el código puede crear un gran fallo en el producto, pero también puede reparar un gran fallo o habilitar excelentes capacidades nuevas. Nos preocupamos por el cambio, porque un desarrollador pícaro puede hacer fracasar el proyecto, sin embargo las brillantes ideas nacidas en la mente de esos pícaros, y un pesado proceso de control de cambios puede disuadirle de hacer un trabajo creativo" (BACH 1998).

Dentro de los procesos de la Gestión de la Configuración se realizan diferentes tareas y entre las fundamentales se encuentra el Control de Cambios. En el mundo se ha planteado que no existe ningún estándar, para el control informal o interno de los cambios, aunque sí hay algunas recomendaciones (IEEE STD 1042 Guide to Software Configuration Management). Es por ello que siempre se cumplirá la 1ra ley de la ingeniería de sistema: **"Sin importar en cuál momento del ciclo de vida estemos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida del proyecto"**

Se pueden mencionar como ejemplos los siguientes tipos de cambios:
- Corrección de un defecto: Los clientes tienden a clasificar todos los cambios en esta categoría.

- Mejora del sistema: Los programadores, sin embargo, los suelen clasificar aquí. Para solucionar el problema de determinar realmente de qué tipo es un cambio es importante la trazabilidad de los requisitos.

En un gran proyecto de desarrollo de software, el cambio incontrolado lleva rápidamente al caos. El control de cambios combina los procedimientos humanos y las herramientas automáticas para proporcionar un mecanismo para el control de cambio.

2.2.1 Representación de las herramientas en los procesos de cambios en el proyecto CICPC

En el Plan de la GCS del proyecto CICPC, se definieron los procesos de cambio tanto semi-formales, como formales. Los Semi-formales cuando se realizan pruebas internas al producto, que serán detectadas tanto por calidad y en un segundo momento cuando el cambio es sobre algún ECS que ya ha sido aprobado con anterioridad, es decir que forma parte de una Línea Base y que se relacione o dependa de otros ECS y los formales, aquellos que los inicialice la parte venezolana cuando se realicen las pruebas por parte de estos o antes una nueva necesidad.

En estos procesos también se hace uso de las herramientas seleccionadas como se muestra seguidamente para los cambios formales:

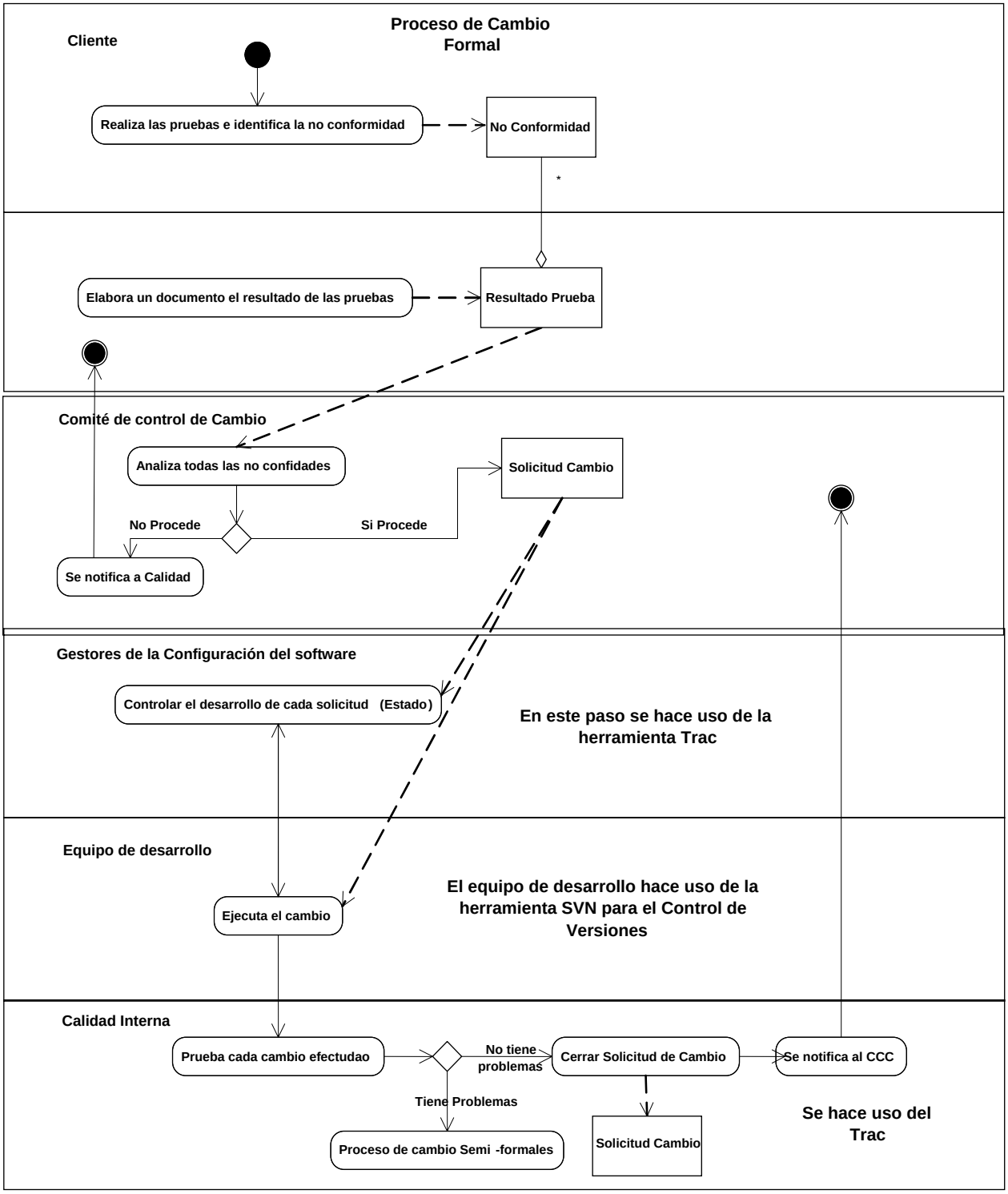


Fig 10. Uso de las herramientas en los procesos de cambio formal

Luego para los cambios Semi-formales:

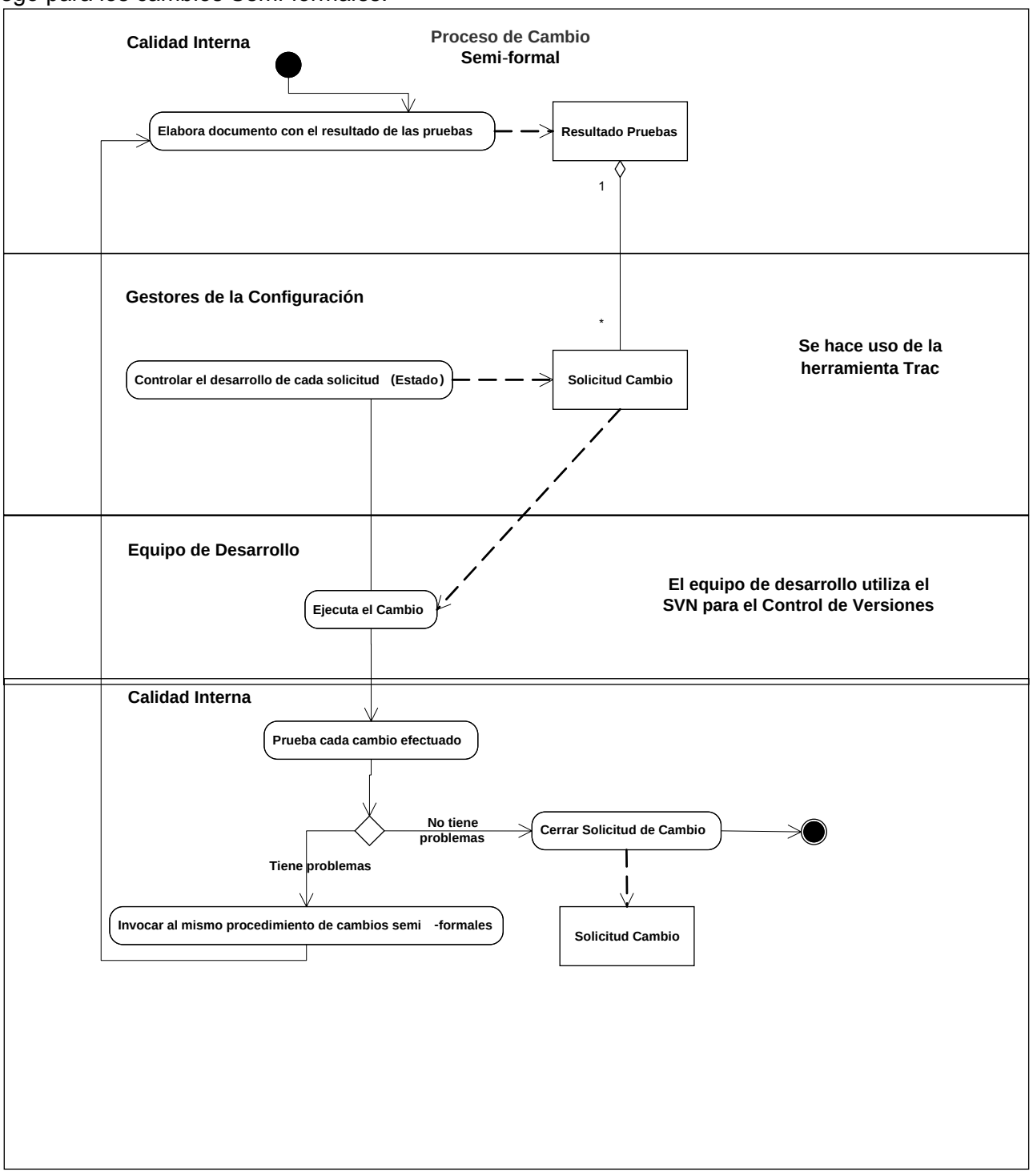


Fig 11. Uso de herramientas en los procesos de cambio Semi-formal

2.3 Funcionamiento del Subversion

En el Capítulo anterior se concluyó el capítulo con la selección de la herramienta Subversion, para el Control de Versiones, el desarrollo y control del software de forma incremetal para el equipo de trabajo. Esta herramienta también se puede representar a través de las siglas svn o SVN, por ser el nombre de la herramienta de línea de comandos.

Como se planteó en el Capítulo 1, Subversion maneja archivos y directorios en el tiempo, dichos archivos son puestos en un repositorio central. Este repositorio es como un servidor de archivos ordinario, con la salvedad de que recuerda todos los cambios realizados en los archivos y directorios que aloja. Esto permite mantener un historial de los cambios realizados en dichos archivos, y volver a versiones anteriores, en el caso de que sea necesario y la posibilidad de ser accedido remotamente.

2.3.1 Subversion en Acción

Una de las opciones que tiene el SVN son las Copias de Trabajo. Una copia de trabajo no es más que un árbol de directorios ordinario en el sistema local, conteniendo una colección de ficheros. Estos se pueden editar como se desee, y si son ficheros de código fuente, se puede compilar el programa de la forma habitual. La copia de trabajo es la propia área de trabajo privada, Subversion nunca incorporará los cambios de otra persona, ni hará que los cambios personales estén disponibles para los demás, a menos que el usuario lo pida explícitamente.

Después de que se haya hecho algunos cambios en los ficheros dentro de la copia de trabajo y se verifique que funcionan correctamente, Subversion provee de comandos para "publicar" los cambios para las demás personas que trabajan en el proyecto (escribiendo en el repositorio). Si los restantes miembros publican sus propios cambios, Subversion provee de comandos para fusionar esos cambios dentro de la copia de trabajo personal (leyendo desde el repositorio). Una copia de trabajo también contiene algunos ficheros extra, creados y mantenidos por Subversion, para ayudar a llevar a cabo esos comandos. En particular, cada directorio dentro de su copia de trabajo, contiene un subdirectorio llamado .svn, también conocido como el directorio administrativo de la copia de trabajo. Los ficheros dentro de los directorios

administrativos ayudan a Subversion a reconocer qué ficheros contienen cambios no publicados, y qué ficheros están desactualizados respecto al trabajo de los demás miembros.

Un repositorio típico de Subversion a menudo contiene los ficheros (o el código fuente) de varios proyectos; usualmente, cada proyecto es un subdirectorio en el árbol de ficheros del repositorio. Con esta disposición, una copia de trabajo de un usuario normalmente corresponderán a un subárbol particular del repositorio. Por ejemplo, se puede suponer que se tiene un repositorio que contiene dos proyectos de software.

El Sistema de Ficheros del Repositorio:

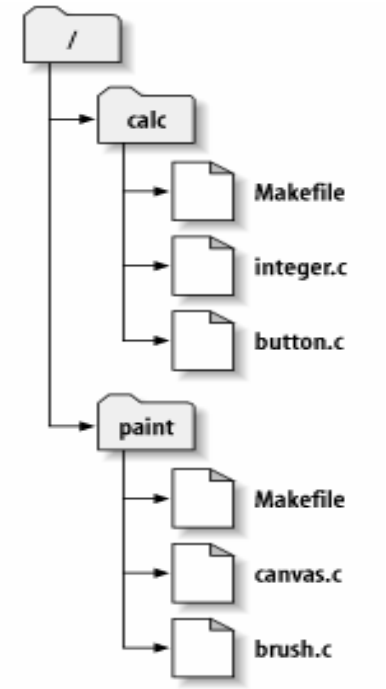


Fig 12. Sistema de Ficheros del Subversion

En otras palabras, el directorio raíz del repositorio tiene dos subdirectorios, paint y calc. Para obtener una copia de trabajo, primero se debe obtener algún subárbol del repositorio. (El término "obtener" puede sonar con que se tenga algo que ver con el bloqueo o la reserva de recursos, pero simplemente crea una copia privada del proyecto).

2.3.2 URLs de Repositorio

Los repositorios de Subversion pueden ser accedidos por muchos métodos diversos en discos locales, o a través de varios protocolos de red. La ruta de un repositorio es siempre, sin embargo, una URL. El esquema URL indica el método de acceso.

Tabla 5. URL de acceso al Repositorio

Esquema	Método de acceso
file://	Acceso directo al repositorio en el disco local o de red.
http://	Acceso utilizando el protocolo WebDAV a un servidor Apache configurado para Subversion.
https://	Lo mismo que http://, pero con encriptación SSL.
svn://	Acceso TCP/IP sin autentificación utilizando un protocolo personalizado a un servidor svnserve.
svn+ssh://	acceso TCP/IP autentificado y encriptado utilizando un protocolo propio a un servidor svnserve

En su mayoría, las URLs de Subversion utilizan la sintaxis estándar, permitiendo que se especifiquen nombres de servidor y números de puertos como parte de la URL. La URL entonces toma la forma file://nombredeequipo/ruta/al/repositorio. Para la máquina local, la parte del nombre de equipo en la URL debe estar o ausente o ser localhost. Por esa razón, las rutas locales normalmente aparecen con tres barras, file:///ruta/al/repositorio.

Además, los usuarios del esquema file: en las plataformas Windows necesitarán utilizar una sintaix “estándar” no oficial para acceder a los repositorios que están en la misma máquina, pero en una letra de unidad diferente de la unidad actual del cliente. Cualquiera de las siguientes sintaxis de URL funcionarán, donde X es la unidad en la que reside el repositorio:

file:///X:/ruta/al/repositorio

...

file:///X|/ruta/al/repositorio

...

