CENTRO DE CIENCIAS BÁSICAS

MAESTRÍA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES

TESINA

Para obtener el grado de Maestría

PROPUESTA DE UN PROCESO DE IMPLEMENTACIÓN Y MANEJO DE CONTROL DE VERSIONES EN EL INEGI

PRESENTA

LI. ALEJANDRO NAVARRO CASILLAS

DIRECTOR DE TESIS

M.C. CÉSAR EDUARDO VELÁZQUEZ AMADOR

SINODALES

M.C. JORGE EDUARDO MACÍAS LUÉVANO

M.C. FRANCISCO JAVIER PINALES DELGADO

CD. UNIVERSITARIA, JUNIO DE 2010

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES
Conmemoración del Bicentenario del inicio de la Independencia de México y del Centenario de la Revolución Mexicana

Centro de Ciencias Básicas

L.I. ALEJANDRO NAVARRO CASILLAS
PASANTE DE LA MAESTRÍA EN INFORMÁTICA
Y TECNOLOGÍAS COMPUTACIONALES
P R E S E N T E .

Estimado (a) Alumno (a) Navarro:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis y/o trabajo práctico titulado: **"Propuesta de un proceso de implementación y manejo de control de versiones en el INEGI",** hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 4 de junio de 2010
"LUMEN PROFERRE"
EL DECANO

DR. FRANCISCO JAVIER ÁLVAREZ RODRÍGUEZ

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES
CENTRO DE CIENCIAS BÁSICAS

c.c.p.- Archivo

Por este conducto autorizamos al tesista:

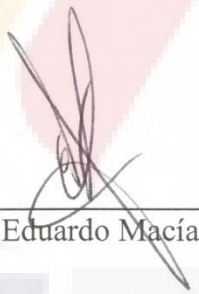
L.I. Alejandro Navarro Casillas

La impresión de su documento final de Tesis, ya que cumple con los requisitos de contenido y forma exigidos en la Universidad Autónoma de Aguascalientes.

Asesor

M.C. César Eduardo Velázquez Amador

Sinodales



M.C. Jorge Eduardo Macías Luévano

M.C. Francisco Javier Pinales Delgado

Agradecimientos:

"*Quiero agradecer en primer lugar a Dios por la oportunidad que me brindó de realizar mis estudios de maestría"*

"A Nancy, mi esposa, por su gran apoyo y comprensión los cuales me sirvieron para darme ánimos en los momentos difíciles."

Resumen

El desarrollo de software en nuestros días no es una tarea fácil, atrás quedaron los sistemas que constaban solamente de un ejecutable y el cual se distribuía por medio de un floppy o se descargaba de la red. Ahora en nuestros días, las empresas y organizaciones demandan de aplicaciones que abarquen mas niveles y satisfagan diferentes servicios. Este tipo de desarrollo de software incluye telecomunicaciones, outsourcing, y equipos de desarrollo geográficamente distribuidos (Hundhausen, 2006), sumándole a todo esto, la necesidad que ahora se tiene de dar cumplimiento a mayores y mejores estándares de calidad que sirvan de garantía de desempeño y funcionalidad hacia nuestros clientes o usuarios, para cada producto desarrollado.

En el presente trabajo de tesis se expone la propuesta de un proceso para la implementación y manejo de un sistema controlador de versiones dentro del Instituto Nacional de Estadística y Geografía (INEGI), que ayude a controlar los cambios que surjan durante el ciclo de vida de desarrollo de los sistemas, para tal efecto se hace una breve documentación de los marcos de referencia enfocándose en la administración de los cambios y/o la administración de configuración, las características básicas que debe contener un sistema administrador de configuraciones o sistema controlador de versiones, así como un análisis de los proyectos en los cuales se han utilizado estos tipos de sistemas, y cuáles han sido las herramientas implementadas dentro del Instituto.

Índice

Índice de figuras e ilustraciones

Índices de tablas

Título

Propuesta de un proceso de implementación y manejo de control de versiones en el INEGI.

Tema

Proponer un proceso que permita la implementación y manejo de un sistema controlador de versiones dentro del Instituto, enfocándose en el proceso de desarrollo de aplicaciones, y que se encuentre sustentado de acuerdo a lo que dicte el marco de referencia CMMI Nivel 2.

Palabras clave:

Control de versiones, Gestión de configuraciones, CMM, CMMI

Introducción

En todo proyecto de desarrollo de software existen cambios, y estos surgen en la medida en que el desarrollo se va llevando a cabo, algunas de las fuentes más comunes de los cambios dentro del ciclo de vida de desarrollo son:

- Cambio de los requerimientos.
- Avances en la Tecnología.
- Restricciones en los tiempos de entrega.
- Las expectativas de los usuarios.
- Oportunidades de mejora.

La administración de los cambios es el proceso de documentar y controlar los cambios que se realicen durante el ciclo de vida de los sistemas, para facilitar esta administración se recomienda el uso de sistemas controladores de versiones en el desarrollo de sistemas (Software Technology Support Center, 2005).

Cuando se trabaja con un equipo de desarrolladores que modifican concurrentemente el código, nos damos cuenta que es muy complicado juntar las distintas versiones del sistema que se encuentran en cada una de las máquinas de cada desarrollador, con la finalidad de realizar una liberación del sistema. Por más que se organicen correctamente las actividades o módulos que cada programador tendrá que desarrollar, es inevitable que alguno de ellos tengan que modificar el mismo archivo de código, inclusive en la misma rutina, procedimiento o método, llegando a un grado de concurrencia tal, que hayan modificado la misma line de código por más de una persona (Meli, 2005).

Es por esto que en el presente trabajo de tesis se realizará la propuesta de un proceso que permita la implementación y manejo de un sistema controlador de versiones como una herramienta de apoyo al ciclo de vida de los sistemas, enfocándose en el proceso de desarrollo de aplicaciones y que se encuentre sustentado bajo el marco de referencia de CMMI.

CAPÍTULO I FORMULACIÓN DEL PROBLEMA

I.1 Antecedentes

El desarrollo de sistemas es una tarea altamente creativa, cambiante y por lo general no es una tarea individual (Meli, 2005), un programador o equipo de programadores emplean la mayor parte de su tiempo haciendo pequeñas o grandes modificaciones para después deshacer esos cambios al día siguiente (Collins-Sussman, W. Fitzpatrick, & Pilato, 2004).

Al inicio la forma de trabajar era con la utilización de hojas de papel, pizarrones, un equipo de desarrolladores y un integrador, el integrador pasaban de un lugar a otro tratando de dar seguimiento al desarrollo, o para ver que módulo se encontraba en desarrollo y por cual desarrollador, revisando que tantos errores se habían corregido, para su vez, encontrar aún más. Por todo esto sobra decir que este proceso no estaba libre de errores y fue la razón por lo que los sistemas administradores de código fueron creados (Garvin, 2007).

Los cambios son una característica constante dentro del proceso de desarrollo de software. Eliminar los cambios sería eliminar la oportunidad de aprender, mejorar e incorporar nuevas y mejores tecnologías. Hacer a un lado la incorporación de los cambios puede significar caer en limitaciones tecnológicas y una rápida obsolescencia de los sistemas, lo cual en este, el mundo del la tecnología puede significar el fracaso de los desarrollos, inclusive antes de que estos sean terminados o liberados (Software Technology Support Center, 2005).

El modelo internacional CMMI SW/SE (Capability Model Integrated Software and Systems Engineering), comienza a desarrollarse en el año 2002, mejorando el CMM como consecuencia de una evolución lógica en el ámbito de la tecnología, ya que analiza el software unido a la gestión de proyectos tecnológicos.

Dicho modelo, permite a las organizaciones medir e incorporar mayores niveles de eficacia y madurez en sus procesos de desarrollo y mantenimiento de software, y está considerado como uno de los mayores referentes mundiales en cuanto a producción de software.

Al no utilizar CMMI, los proyectos pueden desviarse en plazos, incrementar sus costos y por supuesto, la satisfacción del cliente no es la esperada.

Por otra parte dentro del INEGI se cuentan con "Lineamientos sobre el Desarrollo de Software" el cual menciona la “Guía para el Desarrollo y Documentación de Software”, y que es una adaptación por parte del Instituto a la metodología Proceso de Desarrollo Unificado de Software o RUP, con la finalidad de estandarizar el desarrollo y la documentación de los sistemas dentro del INEGI (INEGI, 2006). Esta metodología adolece del uso de un sistema controlador de versiones como una herramienta estandarizada en el proceso de desarrollo de software, ya que solo propone el llenado de un único formato para la gestión del cambio[1], el cual no es suficiente para el adecuado seguimiento que se le debe de dar a los cambios que surgen dentro del un proyecto.

Los desarrollos en el Instituto cuentan con tiempos muertos, los analistas y arquitectos trabajan en el dar forma a un proyecto teniendo como base requerimientos no bien definidos, tratando de implementar una arquitectura de algo y sobre un desarrollo que probablemente cambie en el trascurso del tiempo, el grupo de pruebas se encuentra en espera de realizar una actividad o de que se les entreguen corregidas las versiones de código conforme a sus observaciones para poder seguir probando, los programadores pasan la mayor parte del tiempo ocupados añadiendo características que no se encuentran definidas en ningún documento, el grupo de infraestructura no tiene los elementos necesarios para asignar los recursos adecuados al proyecto y a su vez añaden restricciones a este, y por si todo esto no fuera poco, los usuarios llaman a cada momento añadiendo presión al desarrollo.

[1] **Apéndice E**, Formato control de cambios actual dentro del Instituto (situación actual)

I.2 Definiendo el Problema

I.2.1 Problemática

Se puede decir que una práctica generalizada en la mayoría de los desarrolladores de cualquier organización es que cuando inician un nuevo proyecto empieza a programar sin antes analizar y planear que es lo que se tiene que hacer, la mayoría de los programadores traducen su entendimiento de lo que se tiene que hacer en cientos o miles de líneas código, en ocasiones, este código se transforma y llega a ser algo funcional, pero por lo general se vuelve difícil de mantener, en otras muchas ocasiones los desarrollos ni siquiera logran su objetivo. Si al contrario de empezar a programar se invirtiera un poco más de tiempo y esfuerzo en la planeación antes de escribir siquiera una simple línea de código, entonces sería mucho más probable alcanzar los resultados esperados y más parecidos al requerimiento inicial (J Murphy, 2007).

El Instituto Nacional de Estadística y Geografía INEGI, tiene amplia experiencia en el desarrollo sistemas de software para el procesamiento de la información estadística y geográfica que capta la Institución (INEGI, 2006). Esta actividad, llevada a cabo durante años por el personal del Instituto, es una actividad clave para el logro del objetivo y las estrategias del INEGI, así mismo esta actividad debe estar alineada con la política de calidad con la que el INEGI cuenta, y la cual se refiera a:

> *"Todo producto o servicio que se genere en el INEGI debe tender a la plena satisfacción de las necesidades de información estadística y geográfica de la sociedad mexicana mediante el desarrollo de su personal y la mejora continua, privilegiando la integración de metodologías y tecnologías en sus procesos y proyectos."* (INEGI, 2000)

Es por esto que una simple falta u omisión en el proceso de desarrollo de software pudiera llegar a tener grandes consecuencias durante la operación o ejecución de los programas, aplicaciones o simplemente, durante la consulta de la información a través del sitio Web del Instituto.

Por lo tanto resulta necesaria la implementación de tecnologías que nos permitan llevar un control y administración del proceso de desarrollo de software.

Una barrea para la implementación de una nueva tecnología es el cómo los usuarios de esta la llegan a percibir, ya que en lugar de ser vista como una herramienta que facilite o ayude directamente en el desarrollo de su trabajo, esta puede mal interpretarse y ser tomada como un instrumento que entorpezca la forma en la cual vienen trabajando.

Con los sistemas controladores de versiones ocurre lo mismo, al inicio son vistos solamente como auditores de nuestro trabajo, que en lugar de ventajas acarrean desventajas, así como una herramienta que viene a agregar mayor complejidad y trabajo a nuestros cortos tiempos de desarrollo.

“El control de cambios es vital. Pero las fuerzas que hacen que sea necesario también lo hacen que sea molesto. Nos preocupamos por el cambio debido a que una perturbación diminuta en el código puede crear un gran fracaso en el producto. Pero también puede corregir una gran falla o permitir nuevas capacidades maravillosas. Nos preocupamos por el cambio debido a que un único desarrollador pícaro podría hundir el proyecto, sin embargo, las ideas brillantes se originan en las mentes de los pícaros, y un pesado proceso de control de cambio efectivamente podría disuadirlos de realizar un trabajo creativo.” (Bach, 1998)

Para grandes proyectos de software, los cambios no controlados rápidamente conducen al caos. Para tales proyectos el control de versiones combina procedimientos humanos y procedimientos automatizados, herramientas que proveen el mecanismo para controlar los cambios (Pressman, 2001).

1.2.2 Relevancia del caso

Demostrar en qué medida el uso de sistemas de control de versiones y su adecuada documentación dentro del Instituto resulta benéfico para los procesos de desarrollo de software.

Para tal efecto se realizará el estudió de los marcos de referencia CMM, CMMI, SPICE, etc. para la adecuada documentación y una comprensión adecuada de lo que implica el control de versiones, la gestión del cambio y/o gestión de la configuración de acuerdo a como lo definen dichos marcos de referencia.

Se realizará un análisis de la situación que guarda actualmente el Instituto para determinar el grado de madurez en materia de gestión de los cambios, que guardan los actuales desarrollos de software dentro del Instituto.

Paso seguido se realizará la evaluación de los sistemas automatizados que se encuentran actualmente en el mercado y que pueden ser implementados dentro del Instituto.

Ya para finalizar, se dará un dictamen y recomendaciones de las herramientas evaluadas, así como se propondrá el llenado de formatos para la adecuada documentación de la gestión de los cambios, todo, con base los marcos de referencia.

I.3 Justificación

Los cambios son inevitables cuando se desarrollan sistemas de software. Y estos cambios van incrementando el nivel de complejidad en el desarrollo a medida en que van aumentando los involucrados dentro del proyecto, La confusión surge cuando los cambios no son analizados de manera adecuada antes de que se realicen, o cuando se generan cambios antes de su implementación, no reportando aquellos cambios necesarios de saber, o que sean controlados de manera tal que mejoren la calidad y reduzcan el error en el proceso de desarrollo de software (Pressman, 2001).

La administración de los cambios es una actividad que se debe aplicar a través del proceso de desarrollo de software. Esto debido a que los cambios pueden ocurrir en cualquier momento, dichas actividades nos permiten (1) identificar el cambio, (2) controlarlo, (3) asegurarnos de que el cambio es apropiadamente implementado, y por ultimo (4) reportar el cambio a cualquier otro que esté interesado en el mismo (Pressman, 2001).

Con la implementación y fomento de uso de un sistema controlador de versiones dentro del Instituto, se pretende reducir el esfuerzo necesario para hacer frente a los cambios que surjan dentro del ciclo de vida de desarrollo de sistemas, permitiendo realizar las entregas de los sistemas de manera controlada, reduciendo al máximo los errores de integración y programación, fomentando el trabajo en equipo y la colaboración y comunicación entre el equipo de desarrolladores.

