

# **APLICACIÓN DE UNA HERRAMIENTA DE CONTROL DE VERSIONES PARA MEJORAR LA VISIBILIDAD DE LOS PROYECTOS DE SOFTWARE**

**Rodolfo Villarroel Acevedo**  
[rvillarr@spock.ucm.cl](mailto:rvillarr@spock.ucm.cl)

**Paola González Rodríguez**  
[pgonzale@spock.ucm.cl](mailto:pgonzale@spock.ucm.cl)

**Departamento de Computación e Informática  
Universidad Católica del Maule  
Avenida San Miguel 3605  
Talca, Chile**

## **Resumen**

El control de versiones es un aspecto crítico del desarrollo de software. Cuando se realiza correctamente, no sólo se salvan los elementos de configuración de potenciales desastres, sino que también ayuda a reforzar un buen proceso de ingeniería de software, ya que incentiva a los diseñadores a documentar lo que hacen, de modo que quede una huella de la actividad que es repetitiva y establece una mejor comunicación entre las personas del equipo. Este artículo describe la aplicación de una herramienta de control de versiones en el desarrollo de un proyecto de software. Con la aplicación de la herramienta se mantuvo un control y una organización mucho más rigurosa de los sistemas, logrando mejorar la visibilidad de los proyectos de software.

**Palabras Clave:** Control de versiones, gestión de configuración de software, herramientas de gestión de configuración de software, control de cambios.

## 1. Introducción

A medida que la tecnología avanza, tanto los procesos como los proyectos de desarrollo de software aumentan en complejidad, lo que causa confusión y disminución de la productividad en una empresa. Esto se debe al poco tiempo empleado en tareas tales como diseñar, codificar y depurar productos, que es lo que compone el trabajo real del desarrollo de software, y mucho en organizar lo que se está programando. Cuando el tamaño del grupo de trabajo se incrementa, la caída de productividad suele ser fatal, ya que grupos de tamaño medio o grande pueden no llegar a producir absolutamente nada útil.

Tal decadencia, se puede atribuir a los problemas de comunicación y coordinación entre los miembros del grupo de trabajo. Mientras más personas involucradas, es necesario utilizar más tiempo en la comunicación entre los miembros, y a medida que va creciendo, la comunicación entre los miembros del equipo se hace cada vez más importante y a la vez más costosa.

Los típicos problemas de coordinación llevan como consecuencia el doble de mantenimiento de los productos de software, la necesidad de disponer de datos compartidos, los problemas de la actualización simultánea sobre varias versiones de código fuente, problemas de recompilación incompleta, dependencia entre módulos que obliga a afinar las interfaces de comunicación y los problemas derivados de actualización de herramientas.

Durante el proceso de ingeniería de software, se debe gestionar una amplia variedad de elementos de software. No sólo son importantes los elementos que componen un proyecto, sino que lo son aún más las relaciones entre ellos, y es precisamente ahí donde reside la complejidad de la gestión de los sistemas de software.

La Gestión de Configuración de Software (*Software Configuration Management*, SCM) es una disciplina de gestión que permite controlar formalmente la evolución del software, garantizando la visibilidad en el desarrollo y en el producto, y la trazabilidad en el producto [6]. Los gestores de la configuración son responsables de llevar los registros de las diferencias entre las versiones de software, para asegurar que las nuevas versiones se deriven de forma controlada y para entregar las nuevas versiones a los clientes correctos en el momento justo [9].

Actualmente, resulta casi imposible desarrollar buen software, sin contar con métodos y herramientas adecuados que permiten superar los problemas de complejidad característicos del software. Sin embargo, antes de llevar a cabo un proceso de SCM automatizado, es importante que el gerente de la organización, los diseñadores, y el personal involucrado en el proceso de SCM, pongan en práctica las políticas, procedimientos, e instrucciones necesarias para evaluar y seleccionar una herramienta, tomando en cuenta que la herramienta finalmente seleccionada para ser usada en un proyecto, debe ser compatible con el ambiente de ingeniería de software en el cual se dará a lugar el desarrollo y mantenimiento del producto.

Hoy en día, las opciones para desarrollar aplicaciones de software sobran, ya que se cuenta con poderosas herramientas y técnicas que facilitan su construcción. No obstante, estar bien situado para desarrollar software en estos tiempos requiere una mezcla correcta de elementos, entre los cuales se pueden mencionar [8]:

- Comprender el rol del proceso de ingeniería de software, así como los cambios en los roles de los usuarios y desarrolladores derivados del cambio de paradigma.
- Adoptar y adaptar herramientas apropiadas para las tareas concretas.
- Tener la educación y habilidades necesarias para los cambios en procesos y tecnologías.
- Distribuir eficazmente la tecnología apropiada.