Para publicar los cambios, se puede utilizar el comando de Subversion commit.

Actualemnte la ruta para acceder al Repositorio Central de SVN del proyecto CICPC es //10.35.11.31:5800/svn/Nombre del fichero o archivo. Con el objetivo de hacer mucho más facil el acceso al Repositorio, y por cuestiones de seguridad se ha planteado la necesidad de agregar dicho servidor al dominio UCI, para que se pueda acceder en la parte del host, por el nombre que le sea asignado, pues el nombre cambia menos que el número ip de los equipos. En caso de que se hagan cambios en el servidor de SVN y que influyan en el trabajo de los equipos de desarrollo, este se tendrá que informar con dos días de antelación.Cada una de las acciones que se tendrán que seguir, y la información inmediata de la nueva dirección que se tenga que comunicar a los miembros del proyecto CICPC sobre la nueva dirección del servidor, serán responsabilidad de los Administradores de la Configuración.

2.3.3 Revisiones

La operación commit dentro del SVN, es con la cual se pueden publicar los cambios de cualquier número de ficheros y carpetas como una única transacción atómica. En la copia de trabajo, puede cambiar el contenido de los ficheros, crear, borrar, renombrar, copiar ficheros y directorios, y luego confirmar el conjunto completo de cambios como una unidad.

En el repositorio, cada confirmación se trata como una transacción atómica: o bien todos los cambios de la confirmación se llevan a cabo, o bien ninguno de ellos se realiza. Subversion intenta retener esta atomicidad en caso de errores en el programa, errores del sistema, problemas de red, y otras acciones del usuario.

Cada vez que el repositorio acepta una confirmación, crea un nuevo estado del árbol de ficheros, llamado revisión. A cada revisión se le asigna un número natural único, un número mayor que la revisión anterior. La revisión inicial de un repositorio recién creado se numera como cero, y consiste únicamente en un directorio raíz vacío.

Una buena forma de visualizar el repositorio es como una serie de árboles, con una fila de números de revisiones, empezandose en 0, de izquierda a derecha. Cada número de revisión tiene un árbol colgando debajo, y cada árbol es una “foto” de cómo estaba el repositorio tras cada confirmación. (**Anexo 13**)

Los **Números Globales de Revisión** es una de las opciones que hace a SVN casi único ante la mayoría del resto de los Sistemas de Control de Versiones. Los números de revisión de Subversion se aplican a árboles completos, no a los ficheros individuales. Cada número de revisión selecciona un árbol entero, un estado particular del repositorio tras algún cambio confirmado, es decir la revisión N representa el estado del repositorio tras la confirmación N-ésima. Un ejemplo es que la "revision 5 de foo.c", realmente significa: "foo.c, que está en la revisión 5". Es importante tener en cuenta que las copias de trabajo, no siempre se corresponden a una única revisión en el repositorio; pueden contener ficheros de varias revisiones.

En el proyecto CICPC, cada uno de los integrantes de los equipos de desarrollo, tendrán asignados permisos que determinarán qué acciones podrán realizar y sobre cuáles ECS. Cada integrante del proyecto podrá acceder al repositorio Central y hacer una copia de trabajo personal que a partir de las necesidades de cada miembro, se tendrá que actualizar al finalizar el trabajo, si este se ha concluido y si ha sido revisado por el Jefe del Equipo, y al comenzar el trabajo. Cada uno de los permisos de trabajo serán dados por los Administradores de la GCS y asignados por parte de los Jefes de Sistemas que tendrán que informarlo 24 horas antes, de comenzar el trabajo.

Unión de las Copias de Trabajo al Repositorio

Por cada fichero en un directorio de trabajo, Subversion grabará dos piezas esenciales de información en el área administrativa .svn/:

- en qué revisión se basa su fichero de trabajo (lo que se denomina la revisión de trabajo)
- una fecha que indica cuándo se actualizó por última vez la copia local por el repositorio.

Con esta información, Subversion puede indicar en cuál de los siguientes cuatro estados se encuentra un fichero de trabajo:

Sin cambios, y actualizado:

El fichero no se ha cambiado en el directorio de trabajo, y no se han confirmado cambios a ese fichero en el repositorio desde su revisión de trabajo. Una confirmación de ese fichero no hará nada, y una actualización de ese fichero no hará nada.

Cambiado localmente, y actualizado:

El fichero ha sido cambiado en el directorio de trabajo, y no se ha confirmado ningún cambio a ese fichero en el repositorio desde su revisión base. Hay cambios locales que no se han confirmado al repositorio, por lo que al confirmar el fichero se conseguirá publicar sus cambios, y al actualizar el fichero no se realizará nada.

Sin cambios, y desactualizado:

El fichero no ha sido cambiado en el directorio de trabajo, pero ha sido cambiado en el repositorio. El fichero deberá ser actualizado en algún momento, para actualizarlo con la revisión pública. Un comando confirmar sobre el fichero no hará nada, y al actualizar el fichero se traerán los últimos cambios a su copia de trabajo.

Localmente cambiado, y desactualizado:

El fichero se ha cambiado tanto en el directorio de trabajo como en el repositorio. Un comando confirmar sobre el fichero fallará con un error "desactualizado". El fichero debería ser actualizado primero; al actualizar se intentará fusionar los cambios públicos con los cambios locales. Si Subversion no puede completar la fusión de una forma plausible automáticamente, le dejará al usuario la tarea de resolver el conflicto.

Subversion, CVS y otros sistemas de control de versiones utilizan un modelo *copiar-modificar-fusionar* como alternativa al bloqueo. En este modelo, el cliente de cada usuario lee el repositorio y crea una *copia de trabajo* personal del fichero o del proyecto. Luego, los usuarios trabajan en paralelo, modificando sus copias privadas. Finalmente, las copias privadas se fusionan juntas en una nueva versión final. El sistema de control de versiones a menudo ofrece ayuda en la fusión, pero al final la persona es la responsable de hacer que ocurra correctamente.

2.4 Funcionamiento y características del TortoiseSVN como Cliente del Subversion

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones *Subversion*. Este fue implementado como una extensión de la Shell de Windows y la Interfaz de usuario se encuentra en 28 lenguajes diferentes. Este maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un *repositorio* central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos.

2.4.1 Menú contextual para un directorio bajo el control de versiones con el TortoiseSVN

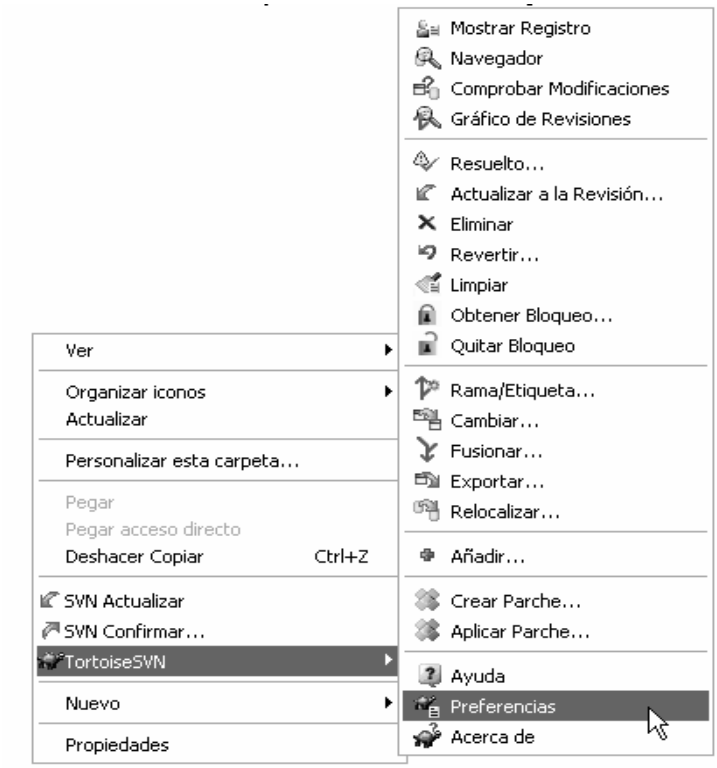


Fig 13. Ventana con las opciones del TortoiseSVN en Windows

Todos los comandos de TortoiseSVN se invocan desde el menú contextual del explorador de Windows. La mayoría se ven directamente, cuando hace click con el botón derecho en un fichero o una carpeta. Los

comandos disponibles dependen de si el fichero o la carpeta, o su carpeta padre, está bajo el control de versiones o no.

2.4.2 Obteniendo Información del Estado.

Sobreimpresión de Iconos

Una vez obtenida una copia de trabajo desde un repositorio de Subversion, se pueden ver sus ficheros en el explorador de Windows con los iconos cambiados. Ésta es una de las razones por las que TortoiseSVN es tan popular. TortoiseSVN añade lo que se llama un icono sobreimpresionado al icono de cada fichero que se superpone al icono original del fichero. Dependiendo del estado en Subversion del fichero, el icono sobreimpresionado es diferente. (**Anexo 14**)

 Una copia de trabajo recién obtenida tiene una marca verde como sobreimpresión. Esto significa que el estado de Subversion es normal.

 Cuando se empiece a editar un fichero, el estado cambia a modificado y el icono sobreimpresionado cambia entonces a una marca de exclamación roja. De esta forma se puede ver fácilmente qué ficheros se han cambiado desde la última vez que se actualizó la copia de trabajo, y que se necesita la confirmación.

 Si durante una actualización ocurre un conflicto, el icono cambia a un signo de exclamación amarillo.

 Si ha establecido la propiedad svn: *needs-lock* en un fichero, Subversion pone a ese fichero como de sólo-lectura hasta que obtenga un bloqueo en dicho fichero. Los ficheros de sólo-lectura tienen esta sobreimprseión para indicar que debe obtener un bloqueo antes de que pueda editar ese fichero.

 Si ha obtenido un bloqueo sobre un fichero, y el estado de Subversion es normal, este icono sobreimpresionado recordará que se debería liberar el bloqueo si no se está utilizando para permitir a los demás que puedan confirmar sus cambios al fichero.

 Este icono muestra que algunos ficheros o carpetas dentro de la carpeta actual se han marcado para ser borrados del control de versiones, o bien que falta un fichero bajo control de versiones dentro de una carpeta.

 El signo más (+) indica que el fichero o carpeta está programado para ser añadido al control de versiones.

Al contrario que TortoiseCVS (la integración shell de CVS) no se muestran iconos sobreimpresionados para los ficheros no versionados, porque el número de iconos sobreimpresionados está limitado por el sistema y debe economizarse su utilización. De hecho, puede ser que no se utilicen todos estos iconos en dependencia del SO. Esto se debe a que el número de sobreimpresiones permitidas por Windows está limitado a 15. Windows utiliza 4 de ellas, y las 11 restantes pueden ser utilizados por otras aplicaciones. Si también está utilizando TortoiseCVS, entonces no hay suficientes huecos de sobreimpresión disponibles, por lo que TortoiseSVN intenta ser un “Buen Ciudadano (TM)” y limita su uso de sobreimpresiones para darles una oportunidad al resto de aplicaciones.

- Normal, Modificado y En conflicto siempre se cargan y están visibles.
- Borrado se carga si es posible, pero se cambia por Modificado si no hay suficientes huecos.
- Sólo-lectura se carga si es posible, pero se cambia por Normal si no hay suficientes huecos.
- Bloqueado se carga sólo si hay menos de 13 sobreimpresiones ya cargadas. Se cambia por Normal si no hay suficientes huecos.
- Añadido se carga sólo si hay menos de 14 sobreimpresiones ya cargadas. Se cambia por Modificado si no hay suficientes huecos.

2.4.3 Viendo Diferencias

A menudo se necesita mirar dentro de los ficheros, para vera lo que ha cambiado. Para esto se selecciona un fichero que haya cambiado, y se selecciona el comando: *Diferenciar* desde el menú contextual de TortoiseSVN. Esto inicia el visor externo de diferencias, que comparará el fichero actual con la copia BASE, que se guardó tras su obtención o tras la última actualización.

Sugerencia:

Se puede mostrar diferencias incluso cuando no está dentro de una copia de trabajo, o cuando tiene múltiples versiones del fichero alrededor:

Se seleccionan los dos ficheros que se desea comparar en el explorador (por ejemplo, utilizando la tecla **Ctrl** y el ratón) y se selecciona *Diferenciar* del menú contextual de TortoiseSVN. El fichero que haya pulsado en último lugar (el que tiene el foco) será tomado como más nuevo.

2.4.4 Manejando datos en un Repositorio

Para llevar a cabo la opción Importar se debe de:

1. Quitar todos los ficheros que no se necesitan para construir el proyecto (ficheros temporales, ficheros que se generan por un compilador como los *.obj, binarios compilados, ...)
2. Organizar los ficheros en carpetas y subcarpetas. Aunque es posible renombrar/mover los ficheros más tarde, y es muy recomendable que tenga la estructura del proyecto antes de importarlo.

Posterior se debe seleccionar la carpeta superior de la estructura de directorios del proyecto en el explorador de Windows, y se hace click con el botón derecho para abrir el menú contextual. Se Selecciona el comando TortoiseSVN → Importar... y aparecerá un cuadro de diálogo (**Anexo 18**)

En diálogo le permite al usuario introducir la URL del repositorio en el que se quiere importar el proyecto. El mensaje de importación se usa como un mensaje de registro. Dado que normalmente acaba de empezar el proyecto, TortoiseSVN lo rellena con un mensaje importación inicial.

Importante:

Los ficheros o directorios que coinciden con el patrón de exclusión *no* se importan, es decir, tan pronto como se presione Aceptar, TortoiseSVN importa el árbol completo de directorios incluyendo todos los ficheros en el repositorio. El nombre de la carpeta que se está importando no aparece en el repositorio, sólo los contenidos de los ficheros. El proyecto ahora se almacenará en el repositorio bajo el control de

versiones. Se debe tener en cuenta que la carpeta que se importa ¡*NO* está bajo el control de versiones! Para obtener una copia de trabajo bajo el control de versiones se necesita Obtener la versión que importó.

Obteniendo una copia de trabajo

Para tener una copia de trabajo se necesita *obtener* una copia de un repositorio. Para esto se debe seleccionar un directorio en el explorador de Windows donde se vaya a poner la copia de trabajo. Se debe hacer click con el botón derecho para mostrar el menú contextual y seleccionar el comando TortoiseSVN → Obtener..., que mostrará un cuadro de diálogo **(Anexo 16).** Si se introduce un nombre de carpeta que no aún no exista, se creará un directorio con ese nombre.

Si se desea obtener sólo la carpeta superior y omitir todas las subcarpetas, se debe utilizar la casilla Sólo obtener la carpeta superior. Si el proyecto contiene referencias a proyectos externos que no desea que se obtengan al mismo tiempo, se tiene que utilizar la casilla Omitir externos. Es recomendable obtener únicamente la parte trunk del árbol de directorios. Si se especifica la ruta padre del árbol de directorios en la URL, al final se acabará con un disco duro lleno pues se obtendrá una copia del árbol completo del repositorio, incluyendo cada rama y etiqueta del proyecto.