I.4 Objetivos, Preguntas y Proposiciones del Caso o Proyecto

I.4.1 Objetivo General

Desarrollar un procedimiento que sirva como base para la implementación y manejo de un sistema controlador de versiones dentro del Instituto, recomendando mediante la investigación y su respectivo análisis, cual sistema controlador de versiones es el más adecuado para su uso dentro del Instituto, aunado a esto, dicho proceso deberá estar debidamente sustentado bajo el marco de referencia CMMI.

Objetivo Específico 1

Desarrollar un procedimiento detallado que permita la implementación de un sistema controlador de versiones con base al marco de referencia CMMI nivel 2 dentro del Instituto.

Objetivo Específico 2

Desarrollar un procedimiento detallado que permita el manejo de un sistema controlador de versiones con base al marco de referencia CMMI nivel 2 dentro del Instituto.

Objetivo Específico 3

Determinar en qué grado cumple el proceso de control de versiones actual con relación a lo que se define en el marco de referencia CMMI nivel 2.

Objetivo Específico 4

Investigar que sistemas comerciales, tanto en su modalidad de software libre o propietario, se recomiendan para su uso como herramienta de control de versiones en el INEGI.

1.4.2 Preguntas

A continuación se describen las preguntas que motivan la presente investigación y que fundamentan el trabajo.

Pregunta 1: ¿Cómo se lleva a cabo el control de versiones en el Instituto y en qué grado cumple el proceso actual con lo que dicta el marco de referencia CMMI nivel2?

Pregunta 2: ¿Qué actividades se deben considerar en el proceso propuesto para la implementación de un sistema controlador de versiones, que de cumplimiento al marco de referencia CMMI nivel 2 en cuanto a gestión del cambio se refiere?

Pregunta 3: ¿Qué actividades se deben considerar en el proceso propuesto para el manejo de un sistema controlador de versiones, que se encuentre debidamente alineado al marco de referencia CMMI nivel 2 en cuanto a gestión del cambio se refiere?

Pregunta 4: ¿Cuál de los sistemas de control de versiones analizado es el más apegado al CMMI?

1.4.3 Proposiciones

Proposición 1: El proceso que se lleva actualmente en el INEGI para control de los cambios no contempla el uso de un sistema de control de versiones.

Proposición 2: El proceso que se lleva actualmente en el INEGI no cumple con lo estipulado en CMMI.

Proposición 3: El proceso propuesto para la implementación de un sistema controlador de versiones complementa al proceso actual.

Proposición 4: El proceso propuesto para el manejo del control de versiones complementa al proceso actual.

Proposición 5: Siguiendo los procedimientos propuestos se tendrá un mayor control de versiones sobre los cambios que surjan en los desarrollos de software dentro del Instituto.

CAPÍTULO II MARCO TÉORICO

II.1 El INEGI

Derivado de la reforma a los artículos 26 y 73 de la Constitución Política de los Estados Unidos Mexicanos, misma que se publicó en el Diario Oficial de la Federación (DOF) el 7 de abril de 2006, se estableció que el Estado Mexicano contará con un Sistema Nacional de Información Estadística y Geográfica (SNIEG), en donde la responsabilidad de normar y coordinar dicho sistema estará a cargo de un organismo con autonomía técnica y de gestión, personalidad jurídica y patrimonio propios.

En consecuencia, y con el fin de reglamentar las modificaciones mencionadas, se publicó el 16 de abril de 2008, en el DOF, la Ley del Sistema Nacional de Información Estadística y Geográfica (LSNIEG), en la cual se establecen las disposiciones generales para organizar y establecer el SNIEG, al mismo tiempo se establece que el Instituto Nacional de Estadística y Geografía (INEGI) obtiene su autonomía técnica y de gestión, personalidad jurídica y patrimonio propios y es el responsable de normar y coordinar al Sistema Nacional.

Los objetivos del Instituto son:

- Generar e integrar información estadística y geográfica sobre el territorio, la población y la economía de México;
- proporcionar a la sociedad el servicio público de información estadística y geográfica; así como
- normar, coordinar y promover el desarrollo de los Sistemas Nacionales Estadístico y de Información Geográfica, con el objeto de
- satisfacer las necesidades de información de los diversos sectores de la sociedad.

Misión

Generar, integrar y proporcionar información estadística y geográfica de interés nacional, así como normar, coordinar y promover el desarrollo de los Sistemas

Nacionales Estadístico y de Información Geográfica, con objeto de satisfacer las necesidades de información de los diversos sectores de la sociedad.

Visión

México pertenece al grupo de países que basan su desarrollo en el uso de la información y en el conocimiento organizado y diseminado electrónicamente al contar con un Sistema Nacional de Información Estadística y Geográfica sustentado en una Red Nacional de Información, que facilita la toma de decisiones de todos los sectores de la sociedad con base en información oportuna y confiable.

El INEGI es responsable de coordinar el Sistema Nacional de Información Estadística y Geográfica, así como la Red Nacional de Información.

Organigrama

Ilustración 1 Organigrama del INEGI

II.1.1 Área del INEGI en donde se va a realiza el estudio

El área dentro del INEGI donde se realizara el estudio es en la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones (DIDTIC) perteneciente a la Dirección General de Innovación y Tecnologías de Información (DGITI).

Dirección General de Innovación y Tecnologías de Información

La Dirección General de Innovación y Tecnologías de Información, perteneciente a la Dirección General de Administración, (ver Ilustración 2) tiene las facultades de:

- Emitir los lineamientos que en materia de informática deberán observar las dependencias y entidades de la Administración Pública Federal, en su carácter de integrantes de los Servicios Nacionales de Estadística y de Información Geográfica y los Sistemas Nacionales Estadístico y de Información Geográfica.
- Promover y Coordinar tecnologías y metodologías que impulsen las competencias de los recursos humanos del Instituto; así como el Sistema de Gestión de Calidad, conforme al plan estratégico institucional y a los lineamientos que establezca el Presidente del INEGI.

Ilustración 2 Dirección General de Administración

Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones.

Objetivo

Dirigir y coordinar los procesos de investigación y desarrollo en tecnologías de información y comunicaciones y de diagnostico y difusión de tecnologías para contribuir a satisfacer las necesidades de la información de los usuarios internos y externos al instituto.

Función

La función principal de la DIDTIC es la de dirigir y coordinar los procesos de investigación y desarrollo de tecnologías de información y comunicación, computo mayor y menor, software de sistemas de información, herramientas y metodologías de desarrollo, herramientas para explotación de la información, de bases de datos y mecanismos de la protección de la información.

Dirigir y coordinar y dirigir los procesos de integración, edición, publicación y divulgación de la investigación tecnológica.

Establecer el plan estratégico de investigación institucional en materia de tecnologías informáticas y de comunicación en acuerdo con las áreas informáticas del instituto.

Dirigir y coordinar las actividades de investigación en materia de tecnología informática y comunicación con otras instituciones.

Dirigir y controlar las actividades para la generación de contenido informático y sistemas para la intranet Institucional y para el sitio del Instituto en Internet.

II.2 Conceptos Teóricos en los que se base el Proyecto

II.2.1 Concepto de Calidad

Es un dicho común el de que no hay dos copos de nieve iguales, ciertamente cuando vemos nevar, es difícil imagina que cada copo de nieve es distinto, no es si no que hasta que se utilizan poderosos lentes de aumento cuando se pueden notar las diferencias entre uno y otro elemento observado.

Este fenómeno "*Variación entre ejemplos*", se aplica a todos los productos o artefactos desarrollados por los humanos.

El control de la variación es el corazón de la calidad y es en la industria manufacturera donde se pretende minimizar la variación entre los productos fabricados (Pressman, 2001).

¿Pero como esto aplica al proceso de desarrollo de software? ¿Qué es lo que tiene que hacer el desarrollador de software para controlar la variación?

II.2.1.1 Calidad

La calidad se puede definir como un conjunto de características o atributos que tiene las cosas. Como un atributo de un elemento, la calidad se refiere a las características que pueden ser medibles, cosas que son susceptibles a comparación (Pressman, 2001).

II.2.1.2 Aseguramiento de la Calidad

El aseguramiento de la calidad consiste en auditarías, reportes y la gestión. El objetivo del aseguramiento de la calidad es el de proveer a la administración con los datos necesarias para que se encuentre informada acerca de la calidad del producto. Por supuesto, si al proveer esta información para el aseguramiento de la calidad se identifican problemas, es responsabilidad de esta manejar estos problemas aplicando lo necesario para que sean resueltos (Pressman, 2001).

II.2.2 Administración del Cambio

La administración del cambio ha sido definida y clasificada en varios estándares y modelos de proceso, con la finalidad de describir y regular las expectativas del proceso de la administración del cambio y como el proceso es implementado en las organizaciones (Rahikkala, 2000).

Administración del cambio es el propósito sistemático de justificar, evaluar, coordinar, aprobar o desaprobar e implementar todos los cambios aprobados de un sistema después de su implantación.

¿Por qué administrar los cambios? Un cambio incontrolado nos lleva al caos, los cambios controlados habilitan al sistema a cambiar de forma que puedan ser acomodados efectiva y eficientemente, ambos en tiempo y costo permitiendo una planeación de defectos reportados y/o peticiones de mejoras al sistema (Jones, 2000).

II.2.3 Control de versiones

II.2.3.1 ¿Qué es un control de versiones?

Un sistema de control de versiones, es un conjunto de herramientas que permiten gestionar los cambios en un archivo o un conjunto de archivos a través del tiempo. Un sistema típico de control de versiones incluye un *repositorio* central que contiene la versión actual del todos los archivos. El *repositorio* también almacena documentación de todos los cambios realizados en los archivos con información necesaria para poder restaurar cualquier *versión* pasada. Cada desarrollador del sistema típicamente mantiene una *copia de trabajo* (o varias) en su equipo (Clifton, Kaczmarczyk, & Mrozek, 2007).

El primer sistema de control de versiones fue introducido como un producto comercial y seguido por dos populares sistemas *Source Code Control System (SCCS) y Revision Control System (RCS)*, La principal característica de estos sistemas que llegaron a ser parte de muchos sucesores son:

1. Soporte para le evolución sobre el tiempo en un esquema de almacenamiento deltas o cambios de una versión a otra, y así consecutivamente.
2. Control sobre concurrencia de versiones, a través de una disciplina de check-out y check-in en los archivos, asegurando una escritura en un tiempo determinado.

La evolución del control de versiones, de un modelo centralizado sobre un solo archivo bajo una estructura de árbol, la cual se encuentra compartida en un tiempo determinado, a un sistema distribuido, ha sido conducido por los cambios en los ambientes computacionales como la migración de los desarrollos de software de las estaciones de trabajo, hacia otros sistemas con interfaces de usuarios mucho más sofisticadas que incluyen herramientas visuales (Wein, Cowan, & Gentleman, 1992).

En casi todos los sistemas controladores de versiones, el *repositorio* es un lugar centralizado que almacena la copia maestra de todas las versiones de nuestros archivos de proyecto. Algunos utilizan bases de datos como repositorio, algunos otros utilizan archivos regulares y algunos otros utilizan una combinación de ambos.

En el pasado, el *repositorio* y todos sus usuarios tenían que compartir en una máquina el sistema de archivos. Convirtiéndose en una limitante, ya que resultaba difícil para los desarrolladores trabajar en sitios diferentes, o en diferentes tipos de máquinas o sistemas operativos.

La Figura 1 muestra de manera gráfica un ejemplo de un control de versiones en su forma más simple.

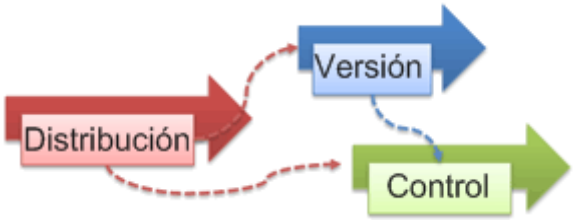


Figura 1 Control de versiones (Azad, 2009)

Muchos desarrolladores de software profesionales consideran el control de versiones una parte esencial del desarrollo de software.

Los cambios, por lo regular son identificados por un número o letra identificados como "numero de revisión", "nivel de revisión" o simplemente "revisión" por ejemplo al inicio, el conjunto de archivos son la "revisión 1". Cuando el primer cambio es realizado, el resultado de ese conjunto de archivos sería la "revisión 2" y así consecutivamente. Cada revisión es asociada dentro de un intervalo de tiempo y con la persona que realizó el cambio. Las revisiones pueden ser comparadas, restauradas y en algunos tipos de controladores de versiones pueden ser fusionadas.

Características de los sistemas controladores de versiones

La mayoría de los esfuerzos del desarrollo que son llevados en el sector comercial, emplean algunos sistemas o ambientes para gestionar documentos, en particular esos sistemas gestionan y mantienen el registro del historial y la evolución de las versiones o revisiones de los documentos. El término sistema administrador de versiones es utilizado para denotar el concepto general de un sistema el cual incluye la siguiente funcionalidad:

- Mantiene el historial de las modificaciones de los documentos, artefactos o sistemas, incluyendo la habilidad de ver una versión previa o el estado del proyecto para ver las diferencias entre dos diferentes versiones, y deshacer los cambios para recuperar la versión previa de un sistema,
- soporte de desarrollo colaborativo, incluyendo resolución de conflictos, mecanismo empleado cuando dos o más desarrolladores realizan cambios que ocasionan conflictos al mismo archivo y
- soporte a la gestión de ramas del sistema bajo el desarrollo dos o más líneas de desarrollos paralelas y una combinación (merging) selectiva de características de los desarrollos paralelos (Beck, 2005).

Tradicionalmente los sistemas controladores de versiones están conformados por un "repositorio central", pero en algunos de los más recientes, cuentan con "repositorios distribuidos"

a) Repositorio central (ver Figura 2) es simple, ya que consta de un único lugar en el cual cada desarrollador en el cual puede obtener (check-out) el archivo con el cual trabajara, modificarlo y subir sus cambios (check-in) al repositorio central para que queden disponibles a todos los miembros del equipo.

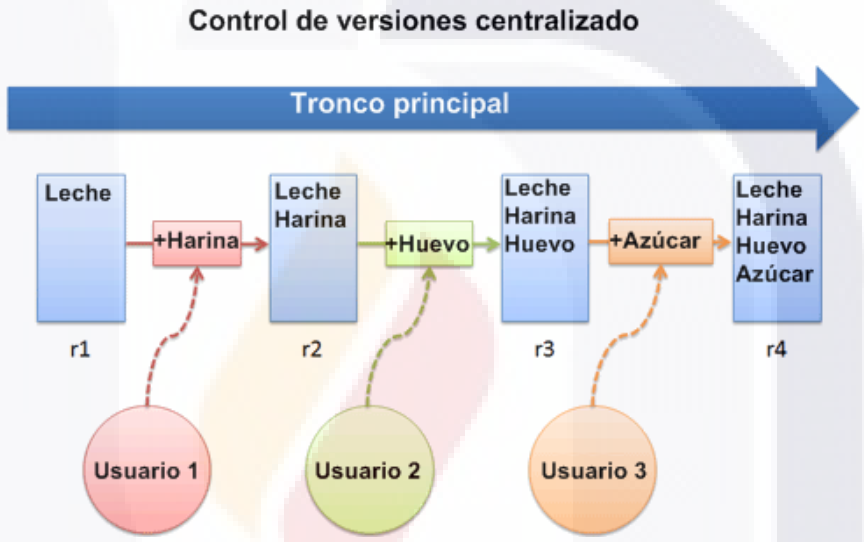


Figura 2 Control de versiones centralizado (Azad, 2009)

En este ejemplo, todos los miembros del equipo suben sus cambios al tronco principal para conformar un archivo final, el *"Usuario 1"* añade a la lista la Harina, el *"Usuario 2"* añade Huevo a la lista y finalmente, el "*Usuario 3"* agrega el Azúcar.