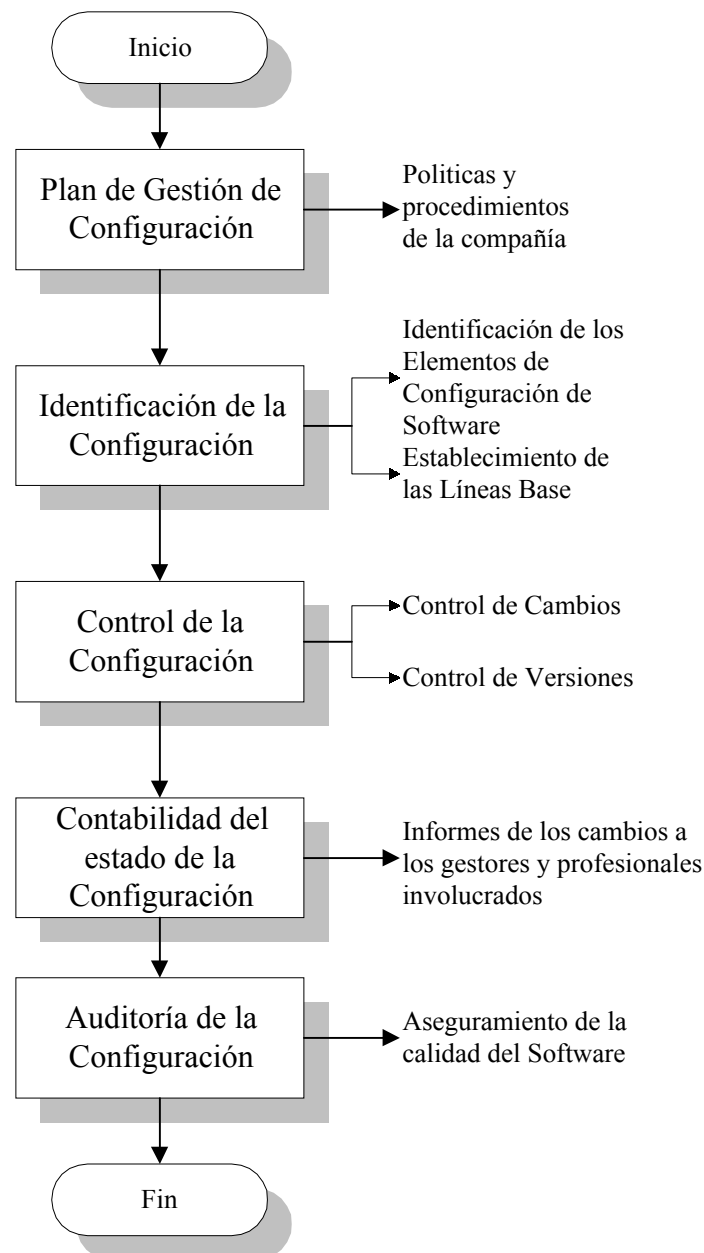
A pesar de la aplicación de técnicas, herramientas y tecnologías, los problemas en el proceso de software no han desaparecido. Esto es, debido a que los desarrolladores ignoran una necesidad simple pero crítica que es: *unir las herramientas con las tareas de desarrollo* [8]. Las herramientas y técnicas usadas en el ciclo de vida estándar para automatizar tareas pueden producir beneficios demostrables.

Las siguientes secciones de este artículo hacen referencia al proceso de SCM, las herramientas de SCM y un caso práctico de aplicación de la herramienta Visual SourceSafe a un proyecto de software.

## 2. El Proceso de SCM

El proceso de SCM está compuesto por un conjunto de actividades y procedimientos que aseguran que el desarrollador de software pueda controlar sistemáticamente los cambios de configuración y mantener la integridad del software a lo largo de todo su ciclo de vida. Lo inevitable y repentino de los cambios hace que aumente el nivel de confusión del personal que está trabajando en el proyecto, ya que cada cambio da lugar a una nueva revisión o versión de software. La confusión se presenta cuando los cambios no se han analizado antes de realizarlos, no se han registrado antes de implementarlos, no se les ha comunicado a aquellas personas que necesitan saberlo o no se han controlado de manera que mejoren la calidad y reduzcan los errores [7].

La figura 1 muestra el proceso de gestión de configuración como un conjunto de subprocesos interrelacionados, el cual comienza con un Plan de SCM y continúa con las actividades propias de SCM [1] [3].