2.4.5 Enviando los cambios al repositorio.

Para enviar los cambios que se le han hecho al repositorio se utiliza la opcion :*confirmar* los cambios. Pero antes se tiene que estar seguro de que la copia de trabajo está actualizada. Para esto se ejecutar TortoiseSVN → Actualizar directamente, o bien ejecutar TortoiseSVN → Comprobar Modificaciones primero, para ver qué se ha cambiado localmente o en el servidor. Si la copia de trabajo está actualizada y no hay conflictos, ya se está preparado para confirmar sus cambios. Se debe seleccionar los ficheros y/o carpetas que se deseen confirmar y se selecciona: TortoiseSVN → Confirmar.

2.4.6 El diálogo de Confirmación

El diálogo de confirmación muestra todos los ficheros cambiados, incluso los ficheros añadidos, borrados o no versionados. Si no se desea que un fichero cambiado se confirme, símplemente se desmarca ese

fichero. Si se desea incluir un fichero no versionado, se debe marcar para que sea añadido a la confirmación. (**Anexo 17**)

Si se ha modificado ficheros que han sido incluidos desde un repositorio diferente utilizando svn:externals, esos cambios no pueden ser incluídos en la misma confirmación atómica. Aparecerá un símbolo de advertencia bajo la lista de ficheros, que indicará si esto ha ocurrido, y el texto de ayuda explicará que esos ficheros externos deben confirmarse de forma separada.

Es importante que al introducir un mensaje de registro este describa los cambios que está confirmando. Esto ayudará a saber qué ocurrió y cuando, según se navega por los mensajes de registro del proyecto en el futuro. El mensaje puede ser tan extenso o escueto como se desee; muchos proyectos tienen directrices sobre qué debe incluirse en ellos, el idioma que debe utilizarse, y a veces incluso un formato estricto.

2.4.7 Actualizar la copia de trabajo con los cambios de otros

Periódicamente, se debe asegurar que los cambios que hacen los demás se incorporen en la copia de trabajo local (personal). El proceso de incorporar los cambios desde el servidor a la copia de trabajo local se conoce como actualización. La actualización puede hacerse en ficheros sueltos, en un conjunto de ficheros, o recursivamente en jerarquías completas de directorios. Para actualizar, se debe seleccionar los ficheros y/o directorios que se deseen, luego hacer click con el botón derecho y seleccionar TortoiseSVN → Actualizar en el menú contextual del explorador. Aparecerá una ventana con el progreso de la actualización según se va ejecutando. Los cambios que los demás miembros hayan hecho se fusionarán con los ficheros, guardando cualquier cambio que se hayan hecho en los mismos ficheros. El repositorio *no* se ve afectado por una actualización. (**Anexo 18**)

El diálogo de progreso utiliza un código de colores para resaltar diferentes acciones de actualización:

Púrpura

Nuevo ítem añadido a la copia de trabajo

Rojo oscuro

Ítem redundante borrado de la copia de trabajo, o ítem faltante reemplazado en la copia de trabajo.

Verde

Cambios del repositorio que se han fusionado satisfactoriamente con los cambios locales.

Rojo brillante

Cambios del repositorio fusionados con los cambios locales, pero que han dado lugar a conflictos que se deben resolver.

Negro

Items sin cambios en la copia de trabajo, actualizados con una versión más nueva desde el repositorio.

Si se obtiene algún conflicto durante una actualización (esto puede suceder si algún integrante del proyecto ha cambiado las mismas líneas del mismo fichero a la vez que otro lo ha hecho y esos cambios no concuerdan), el diálogo muestra esos conflictos en rojo. Se puede hacer doble click en esas líneas para iniciar la herramienta de fusión externa para resolver los conflictos.

TortoiseSVN también permite actualizar la copia de trabajo a una revisión específica, no sólo a la más actual. Este comando se llama TortoiseSVN → Actualizar a la revisión... y primero abre un diálogo donde puede introducir la revisión en cuestión. Por ejemplo: digamos que una copia de trabajo está en la revisión 100, pero se quiere dicha copia de trabajo refleje el estado que tenía en la revisión 50 - entonces símplemente se actualiza a la revisión 50. En ese diálogo, también se puede decidir si desea actualizarse sólo la carpeta actual (sin sus subcarpetas) y/o si desea ignorarse todos los proyectos externos en la actualización (por ejemplo, los proyectos referenciados utilizando svn:externals).

Atención:

Si se actualiza un fichero o una carpeta a una revisión en concreto, no debería cambiarse esos ficheros. Y se Obtendrán mensajes *out of date* (desactualizado) al intentar confirmarlos. Si se desea deshacer los cambios de un fichero y empezar de nuevo desde una revisión anterior, se debería utilizarse alguno de los siguientes comandos en vez de éste. El comando Menú Contextual → Revertir, los cambios de esta revisión del diálogo de registro deshará únicamente los cambios hechos en la revisión seleccionada. Los cambios que se hicieron después se mantendrán. Para revertir múltiples revisiones, se debería utilizar el

comando del menú TortoiseSVN → Fusionar..., donde se puede especificar el rango de revisiones que se desean deshacer.

La actualizacion de una revisión puede ser útil a veces , pues posibilita ver cómo esta un proyecto en un momento anterior en su historia. Pero en general, actualizar ficheros individuales a una revisión anterior no es una buena idea, ya que deja la copia de trabajo en un estado inconsistente. Si el fichero que se está actualizando ha cambiado de nombre, incluso se puede encontrar que ese fichero ha desaparecido de dicha copia de trabajo porque en esa revisión no había ningún fichero con ese nombre. Si simplemente se desea una copia local de una versión antigua de un fichero, es mejor utilizar para ese fichero el comando Menú Contextual → Grabar la revisión como... desde el diálogo de registro.

2.4.8 Múltiples Ficheros y/o Carpetas

Si se seleccionan múltiples ficheros y carpetas en el explorador y luego se selecciona Actualizar, todos esos ficheros/carpetas se actualizan uno a uno. TortoiseSVN se asegura de que todos los ficheros/carpetas del mismo repositorio se actualicen exactamente a la misma revisión. Incluso si entre esas actualizaciones ocurrió alguna confirmación.

El fichero local ya existe

A veces cuando se intenta actualizar, ocurren fallas y aparece un mensaje comunicando que ya existe un fichero local con el mismo nombre. Esto típicamente ocurre cuando Subversion intenta obtener un fichero recién versionado, y se encuentra un fichero no versionado del mismo nombre en su copia de trabajo. Subversion nunca sobreescribirá un fichero no versionado - puede contener algo en lo que está trabajando, y que casualmente tiene el mismo nombre de fichero que otro desarrollador ha utilizado para confirmar el fichero.

Si obtiene este mensaje de error, la solución será simplemente renombrar el fichero local sin versionar. Tras completar la actualización, se podrá comprobar si el fichero renombrado sigue siendo necesario. Si se siguen obteniendo mensajes de error, lo mejor es utilizar el comando TortoiseSVN → Comprobar

modificaciones, para mostrar todos los ficheros con problemas. De esa forma se pueden lidiar con ellos de un golpe.

2.4.9 Resolviendo conflictos

De vez en cuando, se obtendrán conflictos cuando se actualizan los ficheros desde el repositorio. Un conflicto ocurre cuando dos o más desarrolladores han hecho cambios en las mismas líneas de un fichero. Dado que Subversion no sabe nada del proyecto, delegará la resolución de los conflictos en los desarrolladores. Cuando se informa de un conficto, se debe abrir el fichero, y buscar líneas que empiezan con el texto <<<<<<<. El área conflictiva se marca así:

```
<<<<<<< nombre-del-fichero
   sus cambios
=======
   código fusionado del repositorio
>>>>>>> revisión
```

Además, para cada fichero en conflicto Subversion deja tres ficheros adicionales en su directorio:

nombre-del-fichero.ext.mine: Este es el fichero tal y como estaba en la copia de trabajo antes de que actualizara la copia de trabajo esto es, sin marcadores de conflicto. Este fichero tiene sus últimos cambios en él y nada más.
nombre-del-fichero.ext.rREV-ANTIGUA: Este es el fichero que era la revisión BASE antes de que actualizara la copia de trabajo. Esto es, el fichero que se obtuvo antes de empezar a hacer sus últimos cambios.
nombre-del-fichero.ext.rREV-NUEVA : Este es el fichero que el cliente de Subversion acaba de recibir desde el servidor del que se actualizó la copia de trabajo. Este fichero corresponde a la revisión HEAD del repositorio.

Para esto se puede o bien lanzar una herramienta externa de fusiones / editor de conflictos con el menú contextual TortoiseSVN → Editar Conflictos o bien utilizar otro editor manualmente para resolver el

conflicto. Debe decidirse cómo tiene que quedar el código, hacerse los cambios necesarios, y grabarse el fichero.

Posterior a esto, se debe ejecutar el comando TortoiseSVN → Resolver y confirmar sus modificaciones al repositorio. Se toma nota de que el comando Resolver realmente no resuelve el conflicto. Simplemente elimina los ficheros filename.ext.mine y filename.ext.r*, dejándo confirmar sus cambios.

Si se tiene conflictos con ficheros binarios, Subversion no intentará mezclar dichos ficheros por si mismo. El fichero local se mantendrá sin cambios y se obtendrá unos ficheros nombrefichero.ext.r*. Si se desea descartar sus cambios y quedarse con la versión del repositorio,se debe utilizar el comando Revertir. Si se desea mantener la versión y sobreescribir la versión del repositorio, debe utilizarse el comando Resuelto y luego se confirma dicha versión.

Se puede utilizar el comando Resuelto para múltiples ficheros si se pulsa con el botón derecho en la carpeta padre y se selecciona TortoiseSVN → Resuelto... Esto mostrará un diálogo con todos los ficheros en conflicto dentro de esa carpeta, y permitirá seleccionar cuáles marcar como resueltos.

2.4.10 Borrando, Renombrando y Moviendo

Al contrario que CVS, Subversion permite renombrar y mover ficheros y carpetas. Por tanto hay entradas de menú para borrar y renombrar en el submenú TortoiseSVN.

Si se borra un fichero/directorio utilizando TSVN, el fichero se elimina de su copia de trabajo y se marca para ser borrado. La carpeta padre del fichero muestra un icono sobreimpresionado "borrado". Siempre se puede recuperar de nuevo el fichero borrado, si llama a TortoiseSVN → revertir en la carpeta padre.

Si se quiere mover ficheros dentro de una copia de trabajo, puede usarse de nuevo el manejador de arrastrar-y-soltar:

1. se debe seleccionar los ficheros o directorios que se desean mover
2. se arrastrarán con el botón derecho al nuevo destino dentro de la copia de trabajo
3. se soltará el botón derecho del ratón
4. en el menú contextual seleccionar Menú Contextual → Mover ficheros en Subversion aquí.

No SVN Mover Externos:

No se deberían de utilizar los comandos TortoiseSVN Mover o Renombrar en una carpeta que ha sido creada utilizando svn:externals. Esta acción puede provocar que el ítem externo se elimine del repositorio padre, probablemente molestando a muchas otras personas. Si se necesita mover una carpeta externa, deberían moverse como se hace con el resto de ficheros sin versionar, y luego se ajustarían las propiedades svn:externals de las carpetas padres origen y destino.

Si se borra un *fichero* utilizando el Explorador en vez de usar el menú contextual de TortoiseSVN, el diálogo de confirmación muestra esos ficheros y deja eliminarlos del control de versiones antes de confirmar. Sin embargo, si se actualiza la copia de trabajo, Subversion se dará cuenta de que falta un fichero y lo reemplazará con la última versión del repositorio. Si se necesita borrar un fichero bajo control de versiones, utilice siempre TortoiseSVN → Eliminar para que Subversion no tenga que averiguar qué es lo que realmente quiere hacer usted.

Si se borra una *carpeta* utilizando el Explorador en vez de utilizar el menú contextual de TortoiseSVN, la copia de trabajo se romperá y no se podrá confirmar. Si se actualiza la copia de trabajo, Subversion reemplazará la carpeta que falta con la última versión del repositorio, y luego podrá se eliminada de la forma correcta, utilizándose TortoiseSVN → Eliminar.

Confirmar la carpeta padre:

Dado que las operaciones mover y renombrar se realizan como un borrado seguido de un añadir, se debe confirmar la carpeta padre de los ficheros movidos/renombrados para que se muestre la parte de borrado de la operación en el diálogo de confirmación. Si no se confirma la parte eliminada de los ficheros movidos/renombrados, se quedarán en el repositorio y cuando algún compañero la actualice no se eliminará el fichero antiguo, es decir, que se tendrá *ambas* copias, la antigua y la nueva.

Se *deben* confirmar los cambios de nombre de carpeta antes de cambiar cualquier fichero dentro de la carpeta; si no, la copia de trabajo realmente puede quedar estropeada.

2.4.11 Recuperando un fichero o una carpeta borrados

Si se ha borrado un fichero o una carpeta y ya se ha confirmado esa operación de borrado en el repositorio, entonces un comando TortoiseSVN → Revertir normal no lo podrá recuperar. Pero el fichero o la carpeta borrados no están perdidos para siempre. Si se sabe la revisión en la que se borraron el fichero o la carpeta (si no se sabe, se debe utilizar el diálogo de registro para averiguarlo), se abre el navegador de repositorios y se cambia a esa revisión. Luego se selecciona el fichero o la carpeta que se ha borrado, haciendo click con el botón derecho y seleccionando Menú Contextual → Copiar a..., y como destino de esa operación de copia se selecciona la ruta de la copia de trabajo.

Deshacer Cambios:

Si se desea deshacer todos los cambios que se ha hecho en un fichero desde la última actualización, se tiene que seleccionar el fichero, hacer click con el botón derecho para sacar el menú contextual, y luego seleccionar el comando TortoiseSVN → Revertir, de esta forma aparecerá un diálogo que muestra los ficheros que ha cambiado y que se pueden revertir. Se deben seleccionar los que se deseen revertir, posterior a esto se debe pulsa: Aceptar. (**Anexo 19**)

Limpieza:

Si un comando de subversion no puede completarse de forma correcta, quizás por problemas en el servidor, la copia de trabajo puede quedarse en un estado inconsistente. En ese caso se deberá utilizar el TortoiseSVN → Limpieza en la carpeta. Es una buena idea hacerlo en la rama superior de la copia de trabajo. La limpieza tiene otro efecto secundario muy útil. Si se han cambiado la fecha de un fichero pero no su contenido, Subversion no puede determinar si el fichero ha cambiado realmente excepto haciendo una comparación byte-a-byte con la copia prístina. Si se tiene muchos ficheros en este estado, comprobar su estado en Subversion será muy lento, lo que hará que muchos diálogos ralenticen su respuesta. Ejecutando una Limpieza en la copia de trabajo repararán esas fechas “rotas” y restaurarán las comprobaciones de estado a la máxima velocidad.

2.5 Quién cambió qué línea

A veces es necesario conocer no sólo qué lineas han cambiado, sino también exactamente quién cambió líneas específicas en un fichero. Entonces es cuando el comando TortoiseSVN → Autoría..., a veces conocido como comando de *anotar* tiene su utilidad (**Anexo 20**).