Puntos a considerar:

- Cuando un desarrollador realiza un *check-out* sobre el archivo, para realizar un cambio en este, los demás desarrolladores no podrán editarlo hasta que el primer desarrollador suba o comprometa sus cambios al repositorio central (tronco principal).

- Bajo este modelo, cada desarrollador contará con la versión actualizada del proyecto.

b) Repositorio distribuido: Bajo este modelo cada desarrollador (ver Figura 3) cuenta con su propio repositorio y los cambios están presentes al inicio solamente dentro del repositorio de cada usuario. Por lo tanto, este modelo se enfoca en la compartición de los cambios, cada cambio tiene un único identificador o un identificador grupal.

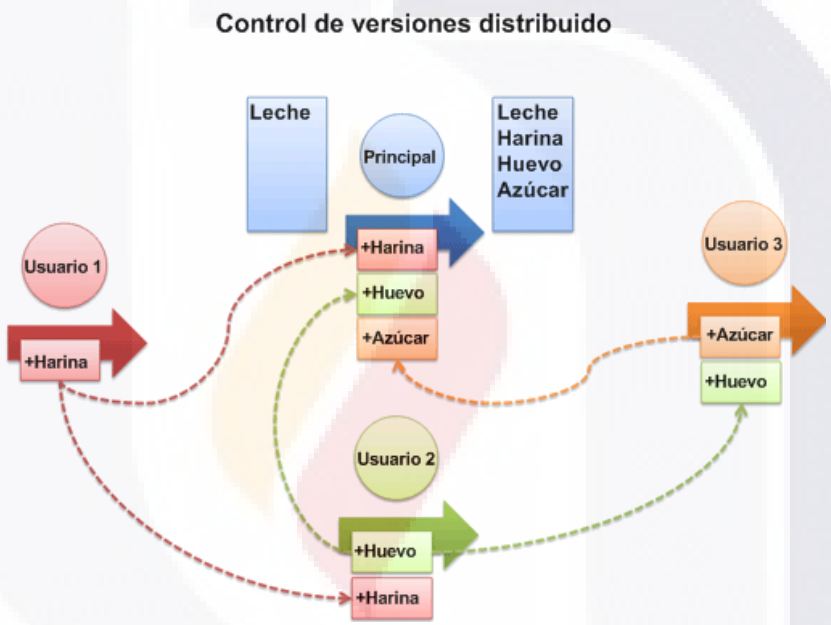


Figura 3 Control de versiones distribuido (Azad, 2009)

En este ejemplo el *"Usuario 1"* puede cambiar o deshacer los cambios en su repositorio local, y no es hasta que sube sus cambios al repositorio común y cuando los demás miembros del equipo se sincronizan con el repositorio común cuando se ven reflejados los cambios de cada usuario.

La elección de un modelo dependerá en gran medida a la forma del proyecto y el equipo de desarrollo, cada uno de los modelos serán abordados con mayor detalle en puntos posteriores (Vesperman, 2006).

1. El sistema controlador de versiones deberá proveer soporte a múltiples desarrolladores.

2. El sistema deberá soportar el "acceso remoto" al repositorio o en su caso la conexión remota hacia todos los repositorios que se encuentren distribuidos.

3. Exportación. Un conjunto de datos exportados son aquellos que no deberían contener datos de administración del sistema controlador de versiones, la exportación crea un “liberación” del código del proyecto la cual puede ser una publicación.

4. "Hooks": Un hook es un lugar en donde se almacenan scripts, estos scripts por lo regular se ejecutan al momento de comprometer (*commit*) algún cambio ayudando así a la fase de procesamiento del control de versiones. Este concepto será desarrollado mas a detalle en secciones posteriores de este documento.

5. Otra característica que debe tener un sistema controlador de versiones es la utilización de "etiquetas" para identificar claramente de una versión a otra.

6. "Branches" o ramas. Esta característica permite que se pueda manejar simultáneamente dos o más versiones del mismo proyecto.

7. Commits atómicos o compromisos atómicos, esta característica asegura que mientras una persona está comprometiendo un cambio en el repositorio (subiendo un cambio), cualquier otro desarrollado no pueda realizar *commit* en ese mismo tiempo.

8. Autenticación. En un ambiente de desarrollo con múltiples desarrolladores la característica de autenticación es necesaria para poder otorgar permisos de acceso según el usuario, y así poder llevar un mejor control de nuestro código.

Usos de sistemas controladores de versiones en el desarrollo de software

Cuando se desarrollan sistemas de información, cualquier elemento que forma parte del proyecto de desarrollo puede ser almacenado dentro del *repositorio* del sistema controlador de versiones. Muchos desarrolladores o lideres de proyecto piensan que gestionar solamente el código fuente es la parte más importante de los sistemas controladores de versiones, “ya que de ahí viene su nombre” (Thomas & Hunt, 2003). Por una parte es cierto, pero muchos cometen el error de olvidar algunas otras cosas que son necesarias y que también deben ser gestionadas bajo un sistema controlador de versiones (Thomas & Hunt, 2003), como lo pueden ser los script necesarios para la correcta compilación del proyecto, los archivos de configuración, o bien los script de la base de datos en los cuales se determina el estado de nuestros objetos a un determinado tiempo de nuestro desarrollo. Estos scripts son una parte fundamental para la construcción y configuración de nuestro proyecto, sin poder recuperar esto, sería casi imposible ensamblar la aplicación, por lo tanto y en medida de lo posible estos deben ser también gestionados en el sistema controlador de versiones.

La fase de desarrollo es la más común para el uso de sistemas controladores de versiones. Después del liberar una versión inicial de un programa, dos versiones usualmente necesitan de ser mantenidas:

a) La nueva versión que podría eventualmente convertirse en próximo liberación y
b) La versión de bugfix o arreglo de errores de la versión actual.

II.2.3.2 Modelos de control de versiones

La misión principal de un sistema de control de versiones es permitir la edición colaborativa y la compartición de los datos. Sin embargo, existen diferentes sistemas que utilizan diferentes estrategias para alcanzar este objetivo (Collins-Sussman et al., 2004).

Bloqueo-modificación-desbloqueo

Muchos sistemas de control de versiones utilizan un modelo de bloqueo-modificación-desbloqueo para atacar este problema. En un sistema como éste, el repositorio sólo permite a una persona modificar un archivo al mismo tiempo.

El problema con el modelo bloqueo-modificación-desbloqueo es que es un tanto restrictivo y a menudo se convierte en un obstáculo para los usuarios (Collins-Sussman et al., 2004).

Copiar-modificar-mezclar

En este modelo, el cliente de cada usuario se conecta al repositorio del proyecto y crea una copia de trabajo personal—una réplica local de los archivos y directorios del repositorio. Los usuarios pueden entonces trabajar en paralelo, modificando sus copias privadas. Finalmente, todas las copias privadas se combinan (o mezclan) en una nueva versión final. El sistema de control de versiones a menudo ayuda con la mezcla, pero en última instancia es un ser humano el responsable de hacer que esto suceda correctamente.

La solución copiar-modificar-mezclar puede sonar un tanto caótica, pero en la práctica funciona extremadamente bien. Los usuarios pueden trabajar en paralelo, sin tener que esperarse el uno al otro. Cuando trabajan en los mismos archivos, sucede que la mayoría de sus cambios concurrentes no se solapan en absoluto; los conflictos son poco frecuentes. El tiempo que toma resolver los conflictos es mucho menor que el tiempo perdido por un sistema de bloqueos (Collins-Sussman et al., 2004).

II.2.3.3 Algunos conceptos básicos

El presente apartado trata de cubrir los conceptos básicos de las tareas y operaciones que se realizan de manera similar en la mayoría de los sistemas controladores de versiones analizados.

Aplicando cambios

El comprometer un cambio dentro del repositorio es quizá una de las tareas más simples (ver Figura 4), ya que se trata de guardar los cambios que se realicen a nuestro código, dentro del repositorio, cada vez que se comprometa un cambio se generará una nueva revisión de nuestro proyecto (r1, r2, r3, r4).

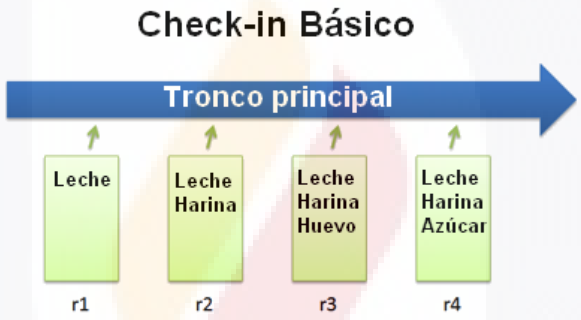


Figura 4 Check-in básico (Azad, 2009)

Una buena práctica que se debe tener presente al momento de comprometer un cambio, es la de agregar un comentario que sea lo suficientemente descriptivo acerca del movimiento que se está realizando.

Obtención de la última versión para su edición.

Esta acción quizá representa la funcionalidad básica de cualquier sistema controlador de versiones, la Figura 5 muestra de forma gráfica esta actividad.

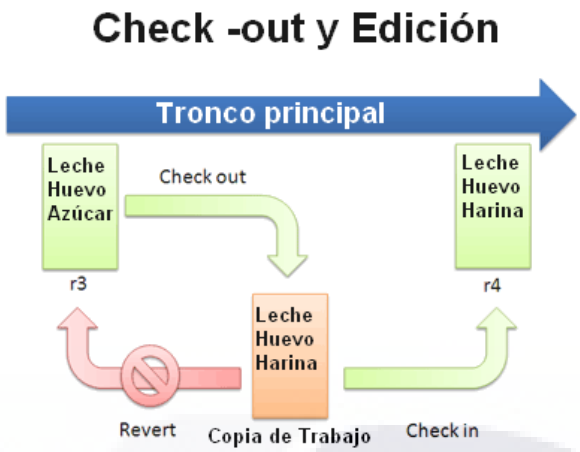


Figura 5 Obteniendo y editando (Azad, 2009)

Con esta acción se obtienen los cabios que se hayan generado sobre un archivo en especifico o sobre todo el proyecto desde la última vez que se sincronizó nuestra copia de trabajo con el repositorio, dándonos la oportunidad de realizar cambios y comprometerlos para solucionar algún problema, o bien simplemente aplazar o revertir los cambios si es que estos no satisfacen del todo a las políticas de calidad y/o las necesidades propias del cambio que se requiere realizar.

Historial de cambios

Al editar un archivo y comprometer sus cambios, por una o varias personas, el sistema controlador de versiones mantiene un historial de los cambios (ver Figura 6), en el cual nos muestra él cómo el archivo va evolucionando a través del tiempo.

Figura 6 Historial de cambios (Azad, 2009)

Por ejemplo, en la imagen se muestra la diferencia que existe entre la primera revisión del archivo (r1) y la segunda revisión (r2) es de que a la lista de ingredientes se le agregó la palabra “Huevo”. De la tercera revisión (r3) a la cuarta revisión, podemos observar que de la lista de ingredientes se eliminó la palabra “Azúcar” y se agregó la palabra “Harina”

Muchos sistemas controladores de versiones almacenan solamente las diferencias, en lugar de las copias completas de los archivos, logrando con esto una mejor administración de espacio en disco.

Ramas o bifurcaciones

Hacer una rama de un proyecto u archivo, da la facilidad de poder trabajar con una copia de nuestro mismo código (ver Figura 7), sin tener que alterar la versión estable o probada de nuestro código, dándolos la facilidad de añadir nuevas características a nuestro desarrollo (Thomas & Hunt, 2003).

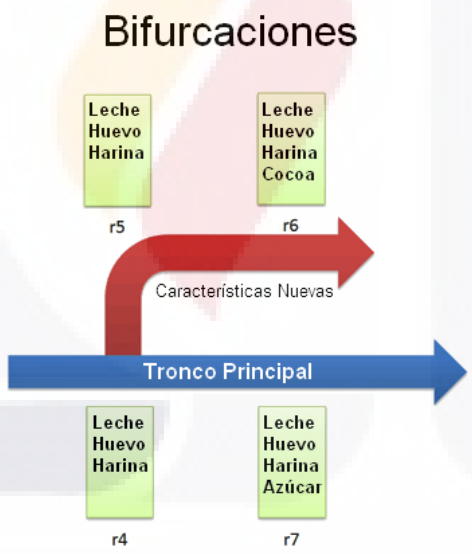


Figura 7 Bifurcaciones (Azad, 2009)

Por eso se tiene que pensar en las bifurcaciones como versiones beta de nuestros archivos o proyecto, las cuales tarde que temprano tendrán que ser probadas y fusionadas a nuestra versión principal de nuestro proyecto.

Fusionando cambios

Las bifurcaciones suenan simples, pero que sucede cuando las nuevas características se requieren que sean incluidas dentro de la versión principal, para esto los sistemas controladores de versiones fusionan los cambios realizados en las ramas o bifurcaciones con los cambios realizados en la versión principal (Thomas & Hunt, 2003), de forma gráfica esto se puede ver en la Figura 8.

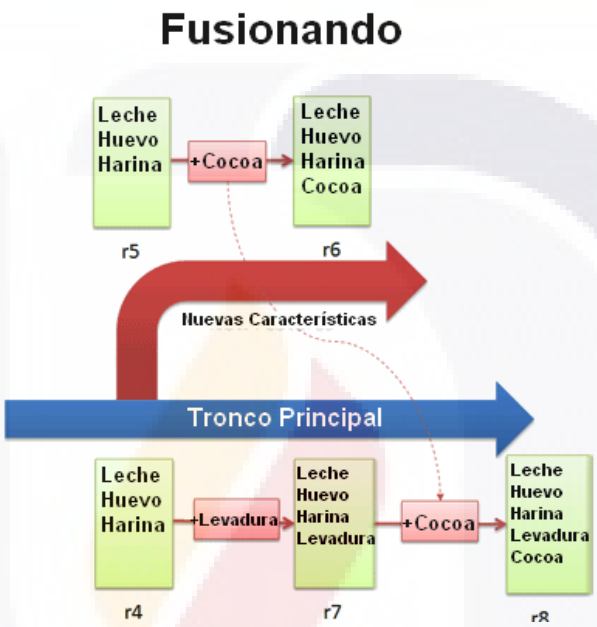


Figura 8 Fusionando cambios (Azad, 2009)

Se debe tener en cuenta que no todos los sistemas controladores de versiones cuentan con herramientas automáticas para poder realizar esta actividad, siendo necesaria la intervención del propio desarrollador.