**Figura 1. El Proceso de Gestión de Configuración de Software**

### 3. Herramientas de SCM

Para conseguir mayor calidad del software es necesario disponer de herramientas que permitan gestionar de manera automatizada las etapas por las que pasa un producto de software a lo largo de su ciclo de vida, por esta razón, las herramientas de SCM se hacen especialmente necesarias en entornos de desarrollo complejos en los cuales se utilizan diversas herramientas CASE (*Computer-Aided Software Engineering*) y el producto de software dispone de numerosas versiones y variantes [6].

Las herramientas de SCM han sido consideradas uno de los mayores logros de la ingeniería de software en las últimas décadas, debido a lo que éstas pueden hacer por el desarrollo de software. Tales herramientas deben ser compatibles con el proyecto que se desea desarrollar y con las necesidades descritas en el plan de gestión de configuración para así apoyar a cualquier actividad involucrada en el proceso de SCM.

Las herramientas de SCM han madurado junto con la tecnología de información en los últimos años. Hoy en día, hablar de herramientas de SCM no sólo significa hablar de control de versiones, ahora hay muchas herramientas que automatizan tanto el proceso como el desarrollo de un sistema de software. A continuación se describirán las tres clases de herramientas según su sistema de automatización [4]:

- **Herramientas de Control de Versiones:** la función principal de estas herramientas es el control de versiones sobre elementos tales como, código fuente, ejecutables, gráficos, documentos, etc. Ejemplos típicos de esta clase de herramientas son: *Visual SourceSafe* de Microsoft; *PVCS* de *Intersolv*; *Source Integrity* de *MKS*.
- **Herramientas Orientadas al Desarrollo:** Estas herramientas tienen capacidades de control de versiones, y también apoyan el ambiente de desarrollo distribuido. Estas herramientas permiten que desarrolladores y gerentes de proyecto trabajen en forma paralela para crear, unir, cambiar y entregar productos de una manera distribuida. Ejemplo típicos de esta clase de herramientas son: *ClearCase* de *Rational*, *ADC/Pro* de *True Software*.
- **Herramientas Orientadas al Proceso:** Estas herramientas incluyen las capacidades de control de versiones, apoyan el ambiente orientado al desarrollo, y tienen la capacidad de automatizar el flujo del ciclo de vida del software, los roles y sus responsabilidades. Estas herramientas se pueden usar para personalizar el modelo de proceso, y proporcionar integridad en la gestión de cambios donde el problema es asociado al código. Ejemplos típicos de esta clase de herramientas son: *Continuus* de *Continuus Software*; *Harvest* de *Platinum*; *PCMS* de *SQL*.

Típicamente, una herramienta de control de versiones será la primera elección para una compañía pequeña o para un grupo de desarrollo que tenga un número pequeño de descargas de software que probablemente no tengan variantes. Las herramientas orientadas al desarrollo satisfacen a las compañías medianas o grandes que no tengan tantos procesos formales definidos y no se enfoquen en certificaciones estándar. Tales compañías deben tener muchas descargas variantes y necesitan un fuerte apoyo para el desarrollo paralelo y gestión de proyectos, así como más fiabilidad en el almacén de datos de SCM. Las herramientas orientadas al proceso satisfacen normalmente a las grandes empresas con procesos formales que necesitan ser automatizados, ya que éstas se enfocan en el mejoramiento de los procesos en general.

Existen muchas herramientas de SCM que cubren las necesidades de organizaciones que apuntan a una gestión de configuración de software automatizada. Durante estos últimos años, las herramientas han ido madurando significativamente en términos de funcionalidad, mejoras de usabilidad y fiabilidad, y también en una amplia cobertura de plataformas. Cuando se ha evolucionado de un mundo de papel al mundo de las herramientas CASE, nuevas definiciones de gestión de configuración surgen [2]:

- SCM es una disciplina activa, que es complementaria al proceso de ingeniería de software y debe apoyar la definición e implementación del propio proceso de software.
- SCM debe manejar cambios a todos los componentes del proyecto a lo largo de su ciclo de vida, y satisfacer las necesidades de ingenieros y gerentes.
- SCM lleva como requisito implícito automatizarse, de este modo, puede manejar con mayor eficiencia y rapidez la evolución de los elementos bajo control de configuración y el proceso de ingeniería.

En consecuencia, las herramientas de SCM modernas deben ir mas allá de la funcionalidad de sistemas de SCM tradicionales, deben asegurar las versiones de los archivos fuentes y apoyar el desarrollo y gestión del producto,

solucionar problemas de rastreo y las tareas de gestión, apoyar el desarrollo distribuido múltiple y, lo más importante, apoyar la gestión del proceso.