Este comando muestra, por cada línea en un fichero, el autor y la revisión en la que se cambió la línea. Para ver la autoría de *cada* revisión se debe poner el valor a 1. Por defecto, el fichero de autoría se ve utilizando *TortoiseBlame*, que remarca las diferentes revisiones para hacerlas más fáciles de leer. Si se desea imprimir o editar el fichero de autoría, se debe seleccionar Utilizar visor de texto para ver autorías. Una vez que se pulse Aceptar, TortoiseSVN empieza a recoger la información para crear el fichero de autoría. Esto se debe tener en cuenta: pues puede llevar varios minutos para completarse, dependiendo en cuánto haya cambiado el fichero y su conexión de red con el repositorio. Una vez que el proceso de autoría ha terminado, el resultado se escribe en un fichero temporal y se pueden ver los resultados.

2.6 Trac: herramienta para la Gestión de la Configuración

El TRAC es un gestor de SVN (esta definición no es oficial) que permite mediante el uso de la Web conocer el estado del proyecto, ver los archivos, los cambios producidos sobre estos en las diferentes revisiones. Además permite crear *líneas de tiempo* basadas en *ticket* para, previa planificación, comprobar de forma visual cuando se ha llegado a un objetivo o versión determinada.

De forma general se pueden mencionar aspectos de la definición tales como:

- Herramienta Web.
- Apoyo a la gestión de la configuración dentro del desarrollo de software.
- Apoyo a la administración y planificación de procesos.
- Sistema de seguimiento (Tracking System).
- GPL.

Utilidades:

- Plena integración con Subversion.

- Contenido orientado a componentes.
- Bug-Tracking.
- Wiki integrada.
- Línea de tiempo (Time Line).
- Diagrama Gantt para la planificación.
- Interfaz de administración para el control de versiones (Subversion).

En la base:

- Desarrollado en python.
- Se encuentra actualmente en la versión 0.10.2.
- Diferentes plataformas (Windows y Linux).
- Apache.
- Seguridad integrada.

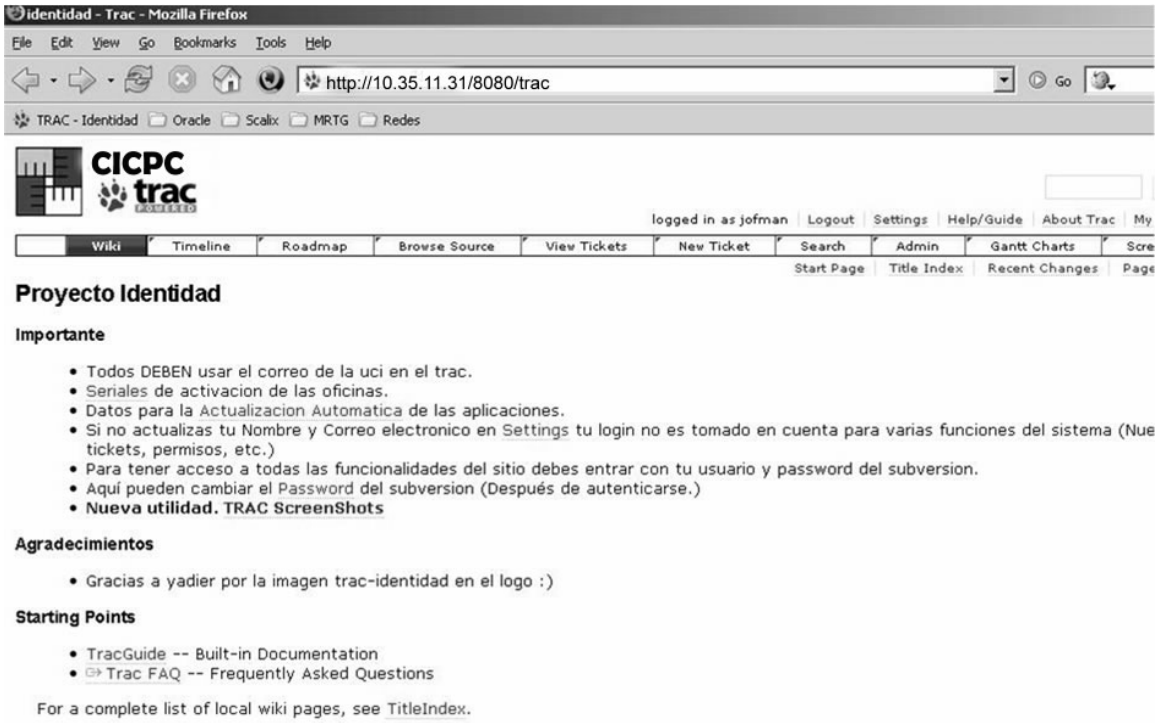


Fig 14. Interfaz gráfica del Trac

Trac es una wiki reforzada, que permite rastrear los problemas en los sistemas de aquellos proyectos de desarrollo del software o productivos. Este llega a imponer tan pequeño como sean posibles los procesos y políticas establecidas para los equipos de desarrollo.

Este provee una interfaz que se integra con el Subversión, una Wiki integrada y además convenientes y fáciles reportes. Trac permite un acercamiento mediante la Wiki a la descripción de los problemas y a obtener los mensajes, que se relacionen con los mismos, creando links y referencias entre los bugs, tareas, cambios introducidos, los archivos y las páginas de la wiki. También tiene disponible la Timeline (horarios), que muestran todos los eventos del proyecto en orden, por lo que se puede ir haciendo una adquisición o apreciación global del proyecto, y se puede llevar un seguimiento del progreso que se hace de forma fácil.

2.6.1 TracGuide (Guía). Punto de partida para la documentación del Trac

Los usuarios del Trac y la Guía de administración:
La Guía del Trac sirve como punto de partida para toda la documentación que tiene el Trac, con respecto al uso y desarrollo de este. Esta guía es un documento de licencia libre, es un esfuerzo colaborado y una parte del los proyectos que utilizan el Trac.

Tabla de Contenido:
TracGuide
Guía de Administración:

- Instalación del Trac (Cómo instalar el Trac y hacerlo funcionar).
- Actualización del Trac (Cómo actualizar la instalación existente).
- Administración del Trac (Cómo administrar los proyectos).
- TracImport (Importar los boletos o ticket de otras base de datos de bug).
- IniTrac (Configurar las referencia de los archivos).
- TracPermissions (Permisos del Trac, abordándose acerca del control de acceso y de permisos).
- TracInterfaceCustomization (Interfaz, Personalizar la Interfaz).
- TracPlugins (Instalar y manejar las extensiones del Trac).

- TracLogging (Anotar fácilmente).
- TracNotification (Notificación mediante e-mail)

Conclusiones

- El uso de herramientas en los procesos de Control de Cambio que han sido representados responden a las necesidades reales, y características del proyecto CICPC.
- Si se controlan los cambios en el proyecto CICPC combinando los procesos humanos y las herramientas automáticas, se evitará caer en el caos.
- El uso de la herramienta SVN en el proceso de Control de Versiones permite hacerlo más eficiente y tributar a un aumento de la eficiencia dentro del proceso.
- El Uso del TortoiseSVN como cliente del SVN integrado a la interfaz de Windows, constituye una mejora importante para los usuarios, ya que permite agilizar la interacción de estos con el Repositorio.
- Con el uso de la herramienta Trac desde la Web se logrará controlar el estado de los archivos y será más eficiente el control de cambio sobre los mismos.

CAPÍTULO 3: APLICACIÓN DE LAS HERRAMIENTAS SELECCIONADAS A UN CASO DE ESTUDIO. RESULTADOS DE LA INVESTIGACIÓN

En este Capítulo 3 se aplica el proceso de Investigación a un Caso de Estudio desarrollado en el proyecto CICPC, presentándose las experiencias del equipo del Portal Web, las Políticas de Integración del Subversion, así como las de Uso, que determinan la interacción con el servidor. Además del análisis de los objetivos propuestos, y las conclusiones, entre otros temas.

3.1 Portal Web en CICPC

En el Proyecto CICPC se han evidenciado problemas debido a la carencia de un punto de partida, que de una manera ágil y fácil, brinde información sobre el proyecto, así como la forma de acceder a la misma y a sus servicios, lo que provoca un desconocimiento del alcance del mismo, del producto ofrecido por el proyecto CICPC y para mantener actualizados a todos los integrantes.

Por otro lado una parte importante dentro de la estrategia comunicacional es precisamente el desarrollo de un Portal Web, que acompañe al software de gestión, a través de servicios adicionales que estén orientados al uso de la población en general, tanto venezolana como extranjera y que será parte de la solución.

Entre los objetivos que se buscan se encuentran: reflejar la imagen e identidad del Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República de Venezuela (CICPC), mejorar los niveles de información que actualmente las personas tienen de las labores que se realizan en la organización, tanto policiales como de relaciones inter-institucionales, abrir una nueva forma de comunicación con la población para escuchar sus inquietudes, sus sugerencias, y poder sondear de esta forma el estado de opinión con respecto al trabajo de la institución, entre otros, y en el que se brindarán diferentes servicios como información general institucional, localización geográfica y de contactos, información relacionada a la historia del CICPC (galería de directores, cambios más importantes realizados desde la creación de la institución, etc.), publicación de boletines y noticias de diferentes temas, entre otros servicios.

Todo lo anterior evidencia la importancia colosal que reporta el Portal Web al proyecto CICPC, ya que desde un punto de vista simple constituye la imagen del mismo, por lo que el equipo del Portal Web del proyecto CICPC, como parte de su preparación y ante una necesidad real de que los miembros del proyecto tuvieran un medio de información, realizó un Caso de Estudio, donde desarrollaron un Portal Web, para el cual hicieron uso de la herramienta SVN para el Control de Versiones, el TortoiseSVN como cliente del SVN y el Trac para la GCS pero solamente en el inicio del Caso de Estudio.

3.2 Repositorio SVN en el Portal Web

Toda aplicación requiere de un repositorio, en el caso del proyecto CICPC el repositorio utilizado es: SVN y como cliente el TortoiseSVN, que fueron las herramientas propuesta en el Capítulo 1 y analizadas en el Capítulo 2. Fue de gran importancia para el desarrollo del portal el uso del SVN, pues este mantuvo el servidor actualizado y disponible para todo el equipo.

3.3 Experiencias del Caso de Estudio del Portal Web

Entre las experiencias que se obtuvieron del uso de las herramientas en el Caso de Estudio del Portal Web, estuvieron que primeramente contaron con la herramienta SVN, la cual centralizó el trabajo, y a partir de las copias de trabajo personales, y las opciones de actualización y confirmación, los integrantes de los equipos pudieron contar siempre con la última versión del trabajo.

El uso del SVN propició que cada uno de los miembros del equipo, pudieran estar al día de los estados de sus archivos a partir de las opciones del TortoiseSVN con la sobreimpresión de iconos. Además de que tuvieron un fácil acceso al Repositorio de forma personalizada, haciendo más amena la interacción con el mismo.

El SVN también tuvo un gran aporte, al tratar cada una de las confirmaciones de forma atómica, permitiendo que los desarrolladores pudieran construir y confirmar cambios como una unidad lógica, por lo que cada vez que se buscó una versión anterior, podían ver todos los cambios realizados como una unidad lógica. También permitió restringir el acceso a las carpetas que se encontraban dentro del

Repositorio y que almacenaban información, código fuente del proyecto CICPC y que un mal cambio, traería problemas para el desarrollo continuo del mismo, lográndose a partir de opciones como autorización, autentificación, navegación del repositorio entre otras, al permitir especificar quiénes tenían accesos y a qué carpeta del repositorio.

Uno de los problemas surgidos en la confección del portal Web, y en el que el Subversion demostró una vez mas su genialidad, fue el dado entre dos componentes dependientes: una Pagina Web y un Archivo de Estilo. Se fue modificando el Cascade Sheet Style(CSS) hasta que el sitio tuviera un estilo común tanto en Internet Explorer como en Mozilla Firefox. Uno de los desarrolladores debido a su tardía incorporación al proyecto solamente se había superado en Internet Explorer y apenas comenzaba con un conocimiento mínimo en Mozilla Firefox. Tratando de optimizar el archivo de estilo obvió y borró una sentencia que a su entender no era significativa a su Internet Explorer, y actualizó el archivo, sin embargo dicha sentencia era necesaria para el Mozilla Firefox utilizado. Esto desató un conjunto consecuencias que se evidenciaron en el período de prueba en el cual se detectó que la página Web se deformaba en el navegador del software libre, lo que no pasaba antes. Gracias al SVN, y por el sistema de almacenamientos en cada una de las revisiones que se realicen, se pudo tener acceso a la versión correcta retrocediendo y lográndose de esta forma restaurar el código perdido, y que no se comenzara el trabajo desde los inicios, evitando que se atrasara el tiempo en que tenía que cumplirse la tarea, y que de estas dependieran otras tareas.

Ante tal situación, apareció un nuevo problema: un retrazo del tiempo que se le había asignado inicialmente a dicha tarea, y que fue dado por el cambio efectuado en la línea de código. Con el objetivo de aminorar el tiempo de recuperación, el SVN permitió reutilizar el código, mediante el uso de versiones anteriores.

Otros de los problemas identificados fue cuando uno de los miembros del equipo del Portal Web, trabajó en una función determinada, y el Jefe des Sistema se percató que dicha funcionalidad no correspondía con los requerimientos funcionales solicitados, y con el uso de la herramienta Trac el Jefe de Sistema pudo notificar al estudiante del error cometido y en un posterior momento junto con este, pudieron comparar las versiones, por lo que también sirvió como medio de estudio.

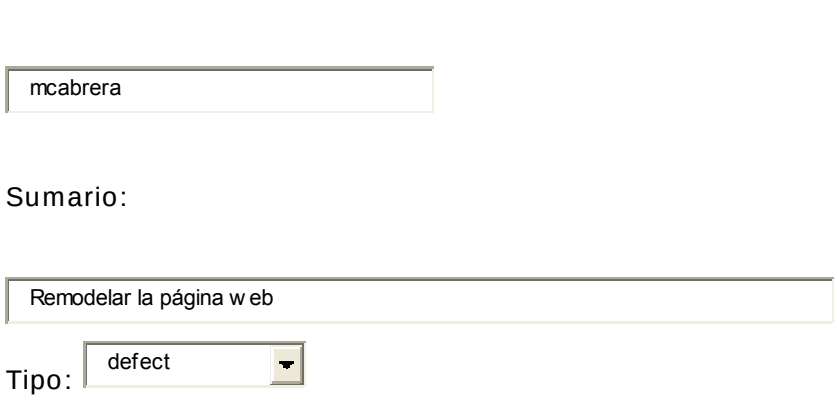
También otras de las dificultades dadas fue cuando se presentaron problemas con el cumplimiento de las tareas por parte de algunos miembros, por el cúmulo de trabajo, la mala distribución del mismo entre los miembros del equipo, y que a partir del uso del Trac se pudo contabilizar y hacer una mejor distribución de las tareas.

3.4 Configuración del Trac para el desarrollo del proyecto CICPC

El Trac como se ha definido en los capítulos anteriores, es un sistema para controlar los cambios, que permite a muchos desarrolladores que trabajan en partes de un sistema complejo gestionar los errores, las mejoras deseables y de una manera sencilla.

En el proyecto CICPC, en aquellos momentos se estaban dando los primeros pasos en el uso de esta herramienta, por lo que solo acompañó al Caso de Estudio en sus inicios y luego el trabajo se basó en analizar las características fundamentales que era necesario comprobar, como el uso de Ticket, el Diagrama de Gantt, entre otras funcionalidades.