Resolución de conflictos

En muchas ocasiones el sistema controlador de versiones pude fusionar automáticamente varios cambios, sin embargo cuando esto no sucede, surgen los conflictos, y estos deben ser solucionados manualmente por el usuario. Un conflicto es cuando más de un usuario/desarrollador modifica la misma línea de código en la misma revisión, la Figura 9 muestra de forma gráfica esta actividad.

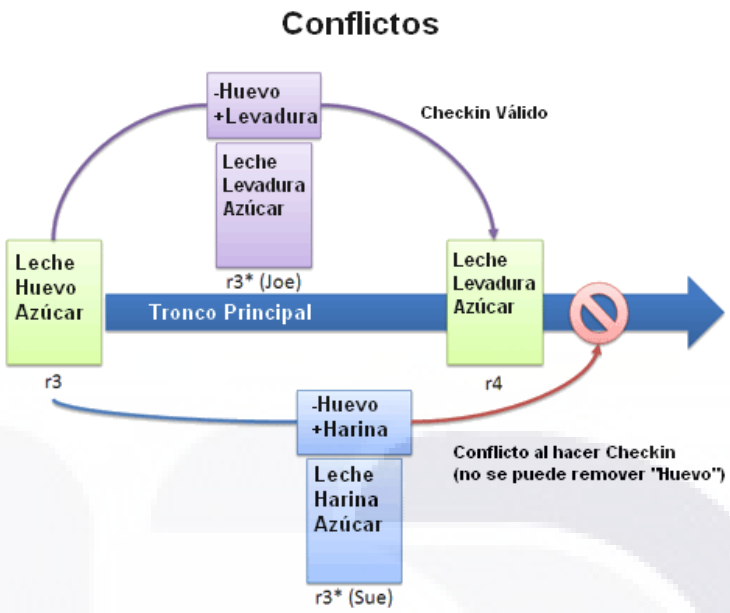


Figura 9 Resolución de conflictos (Azad, 2009)

Al igual que la fusión de cambios, la resolución de conflictos es por lo general una actividad en la cual debe intervenir el criterio del desarrollador, aceptando o descartando los cambios que causen conflicto sobre la revisión que se esté modificando.

Etiquetado

El etiquetado consiste en básicamente nombrar cualquier revisión para su fácil referencia, la Figura 10 muestra de forma gráfica dicha actividad.

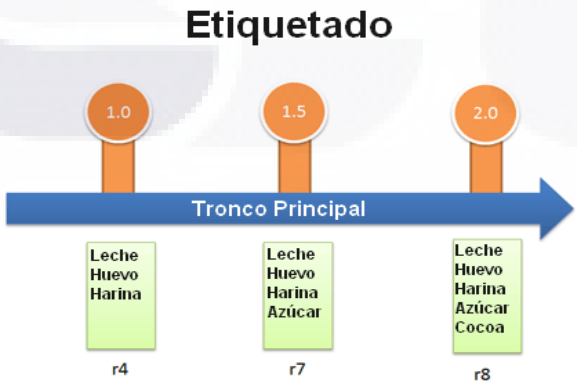


Figura 10 Etiquetado (Azad, 2009)

II.2.4 Marcos de referencia

El software es una herramienta que establece las dinámicas laborales, de producción y hasta de convivencia en todo el mundo. Los múltiples desarrollos que en este ámbito se dan, generan como consecuencia la necesidad de establecer cánones de calidad para cada producto, para así garantizar que su desempeño y funcionamiento cubran las expectativas de sus consumidores y que a la par cumplan con su cometido satisfactoriamente (Oktaba, 2003).

Los marcos de referencia surgen con la misión de proporcionar liderazgo y promocionar el estado de la práctica de la ingeniería de software mediante la mejora de la calidad de los sistemas que dependen del software. En el SEI ha habido un fuerte énfasis en el tratamiento de las tareas de desarrollo de software como procesos que pueden ser definidos, practicados, medidos y mejorados (Dunaway & Masters, 1996).

El propósito de contar con un marco de referencia es apoyar al proceso de desarrollo de software dentro del Instituto, en su tránsito de su estado actual, en el cual, la calidad de los desarrollos depende de las habilidades de los individuos, a el estado deseado, en donde la calidad de los desarrollos de software serán la consecuencia de la definición de procesos (Oktaba, 2003) y usos de buenas prácticas en el INEGI.

II.2.4.1 CMM (The Capability Maturity Model for Software)

El modelo fue generado a través de la experiencia colectiva de los proyectos más exitosos de *software*, generando así un conjunto de prácticas importantes que deben ser implantados por cualquier entidad que genera o mantiene *software* (Álvarez Rodríguez et al., 2008).

El modelo CMM fue desarrollado por el Software Engineering Institute (SEI), dándolo a conocer en 1994 y se ha popularizado por la noción de medir la madurez de los procesos de software de las organizaciones.

Objetivo

El objetivo de este modelo es evaluar los procesos en sus distintos niveles de madurez, identifica los niveles a través de los cuales una organización debe formarse para establecer una cultura de excelencia en la Ingeniería de Software, partiendo de la premisa de que los procesos sin una apropiada fundamentación fallan (Álvarez Rodríguez et al., 2008).

Características del modelo

CMM describe los principios y prácticas que ayudan a las organizaciones de desarrollo de software a mejorar su madurez en sus procesos de software en términos de direccionamiento evolutivo, desde un proceso ad hoc y caótico a posesos disciplinados (Paulk, Michael D. Konrad, & Garcia, 1995), CMM presenta un marco de trabajo representado por guías de recomendaciones para organizaciones de *software* que buscando las siguientes actividades.

- ✓ Identificar fortalezas y debilidades en la organización.
- ✓ Identificar los riesgos de seleccionar entre diferentes contratos y monitorear los mismos.
- ✓ Entender las actividades necesarias para planear e implementar los procesos de software.

- ✓ Ayudar a definir e implementar proceso de software en la organización a través de una guía.

Los procesos son evaluados a través de distintos niveles de madurez, que van desde prácticas desordenadas o descoordinadas, hasta lograr una mejora continua de procesos (Álvarez Rodríguez et al., 2008), para lo cual se clasifican en cinco niveles de madurez:

1. **Inicial:** Describe a la organización dentro de un inmaduro e indefinido proceso.
2. **Repetible:** Administración de requerimientos, planeación de proyectos de software, seguimiento a proyectos de software, administración de la subcontratación de software, aseguramiento de la calidad, administración de la configuración del software.
3. **Definido:** Enfocado al proceso organizacional, definición de procesos de la organización, programas de capacitación, administración integral de software, productos de ingeniería de software, puntos de revisión, coordinación inter grupal.
4. **Administrado:** Administración y medición del proceso, administración de calidad, prevención de defectos.
5. **Optimizado:** Innovación tecnológica, administración del proceso del cambio.

Para una mayor comprensión, la Figura 11 muestra de manera gráfica los cinco distintos niveles de madurez definidos en este marco de referencia.

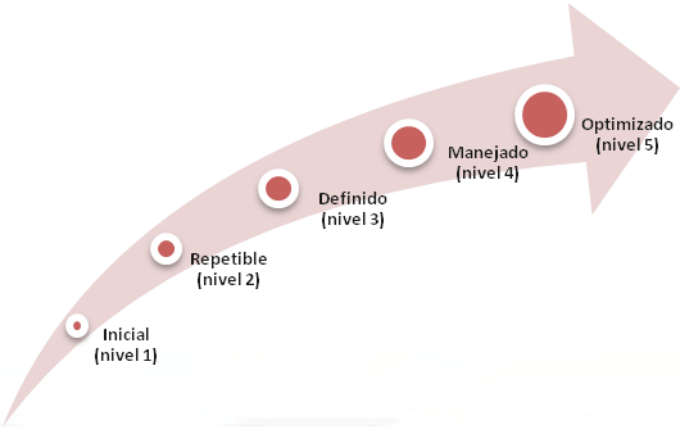


Figura 11 Niveles de madurez de los procesos de software

El principal objetivo para la mayoría de las organizaciones es alcanzar el nivel 3 de madurez.

Así es como el modelo CMM establece una medida del progreso, conforme al avance en niveles de madurez. Cada nivel a su vez cuenta con un número de áreas de proceso que deben lograrse. Cada uno de los restantes Niveles de Madurez están compuesto por un cierto número de Áreas Claves de Proceso (ver Tabla 1), conocidas a través de la documentación del CMM por su sigla inglesa: KPA.

Cada KPA identifica un conjunto de actividades y prácticas interrelacionadas, las cuales cuando son realizadas en forma colectiva permiten alcanzar las metas fundamentales del proceso. Las KPAs pueden clasificarse en 3 tipos de proceso: Gestión, Organizacional e Ingeniería.

Nivel	Key Process Area	Definición
Inicial		Algunas prácticas son estables El éxito depende del esfuerzo heroico de los individuos
Repetible	Administración de requisitos. Planeación de proyecto de software Seguimiento y control del proyecto de software Administración de subcontratos de software Aseguramiento de calidad de software Administración de la configuración de software	Se establecen y documentan las estimaciones y la planeación de los proyectos, es decir se centra en los proyectos y su administración. Los problemas de los proyectos son identificados y corregidos
Definido	Enfoque en procesos de la organización Definición de procesos de la organización Programación de capacitación Administración integral del software Ingeniería de productos de software Coordinación inter grupal Revisión entre colegas	Integración de los procesos de administración e ingeniería para la organización. Los problemas se anticipan y se previenen o bien su impacto es minimizado
Administrado	Administración cuantitativa del proceso Administración de la calidad del software	Los procesos son entendidos y establecidos cuantitativamente. Las fuentes de problemas individuales se conocen y se eliminan
Optimizado	Prevención de defectos Administración del cambio de tecnología Administración de cambio de proceso	Los procesos son sistemáticamente mejorados. Las fuentes de los problemas comunes son entendidas y eliminadas.

Tabla 1 CMM áreas clave de proceso

Como vemos, las áreas claves de proceso dentro del conjunto del nivel de madurez 2 de CMM (Repetible) incluyen los requisitos de gestión, la planificación de proyectos de software, el seguimiento de la planificación de proyectos y de supervisión, y software de gestión de subcontratos. Prácticas integrales son la gestión de la configuración del software y la garantía de calidad de software (Mallette, 2005).

Administración de la Configuración del Software (SCM)

El modelo Capability Maturity Model (CMM) define a la administración de la configuración de software *Software Configration Managment (SCM)* como una área clave de proceso (KPA), la cual provee practicas detalladas de control, gestión, y gestión de los desarrollos de software, teniendo como propósito establecer y mantener la integridad de los productos del proyecto de software durante todo su ciclo de vida.

La administración de la configuración del software involucra la identificación de la configuración del mismo (los productos de trabajo de software seleccionados y sus descripciones) en determinado momento, controlado cambios a la configuración y manteniendo la integridad y la posibilidad de dar seguimiento a la configuración durante el ciclo de vida del proyecto de software. Los productos de trabajo colocados bajo la SCM incluyen los productos de software que son entregados al cliente (el documento de requerimientos y el código) y los productos que se identifican o requieren para crear esos productos de software.

Se establece una bibliotecas de líneas base del software conforme vayan siendo desarrolladas. Los cambios a la línea base y la liberación de los productos de software a partir de la creación de la biblioteca de líneas base del software son controlados mediante las funciones de control de cambios del SCM (Álvarez Rodríguez et al., 2008).

Metas del SCM:

1. Planear las actividades de la administración.
2. Identificar, controlar y poner a disposición los productos de trabajo de software seleccionados.
3. Controlar los cambios a los productos de trabajo de software identificados.
4. Informar el estado y contenido de las líneas base del software a los grupos y a las personas afectadas.

Actividades (Ilustración 3) de la KPA de Administración de la Configuración del Software.

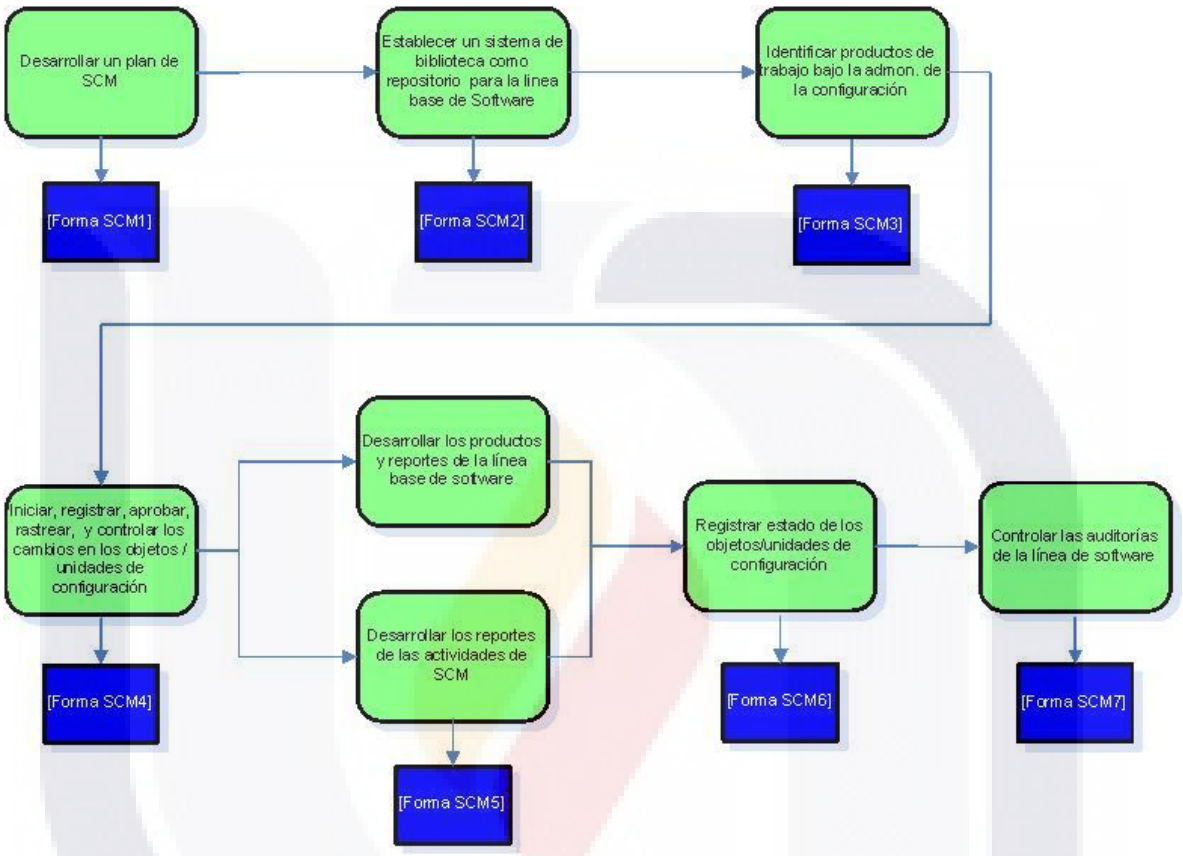


Ilustración 3 Actividades CM en CMM (Álvarez Rodríguez et al., 2008)

Los formatos SCM1 al SCM7 sugeridos en la Ilustración 3 Actividades CM en CMM (Álvarez Rodríguez et al., 2008) podrán ser consultados en el *apéndice F* del actual trabajo de tesis.