Hay muchísimas herramientas de SCM, de las cuales alrededor de 8 son de dominio público y las demás son de carácter comercial. Las herramientas de dominio público son generalmente gratuitas, sin embargo tienen algunos inconvenientes, puesto que muchas de estas herramientas no proporcionan soporte ni tampoco son aconsejables para ser usadas en algunos proyectos. A continuación, se dará una pequeña descripción de algunas herramientas de dominio público:

**Aegis:** Corresponde a una herramienta orientada al desarrollo y no a la gestión, proporciona almacén de datos, líneas base, además de revisión y *testing* obligatorios de los cambios.

**CVS:** *Concurrent Version System*, corresponde a una extensión de RCS. Controla las concurrencias de varios usuarios que trabajan en un mismo proyecto.

**ICE:** *Incremental Configuration Engine*, herramienta que apoya la mayoría de las áreas de gestión de configuración, tales como control de versiones, control de cambios, reportes. Sin embargo, se han reportado problemas con sus líneas de comandos y con caídas en su interfaz gráfica.

**RCS:** *Revisión Control System*, herramienta que trabaja con un conjunto de archivos y directorios como una entidad.

**SCCS:** *Source Code Control System*, herramienta de control de versiones que soporta diferentes plataformas.

**ShapeTools:** herramienta que consta de un conjunto de comandos para el control de cambios y que proporciona un programa de control de versiones, un sistema de gestión de release, entre otros atributos.

**TkCVS:** Herramienta basada en la interfaz gráfica de CVS.

Cuando una organización comienza a mejorar sus problemas de gestión de configuración, la elección de una herramienta es fundamental para que nuevos procesos puedan ser construidos. La selección de una herramienta debe ser hecha con cuidado y debe basarse en las necesidades de la organización y la solución global de SCM. El proceso de adopción, debería determinar la herramienta, no viceversa.

#### 4. Caso Práctico: Aplicación de una Herramienta de SCM a un Proyecto de Software

El proyecto consiste en dos sistemas computacionales que manejan y administran la información referente a los ingresos y recursos para empresas insertas en el rubro de la construcción y específicamente en el ámbito de proyectos inmobiliarios.

El sistema *Cuentas Corrientes de Clientes* maneja la información correspondiente a los ingresos en dinero y a la creación de proyectos de ingeniería de construcción. Su desarrollo surge de la necesidad de obtener un registro y procesamiento de datos de cada uno de los proyectos de ingeniería que se van creando en las sucursales, al igual que las etapas con sus respectivos lotes. También maneja información de los clientes que se van asociando y sus contratos respectivos, junto con los ingresos que el cliente proporciona por la compra de un lote determinado. Además, permite la emisión de informes (ingresos, cuadro de cobranzas, cuentas, etc.) para mantener al usuario al tanto de la información que sea necesaria.

El sistema de *Control de Presupuestos* maneja la información correspondiente a los centros de costo con relación a los presupuestos asignados, los insumos ocupados en cada presupuesto de obra o urbanización, los proveedores de servicios o materias primas para los proyectos, las sucursales donde se realizan dichos proyectos, los documentos emitidos o recibidos por la empresa (facturas, boletas, órdenes de compra, etc.); además, los listados o reportes emitidos por el sistema para verificar el grado de avance de obras y administrar el uso de los recursos destinados a cada proyecto. También permite la emisión de informes (avance de obra, reporte de documentos, libros contables) para mantener al usuario al tanto de la información que sea conveniente.

Ambos sistemas fueron construidos en su primera etapa por un tiempo no menor a un año y con múltiples versiones del producto. Debido a que el sistema fue preparado sin un proceso de ingeniería de software debía estar constantemente en mantenimiento. Más tarde se consideró que el sistema tenía que dar un giro y ser replanteado para satisfacer nuevas necesidades de la empresa, por lo que se comenzó a desarrollar cada sistema nuevamente, los

cuales mantenían algunos de sus requerimientos anteriores, y los otros se adecuaron a la nueva estructura, a la vez que se agregaban nuevos requisitos.

#### **4.1 Actividades de SCM a realizar**

Al momento de analizar los cambios que luego fueron aplicados al sistema, se llegó a la determinación de implantar las siguientes actividades de SCM:

##### **a) Identificación de la configuración**

La identificación de los elementos de configuración debe involucrar todos las unidades del sistema que son vulnerables a cambios. Estos deben ser controlados durante todo el ciclo de desarrollo del proyecto hasta su mantenimiento. A continuación se listan los elementos de configuración a considerar en el proyecto los cuales pueden ser considerados como líneas base para futuros desarrollos:

- Especificación global del sistema y plan del proyecto de software.
- Especificación de requisitos del software.
- Especificación de diseño, que incluye la descripción del diseño de datos, descripción del diseño arquitectónico, descripciones del diseño de los módulos y descripciones del diseño de las interfaces.
- Manual preliminar de usuario.
- Especificación de las pruebas, que incluye un plan con los procedimientos de prueba y casos de prueba con el registro de sus resultados.
- Manuales de operación y de instalación.
- Programa ejecutable, con el código ejecutable del módulo y módulos enlazados.
- Descripción de la base de datos, esquema, estructura de archivos y contenido inicial.
- Manual de usuario final.
- Documentos de mantenimiento, con informes de problemas del software y peticiones de mantenimiento.