El Trac tiene varias secciones, su unidad central es el ticket. Un ticket es un pedazo de texto que describe algo que hay que hacer (similar a una orden de trabajo) **(Anexo 21)**. Es la unidad básica del trabajo, por ejemplo: crear un fondo de escritorio, o un archivo de configuración, o corregir un defecto (también llamado *bug*). Un ticket tiene una estructura en la cual se pueden identificar varios campos como: e-mail, resumen, el tipo de ticket, como se muestra a continuación:



Toda la descripción:

Necesitamos renovar la página web:
- Convertir los HTML en un PHP con includes HECHO
- Logos HECHO
  - Logo de la portada
  - Fondo de la portada
  - Banner superior
- Estilo HECHO
  - En lugar de naranja -> violeta
- Contenidos
  - Reescribir la portada (en lugar de Debian -> Xubuntu)

Los ticket tienen también propiedades, y campos que es necesario llenar entre los que se encuentran la prioridad de la orden de trabajo, el tipo de componente, la severidad, a quién ha sido asignada, la versión, la palabra clave, entre otras como se muestra en la siguiente imagen:

Priority: (Prioridad)	normal	**Milestone: (Hito)**	Final
Component: (Componentes)	web	**Version: (Version)**	1.0
Severity: (Severidad)	normal	**Keywords: (Palabra clave)**	
Assign to: (Asignado a:)		**Cc:**	

☐ I have files to attach to this ticket (Indicar si hay archivos que se tengan que adjuntar al ticket)

Preview Submit ticket

Fig 15. Estructura de una orden de Trabajo en el Trac

En la Vista de un Ticket se podrán listar todos los tickets generados, permitiendo observar los datos específicos de cada una de las órdenes de trabajo que se hayan incluido:

Reporte de la página Web

Reported by:	yperez	Assigned to:	txopi (accepted)
Priority:	highest	Milestone:	Beta
Component:	web	Version:	
Severity:	critical	Keywords:	
Cc:			

Fig 16. Vista de un ticket

Con la puesta en práctica de la herramienta Trac se permitió comprobar que esta permite apoyar y orientar en el momento que se tenga problemas, con el fin de maximizar la eficiencia y calidad en el menor tiempo posible. El proceso de reporte de fallas tiene como objetivo brindarle al equipo de trabajo una respuesta en el momento oportuno con la calidad de servicio que se merecen, para lograr del éxito en el producto.

Pero su importancia reside en que se utiliza para la planificación de las tareas de cada miembro del equipo, que lleva un control de las tareas que se han cumplido y las que no, el uso de la Wiki, y que Controla los Cambios. Lo que tiene de particular el Trac es que te permite vincular un ticket con la revisión en un controlador de versiones (SVN) que corrige lo reportado en el ticket.

Cada una de las características que se mencionaron de la herramienta, fueron comprobadas y permitió que el equipo de trabajo tuviera una herramienta que le ayudara a controlar el tiempo que tenían para cada tarea, y a los administradores les posibilitó llevar datos actualizados del estado de cumplimiento de las tareas asignadas y seguir el los cambios.

3.5 Políticas de Integración y Uso de las Herramientas a utilizar

Herramientas

El cliente del control de versiones se encuentra integrado a la herramienta Eclipse IDE (herramienta definida como IDE estándar de desarrollo), por lo que el integrarse a la estructura organizativa del proyecto bajo el control de versiones requiere completar los pasos descritos en el punto **Integración al control de versiones.** Para el Control de Versiones: Subversión-1.3.1 para con TortoiseSVN-1.1.1m como cliente del mismo. Como herramienta para el control de cambios se propone el *Trac*.

Conexión al servidor:

Para establecer una conexión con el servidor es necesario contar con 3 elementos importantes: dirección, cuenta de identificación (autenticación) y el mecanismo de conexión.

- Dirección. Esta formada por la dirección IP del servidor y el nombre o ruta del repositorio.
- Cuenta de identificación. Formada por el identificador del usuario y la clave de acceso
- Mecanismo de conexión. Identifica al tipo de conexión y el puerto a utilizar. Para este proyecto se utilizará una conexión de tipo “svn” y el puerto a utilizar es el que se utiliza por defecto en los servidores subversión (5800, 5802).

Integración al control de versiones desde eclipse con subeclipse:

Para poder integrarse al control de versiones de la aplicación se requiere que cada persona involucrada tenga en su máquina el IDE de desarrollo con sus respectivos plugins (WTP 1.0, spring IDE 1.2.5, subeclipse) y poder así sincronizarse a la estructura del proyecto almacenada en el servidor de control de versiones.

3.5.1 Instrucciones de integración

- Definir las siguientes configuraciones iniciales en Eclipse IDE 3.2.x + WTP 1.x + Spring IDE 1.2.6:
Seleccionar el menú Window y escoger la opción Preferences:

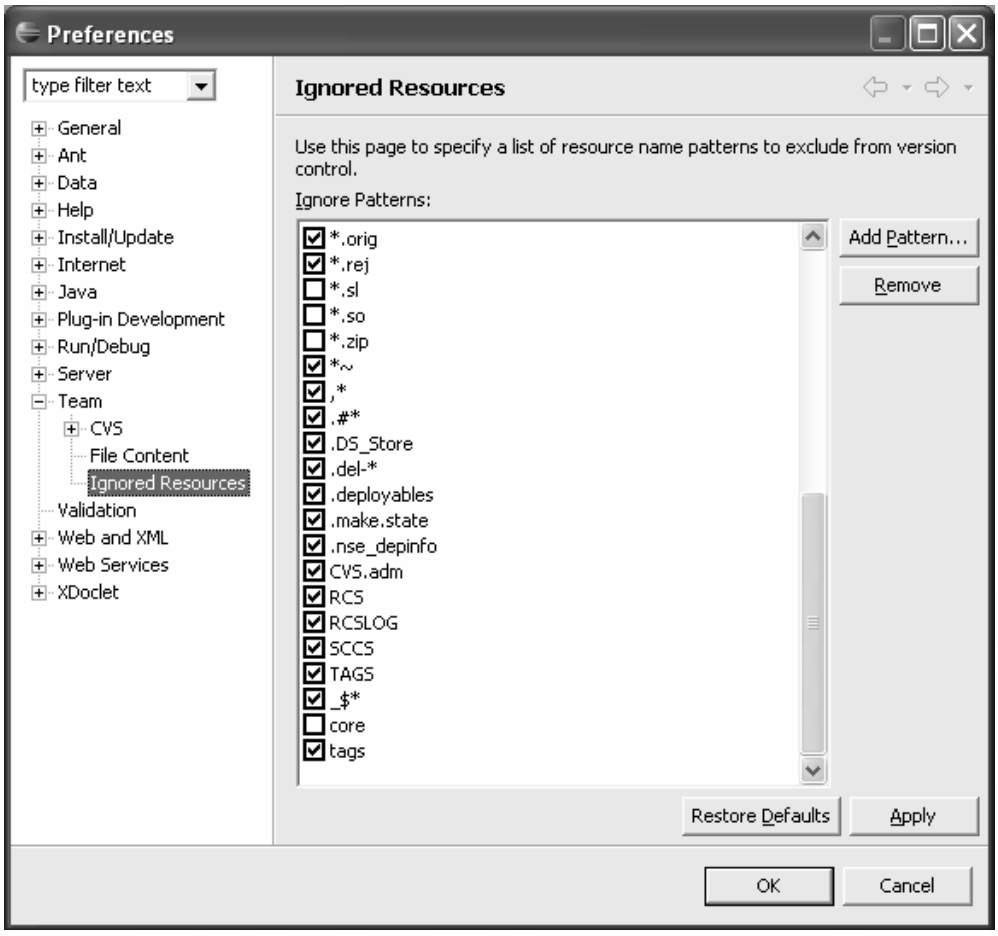


Fig 17. Ventana de selección de recursos

Luego se deben agregar las siguientes entradas en Ignored Resources de la opción Equipo:

*.settings

*.classpath

*.project

*.springBeans

build

Permite que no se eliminen aquellos directorios que se crearon en la estructura de la aplicación y que se encuentren vacíos.

Windows // Preferences // Team // CVS... en el tab Files and Folders…. y aquí desactivar la opción:

- Prune empty directories

Agregar las siguientes configuraciones:

Preferences // Java/ Installed JREs... y configurar el JDK en ves del JRE.

Preferences // Server // Installed Runtimes... y agregar el servidor web (tomcat 5.5.x)

- Crear una ubicación de depósito desde la perspectiva SVN Repositories:

Click derecho // New // Repository Location... y llenar con los datos:

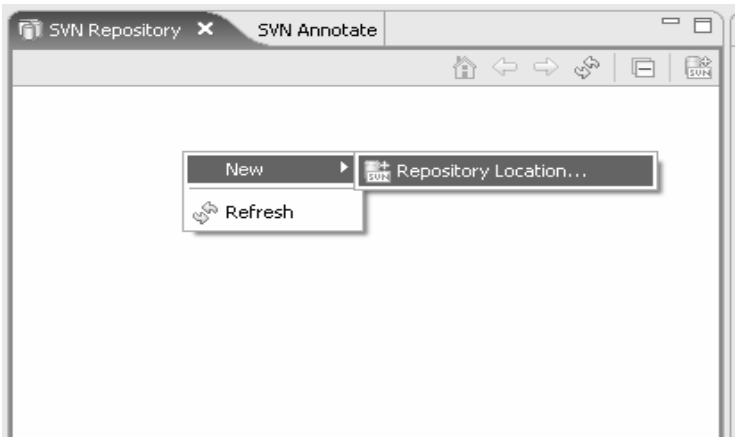


Fig 18. Ventana del Repositorio SVN

Los parámetros tienen la siguiente estructura:

svn://10.32.13.155/SIGEP/trunk/**workspace**

Fig 19. Arquitectura de carpetas

El **workspace** es en dependencia del rol de cada desarrollador. Los programadores trabajarán exclusivamente con el workspace **development. Ejemplo:**

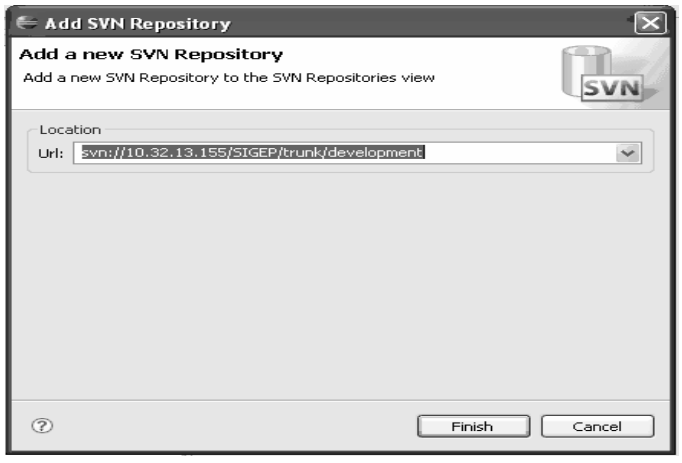


Fig 20. Adicionando el repositorio.

Cuando se le de OK saldrá la ventana para autentificarse:

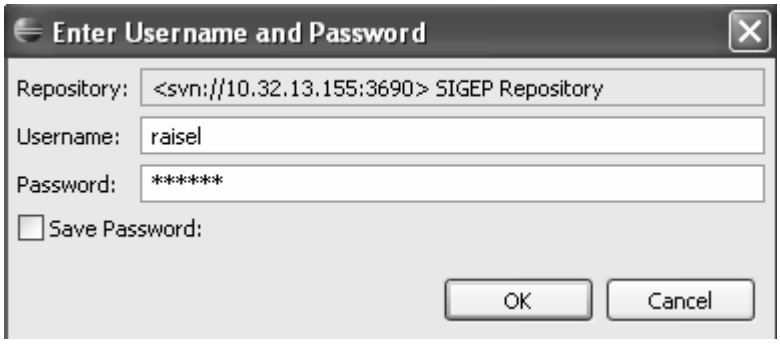


Fig 21. Ventana de autentificación del SVN

Importante: En caso que se haya dado recordar contraseña y esta cambie dará un error. La solución es borrar el archivo donde se guarda esta contraseña:

Window: C:\Documents and Settings\\<**USUARIO>**\Application Data\Subversion\auth\svn.simple

Linux: home/**<USUARIO>/.**subversion/auth/svn.simple

Una vez estando dentro de estas carpetas hay que borrar todos los ficheros que hay dentro. Para obtener la carpeta dar click derecho sobre la carpeta seleccionada y elegir la opción de Check Out As…

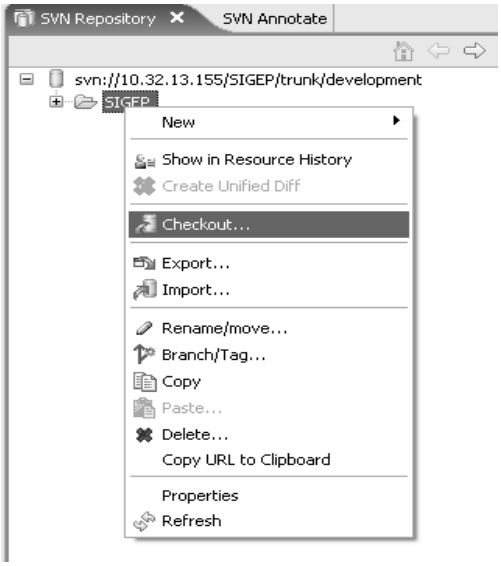


Fig 22. Ventana donde se obtienen las copias de trabajo

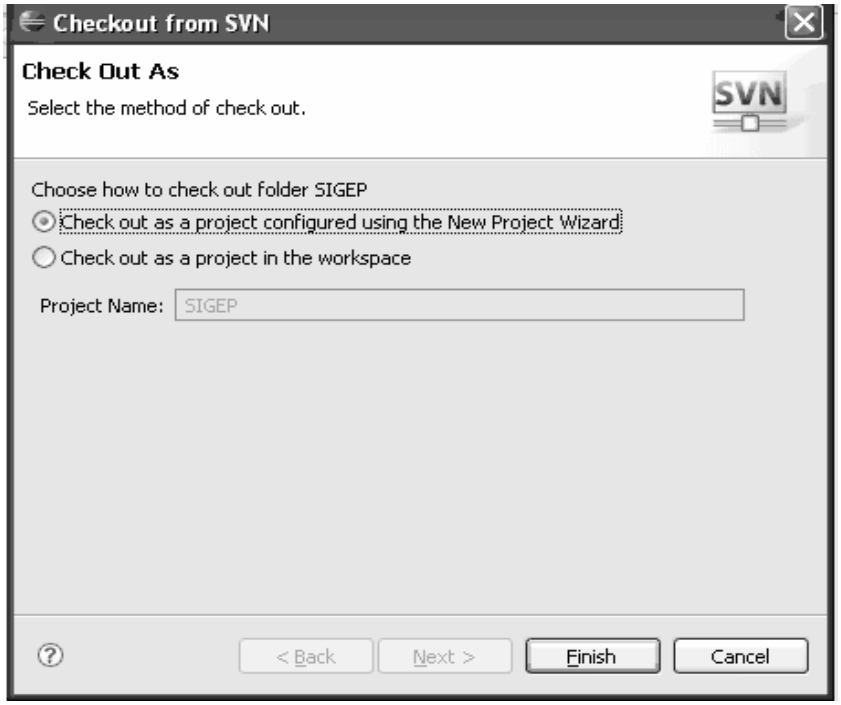


Fig 23. *Check Out as*

Luego se debe dar click en Finish. Después Seleccionar en Web // Dinamic Web Project