II.2.4.2 CMMI (Capability Maturity Model Integration)

Desde 1991, los CMM se han desarrollado para innumerables disciplinas. Algunas de las más notables comprenden modelos para la ingeniería de sistemas, la ingeniería del software, la adquisición del software, el desarrollo y la gestión del personal, y el desarrollo integrado de productos y procesos.

Aunque estos modelos han probado ser útiles para muchas organizaciones en el seno de diferentes industrias, el uso de múltiples modelos ha sido problemático. Sin embargo, las diferencias entre los modelos de disciplinas específicas utilizados por cada grupo, incluyendo su arquitectura, contenido y aproximación, han limitado las capacidades para generalizar con éxito sus mejoras (Chrissis, Mike Konrad, & Shrum, 2006).

El proyecto de integración de CMM ha sido realizado para regular el problema de utilizar múltiples CMM. La misión inicial del equipo del producto CMMI (CMMI Product Team) fue combinar tres modelos fuente:

1. SW-CMM (Capability Maturity Model for Software), version v2.0 draft C .
2. SECM (Systems Engineering Capability Model.
3. IPD-CMM (Integrated Product Development Capability Maturity Model), version v0.98.

Estos tres modelos fuente fueron seleccionados debido a su extensa adopción por las comunidades de desarrollo de sistemas y de software y también porque proponen diversos acercamientos a la mejora de procesos en el seno de una organización.

Apoyarse en estos modelos populares y largamente apreciados ha permitido al equipo de producto del CMMI crear un conjunto coherente de modelos integrados, que pueden adoptarse tanto por aquellos que usan actualmente los modelos fuente como por aquellos nuevos al concepto de CMM. Así, el CMMI resulta directamente de la evolución de los modelos SW-CMM, SECM e IPD-CMM (Chrissis et al., 2006).

Capability Maturity Model Integration (CMMI) es un modelo para la mejora de procesos de software que proporciona a las organizaciones los elementos esenciales para procesos eficaces.

El modelo CMMI cuenta con dos representaciones distintas: continua y escalonada o por etapas.

Cada organización puede optar por la que se adapte a sus características y prioridades de mejora. La visión continua de una organización, mostrará la representación de nivel de capacidad de cada una de las áreas de proceso del modelo. La visión escalonada definirá a la organización, dándole en su conjunto un nivel de madurez del 1 al 5.

La representación escalonada

La representación escalonada (ver Ilustración 4) se centra en la mejora de la capacidad de los procesos en una organización. Cuenta con cinco niveles de madurez, cada nivel provee los fundamentos para la mejora del siguiente nivel (Kulpa & Johnson, 2003).

1. **Inicial:** Representa un proceso de madurez representado por resultados impredecibles.

   En el nivel de madurez 1, los procesos suelen ser ad hoc y caóticos. La organización no suele proporcionar un entorno estable. El éxito en estas organizaciones depende de la competencia y el heroísmo de la gente en la organización y no en el uso de procesos probados. A pesar de este ad hoc y caótico medio ambiente representativo del nivel de madurez 1, las organizaciones suelen producir productos y servicios funcionales, sin embargo, con frecuencia superan el presupuesto y el calendario de sus proyectos (CMMI Product Team, 2002).

2. **Manejado:** Representa un proceso de madurez caracterizado por un repetible desempeño del proyecto. La compañía utiliza disciplinas de la fundación para la administración de requerimientos, planeación de

proyectos, monitoreo y control de proyectos, administrador de proveedores, aseguramiento de la calidad de procesos y productos, gestión de la configuración y análisis/administración. Para el nivel 2, el proceso clave es a nivel de proyecto y prácticas.

Se dice que una organización se encuentra en el nivel de madurez 2 *manejado* cuando ha logrado todos los objetivos específicos y genéricos de las áreas de proceso del nivel de madurez 2. En otras palabras, los proyectos de la organización han asegurado que los requisitos son gestionados y que los procesos son planificados, realizados, medidos y controlados (CMMI Product Team, 2002).

La disciplina del proceso se refleja en el nivel de madurez 2 ayudando a garantizar que las prácticas actuales se mantienen durante momentos de estrés. Cuando estas prácticas están en su lugar, los proyectos se efectúan y gestionan de acuerdo con sus planes documentados (CMMI Product Team, 2002).

3. **Definido:** Representa el nivel de madurez caracterizado por mejorar el desempeño de los proyectos dentro de la organización.

   Una organización se encuentra en el nivel 3 de madurez, cuando ha logrado todos los objetivos específicos y genéricos de las áreas de proceso asignadas a los niveles de madurez 2 y 3. En el nivel de madurez 3, los procesos están bien caracterizados y comprendidos, y se describen en las normas, procedimientos, herramientas y métodos (CMMI Product Team, 2002).

4. **Cuantitativamente administrado:** Representa el proceso de madurez caracterizado por mejorar el desempeño de la organización.

   Se dice que una organización se encuentra en el nivel de madurez 4, cuando ha logrado todos los objetivos específicos de las áreas de proceso asignados a los niveles de madurez 2, 3 y 4 y los objetivos genéricos

asignados a los niveles de madurez 2 y 3. Subprocesos son seleccionados, que contribuyen significativamente al rendimiento global del proceso. Estos subprocesos seleccionados son controlados mediante técnicas cuantitativas y de estadísticas.

Los objetivos cuantitativos para la calidad y el rendimiento del proceso se han establecido y son utilizados como criterios en la gestión de procesos. Los objetivos cuantitativos se basan en las necesidades de los clientes, los usuarios finales, la organización, y en la ejecución del proceso. Calidad y rendimiento de los procesos son comprendidos en términos estadísticos y son administrados durante toda la vida de los procesos (CMMI Product Team, 2002).

5. **Optimizado:** Representa el nivel de madurez caracterizado por una rápida reconfiguración de la organización como mejoramiento continuo de los procesos.

   En el nivel de madurez 5, una organización ha logrado todos los objetivos específicos de las áreas de proceso asignados a los niveles de madurez 2, 3, 4 y 5 y los objetivos genéricos asignados a los niveles de madurez 2 y 3. Los procesos son continuamente mejorados, basándose en una comprensión cuantitativa de las causas comunes de variación inherentes a los procesos.

   El nivel de madurez 5 se centra en la mejora continua del rendimiento de los procesos a través de mejoras tecnológicas, ya sea incremental e innovadoras. Proceso cuantitativo de los objetivos de mejora de la organización se han establecido, modificándose constantemente para reflejar los cambios de los objetivos de negocio, y se utiliza como criterio en la gestión de mejora de procesos. Los efectos de la mejora de los procesos utilizados son medidos y evaluados en el proceso de los objetivos cuantitativos de mejora. Tanto los procesos definidos, como el conjunto

procesos estándar establecidos en la organización son los objetivos de las actividades de mejora (CMMI Product Team, 2002).

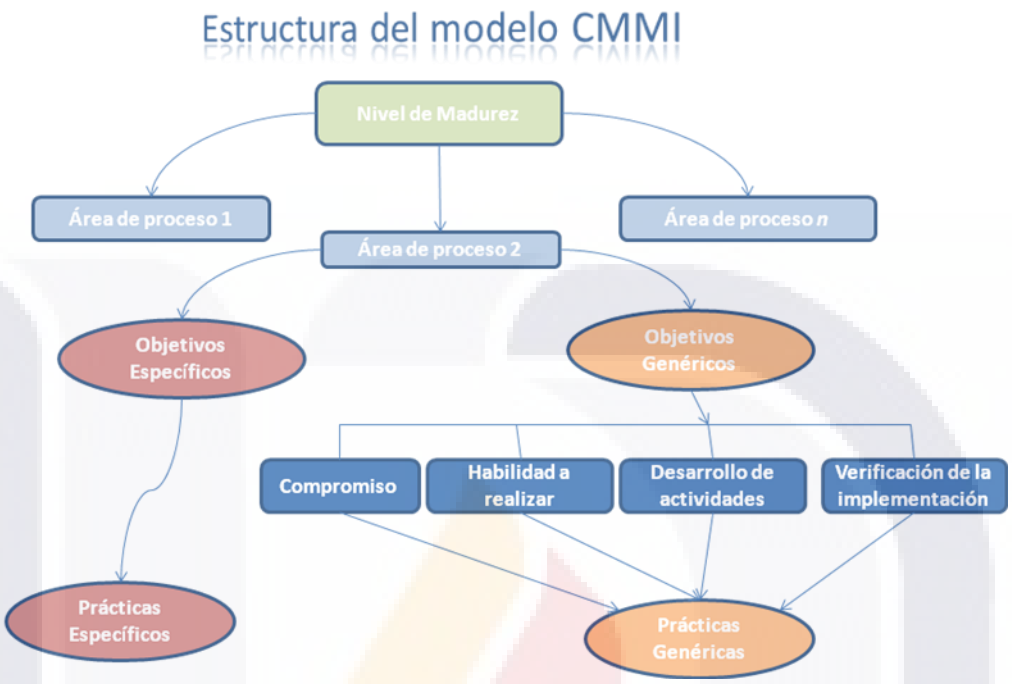


Ilustración 4 CMMI representación escalonada. (Kulpa & Johnson, 2003)

La representación continua

La representación continua (ver Ilustración 5) cuenta básicamente con la misma información que la representación escalonada, aunque un poco diferente. La representación continua se enfoca en la mejora de los procesos, en acciones a ser completadas dentro de las áreas de proceso, el proceso y sus acciones pueden abarcar diferentes niveles. Estos niveles son llamados *Niveles de capacidad* y existen seis niveles (Kulpa & Johnson, 2003).

- **Nivel 0 Incompleto:** Un proceso incompleto no implementa toda la especificación del nivel de capacidad 1
- **Nivel 1 Realizado:** Un nivel de capacidad 1 es un proceso del cual se espera que desarrolle todas las especificaciones del nivel 1 de capacidad y sus prácticas genéricas. El desempeño quizá no sea estable y no conocen

objetivos específicos como calidad, costos y calendarización, pero regularmente el trabajo es hecho.

- **Nivel 2 Manejado:** Un proceso manejado es planeado, desarrollado, monitoreado y controlado por proyectos individuales, o por proyectos standalone para lograr la obtención de un propósito.
- **Nivel 3 Definido:** El proceso está definido en la organización y se ejecuta siempre.
- **Nivel 4: Cuantitativamente Manejado:** Un proceso cuantitativamente manejado es un proceso definido que es controlado usando estadísticas y otras técnicas cuantitativas.
- **Nivel 5 Optimizado:** Un proceso optimizado es cuantitativamente manejado y es mejorado, basado en un entendimiento de causas comunes de inherentes a la variación del proceso. Se centra en la mejora continua del desempeño de los procesos a través del la mejora incremental y la innovación.

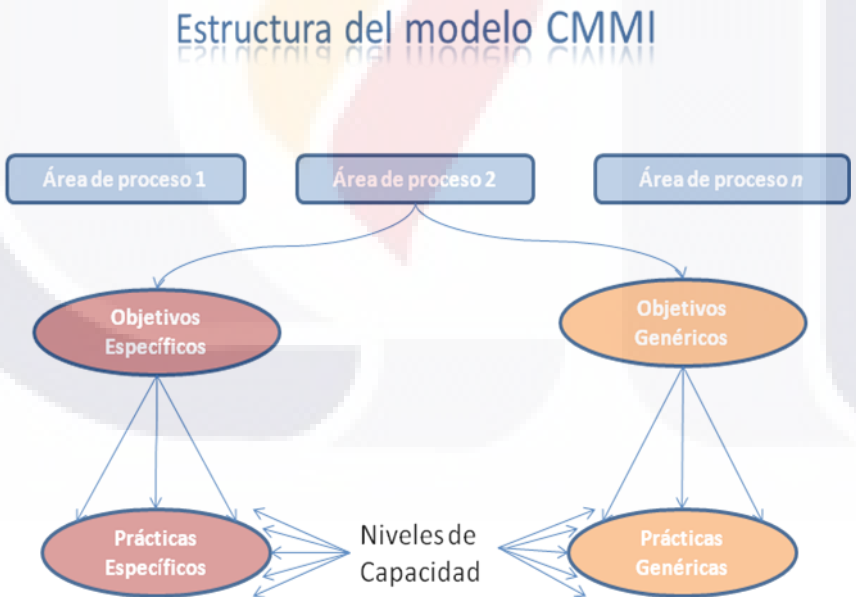


Ilustración 5 CMMI representación continua.(Kulpa & Johnson, 2003)

Aéreas de proceso

Un área de proceso es un conjunto de prácticas relacionadas en un área que, cuando se realiza colectivamente, satisfacen un conjunto de objetivos considerados importantes para hacer una mejora significativa en esa área.

Objetivos Específicos

Los objetivos específicos se aplican a un área de proceso y dirigen las características únicas que describen lo que deben aplicarse para satisfacer el área de proceso. Los objetivos específicos son componentes del modelo requeridos y se utilizan en las evaluaciones para ayudar a determinar si se cumple un área de proceso.

Prácticas específicas

Una práctica específica es una actividad que se considera importante en la consecución del objetivo específico asociado. Las prácticas específicas describen las actividades que se espera que resulte en el logro de los objetivos específicos de un área de proceso.

Implementación de software para gestión de configuraciones en CMMI

Administración de la configuración

El objetivo de la administración de la configuración es establecer y mantener la integridad de los productos de trabajo mediante la identificación de configuración, control de la configuración, lo que representa el estado de configuración, y auditorías de configuración (CMMI Product Team, 2002).

El primer paso para la implementación y manejo de sistema controlador de configuraciones como una campaña para alcanzar el Nivel 2 dentro de la organización, es dar seguimiento a las siguientes áreas de proceso:

- Identificar los elementos de configuración. Los elementos de configuración pueden consistir de: archivos de ayuda, documentos, documentación de

requerimientos, código fuente, etc. que componen las líneas base en los puntos que figuran en el tiempo.

- El control de cambios en la configuración de elementos.
- Construyendo o proveyendo especificaciones para crear productos de trabajo desde el sistema de gestión de la configuración.
- Mantener la integridad de las líneas de base.
- Proporcionar situación exacta y datos de configuración actuales para los desarrolladores, usuarios finales y clientes.

El trabajo de poner los productos bajo una administración de la configuraciones, incluyen los productos que se entregan a los clientes, los productos designados a trabajos internos, productos adquiridos, herramientas y otros elementos que se utilizan en la creación y descripción de estos productos de trabajo (CMMI Product Team, 2002).

Una vez identificados los elementos de configuración, las organizaciones necesitan almacenarlas en un sistema de gestión de configuraciones o controlador de versiones. Donde el sistema de gestión de configuraciones sea el responsable de seguir y controlar los cambios en los elementos de configuración en un repositorio central (Jacobs, 2004).

Ejemplos de productos de trabajo que deben ser manejado bajo un sistema de gestión de la configuración:

- Planes.
- Descripciones de procesos.
- Requerimientos.
- Datos de diseño.
- Gráficas.
- Especificaciones de los productos.
- Códigos.
- Compilados.

- Archivos de datos de los productos.
- Publicaciones técnicas de los productos.

Objetivos de la gestión de la configuración

La Gestión de la Configuración es considerada parte de la categoría de apoyo. Su propósito es de establecer y mantener la integridad de los productos de trabajo mediante la configuración de identificación, control de configuración, lo que representa el estado de configuración, y auditorías de configuración.