Los elementos de la lista deben ser desarrollados en ese orden y ser incluidos en la herramienta de SCM para controlar los cambios producidos y sus versiones correspondientes.

##### **b) Control de versiones**

El control de versiones del código fuente es una de las tareas más importantes a realizar. Las versiones deben cubrir tanto a archivos individuales como al sistema en general. Una nueva versión de un archivo es creada siempre y cuando se hayan llevado a cabo los cambios impuestos. La herramienta de SCM debe obtener las últimas versiones del sistema al abrir el proyecto, controlar la protección y desprotección de los archivos de código cada vez que se cierre el proyecto y agregar o eliminar archivos dentro del control de código fuente de la herramienta cada vez que se cree o elimine en el proyecto.

##### **c) Revisiones formales**

A medida que se va desarrollando, se deben revisar los elementos de configuración para comprobar que las especificaciones, diseño, estructura del sistema, entre otros requisitos, han sido correctamente establecidos dentro de la implementación del sistema. Mediante revisiones formales se debe corregir cualquier cambio en los elementos de configuración que no sea elemental.

##### **d) Otras tareas**

Es importante volver a considerar que para llevar a cabo un proceso de SCM exitoso, se debe escribir un plan de SCM que cubra responsabilidades y recursos disponibles para su ejecución, determinando las actividades de SCM a realizar más importantes. Una vez realizada esta tarea es posible que puedan surgir otras actividades de SCM para el proyecto.

#### **4.2 Selección de la Herramienta de SCM a utilizar.**

Tanto Cuentas Corrientes como Control de Presupuestos fueron desarrollados en su primera etapa por una persona y no se apoyó de ninguna herramienta de SCM ni tampoco se apoyó en ningún procedimiento de gestión de cambios.

En el nuevo plan de proyecto de ambos sistemas se especifica que tanto la definición de los requisitos como el desarrollo tenían que efectuarse en un plazo de 3 meses. Tal período de tiempo presionó al equipo de desarrollo para que empezara de inmediato con la implementación utilizando los recursos que disponía.

La herramienta de SCM disponible en la empresa correspondía a *Visual SourceSafe*, la cual encajaba en gran parte con las actividades de gestión de configuración que se deseaban aplicar. *Visual SourceSafe* (VSS), a pesar de no poseer rasgos de control de cambios, ofrece un buen modelo de control de versiones ideal para organizar múltiples versiones de un proyecto, por lo cual se consideró que podría ser una buena herramienta para ser usada en el sistema. Sin embargo, independientemente de estas opiniones favorables, se decidió hacer una evaluación más profunda de la herramienta, para verificar si realmente satisfacía las necesidades, o existía la posibilidad de hacer uso de alguna herramienta de dominio público. Sólo se consideraría la posibilidad de adquirir una nueva herramienta de SCM si no se satisfacían los requerimientos con VSS ni con alguna herramienta de dominio público.

Las ventajas y desventajas que se apreciaron en la herramienta VSS, en una primera instancia, fueron las siguientes:

- Ventajas: Buena interfaz de usuario, rápida y simple de instalar y usar, compatibiliza con el lenguaje del sistema, informativa, visualización de los archivos del proyecto en forma de árbol.
- Desventajas: control de acceso no es muy asegurado, sólo es una herramienta de control de versiones.

La siguiente tabla muestra la evaluación realizada a VSS, basándose en las categorías definidas por Mosley [5]. La tabla 1 fue diseñada clasificando la calidad de cada categoría en 4 tipos; Alta, Media, Baja y No tiene, entregando el resultado de la evaluación realizada.