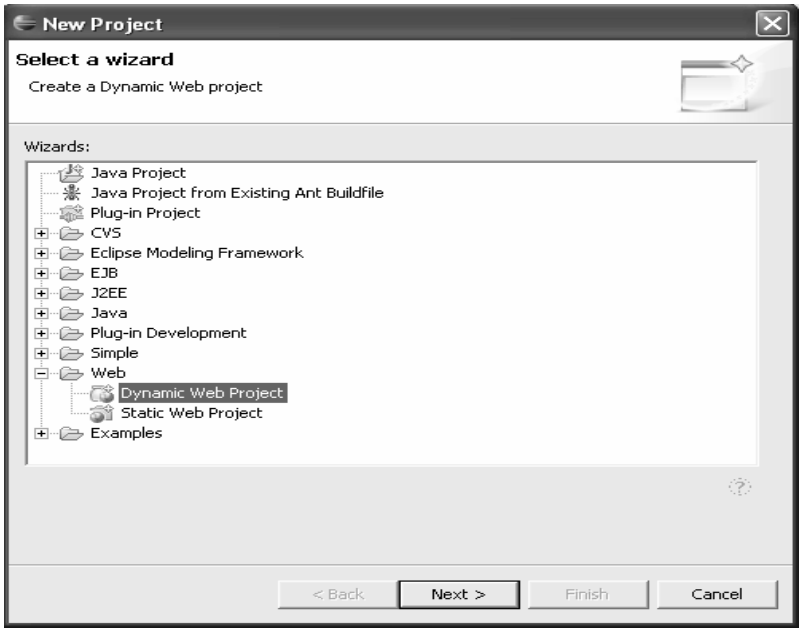


Fig 24. Ventana donde se configura los nuevos proyectos

Luego se debe escribir en Project Name → Proyecto

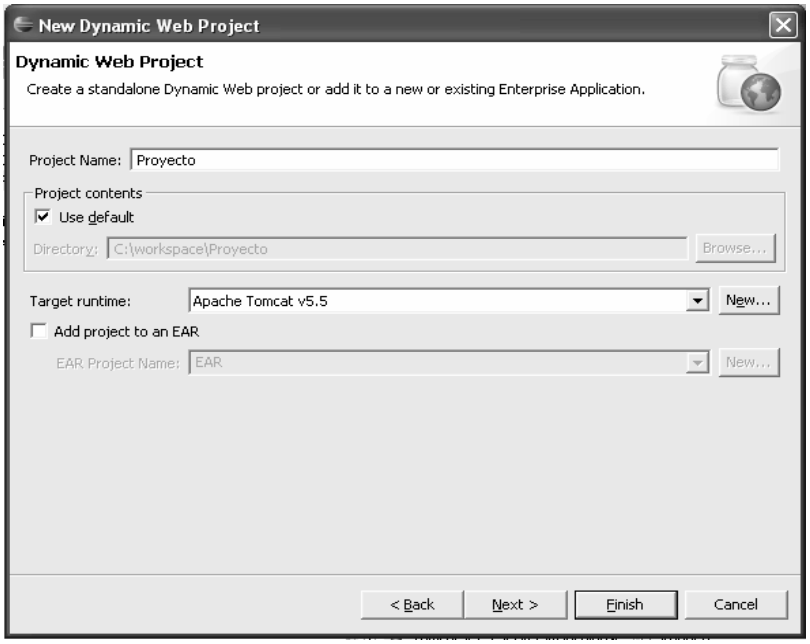


Fig 25. Ventana donde se adiciona una nueva o existente aplicación

Y se presiona el botón Finish.

Sincronización con el Control de versiones:

Para contar siempre con los últimos cambios y nuevos elementos del sistema, así como registrar en el control de versiones los cambios (casos de uso, análisis, diseño, implementación y despliegue) se debe mantener sincronizada la copia local de los proyectos con las carpetas del servidor de control de versiones.

Sincronizar siempre desde las vistas Project Explorer o Package Explorer cada uno de los proyectos, ya que esto **asegura obtener la última versión tanto en código como en modelado y todos sus elementos.**

Para sincronizar sólo es necesario dar clic derecho al proyecto seleccionado y seleccionar Team – Synchronize with Repository:

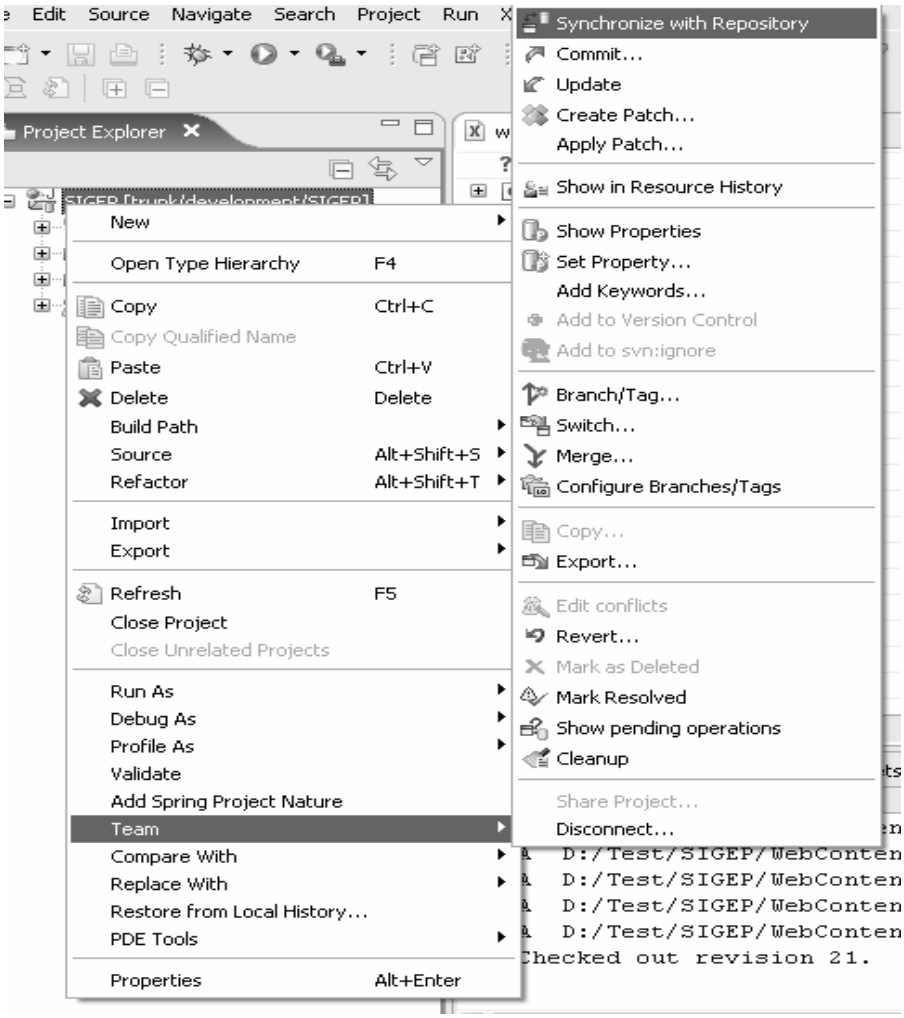


Fig 26. Ventana donde se muestran las Opciones del SVN en el Eclipse

Y seleccionar los recursos a comprometer (commit) en el CVS en la Modalidad Saliente de la vista Synchronize, así como los recursos a actualizar desde el repositorio (update) en la Modalidad Entrante de la vista Synchronize.

3.6 Integración al control de versiones desde Windows con TortoiseSVN.

1-Para bajar el proyecto (check out)

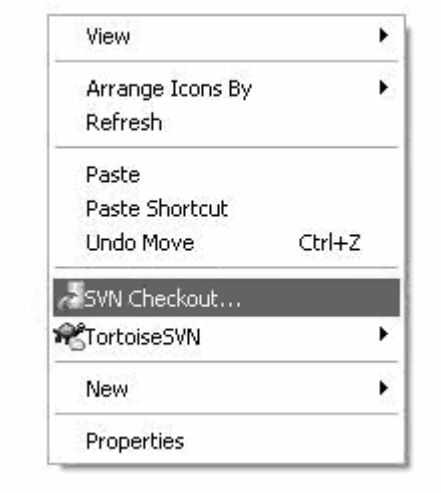


Fig 27. Ventana en Windows

2- Donde aparece "URL of Repository" se debe poner la dirección del repositorio y debajo en "*Checkout directory*" se pone el nombre de la carpeta para donde se va a bajar el proyecto.

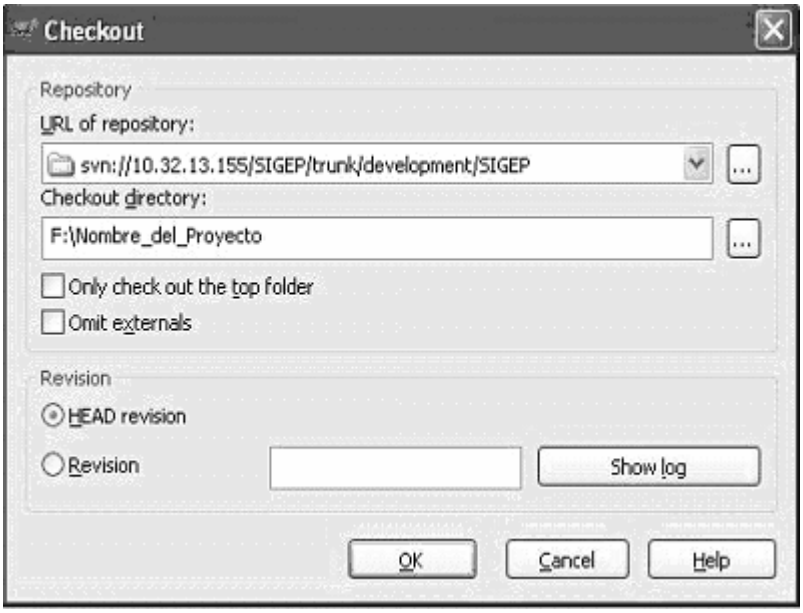


Fig 28. Ventana *Checkout* del SVN

3-Después aparece una ventana para autentificarse y acto seguido se debe comenzar a bajar el proyecto.

4-Para actualizar el proyecto se debe dar click derecho dentro del proyecto (Ver figura debajo)

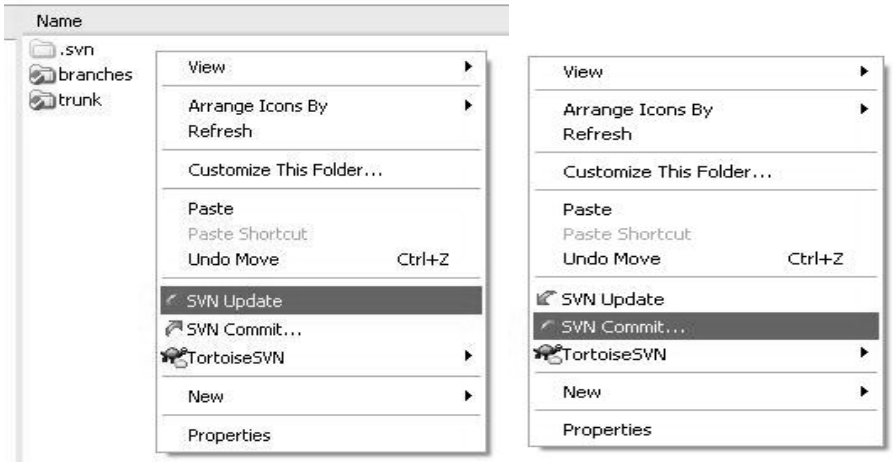


Fig 29 Pasos 4 y 5 para actualizar en el SVN

5- Para subir los cambios se da clic derecho dentro del proyecto (ver ejemplo debajo) y se selecciona SVNCommit

6-Después se seleccionan los cambios y se aceptan como aparece a continuación:

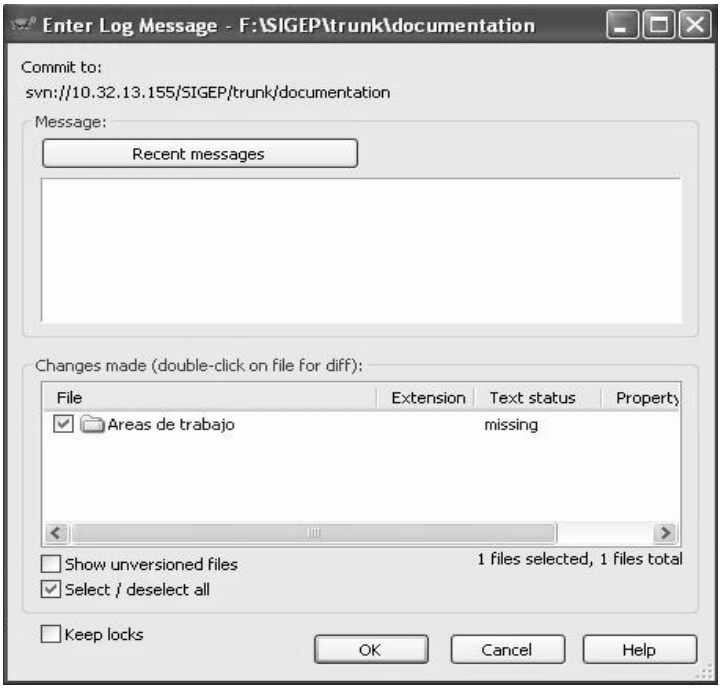


Fig 30. Ventana donde se suben los cambios al Repositorio

La PC en la que quedó instalado el Servidor Central es en la: 10.35.12.31, la cual se conectarán por el número de IP y en segundo momento por el nombre del Host.

3.7 Política de operaciones

Ante la necesidad de definir los pasos a seguir a la hora de trabajar con el Repositorio, se propone que del proyecto CICPC sean los siguientes:

Políticas de Operación:

Las políticas de operación tienen como objetivo establecer el marco de referencia sobre el cual se basarán las actividades de coordinación del proyecto en términos del control de versiones.

- La versión que comanda en todo momento es la que se encuentra en el Servidor Central de Control de Versiones.
- No se deben utilizar cuentas de acceso genéricas. El servidor registra los movimientos a los archivos con el usuario que ejecutó las modificaciones, por lo tanto, es importante que cada usuario que accede al servidor utilice la clave de acceso que se le asignó, para asumir las responsabilidades o disfrutar de los beneficios del histórico de movimientos. No se deben compartir las claves de acceso que se asignaron en forma individual.
- Sincronizar con el servidor de control de versiones modificaciones completas. Esto implica que solo se deben integrar al servidor los cambios que hayan sido probados y que se consideren, hasta cierto punto, estables. Así mismo, se deben sincronizar todos los elementos de que dependa una modificación. En este caso cada uno de los jefes de sistema, revisará antes de actualizar el archivo de trabajo.
- Utilizar el repositorio que se menciona en este documento aplicando los criterios necesarios para ubicar o clasificar el(los) proyecto(s) o archivos a subir en el repositorio que les corresponda.
- Que solamente en el repositorio se almacene la información y los elementos que son establecidos por cada Jefe de Sistema o miembro del Consejo Técnico.
- Cada uno de los Jefes de Sistemas pueden solicitar que determinado personal tenga acceso a carpetas con acceso restringido y esto se hará mediante algún miembro del equipo de GCS y con más de 36 horas de anterioridad.