Objetivos

- SG1 Establecimiento de la línea base
  - SP1.1 Identificación y control de elementos
  - SP1.2 Establecer un sistema de gestión de la configuración
  - SP1.3 Crear o lanzar líneas bases.
- SG2 Seguimiento y control de los cambios
  - SP2.1 Rastreo de solicitudes de cambio
  - SP2.2 Control de elementos de configuración
- SG3 Establecimiento de la integridad.
  - SP3.1 Establecer registros de administración de la configuración
  - SP3.2 Realizar auditorías de configuración.

SG1 Establecimiento de línea base

SP1.1 Identificando elementos de configuración. Un “elemento de configuración” es una entidad designada por la gestión de la configuración, el cual debe consistir de múltiples productos de trabajo relacionados que forman la línea base.

Productos de trabajo típicos

1. Identificación de elementos de configuración.

Sub prácticas

1. Selección de los elementos de configuración y productos de trabajo que los componen basándose en criterios documentados.
2. Asignación de identificadores únicos para los elementos de configuración.
3. Especificar las características importantes de cada elemento de configuración.
4. Especificar cuando cada elemento de configuración es puesto bajo el administrador de configuración.
5. Identificar el responsable para cada elemento de configuración.

SP1.2 Establecer un sistema de control de la configuración. Un sistema de gestión de la configuración incluye los medios de almacenamiento, los procedimientos, y las herramientas para acceder al sistema de configuración.

Productos de trabajo típicos

1. Sistema administrador de la configuración que controle los productos de trabajo.
2. Sistema administrador de la configuración.
3. Bases de datos de peticiones de cambios.

Sub prácticas

1. Establecimiento de un mecanismo para administrar múltiples niveles de control.
2. Almacenamiento y recuperación de elementos de configuración en el sistema de administración de la configuración.
3. Compartir y transferir elementos de configuración entre los niveles de control dentro del sistema administrador de la configuración.
4. Almacenar y recobrar versiones archivadas de elementos de configuración.
5. Almacenamiento, actualización y recuperación de registros.
6. Creación de reportes de administración de la configuración desde el sistema administrador de configuración.

7. Preservar el contenido del sistema administrador de la configuración.
8. Revisión como sea necesario de la estructura del administrador de configuración.

SP1.3 Crear o liberar líneas base. Una línea base es un conjunto de especificaciones o productos de trabajo que han sido formalmente revisados y acordados que posteriormente sirven como base para un mayor desarrollo, y que sólo se puede modificar a través de procedimientos de control de cambios.

Productos de trabajo típicos

1. Línea base.
2. Descripción de líneas base.

Sub prácticas

1. Obtención de autorización del control de configuración antes de crear o liberar líneas base o elementos de configuración.
2. Creación o liberación de líneas base solamente dese elementos de configuración en el sistema administrador de configuración.
3. Documentar el conjunto de elementos de configuración que son contendidos en una línea base.

SG2 Seguimiento y control de los cambios.

Los cambios en los productos de trabajo en el marco de gestión de configuración son seguidos y controlados

SP2.1 Seguimiento de las solicitudes de cambios

Las solicitudes de cambio no solo tratan las nuevas necesidades o modificaciones, sino que también los fracasos y defectos en los productos de trabajo.

Las solicitudes de cambio son analizadas para determinar el impacto que el cambio tendrá en el producto de trabajo, productos relacionados con el trabajo, y el calendario y el costo.

Productos de trabajo típicos

1. Cambios de solicitudes.

Sub prácticas

1. Iniciar y registrar las peticiones de cambio en la base de datos de peticiones de cambio.
2. Analizar el impacto de los cambios y correcciones propuestas en las peticiones de cambio.
3. Revisar las peticiones de cambio, que serán tratadas en la siguiente línea base, con las partes interesadas relevantes y conseguir su acuerdo.
4. Seguir el estado de las solicitudes de cambio hasta su cierre.

SP2.2 Control de elementos de configuración

El control se mantiene sobre la configuración de la línea de base del producto de trabajo. Este control incluye el seguimiento de la configuración de cada uno de los elementos de configuración, la aprobación de una nueva configuración, si es necesario, y la actualización de la línea base.

Productos de trabajo típicos

1. Historial de revisiones de los elementos de configuración.
2. Archivos de la línea base.

Sub prácticas

1. Control de cambios de los elementos de configuración durante toda la vida del producto.
2. Obtener la autorización apropiada antes de que los elementos de configuración cambiados sean introducidos en el sistema de gestión de configuración.
3. Check-in y check-out de los elementos de configuración del sistema de gestión de la configuración para la incorporación de los cambios de forma

que mantengan la exactitud y la integridad de los elementos de configuración.

4. Realizar revisiones para garantizar que los cambios no han causado efectos no deseados sobre las líneas de base.
5. Se registran los cambios de los elementos de configuración y las razones de los cambios, según proceda.

SG3 Establecimiento de la integridad

La integridad de las líneas base, establecida por los procesos asociados con la meta específica "*Establecer líneas base*", y mantenida por los procesos asociados con la meta específica "*Seguir y controlar los cambios*", se proporciona por las prácticas específicas de esta meta específica.

SP3.1 Establecer registros de administración de la configuración.

Establecer y mantener los registros que describen los elementos de configuración.

Productos típicos de trabajo:

1. Historial de revisión de los elementos de configuración.
2. Registro de cambios.
3. Copia de las peticiones de cambio.
4. Estado de los elementos de configuración.
5. Diferencias entre líneas base.

Sub prácticas

1. Registrar las acciones de gestión de configuración con el suficiente detalle para que el contenido y el estado de cada elemento de configuración sea conocido y las versiones anteriores se pueden recuperar.
2. Asegúrese de que los interesados tengan acceso a los conocimientos del estado de la configuración de los elementos de configuración.
3. Especificar la versión más reciente de las líneas de base.

4. Identificar la versión de los elementos de configuración que constituyen una línea base particular.
5. Describir las diferencias entre las líneas de base sucesivas.
6. Revisar el estado y el historial (es decir, los cambios y otras acciones) de cada elemento de la configuración como sea necesario.

SP3.2 Realizar auditorías de configuración

Las auditorías de configuración confirman que el resultado de las líneas base y de la documentación están conformes con un estándar o requerimiento especificado.

Productos de trabajo típicos

1. Resultados de la auditoria de configuración.
2. Elementos de acción.

Sub prácticas

1. Evaluar la integridad de la línea base.
2. Confirmar que los registros de configuración identifican correctamente los elementos de configuración.
3. Revisar la estructura y la integridad de los elementos en el sistema de gestión de la configuración
4. Confirmar la integridad y exactitud de los elementos en el sistema de gestión de la configuración.
5. Confirmar el cumplimiento con las gestión de configuración aplicables a los procedimientos y estándares.
6. Seguimiento a los puntos de acción de la auditoria del cierre.

Lista de Métricas

1. Número de peticiones de cambio o junta de cambio procesados por unidad de tiempo.
2. Milestones alcanzados por la actividad de gestión de la configuración comparado con el plan.

3. Trabajo completado, esfuerzos esperados y fundamentos esperados en las actividades de gestión de la configuración.
4. Número de cambios de elementos de configuración.
5. Número de arreglos regresados como “Aun no reparados”.
6. Número de arreglos regresados como “No se pudo reproducir el error”.
7. Número de reportes de problemas pendientes contra la taza de reparación.
8. Número de veces que las personas cuentan con una versión equivocada de la línea base.
9. Número de cambios por categoría al código fuente y a la documentación de soporte.
10. Número de cambios por categoría, tipo y severidad.
11. Líneas de código almacenadas en librerías almacenadas bajo el control de la configuración.

La gestión de la configuración no es solo el establecimiento de librerías para los archivos para poder migrar los archivos de etapas anteriores a posteriores. No es solamente comprar herramientas que nos puedan decir cuando los archivos se han movido, o que un cambio ha ocurrido. La gestión de la configuración es acerca de la definición de elementos de configuración. Cuando un sistema está comenzando desde su diseño, este puede ser descompuesto en muchas piezas. Estas piezas pueden ser los archivos, los datos dentro de esos archivos, programas, reportes, y módulos de integración.

La gestión de la configuración comprende también el establecimiento de la línea base. La línea base es básicamente cuando uno dibuja una línea en la arena y dice, “está bien, cualquier desarrollo que pase esta punto o cualquier cambio que pase este punto debe de ir a través de alguna corta revisión oficial antes de que sea incorporada dentro del resto del sistema”. CMMI no nos dice cuando se tiene que establecer la línea base, solamente nos fomenta a hacerlo (Kulpa & Johnson, 2003).

Una herramienta puede ser utilizada para manejar aquellos cambios que realicemos, pero una herramienta no es tan sofisticada como para decirnos que cambios fueron y como estos cambios afectan el resto del sistema. Por lo tanto las personas deben ser involucradas en la revisión y en el análisis de los efectos de cualquier cambio.

La gestión de la configuración incluye la definición de los elementos de configuración, desarrollando o implementando herramientas de soporte, técnicas, procedimientos y medios de almacenamiento, generando y manteniendo la línea base, manejando los cambios solicitados y controlándolos, documentando los resultados de los esfuerzos de la gestión de la configuración, y desarrollando auditorias de configuración (Kulpa & Johnson, 2003).

II.2.4.3 SPICE (Software Process Improvement and Capability dEtermination)

SPICE Software Process Improvement and Capability dEtermination, es un modelo de mejora de procesos software desarrollado en el ESI que integra en sus procesos, las prácticas necesarias para cumplir los requisitos de la norma ISO 9001:2000, proporcionándole a la industria la habilidad de poder utilizar estos estándares en muchas formas (Paulk et al., 1995).

- En la capacidad del modo de determinación para ayudar a la organización en sus compras para determinar la capacidad de un potencial proveedor de software.
- En el modo de mejora de procesos, para ayudar a las organizaciones de desarrollo de software para mejorar sus propios desarrollos de software y mantener el proceso y
- En el modo de auto aseguramiento, para ayudar a una organización a determinar si cuenta con la habilidad para implementar nuevos proyectos de software.

La dimensión de procesos de SPICE se compone de cinco categorías de procesos. (ver Ilustración 6) Una categoría de procesos es un conjunto de procesos relacionados.

- Cliente - Proveedor: Consiste en un conjunto de cinco procesos que se refieren a la relación con el cliente como por ejemplo dar soporte al cliente.
- Ingeniería: Agrupa aquellos procesos directamente relacionados con la implementación o mantenimiento de sistemas y productos de software y su documentación (7 procesos).
- Soporte: Consiste en un grupo de procesos, que sirven de ayuda a los demás procesos a lo largo de todo el ciclo de vida (8 procesos).
- Gestión: Es un conjunto de procesos que contiene practicas para gestionar los productos y procesos (4 procesos).

- Organización: Son procesos que establecen los objetivos de la organización donde se aplican el resto de los procesos (5 procesos).

El modelo de referencia en la dimensión de procesos nos indica el propósito y los resultados de cada uno de ellos.

Ilustración 6 El modelo SPICE

SPICE maneja seis niveles de madurez, cada nivel de capacidad o madurez tiene dos atributos de procesos (Process attributes). Menos el nivel 1 que solo tiene un atributo de proceso.

Los atributos de procesos son los indicadores que ayudan al evaluador a puntuar el nivel de madurez de los procesos.

- Nivel 0: Proceso Incompleto, el proceso no está implementado, o no consigue sus propósitos.
- Nivel 1: Proceso realizado, el proceso implementado consigue los propósitos definidos.
- Nivel 2: Proceso Gestionado, el proceso genera productos de calidad aceptable dentro de los plazos definidos y los recursos asignados.
- Nivel 3: Proceso Establecido, el proceso se realiza usando un proceso definido basado en los principios de la ingeniería de software.
- Nivel 4: Proceso Predecible, el proceso se realiza consistentemente dentro de los límites de control definidos para conseguir sus objetivos.
- Nivel 5: Proceso Optimizado, el proceso optimiza sus ejecuciones para alcanzar necesidades actuales y futuras de negocio y conseguir repetidamente los objetivos de negocio definidos.

Gestión del cambio en SPICE

Dentro del marco de referencia de SPICE podemos encontrar la gestión del cambio o de la configuración dentro del *proceso de soporte*, y es dicha categoría la que consta de los procesos que permiten y apoyan al rendimiento de los otros procesos en un proyecto.

Es dentro del proceso de Soporte o Apoyo donde se realizaran las siguientes actividades:

- Desarrollo de la documentación
- Realización de la Gestión de la Configuración
- Realización del Aseguramiento de la Calidad
- Verificación
- Validación
- Llevar a cabo auditorias
- Realización de la Resolución de Problemas

II.2.4.4 ITIL

Las Librerías de Infraestructura de Tecnologías de Información, ITIL, Son un conjunto de herramientas clave en el uso de tecnologías de información soportadas en los procesos organizacionales y la filosofía del negocio que permiten a las organizaciones proyectarse en la estandarización y en la reestructuración de los modelos ya implementados de un área de tecnologías de información.

ITIL, son un amplio conjunto de documentos de asesoramiento sobre las mejores prácticas de gestión elaborado por la Oficina de Comercio de Reino Unido (OGC) para la prestación de la calidad de los servicios de TI, tomando como base el Manejo del Servicio en toda su dimensión. Actualmente, se ha difundido en el nivel internacional en todos los sectores económicos como herramienta administrativa y de gestión necesaria para incursionar exitosamente en múltiples ámbitos de competitividad (Ortiz Núñez & Hoyos Franco, 2005).

Gestión del Cambio

El proceso de gestión del cambio proporciona un mecanismo para controlar y gestionar la iniciación, ejecución y revisión de todos los cambios propuestos a la infraestructura operativa, a fin de minimizar el impacto del cambio sobre los incidentes relacionados con la prestación de servicios.

ITIL define seis pasos claves incluidos en el proceso de gestión del cambio, estos procesos son:

1) Surgimiento y registro del cambio: Este paso incluye la creación de la petición del cambio y el registro del cambio *(para lo cual recomienda el uso de herramientas SCM)*.
2) Revisión del cambio: Es un “intenso chequeo” que usualmente es desempeñado por el administrador de cambios, capturando todos los detalles para el aseguramiento de que las peticiones no se dupliquen.

3) Asignar cambios: Se debe revisar el impacto de los cambios, observando todos los recursos involucrados, riesgos y los planes. A los cambios se les asigna la prioridad y su plan de solución de riesgo es confirmado.
4) Autorización del cambio: bajo ITIL el cambio es formalmente autorizado por el administrador de cambios, Él o Ella, deben comunicar, deben aprobar o rechazar la decisión.
5) Coordina la implementación: El contacto Técnico del cambio (usualmente el solicitante) es responsable por reunir los recursos solicitados y completar los cambios de acuerdo a los detalles aprobados. Es responsable de asegurarse que el cambio es propiamente probado antes de la implementación.
6) Revisión y cierre: Una vez el cambio es completado, se debe verificar que ese cambio fue completado como fue planeado y cualquier ítem asociado al está cerrado. El administrador del cambio entonces cierra el registro del cambio para indicar que el cambio está completado (Beth-Anne Sullivan, 2008).