Categoría	Calidad	Resultado de la Evaluación
Lenguaje	Alta	VSS compatibiliza con <i>Visual Basic</i> , <i>Visual C</i> , <i>Visual Test</i> , <i>Fortran</i> , <i>PowerStation</i> , <i>PowerBuilder</i> , entre otros.
Control de procesos	Baja	VSS controla parte del proceso de gestión de configuración, el cual sólo incluye control de versiones.
Versiones/Líneas base	Alta	VSS es capaz de generar versiones de múltiples líneas base, así como acceder a líneas base previas sin mayores problemas.
Control de accesos	Media	VSS permite controlar el acceso de los usuarios a los diferentes elementos de configuración. Sin embargo, no es una tarea que se realiza en forma 100% confiable.
Gestión del Cambio	No tiene	VSS no es capaz de dar soluciones en caso de haber problemas con los cambios efectuados o de realizar análisis del impacto.
Reportes	Alta	VSS permite mostrar el estado de cada elemento de configuración y generar reportes sobre las versiones y las diferencias de los archivos.
Misceláneo	Baja	VSS soporta proyectos con múltiples base de datos, pero con problemas al momento de organizar bases de datos automáticamente.
Plataformas	Baja	VSS puede operar en Windows, Unix, Linux. Sin embargo, un desarrollo en varias plataformas es poco probable.
Otros (interfaz con otras herramientas)	No tiene	VSS no interactúa con otras herramientas de SCM.
Costos (adquisición, entrenamiento, etc.)	-----	No hay gastos en la adquisición del software (está disponible), el personal de desarrollo se compromete a aprender la herramienta y a capacitar.

**Tabla 1. Formato de Evaluación de Herramientas de SCM usado con VSS**

Basándose en esta evaluación y, teniendo en cuenta las actividades de SCM implantar, se decidió realizar el proyecto usando VSS. Las razones principales fueron la disponibilidad de la herramienta y una muy buena evaluación en las categorías de Lenguaje, Versiones/Línea Base y Reportes. Se descartó la posibilidad de usar software de dominio público, debido a que tuvo una menor evaluación basada en las categorías del formato utilizado. Aparte de *Visual SourceSafe* se incorporaron otras herramientas *CASE* para la elaboración del proyecto tales como *PowerBuilder 8.0*,

herramienta que resultó bastante útil a la hora de elaborar el *script* que generaba la base de datos de los sistemas en PostgreSQL, además de definir y graficar los modelos conceptuales y relacionales de las tablas de datos.

Ya que la herramienta no permitía controlar los cambios formalmente, se analizó una forma de llevarlos a cabo pero de manera no formal, la cual sólo se llevaba a cabo si el cambio era muy complejo. Los puntos a considerar a la hora del cambio fueron siguientes:

- Análisis de cambios críticos.
- N° de líneas de código estimada o documentación afectada.
- Complejidad del problema.
- Tiempo estimado para reparar, testear e incorporar.
- Valoración de recursos disponibles y costos relacionados.
- Totalidad del impacto.

#### **4.3 Estructura del Proyecto de Software y Proceso de Desarrollo.**

El proyecto ha sido implementado usando *Visual Basic 6.0* como lenguaje de programación. Actualmente consta de un módulo correspondiente a la declaración de variables globales, un módulo de rutinas de carga y un módulo de clase para la verificación de permisos a la base de datos, además del archivo ejecutable. La base de datos de los sistemas está administrada por *PostgreSQL* en un servidor bajo ambiente *Linux*. El nuevo Sistema de Cuentas Corrientes y el nuevo Sistema de Control de presupuestos contienen 64 y 26 archivos de código fuente respectivamente con aproximadamente 50 a 100 líneas de código cada uno, los cuales pertenecen a 30 formularios de pantalla en Cuentas Corrientes y 24 para Control de Presupuestos. Los archivos de código fuente están organizados según el formulario al que pertenecen, de esta forma, un archivo de código tendrá el mismo nombre que su formulario correspondiente.

Los requisitos del proyecto fueron analizados en conjunto con la Gerencia y el Departamento de Contabilidad de la empresa. Las reuniones se establecían normalmente cada vez que había confusión en el grupo de desarrollo con los requerimientos de usuario o cuando éstos sugerían nuevas funciones. Lo fundamental en el desarrollo de ambos sistemas es que en su implementación debía mantener una modalidad estándar, que les permitiera ser vendido y adecuado a las necesidades de otras empresas del rubro. En general se llegó a la conclusión que los cambios en el desarrollo de los nuevos sistemas con respecto a los antiguos debían consistir en tres categorías principales:

##### **a) Cambios a la Base de Datos**

- Remodelación a las tablas de ambos sistemas, lo cual incluye eliminación de tablas innecesarias y adicionar otras para lograr un mejor acceso a la base de datos.
- Campos representativos que permitan ser identificados según la tabla de datos a la que pertenecen. Esto es importante tanto para llevar un adecuado orden de la base de datos como para facilitar la ubicación de los campos dentro de la misma.
- Mantenimiento de datos históricos mediante un campo que permita verificar si el registro está o no en uso. Tal campo debe ser incluido sólo en las tablas donde sea necesario.