- En etapas de cierre se especificará una fecha y hora para cerrar el acceso al Servidor de Control de Versiones y poder hacer un cierre (freeze). Cualquier necesidad extraordinaria de acceso deberá ser solicitada por escrito y dirigida al líder del proyecto o al Equipo de GCS.
- En caso que existan recursos en conflicto al momento de sincronizarse con el svn, solicitar ayuda del líder del equipo correspondiente para tomar la acción pertinente.

Conclusiones

- El proyecto cuenta con herramientas que responden a sus necesidades y que le permiten desarrollar eficientemente los procesos de la Gestión de la Configuración.
- Las políticas que se proponen complementan la seguridad del proyecto, y permiten que se organice el trabajo.
- Las herramientas seleccionadas han demostrado resultados satisfactorios en el Caso de Estudio que se presentó, demostrándose que han sido correctos los principios de selección utilizados para cada uno.
- Dentro del proyecto se ha ganado en eficiencia, y se ha aumentado la calidad del mismo debido a la correcta selección y aplicación de las herramientas.

CONCLUSIONES

Constituye una necesidad para la Institución que recién comienza en el mundo de la Industria del Software, que para el proceso de Gestión de Configuración se utilicen herramientas a partir de la realización de un minucioso estudio y selección, en el que se llegue a definir y aplicar las que se ajusten a las necesidades y características de cada proyecto productivo. Lo cual garantiza que los productos realizados sean confiables y precisos, siendo esto imprescindible ante la creciente exigencia por parte de clientes y usuarios en general.

Al finalizar el presente trabajo se consideran cumplidos los objetivos trazados y demostrada la hipótesis enunciada. A este planteamiento se puede arribar a partir de las siguientes sentencias:

- Actualmente en la UCI se le resta importancia a la Gestión de la Configuración dentro del desarrollo de software, guiados por la falta de conocimiento de los efectos que tienen los cambios incontrolados en las etapas por las que transita el producto.

- La automatización de los procesos de Control de Cambio y Control de Versiones, ha permitido en el proyecto CICPC aumentar la eficiencia, y la calidad.

- Se ha propuesto al equipo de trabajo de software, herramientas que permiten la Gestión de la Configuración en el proyecto CICPC: el Trac para la GCS y Control de Cambios, Subversion para el Control de Versiones y el TortoiseSVN como cliente del mismo.

- Se ha comprobado las ventajas del uso de las herramientas seleccionadas a partir de la aplicación de los resultados de la investigación en un caso de estudio.

- Se cuenta con una herramienta (el Trac), que permite notificar sobre cada cambio que ocurra en el proyecto a cada integrante del equipo de trabajo.

CONCLUSIONES

- Estas herramientas posibilitan a cada integrante del grupo de trabajo del software, mejorar su productividad determinando cuánto tiempo le dedican a un tipo de tarea en específico, llevar el estado de cumplimiento de las mismas y mantener la comunicación dentro del equipo de trabajo del proyecto.

De esta forma se evidencia el cumplimiento de los objetivos propuestos de una manera satisfactoria, ya que en CICPC el proceso de Gestión de Configuración utiliza como herramienta para el control de versiones el Subversion con el TortoiseSVN como cliente del mismo, mientras que para el control de cambios el Trac, permitiendo la automatización de tareas relacionadas con otras actividades, y dando un soporte eficiente a la interrelación existente entre los subprocesos de la GCS.

RECOMENDACIONES

Consecuentemente con el estudio realizado, se demostró la importancia de la Gestión de la Configuración dentro de los proyectos productivos tanto en La Universidad de las Ciencias Informáticas como en las restantes instalaciones que desarrollen software, por lo que se recomienda que el trabajo se publique, para que la comunidad de desarrollo de software, cuente con material bibliográfico, además de servir como referencia no solamente a los nuevos proyectos, sino también a aquellos que están en desarrollo y que puedan presentar problemas actualmente, tanto con la Gestión de la Configuración, como con el uso de herramientas que automaticen los procesos de ésta. Se recomienda también, aplicar los resultados obtenidos en la investigación durante el ciclo de desarrollo del proyecto CICP apoyándose en las experiencias del caso de estudio. Por último, que se siga de cerca los cambios y las nuevas versiones de herramientas como el Bazaar-ng por las opciones que brinda para el Control de Versiones.

REFERENCIA BIBLIOGRÁFICA

1. ALTAMIRANDA. *Plataforma Gforge*, [19/05/2007]. Disponible en: http://sistemas.fsl.fundacite-merida.gob.ve/docman/view.php/16/116/gforge.pdfDireccion
2. AMÉRICA. *Revista América Económica.com*, 2002. [12/12/2006]. Disponible en: http:www.AmericaEconomica.com/articles/comparisson.htm
3. ANTONIO, A. D. *La Gestión de la Configuración del Software*, 2001. [12/12/2006].
4. BABICH, W. *Software Configuration Management*. [12/12/2006], 1996. p.
5. BACH, J. *El Control de Cambio en la GCS*, 1998. [02/03/2007].
6. BERSOFF; H. SIEGEL*, et al. Software Configuration Management*. 1980. p.
7. BITELINA. *Wrike*, [20/05/2007]. Disponible en: http://bitelia.com/2006/12/22/wrike/
8. BROWN, N. and W. E. MICHAEL. *GathmannHobbs*, 1998. [10/12/2006]. Disponible en:
9. BUGZILLA. *Bugzilla*, 2007. [28/04/2007]. Disponible en: http://www.bugzilla.org/
10. CHANG, E. *Wrike*, 2006. [18/05/2007]. Disponible en: http://www.emilychang.com/go/ehub/app/wrike/
11. CHUIDIANG. *Sesión con CVS*, 2007. [20/02/2007]. Disponible en: http://www.chuidiang.com/chuwiki/index.php?title=Sesi%C3%B3n_con_CVS
12. COLLINS-SUSSMAN, B.; B. W. FITZPATRICK*, et al. Control de versiones son Subversion*, 2004. [10/10/2006].
13. ---. *Version Control with Subversion For Subversion 1.3*, TBA, 2006. [11/12/2006].
14. DARCSWIKI. *Darcs*, 2006. [25/02/2007]. Disponible en: http://darcs.net/DarcsWiki
15. --- Darcs Workflows compared with Subversion, 2006, [12/12/2006].
16. DOTPROJECT. *The home of dotProject - the Open Source Project Management tool*, 2005. [16/05/2007]. Disponible en: http://www.dotproject.net/index.php
17. EGROUPWARE. *eGroupware*, 2007. [17/05/2007]. Disponible en: http://es.wikipedia.org/wiki/EGroupware
18. E-GROUPWARE. *whats is eGroupWare*, 2000. [17/05/2007]. Disponible en: http://es.wikipedia.org/wiki/EGroupware
19. FEBLES, A. *Case Corporativo para el proceso de control de cambios*. Cuidad de la Habana, CUJAE, 2000a. [12/12/2006]. p.

20. ---. *MConfig.PM, Modelo de referencia para la Gestión de Configuración en la pequeña y mediana empresa de software* 2004. [12/06/2006]. Disponible en: http://calidadsoft.prod.uci.cu/documentacion/tesis/ConfigCASE%203.0.%20Herramienta%20de%20apoyo%20a%20la%20GCS.%20Propuesta%20arquitectonica.pdf/view
21. ---. *Un sitio de soporte de software, acercando la empresa al cliente, Memorias de la Convención Informática 2002*, 2002 [12/05/2006]. Disponible en:
22. FEBLES, A. and S. ÁLVAREZ. *Modelo de calidad CMM y su aplicación en las empresas, Memorias del primer encuentro internacional de Computación aplicada*. Guadalajara, México, 2000b.
23. FEBLES, A.; S. ÁLVAREZ*, et al. CASE Corporativo para la creación de la línea base de una empresa de Software*. *Revista Ingeniería Industrial CUJAE*, 2000c. [12/05/2006].
24. FERRER, J. A. and A. G. PASTOR. *SubVersioN*, [12/06/2007]. Disponible en: http://prhlt.iti.es/seminars/2007/Andres07b.pdf
25. GERVÁS, P. *Estándares y Gestión de la Configuración*. UCM, 2002. 4 p.
26. GNU. *Software*, 2006. [26/02/2007]. Disponible en: http://www.gnu.org/software/software.es.html
27. IBM. *Helps of Rational* 2001. [21/05/2007].
28. ---. *Helps of Rational*, 2003. [21/05/2007].
29. ---. *Rational ClearCase*, International Buseness Machine, 2005a. [21/05/2007]. Disponible en: http://www.ibm.com/software/awdtools/clearcase/
30. ---. *Rational ClearCase*, 2005b. [21/03/2007]. Disponible en: http://www-306.ibm.com/software/awdtools/clearquest/
31. ---. *Rational ClearCase*. [12/03/2007].
32. IEEE. *IEEE Guide to Software Configuration Management*, American National Standards Institute, 1987. [Disponible en: http://noqualityinside.com/nqi/node/9
33. ---. *IEEE Standard for Software Configuration Management* IEEE Computer Society, 1990. [Disponible en:
34. JACOBSON, I.; M. GRISS*, et al. Software*, 1994. [19/03/2006].
35. ---. *Software Reuse: Architecture, Process, and Organizartions for Business Success*. 1997. p.
36. LIBRESOURCE. *LibreSource*, [20/05/2007]. Disponible en: http://software-1.net/?u=/wiki/LibreSource
37. LÓPEZ, R. M. C. *Introducción a Rational ClearCase LT*, [19/03/2006]. Disponible en:

http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo_ClearCase.pdf

38. MARTÍNEZ, R. D. *ConfigCASE 3.0 Herramienta de apoyo a la Gestión de la Configuración. Propuesta arquitectónica*, CUJAE, 2006. [19/03/2006]. Disponible en: http://calidadsoft.prod.uci.cu/documentacion/tesis/ConfigCASE%203.0.%20Herramienta%20de%20apoyo%20a%20la%20GCS.%20Propuesta%20arquitectonica.pdf/view
39. MINREX. *Informe del MINREX Cumbre Mundial de la Sociedad de la Información*, 2004. [09/03/2006]. Disponible en: http://calidadsoft.prod.uci.cu/documentacion/tesis/ConfigCASE%203.0.%20Herramienta%20de%20apoyo%20a%20la%20GCS.%20Propuesta%20arquitectonica.pdf/view
40. MORFEO-PROJECT. *Gforge*, 2006. [22/05/2007]. Disponible en: http://morfeo-project.org/files/Estudio%20Mejoras%20y%20Nuevas%20Funcionalidades%20para%20GForge.pdf
41. MSDN. *Documentación de Visual SourceSafe*, 2007. [23/01/2007]. Disponible en: http://msdn2.microsoft.com/es-es/library/ms181038(vs.80).aspx
42. ---. *Introducción a Visual SourceSafe*, 2007. [23/01/2007]. Disponible en: http://msdn2.microsoft.com/es-es/library/3h0544kx(VS.80).aspx
43. MURCIA, U. D. *LibreSource*, [17/01/2007]. Disponible en: http://www.um.es/ayudaumu/buscar-resultados.php?hl=es&lr=&ie=UTF-8&q=related:dev.kiwi-cms.org/projects/kiwicms
44. NASA. *Software Configuration Management Guidebook*, Software Assurance Technology Center, NASA, 1995. [12/06/2007]. Disponible en: http://satc.gsfc.nasa.gov/GuideBooks/cmpub.html
45. OLEA, I. and I. ARENAZA. *Tutorial de Uso de CVS*, Free Software Foundation, Disponible en: http://es.tldp.org/Presentaciones/200211hispalinux/iarenaza/cvs-como-html/.
46. PAULK, M. C. and C. V. WEBER. *The Capability Maturity Model, Guidelines for Improving the Software Process*. [22/022007], CMU, 1999. 180-191 p.
47. PRESSMAN. *Ingeniería de Software, un enfoque práctico*. 3. [23/04/2007], Mc GrawHill, 1997. p.
48. ---. *Ingeniería de software. Un enfoque práctico*. Madrid, [25/04/2007], Mc Graw-Hill Interamericana de España S.A, 1998. p.
49. ---. *Ingeniería del Software, un enfoque práctico*. GrawHill Iberoamericana, 2000. p.
50. PRESSMAN, R. S. *Ingeniería de Software, un enfoque práctico*. 4. Mc GrawHill Iberoamericana, 2000. p.

51. ---. *Ingeniería de Software, un enfoque práctico*. 4. Mc GrawHill Iberoamericana 2000. p.
52. PROGRAMACIÓN.COM, M. D. *TortoiseSVN*, [021/02/2007]. Disponible en: http://www.manualesdeayuda.com/manuales/programacion/general/
53. PROGRAMMERS, D. *CVSTrac and Trac Combined Tutorial* 2005. [16/05/2007]. Disponible en: http://www.developingprogrammers.com/index.php/2005/12/15/cvstrac-and-trac-combined-tutorial/
54. RICKABAUGH, J. *Configuration management: the hidden friend in business reengineering. (Business Process Reengineering)*, 1994. [20/02/2007].
55. SENTIDOWEB. *Aplicaciones de gestión de proyectos*, 2006. [26/02/2007]. Disponible en: http://sentidoweb.com/2007/05/17/aplicaciones-de-gestion-de-proyectos.php
56. SÍNTESIS. *Síntesis*, 2005. [13/02/2007]. Disponible en: http:www.isocmex.org.mx/1522jul.html
57. SOFTWAREKM. *Software para Gestión del Conocimiento*, 2005. [06/01/2006]. Disponible en: http://softwarekm.blogspot.com/2007/04/egroupware.html
58. SOTO, N. *Proyecto Térnico del Proyecto CICPC*, 2006.
59. SOURCE, C. O. *The project.net experience*, 2006. [27/04/2007]. Disponible en: http://www.project.net/
60. STATION, M. *Bazaar-NG (bzr), sistema de control de versiones distribuido*, 2007. [13/06/2007]. Disponible en: http://www.marblestation.com/blog/?p=605
61. SVENSSON, H. *Incorporating the Personal software Process into Rational Unified Process*. Rational Edge, 2001. p.
62. TELEFÓNICA. *Estudio de Mejoras y Nuevas Funcionalidades para el Gforge*, 2005. [08/02/2007]. Disponible en: http://morfeo-project.org/files/Estudio%20Mejoras%20y%20Nuevas%20Funcionalidades%20para%20GForge.pdf
63. TIGRIS.ORG. *Subversion*, 2006. [02/02/2007]. Disponible en: http://subversion.tigris.org/
64. ---. *Tortoisesvn*, 2006. [02/02/2007]. Disponible en: http://tortoisesvn.tigris.org/
65. TRAC. *Trac open source project*, 2007. [23/02/2007]. Disponible en: http://trac.edgewall.org/
66. UNIVERSITY, O. *How to use RapidSVN with Subversion*, 2005. [10/06/2006]. Disponible en: http://www.oucs.ox.ac.uk/oucsweb/rapidsvn.xml
67. VALDÉS, S. La Habana, Universidad de las Ciencias Informáticas, 2006. [12/05/2007]. p.
68. VALERIO, G. *Software para Gestión del Conocimiento*, [22/03/2007]. Disponible en:

http://softwarekm.blogspot.com/2007/04/egroupware.html

69. WIKIPEDIA. *Bugzilla*, 2007. [15/04/2007]. Disponible en: http://es.wikipedia.org/wiki/Bugzilla
70. ---. *Control de Versiones*, 2007. [10/06/2007]. Disponible en: http://es.wikipedia.org/wiki/Control_de_versiones
71. ---. *CVS*, 2006. [12/06/2007]. Disponible en: http://es.wikipedia.org/wiki/CVS
72. --- The Free Enciclopedia, 2005, [04/04/2007].
73. ---. *GNU Savannah*, 2007. [16/01/2007]. Disponible en: http://en.wikipedia.org/wiki/GNU_Savannah
74. ---. *Project.net*, 2007. [12/01/2007]. Disponible en: http://en.wikipedia.org/wiki/Project.net
75. ---. *Repositorio*, 2006. [20/11/2006]. Disponible en: http://es.wikipedia.org/wiki/Repositorio
76. ---. *Subversion*, 2007. [20/01/2007]. Disponible en: http://es.wikipedia.org/wiki/Subversion