Gestión de la Configuración

El proceso de gestión de la configuración en ITIL, abarca la identificación, control, contabilidad y verificación de estado de los componentes de TI (configuración de los elementos de infraestructura, los activos) y sus relaciones. El objetivo principal de este proceso es proporcionar información acerca de los componentes que se utilizan en otros procesos de gestión de servicios.

Gestión de versiones (control del software)

La gestión de versiones es la planificación, diseño, construcción, configuración y pruebas de hardware y software para la liberación controlada en el medio ambiente en operativo. La gestión de versiones trabaja en colaboración con la Gestión del Cambio y Gestión de la Configuración procesos.

II.2.4.5 COBIT (Control Objectives Control Objectives for Information and related Technology)

Es el marco aceptado internacionalmente como una buena práctica para el control de la información, TI y los riesgos que conllevan. COBIT se utiliza para implementar el gobierno de TI y mejorar los controles de TI. Contiene objetivos de control, directivas de aseguramiento, medidas de desempeño y resultados, factores críticos de éxito y modelos de madurez.

Las directrices de gestión de COBIT contiene el modelo de madurez, el proceso de descripción, criterios de información y recursos de TI, que indican la mejora potencial, factores críticos de éxito, indicadores de los objetivos clave y los indicadores clave de rendimiento para cada proceso. El framework COBIT, detalla los objetivos de control y directrices de auditoría de TI para mejorar el nivel de control en la organización, la mitigación de los riesgos y mantener el rendimiento.

El valor de cualquier modelo de rendimiento de la conducción mejora, depende de si el modelo se identifica oportunidades de mejora adecuadas para la organización. COBIT se puede utilizar para identificar las debilidades y oportunidades de mejora en la eficiencia, eficacia, confidencialidad, integridad, cumplimiento y confiabilidad. COBIT, también se puede utilizar para optimizar la gestión de personas, aplicaciones, tecnología, instalaciones y datos (Mallette, 2005).

COBIT determina un conjunto de mejores prácticas para la seguridad, la calidad, la eficacia y la eficiencia en TI que son necesarias para alinear TI con el negocio, identificar riesgos, entregar valor al negocio, gestionar recursos y medir el desempeño, el cumplimiento de metas y el nivel de madurez de los procesos de la organización (ver Tabla 2).

0 No Existente	Carencia completa de cualquier proceso reconocible.
1 Inicial	Existe evidencia que la empresa ha reconocido que los problemas existen y requieren ser resueltos. Sin embargo; no existen procesos estándar en su lugar existen enfoques ad hoc que tienden a ser aplicados de forma individual o caso por caso.
2 Repetible	Se han desarrollado los procesos hasta el punto en que se siguen procedimientos similares en diferentes áreas que realizan la misma tarea. No hay entrenamiento o comunicación formal de los procedimientos estándar, y se deja la responsabilidad al individuo, por lo tanto, los errores son muy probables.
3 Definido	Los procedimientos se han estandarizado y documentado, y se han difundido a través de entrenamiento. Sin embargo, se deja que el individuo decida utilizar estos procesos, y es poco probable que se detecten desviaciones. Los procedimientos en sí no son sofisticados pero formalizan las prácticas existentes.
4 Administrado	Es posible monitorear y medir el cumplimiento de los procedimientos y tomar medidas cuando los procesos no estén trabajando de forma efectiva. Los procesos están bajo constante mejora y proporcionan buenas prácticas.
5 Optimizado	Los procesos se han refinado hasta un nivel de mejor práctica, se basan en mejoras continuas y en un modelo de madurez con otras empresas.

Tabla 2 Modelo de madurez COBIT

El modelo de madurez es una forma de medir qué tan bien están desarrollados los procesos administrativos, esto es, qué tan capaces son en realidad. Qué tan bien desarrollados o capaces deberían ser, principalmente dependen de las metas de TI y en las necesidades del negocio subyacentes a las cuales sirven de base. Cuánta de esa capacidad es realmente utilizada actualmente para retornar la inversión deseada en una empresa (IT Governance Institute, 2007).

Para que TI tenga éxito en satisfacer los requerimientos del negocio, la dirección debe implementar un sistema de control interno o un marco de trabajo. El marco de trabajo de control COBIT contribuye a estas necesidades de la siguiente manera:

- Estableciendo un vínculo con los requerimientos del negocio
- Organizando las actividades de TI en un modelo de procesos generalmente aceptado
- Identificando los principales recursos de TI a ser utilizados

- Definiendo los objetivos de control gerenciales a ser considerados

Gestión de la configuración

Garantizar la integridad de las configuraciones de hardware y software requiere establecer y mantener un repositorio de configuraciones completo y preciso Este proceso incluye la recolección de información, de la configuración inicial, el establecimiento de normas, la verificación y auditoria de la información de la configuración y la actualización del repositorio de configuración conforme se necesite. Una efectiva administración de la configuración facilita una mayor disponibilidad, minimiza los problemas de producción y resuelve los problemas más rápido (IT Governance Institute, 2007).

La versión 4.1 de COBIT es mucho más alineada con la arquitectura empresarial que las versiones anteriores.

Para resumir, los recursos de TI son manejados por procesos de TI para lograr metas de TI que respondan a los requerimientos del negocio. Este es el principio básico del marco de trabajo COBIT, como se ilustra en el cubo COBIT (Ilustración 7).

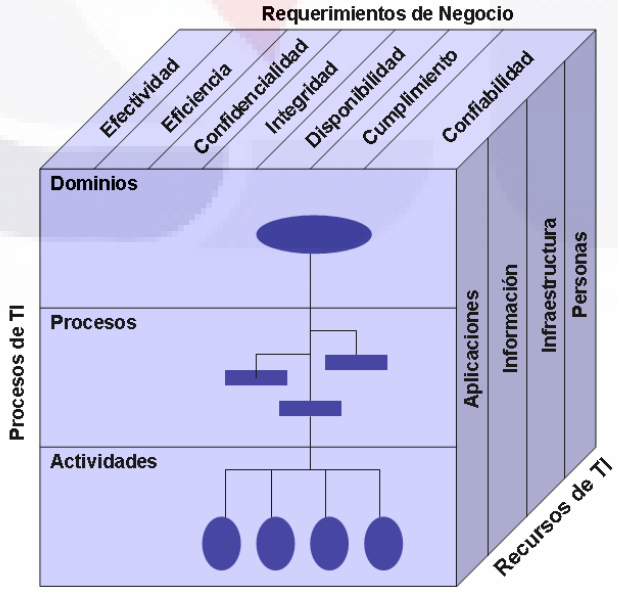


Ilustración 7 El cubo de COBIT

En detalle, el marco de trabajo general COBIT se muestra gráficamente en la Ilustración 8, con el modelo de procesos de COBIT compuesto de cuatro dominios que contienen 34 procesos genéricos, administrando los recursos de TI para proporcionar información al negocio de acuerdo con los requerimientos del negocio y de gobierno.

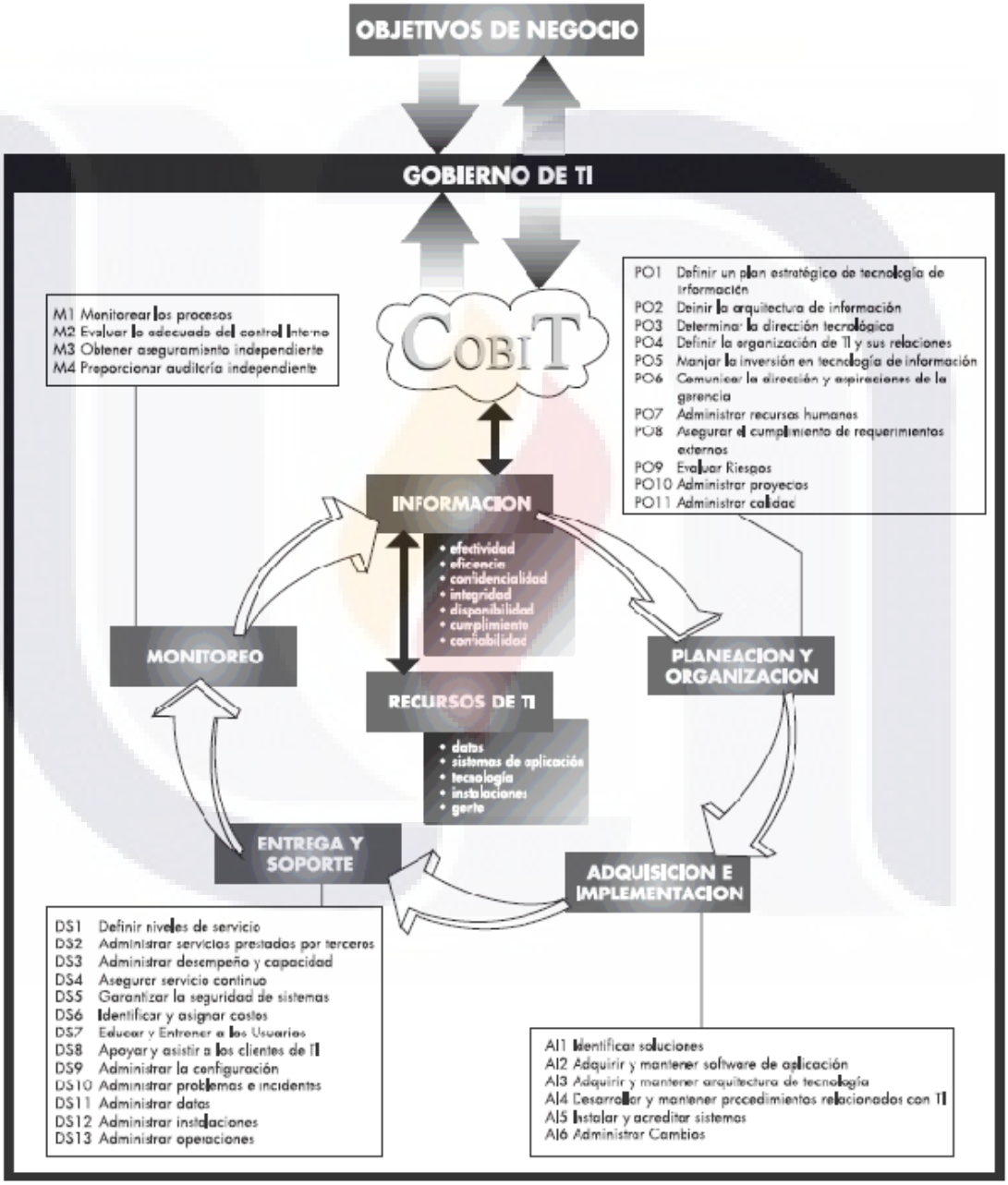


Ilustración 8 Marco de trabajo COBIT

II.2.5 Metodología de Desarrollo de Sistemas de Información INEGI

El Instituto Nacional de Estadística y Geografía (INEGI), tiene amplia experiencia en el desarrollo sistemas de software para el procesamiento de la información estadística y geográfica que capta la Institución. Esta actividad, llevada a cabo durante años por el personal del Instituto, ha utilizado diferentes métodos para el desarrollo de estos programas.

Al paso del tiempo y con el desarrollo de las Tecnologías de Información y Comunicaciones, se han presentado nuevas metodologías que, con diferentes propuestas de solución y control de los procesos de desarrollo de software, han sido utilizadas para resolver la sistematización del procesamiento de la información.

INEGI ha adoptado la metodología Proceso de Desarrollo Unificado de Software o RUP, con la finalidad de estandarizar el desarrollo y la documentación de los sistemas de procesamiento y administración de la información en el Instituto, con el objetivo de facilitar su adopción por parte de los desarrolladores de software (INEGI, 2006).

Desarrollo Iterativo e Incremental

El desarrollo iterativo, se enfoca en el crecimiento del sistema en pasos pequeños, incrementales y planeados.

Cada iteración incluye un ciclo de vida completo, incluyendo análisis, diseño, implementación y pruebas.

Tanto los modelos como el software son construidos incrementalmente, sobre múltiples iteraciones. El mantenimiento de sistemas, es simplemente otra iteración o series de múltiples iteraciones.

II.2.5.1 Proceso Unificado

El Proceso Unificado (ver Ilustración 9) es una versión abierta de la metodología Rational Unified Process (RUP), creada por Booch, Jacobson, and Rumbaugh. Es una metodología cuyo desarrollo es iterativo e incremental.

Tiene 4 fases:

- Gestación o Concepción.- Crea una visión del Software
- Elaboración.- Se definen la mayoría de los casos de uso, así como la arquitectura del sistema.
- Construcción.- Se construye el software.
- Transición.- El software se mueve de una versión Beta a una de producción.

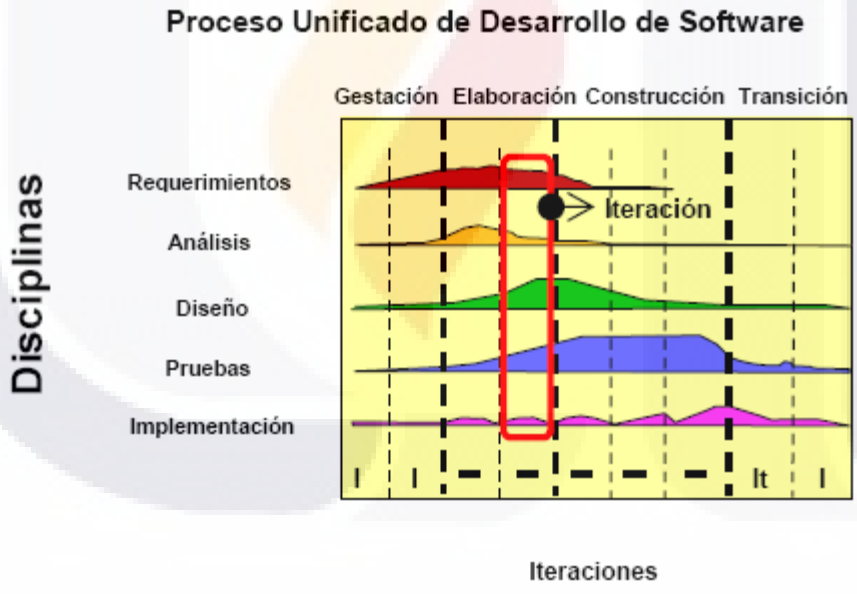


Ilustración 9 Proceso Unificado (INEGI, 2006)

II.2.5.2 Proceso Unificado de Desarrollo INEGI Versión 1.0

Fase de Gestación del Proceso Unificado (INEGI)

Esta fase es el principio de los esfuerzos de desarrollo. Su meta principal es establecer de forma conjunta entre los usuarios y el equipo de desarrollo, los objetivos, alcances y términos del proyecto. Para el caso de sistemas ya existentes, la fase de gestación deberá ser muy breve.

Fase de Elaboración del Proceso Unificado (INEGI)

El objetivo de esta fase es obtener una arquitectura central del sistema y una definición detallada de los casos de uso más significativos, que provean una base fija, principalmente para la parte en la que se realizan el diseño y la implementación, en la Fase de Construcción (INEGI, 2006).