##### **b) Cambios al Código**

- Componentes que puedan ser reutilizables para nuevos cambios o para futuros nuevos sistemas.
- Estandarización de los procedimientos y funciones comunes, que se pueden acceder desde cualquier punto del sistema, y que a su vez permitan que no se repita el mismo código dentro de los diferentes módulos.
- Control de errores en cada procedimiento que permita manejar e informar en forma adecuada las posibles fallas del código.

##### **c) Cambios de Presentación**

- La disminución de formularios, para que permita la disminución del tamaño que ocupa en disco el archivo ejecutable.
- La unión de funcionalidades en un mismo proceso, para que exista un orden de acceso dentro del sistema.
- Tanto la presentación como las pantallas de los procesos deben mantener un estilo común y propio, facilitando en lo posible el trabajo al usuario.



Se planteó una lista con los cambios a realizar en cada categoría, evaluándolos de acuerdo a su necesidad y complejidad. Una vez aprobados tales cambios se especificaron las restricciones a respetar y los criterios a seguir.

Se comenzó con el desarrollo del sistema de Cuentas Corrientes, el cual mantenía la mayoría de los requisitos del antiguo sistema con lo que se reutilizó parte del código antiguo modificándolo y adecuándolo a la nueva estructura del sistema. El desarrollo de Control de Presupuestos resultó más complejo ya que los nuevos requisitos indicaban un manejo más riguroso de los datos por lo que el código antiguo no resultaba útil.

El grupo de desarrollo estaba compuesto de 3 personas; un jefe de proyecto y dos desarrolladores. El jefe de proyecto dispone de un equipo llamado *servdesarrollo* o servidor de desarrollo el cual estaba dispuesto para almacenar todo el código generado a lo largo de la implementación de los sistemas. El equipo de los desarrolladores, además de trabajar con el desarrollo de los sistemas, se disponía para las pruebas en la base de datos y para la elaboración de los documentos respectivamente. Por lo tanto, se dispuso el computador del jefe de proyecto como servidor de desarrollo y fue ahí donde se instaló VSS y se asignaron los permisos de usuario al control de código fuente. Los otros dos equipos de desarrollo accedían al servidor por la red, desde donde se instaló VSS cliente. Se estableció que todos los elementos de desarrollo, tanto el código como la documentación de los sistemas, serían guardados y accedidos desde el equipo servidor.

Se crearon dos bases de datos de VSS, una para el sistema de Cuentas Corrientes y otra para Control de Presupuestos. En cada una de ellas se agregó el código fuente correspondiente y la documentación que se iba elaborando a medida que se procedía con el desarrollo. Los archivos fueron organizados en carpetas dentro del explorador de VSS de acuerdo al proyecto al que pertenecían. Los archivos corresponden a la documentación, código fuente y formularios de cada sistema respectivamente. La organización de las carpetas del proyecto fueron realizadas de esta forma para facilitar el hallazgo de los archivos. Las actividades centrales fueron planeadas y luego implementadas manteniendo un registro de lo realizado en un documento guía, con el propósito de mantener un informe del proceso de SCM realizado.

Los cambios fueron llevados a cabo de acuerdo a los establecido y las actividades correspondientes al control de versiones fueron logradas en su mayoría. VSS usa números de versión para mantener un registro de cada cambio que se realiza en los archivos y proyectos. De este modo, da la capacidad de recuperar cualquier versión de un archivo o proyecto. Los registros de las versiones anteriores se pueden realizar de 3 maneras: mediante el número de versión interno, mediante la fecha, y mediante etiquetas definidas por el usuario. Esto permitió de alguna forma personalizar el proyecto para un entendimiento individual de trabajo. Pocas veces se emitieron informes del explorador de VSS, y las veces que se hacía era cuando el jefe del proyecto requería detalles de usuario y fecha de lo desarrollado, así como de las versiones liberadas.

A pesar que el jefe de proyecto estaba generalmente al tanto de los cambios hechos al sistema, los reportes de la herramienta permitían probar en forma palpable el trabajo efectuado por cada uno de los integrantes del grupo de desarrollo y los cambios realizados a cada uno de los sistemas, considerando los elementos de la configuración que habían sido trabajados y los responsables de cada labor.

## **5. Conclusiones**

Aplicar el proceso de gestión de configuración en un grupo de desarrollo pequeño no es una tarea muy fácil de realizar, sobre todo porque la mayoría de las veces no se cuenta con los recursos para llevarla a cabo. Una de las más grandes dificultades de aplicar gestión de configuración de software en un grupo pequeño de desarrollo es el poco conocimiento de la importancia de esta actividad y a veces, la idea de que es un servicio burocrático que sólo produce retrasos.