BIBLIOGRAFÍA

1. AEDIP. *Biblioteca*, [19/02/2007]. Disponible en: http://www.aedip.org/Actividades/Biblioteca/Otras/DIP_Heredia_Indice.pdf
2. ANTONIO, A. *Gestión de la Configuración*, 2001. [11/04/2007].
3. BABICH. *Software Configuration Management*. Addison-Wesley, 1986. p.
4. BLAIR, S. *A Guide to Evaluating a Bug Tracking System – White Pape*, MetaQuest Software, 2004. [16/04/2007].
5. BLISS, M. *Software configuration management: delivering quality software products*, 1993. [17/05/2007].
6. BUCKLEY, F. J. *Implementing Configuration Management: hardware, Software, and Firmware*. 2ª edición. IEEE Computer Society, 1995. p.
7. CALIDADDELSOFTWARE. *Consideraciones sobre la Gestión de la Configuración.*, 2006. [20/11/2006].
8. CERVANTES, O. H. C. *Tecnología de Información y Herramientas para la Administración de Proyectos de Software*, 2006. [12/06/2007].
9. CLYMAN, J. *Keep Track of Your Documents*. MAGAZINE, P., 2005. [11/05/2007]: 92-184.
10. COLLADO, M. *Control de versiones, configuración y cambios*, 2004. [18/04/2007].
11. DARCS. *FrontPage*, [03/02/2007]. Disponible en: http://darcs.net/DarcsWiki
12. DAVEEATON. *Configuration management Tools Summary*, 2007. [03/04/2007]. Disponible en: http://www.daveeaton.com/scm/CMTools.html
13. DMOZ. *Computers: Software: Configuration Management: Tools: Concurrent Versions System*, 2007. [02/10/2006]. Disponible en: http://dmoz.org/Computers/Software/Configuration_Management/Tools/Concurrent_Versions_System/
14. FEBLES, A. *Modelo de referencia para la Gestión de la Configuración en la pequeña y mediana empresa de software*. La Habana, Cuba, 2004.
15. FOGEL, K. and M. BAR. *Open Source development with CVS*, 2000. [18/05/2007].
16. GSINNOVA. *Rational ClearQuest*, [21/03/2007]. Disponible en: http://www.rational.com.ar/herramientas/clearquest.html
17. IBM. *Ayuda del Rational*, [07/04/2007].

18. ---. *Rational ClearCase LT*, [20/11/2006]. Disponible en: http://www-306.ibm.com/software/info/ecatalog/es_ES/products/A108274F54613Y01.html
19. ---. *Rational ClearQuest*, [15/02/2007]. Disponible en: http://www-306.ibm.com/software/info/ecatalog/es_ES/products/W108274Q56011S81.html
20. INDUDATA. *Rational ClearQuest*, 2007. [16/02/2007]. Disponible en: http://www.indudata.com/1rational_clear_quest.htm
21. INFOSGROUP. *Rational ClearQuest*, [20/11/2006]. Disponible en: http://www.infosgroup.com/paginas/v4/publico/soluciones/soluciones_producto/rational/productosrational.asp?referal=/paginas/v4/publico/soluciones/soluciones_producto/rational.asp#clearcase
22. LÓPEZ, R. M. C. *Introducción a rational ClearCase LT*, [21/06/2007].
23. LOURIDAS, P. *Version Control*, IEEE Software 2006. [20/02/2007].
24. LUGFI. *Introducción a los sitemas de control de versiones*, 2007. [26/01/2007]. Disponible en: http://www.lug.fi.uba.ar/documentos/scms/index.php#darcs
25. OLIN, B. and M. M. HESS. *Reengineering a configuration-management system*. SOFTWARE, I., Thomson Gale. Universidad de las Ciencias Informáticas, 1995. [21/04/2007].
26. PHILLIPS and DWAYNE. *Project management: filling in the gaps.(use of configuration management in software development)*. SOFTWARE, I., 1996. [22/04/2007].
27. PRESSMAN, R. S. *Ingeniería del Software*. 3ª edición. McGraw-Hill, 1993. p.
28. ---. *Ingeniería del software. Un enfoque práctico*. McGraw-Hill, 2000. 151 p.
29. RICKABAUGH, J. *Configuration management: the hidden friend in business reengineering*. *Business Process Reengineering*. ENGINEERING, I., 2007.
30. SCHAMP, A. *CM-tool evaluation and selection*. SOFTWARE, I., 1995. [17/03/2007].
31. SOTO, N. *Proyecto Técnico del proyecto CICPC*, 2006.
32. STATION, M. *Bazaar-NG (bzr), sistema de control de versiones distribuido*, 2007. [12/01/2007]. Disponible en: http://www.marblestation.com/blog/?p=605
33. STELLMAN, A. and J. GREENE. *Applied Software Project Management*, 2004. [20/03/2007].
34. SWEBOK. *Guide to the Software Engineering Body of Knowledge, SWEBOK*, IEEE 2004. [19/05/2007].
35. THURMAN, M. *Protecting the Crown Jewels 39(19)*, Computerworld, [19/02/2007].
36. TIEMPO21. *Comienza en Cuba Covención internacional Informática 2007*. *IslaGrande*, 2007.

37. TIRADO, H. J. F. *Sistemas de control de versiones*, 2006. [22/04/2007]. Disponible en: http://www.enterate.unam.mx/Articulos/2006/agosto/controlver.htm
38. TORRES, P. L. *Introducción a Rational Unified Process*, 2004. [12/11/2006]. Disponible en: http://www.dsic.upv.es/~letelier/pub/p16.ppt
39. VERLANG-SPRINGER. *Ingeniería del Software: un enfoque práctico*, 1989.
40. VESPEMAN, J. *Essencial CVS*, 2003. [11/03/2007].
41. WIILIAMS; C. CHADD*, et al. Automatic mining of source code repositories to improve bug-finding techniques*. ENGINEERING, I. T. O. S., 2005. [21/03/2007].
42. WIKILIBROS. *Informática Educacional/Gestión de la Configuración*, 2005. [21/11/2006]. Disponible en: http://es.wikibooks.org/wiki/Inform%C3%A1tica_Educacional/Gesti%C3%B3n_de_la_Configuraci%C3%B3n
43. WIKIPEDIA. *Atomicidad*, 2007. [20/05/2007]. Disponible en: http://es.wikipedia.org/wiki/Atomicidad
44. ---. *Bug Tracking system*, 2007. [21/05/2007]. Disponible en: http://en.wikipedia.org/wiki/Bug_tracking_system
45. ---. *Bugzilla*, 2007. [15/04/2007]. Disponible en: http://es.wikipedia.org/wiki/Bugzilla
46. ---. *Comparison of issue tracking systems*, 2007. [17/05/2007]. Disponible en: http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems
47. ---. *Configuration management*, 2007. [12/102006]. Disponible en: http://en.wikipedia.org/wiki/Configuration_management#Products_for_software_.26_hardware_configuration_management_.28SCM_.26_HCM.29
48. ---. *Control de Versiones*, 2007. [26/01/2007]. Disponible en: http://es.wikipedia.org/wiki/Control_de_versiones
49. ---. *CVS*, 2007. [26/01/2007]. Disponible en: http://es.wikipedia.org/wiki/CVS
50. ---. *dotProject*, 2007. [16/05/2007]. Disponible en: http://en.wikipedia.org/wiki/DotProject
51. ---. *List of project management software*, 2007. [20/03/2007]. Disponible en: http://en.wikipedia.org/wiki/List_of_project_management_software
52. ---. *Project management software*, 2007. [07/04/2007]. Disponible en: http://en.wikipedia.org/wiki/Project_management_software
53. ---. *Repositorio*, 2007. [26/01/2007]. Disponible en: http://es.wikipedia.org/wiki/Repositorio

54. ---. *Software Libre*, 2007. [12/11/2006]. Disponible en: http://es.wikipedia.org/wiki/Software_libre
55. ---. *Trac*, 2007. [20/01/2007]. Disponible en: http://es.wikipedia.org/wiki/Trac
56. WILLIAMS, A. *When it comes to software, don't just do it*. DATABASE, C., Thomson Gale, 2006. [23/05/2007].
57. XIMBIOT. *CVS--Concurrent Versions System*, [02/02/2007]. Disponible en: http://ximbiot.com/cvs/wiki/index.php?title=CVS--Concurrent_Versions_System_v1.12.12.1
58. ---. *Main Page*, [02/01/2/2007]. Disponible en: http://ximbiot.com/cvs/wiki/index.php?title=Main_Page
59. ---. *Version Management with CVS*, [02/02/2007]. Disponible en: http://ximbiot.com/cvs/manual/
60. XPLORER, I. *IEEE guide to software configuration management*, 2006. [30/05/2007]. Disponible en: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=89631
61. ZIMMERMANN, T.; P. WEIBGERBER*, et al. Mining version histories to guide software changes*. ENGINEERING, I. T. O. S., 2005. [25/02/2007].

GLOSARIO

Actividades: son las tareas asignadas dentro de un proyecto

Artefactos: Una parte de la información que (1) es producida, modificada, o usada por un proceso, (2) define un área de responsabilidad, y (3) está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos.

CCC: Comité de Control de Cambio

CICPC: Cuerpo de Investigaciones Científicas, Penales y Criminalística de la República Bolivariana de Venezuela

CM: (Management Configuration):

Commits: La acción de actualizar el repositorio con los cambios realizados es a lo que se le denomina *commit*, *check in* o *proteger*

CRM: CRM o Gestión de Relaciones con el Cliente es: el conjunto de estrategias de negocio, marketing, comunicación e infraestructuras tecnológicas, diseñadas con el objetivo de construir una relación duradera con los clientes, identificando, comprendiendo y satisfaciendo sus necesidades.

ECS: Elemento de configuración del Software.

Emulador: Un ***emulador*** es un software que permite ejecutar programas de computadora en una plataforma diferente de la cual fueron escritos originalmente.

FSF: Fundación de Software Libre (Free Software Foundation).

GCS: Gestión de Configuración de Software.

GNU: El Proyecto GNU se fundó en septiembre de 1983 por Richard M. Stallman para crear un sistema operativo completo de Software Libre. El Proyecto GNU consta de una serie de pequeños subproyectos mantenidos por voluntarios, empresas o combinaciones de ambos. Estos subproyectos también se denominan «Proyectos de GNU» o «Paquetes GNU».

GPL: La GPL es un texto legal e informáticamente complejo, la Licencia Pública General de GNU pretende garantizarle la libertad de compartir y modificar software libre, para asegurar que el software es libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayor parte del software de la Free Software Foundation.

Interfaz de Usuario: Interfaz de usuario, como conjunto de componentes empleados por los usuarios para comunicarse con las computadoras

GLOSARIO DE TÉRMINOS

Interfaz gráfica: Tipo de interfaz que permite a los usuarios comunicarse con un programa mediante funcionalidades gráficas. Normalmente incluyen una combinación de gráficos, barras de menús e iconos.

Interfaz: Colección de operaciones que son usadas para especificar un servicio de una clase o un componente.

Items: artículo

Modelo: Cosa que ha de servir de objeto de imitación. Objeto, construcción u otra cosa con un diseño del que se reproduce más iguales. Esquema teórico de un sistema o de una realidad compleja que se elabora para facilitar su comprensión y estudio.

Multiplataforma: Esto significa que el hardware o software que es multiplataforma tiene la característica de funcionar de forma similar en distintas plataformas (distintos sistemas operativos por ejemplo).

Open Source: Código abierto (del inglés ***open source***) es el término con el que se conoce al software distribuido y desarrollado libremente. Fue utilizado por primera vez en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (*free software*).

Portal Web: Un **portal de Internet** es un sitio Web cuyo objetivo es ofrecer al usuario, de forma fácil e integrada, el acceso a una serie de recursos y de servicios, entre los que suelen encontrarse buscadores, foros, documentos, aplicaciones, compra electrónica, etc.

Proyecto: Es la entidad que contiene el grupo de tareas necesarias para desarrollar un determinado producto.

Release: Un release es una versión de parte de un software que ha sido

Repositorio : Es un término utilizado en el dominio de las herramientas CASE. El repositorio podría definirse como la base de datos fundamental para el diseño; no sólo guarda datos, sino también algoritmos de diseño y, en general, elementos software necesarios para el trabajo de programación.

Requisitos: Condición que se tiene que cumplir, para que se pueda ejecutar.

Rol: Papel, cometido o función que tiene o desempeña que interpreta un actor.

RTF: Revisión Técnica Formal.

RUP: (Rational Unified Process): Proceso Unificado de Desarrollo de Software de Rational.

Servidor: Un servidor es un ordenador de gran potencia, que se encarga de "prestar un servicio" a otros ordenadores que se conectan a el. Varios servidores juntos, es decir conectados entre si, forman una red IRC. Estas maquinas se encargan de hacer que los mensajes que usted escriba lleguen a su destino y

también de hacer llegar hasta usted, los mensajes del resto de usuarios. En definitiva, se ocupan de gestionar todo el tráfico de información que produce en la red IRC.

Test unitarios: un test unitario ejercita el funcionamiento de una unidad de código—normalmente una clase—aislada del resto de la aplicación. La ejecución de los tests unitarios es sumamente rápida (del orden de los milisegundos). Decimos que estos tests tienen granularidad fina porque validan los detalles de más bajo nivel de la aplicación: el **micro-arquitectura** de las clases.

TIC: Tecnologías de la Información y la Comunicación

Trunk: (tronco) es un directorio para alojar la "línea principal" del desarrollo, un directorio branches (ramas) para que contenga las copias/ramas, y un directorio tags (etiquetas) para contener las copias/etiquetas.

UML: Lenguaje Unificado para el Modelado. Empleado mayormente para la modelación visual de aplicaciones con enfoque orientado a objeto, pero con mecanismos de extensibilidad para emplearlo en otros contextos.