La aplicación de la herramienta para automatizar las actividades propuestas, fue sin duda una ventaja para el proyecto. Con ella se mantuvo un control y una organización mucho mas rigurosa de los sistemas y se pudo realizar una visualización de los cambios efectuados. No obstante, la instalación y posterior uso de VSS presentó al principio problemas de estabilidad debido a la inexperiencia en el uso de herramientas de SCM, ocasionando descoordinación del grupo de desarrollo. Problemas tales como el olvido de la protección y desprotección de elementos de

configuración en la utilización de la herramienta al principio eran frecuentes; sin embargo, se iban conociendo y solucionando a medida que se trabajaba con la herramienta.

Sin una herramienta de control en un equipo de desarrollo que trabaja en red, lo más probable es que la confusión habría sido algo común y el tiempo estimado de desarrollo se habría alargado más de lo estipulado. A continuación, se citan algunas pautas generales para evitar problemas con VSS:

- Todos los archivos requeridos para operar deben estar bajo control de versiones.
- Si un miembro del equipo deja de trabajar sobre un elemento de configuración, debe proteger el archivo.
- Al momento de crear un nuevo elemento de configuración se debe ingresar de inmediato dentro del control de VSS, para así mantener un historial de los cambios del elemento desde el principio.

Una adopción de tecnología de SCM presenta un desafío difícil para muchas organizaciones que se esfuerzan por mejorar su proceso de gestión de configuración. Por esto se hace necesaria una estrategia definida que solucione problemas complejos de adopción de tecnología, y que realice todo el trabajo necesario para asegurar el uso exitoso, óptimo, y eficaz de dicha tecnología. Su adopción es compleja porque impacta datos, procesos, y personas. La tarea de entender los problemas involucrados en una adopción de tecnología de SCM es la razón principal de por qué las organizaciones no utilizan sus herramientas con éxito. Una herramienta por sí sola no resolverá los problemas de gestión de configuración de una organización, ya que deben priorizarse primero otros problemas técnicos y culturales.

La barrera más grande a superar, cuando se introduce SCM en una organización, es cambiar el punto de vista de las personas. La reacción a menudo es negativa hacia esta disciplina, ya que muchos diseñadores de software han experimentado problemas en este sentido con el personal de gestión de configuración, quienes perciben la herramienta como un intruso, y por no entender el proceso y encontrarlo burocrático no siguen sus procedimientos. En muchas organizaciones, SCM tiene un nivel bajo, y el personal no se entrena para realizar las labores correspondientes. SCM integra el trabajo creado por muchas personas a lo largo de una organización, la persona encargada de gestión de configuración necesita comprender los principios de la ingeniería de software y los aspectos culturales de la organización.

Las herramientas de control de versiones y herramientas de gestión de configuración cubren una amplia gama de acercamientos y funcionalidad. Sin embargo, obtener el producto mas avanzado podría ser un error si sólo se intenta controlar una situación simple, por lo que el problema no es qué producto es el mejor, sino cual es el mejor para el proyecto.

Como trabajo futuro, se pretende proponer un plan de acción que cubra todas las actividades de SCM, adaptado a la realidad nacional, e implementarlo en una empresa nacional.

## Referencias

- [1] BERLACK, RONALD. "Software Configuration Management". John Wiley & Sons, 1992.
- [2] BERLACK, RONALD. "Evaluation and Selection of Automated Configuration Management Tools". Noviembre 1995. <http://stsc.hill.af.mil/crosstalk/1995/evaluati.asp>
- [3] BUCKLEY, FLETCHER J. "Implementing Configuration Management". IEEE Computer Society Press, 1996.
- [4] DHARMINDER, PREMI. "Software Process Management". Abril 2001. <http://sern.ucalgary.ca/~dspremi/621/webdoc.htm>
- [5] MOSLEY, VICKY. "Software Configuration Management Tool: Getting Bigger, Better and Bolder". Enero 1996, <http://www.stsc.hill.af.mil/crosstalk/1996/jan/cmttools.htm>
- [6] PIATTINI, MARIO. "Mantenimiento del Software", RA-MA. 1998.
- [7] PRESSMAN, ROGER. "Ingeniería del Software. Un Enfoque Práctico", 5ª. Edición. McGraw-Hill, 2002.
- [8] RUBIN, HOWARD. "Ingeniería de Software, para sobrevivir en los 90". Informática, Vol. 23, Nº 6, Julio 1990, pp. 55-61.
- [9] SOMMERVILLE, IAN. "Ingeniería de Software", 6ª. Edición. Addison Wesley. 2002.