Fase de Construcción del Proceso Unificado (INEGI)

La meta de esta fase consiste en completar el desarrollo del sistema. Las iteraciones dentro de esta fase, son exactamente iguales a las de la Fase de Elaboración, en donde ya existen las bases de una arquitectura. En las iteraciones de la Fase de Construcción, se busca terminar con el producto de software ejecutable. Esto se logra siguiendo la estrategia de división del proyecto en subproyectos de 1 a 3 semanas, en cuanto a su duración y guiados por los casos de uso prioritarios, según lo indique el líder de proyecto (INEGI, 2006).

Es importante hacer énfasis en:

- Administrar recursos y controlar procesos.
- Desarrollar y probar componentes.
- Asegurar la Iteración

Fase de Transición del Proceso Unificado (INEGI)

Esta fase se centra en implantar el producto en su entorno de operación. Los objetivos básicos de esta fase consisten en cumplir los requisitos establecidos en las fases anteriores, hasta la satisfacción de todos los usuarios, así como gestionar todos los aspectos relativos a la operación en el entorno del usuario, incluyendo la corrección de los defectos remitidos por los usuarios de la versión beta o por los encargados de las pruebas de aceptación (INEGI, 2006).

En la Ilustración 10 se muestra de manera gráfica el Proceso Unificado de Desarrollo INEGI Versión 1.0 (metodología INEGI).

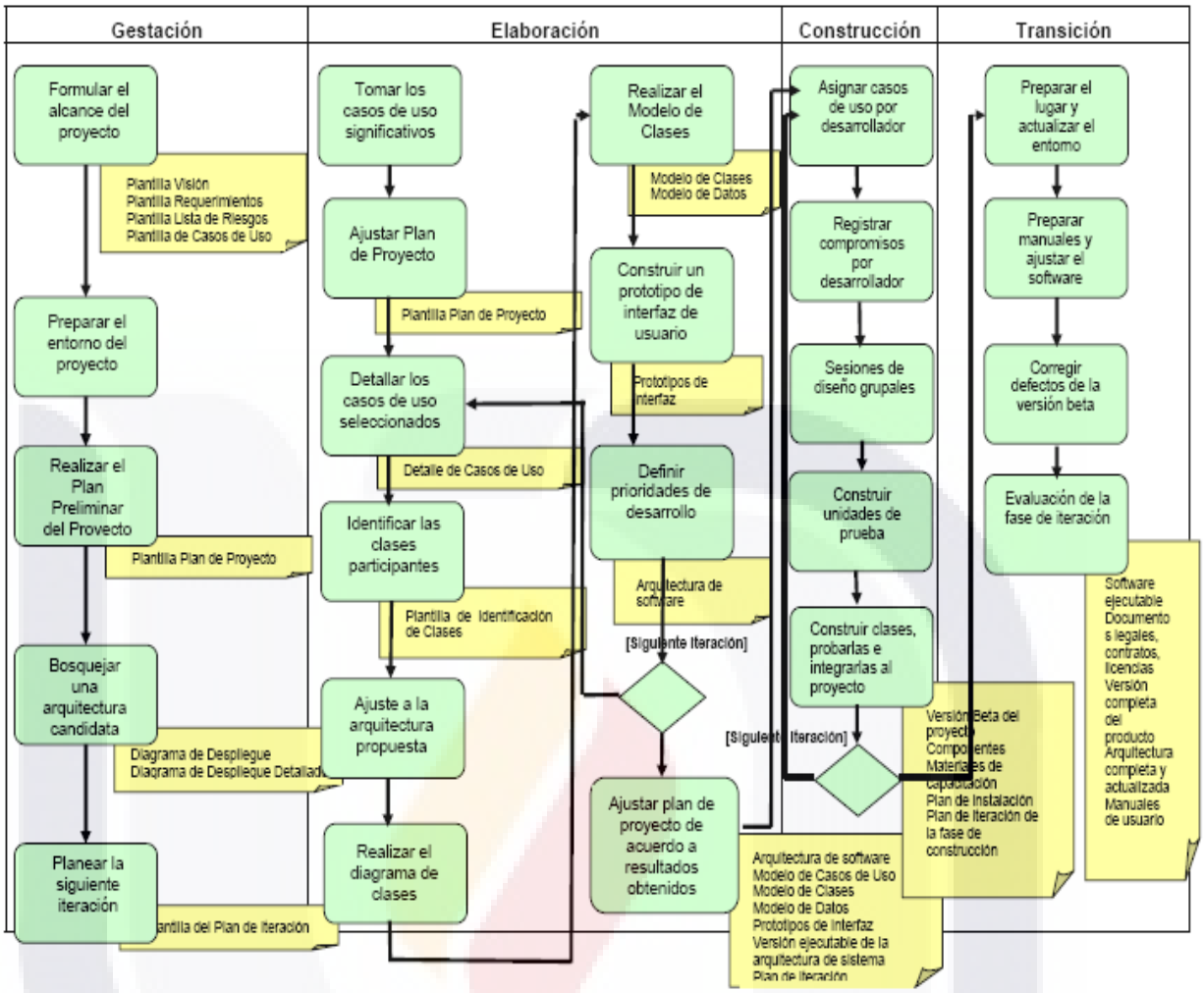


Ilustración 10 Proceso Unificado de Desarrollo INEGI Versión 1.0

Control de cambios en metodología INEGI

Como se ha mencionado, dentro del Instituto, se ha adoptado y adaptado la metodología Proceso de Desarrollo Unificado de Software RUP con la finalidad de estandarizar el desarrollo y documentación de los sistemas, es por esto que el control de cambios se lleva a cabo mediante el llenado de un único formato, en el cual se destacan los siguientes elementos:

- Nombre del módulo involucrado en el cambio.
- Caso de uso involucrado en el cambio.
- Justificación del cambio.

- Descripción detallada del cambio realizado.
- Nombre de la(s) persona(s) que realiza el cambio.
- Firmas del solicitante y quien recibe la petición.

Para mayor información, el formato utilizado para el control de cambios en la actual metodología utilizada en el Instituto, puede ser consultado en el *apéndice E* del presente trabajo de tesis.

El proceso actual de la gestión de los cambios en la metodología INEGI se muestra de forma gráfica en la Ilustración 11

Ilustración 11 El proceso actual de gestión de los cambios en el INEGI

Como se puede ver, la Ilustración 11 muestra el procedimiento que actualmente se realiza para la gestión de los cambios, los cuales consisten: la recepción de la solicitud o identificación del cambio, el registro de la petición del cambio en el formato de control de cambios, implícitamente su corrección, para finalizar con la liberación o publicación de los cambios.

II.3 Sistemas de control de versiones analizados en el Instituto.

II. 3.1 CVS Concurrent Versions System

CVS es un sistema que da seguimiento a versiones. Mantiene el registro de los archivos a través de su desarrollo, permitiendo regresar a cualquier versión del archivo que se tenga almacenada, y soporta múltiples versiones de un mismo archivo. CVS permite el trabajo concurrente en un solo archivo, de muchos desarrolladores sin que se experimente perdida de la información. Cada desarrollador trabaja en su propia copia del archivo, y todos los cambios son posteriormente integrados en un solo copia maestra. CVS puede ser integrado con sistemas de seguimiento a errores y sistemas de seguimiento de características, y proveer características que puedan asistir a la administración de proyectos por seguimiento a cambios de un proyecto a través del tiempo (Vesperman, 2006).

CVS puede ser utilizado en muchos ambientes para múltiples propósitos, no solo para la gestión de código fuente, entre estos usos pueden ser por ejemplo: mantenimiento de archivos de configuración, aliases de correo, archivos FAQ, etc. (Vesperman, 2006).

Ventajas

El uso de CVS como una herramienta de control de versiones tiene varios beneficios:

- Se puede adecuar a cualquier modelo o marco de referencia que el equipo de desarrollo elija.
- Es un lenguaje neutral, independiente de la plataforma. Muchos desarrollos de software son programados y diseñados alrededor de un lenguaje de programación especifico, y a su vez el controlador de versiones solamente puede ser utilizado cuando se utiliza aquel lenguaje.
- Es ampliamente publicado y utilizado.

- Fácil de administrar. CVS cuenta con su propio administrador de sistema, el cual permite administrar el repositorio central el cual es accedido por cada uno de los clientes que se encuentran distribuidos.
- Fácil de usar. Dos comandos, "*login"* y "*check-out"* usualmente necesitan de ser ejecutados solamente una vez por desarrollador. Y la mayoría del desarrollo restante es acompañado por el uso de dos comandos más "*update"* y "*commit"*.
- Cuenta con unas herramientas para solución de conflictos que utiliza un modelo optimista, con lo cual los desarrolladores cuentan con la libertad de trabajar colaborativamente y crear conflictos.
- Es de licencia libre, sin costo. Esto es de gran ayuda más en tiempos de bajos presupuestos (Beck, 2005).

Desventajas

- No cuenta con un entorno gráfico nativo para su administración, para tal efecto se deben utilizar entornos gráficos de terceros.
- Si bien es multiplataforma, se requiere de instalar más programas para su correcta implementación bajo sistemas operativos Windows o Mac OS

Recomendaciones

- Como todo proyecto de software libre es recomendable contar con el último release de la versión de CVS.
- Desproteger solo aquellos archivos que serán modificados.
- Al realizar un commit de un archivo verificar que el código compila al 100% y agregarle una descripción que documente de manera clara el cambio realizado.
- Siempre, al inicio del día o al finalizar este se recomienda obtener la última versión del código fuente.

II.3.2 Visual SourceSafe 2005

Microsoft Visual SourceSafe (también conocido por sus siglas VSS) es una herramienta de Control de versiones que forma parte de Microsoft Visual Studio aunque está siendo sustituida por el Visual Studio Team Foundation Server.

Ventajas

- Integración con Visual Studio 2005.
- Organización a través de carpetas.
- Capacidad de hacer rollback a versiones anteriores.
- Administración por proyecto y/o usuario.

Desventajas

- Visual SourceSafe es solamente un cliente de control de versiones, el cual lee y escribe dentro de una "*base de datos*" de SourceSafe, la cual es una colección de archivos almacenados en una carpeta compartida de la red (St.Jean, 2006).
- Difícil de sincronizar cuando se trabaja de manera desconectado.
- No permite la programación concurrente, puesto que los archivos quedan bloqueados y no se puede realizar ningún cambio hasta que se libere el archivo por quien lo bloqueó o también por el administrador del SourceSafe.

Recomendaciones

- Desproteger solo aquel archivo que se vaya a actualizar.
- Una vez que se termine de actualizar el archivo, protegerlo de forma inmediata.
- Escribir la observación del cambio que se está realizando.

- Verificar todos los días que se encuentren protegidos los archivos que se utilizaron.
- Todos los días en la mañana o antes si se considera necesario, descargar la última versión del proyecto.
- Todos los días al finalizar la jornada laboral proteger (subir) la aplicación y los archivos actualizados al repositorio.
- Si se encuentra desarrollando una aplicación, una vez que subió sus cambios se aconseja descargar la última versión, ejecutar la aplicación y verificar que sus cambios se ejecuten adecuadamente con la versión descargada.
- Cuando se agrega un nuevo archivo a un proyecto que se encuentra enlazado a VSS, todo el proyecto es desprotegido, para ello se aconseja que de forma inmediata se suba (proteja) a SourceSafe todo el proyecto. Esto debido que mientras el proyecto se encuentre desprotegido, nadie más puede agregar archivos.

II.3.3 SVN (Subversion)

A principios del 2000, CollabNet, Inc. (http://www.collab.net) comenzó a buscar desarrolladores para escribir un sustituto para CVS. CollabNet ofrece un conjunto de herramientas de software colaborativo llamado SourceCast, del cual un componente es el control de versiones. Aunque SourceCast usaba CVS como su sistema de control de versiones inicial, las limitaciones de CVS se hicieron evidentes desde el principio, y CollabNet sabía que tendría que encontrar algo mejor. Así CollabNet decidió escribir un nuevo sistema de control de versiones desde cero, manteniendo las ideas básicas de CVS, pero sin sus fallos y defectos (Collins-Sussman et al., 2004).

Después de catorce meses de codificación, Subversion pasó a ser "auto-hospedado" el 31 de agosto del 2001. Es decir, los desarrolladores de Subversion dejaron de usar CVS para la administración del propio código fuente de Subversion, y en su lugar empezaron a usar Subversion (Collins-Sussman et al., 2004).

Así, Subversion nació para ser el reemplazante natural del CVS y para cubrir algunas de sus principales fallas, entre ellas:

- Soporte transaccional para los *commits* y *updates*.
- Posibilidad de Renombrar Directorios y Archivos.
- Agregar Meta-data a los archivos.
- Permitir manejar diferencias en archivos binarios, guardando sólo los cambios y no todo el archivo como en el CVS.
- Manejo eficiente de los Branch y Tags (en CVS eran proporcionales al tamaño del proyecto).

El proyecto tiene una velocidad de maduración asombrosa y cada vez tiene más adeptos. El cliente de SVN más conocido y fácil de usar es el **TortoiseSVN**

El repositorio

Subversion es un sistema centralizado para compartir información. En su núcleo está un repositorio, que es un almacén central de datos. El repositorio almacena información en forma de un árbol de ficheros – una jerarquía típica de ficheros y directorios. Cualquier número de clientes se conectan al repositorio, y luego leen o escriben esos ficheros. Al escribir datos, el cliente hace que la información esté disponible para los otros; al leer los datos, el cliente recibe la información de los demás (Collins-Sussman et al., 2004).

Ventajas

- Se sigue la historia de los archivos y directorios a través de copias y renombrados (Collins-Sussman et al., 2004).
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente.
- Se envían sólo las diferencias en ambas direcciones (al repositorio y/o al usuario).
- Puede ser servido mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios.
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.).
- Permite la programación concurrente y de forma desconectada del repositorio

Desventajas

- No se integra al de manera nativa al IDE de Visual Studio
- No cuenta con una herramienta automática que realice el *auto merge*

Recomendaciones

Al igual que CVS es recomendable tener siempre la última versión de Subversion instalada

- Desproteger solo aquellos archivos que serán modificados
- Al realizar un commit de un archivo verificar que el código compila al 100% y agregarle una descripción que documente de manera clara el cambio realizado.
- Todos los días en la mañana o antes si se considera necesario, descargar la última versión del proyecto.

II.3.4 Team Foundation Server (TFS)

Microsoft Team Foundation Server (TFS) es una parte integral de “Visual Studio Team System” la cual ofrece una variedad de características que permiten la integración de diferentes disciplinas dentro del equipo de desarrollo con el propósito de colaborar efectivamente en la producción de productos de software, el rectángulo azul en la *Ilustración 12 Visual Studio Team System* (St.Jean, 2006) es solo una pequeña parte de la plataforma total de TFS, pero juega un rol importante (St.Jean, 2006) en el desarrollo de software.

Control de código fuente de Team Foundation

Control de código fuente Team Foundation proporciona la funcionalidad de control de versiones de código fuente estándar, que se puede escalar para controlar a miles de desarrolladores. Además de la típica funcionalidad de control de código fuente, Team Foundation también es un producto de administración de configuración de software para la empresa que proporciona un control de versión integrado, seguimiento de problemas y administración de procesos para los equipos de desarrollo (Hundhausen, 2006).

Además de estar integrado en el entorno de Visual Studio con otras tecnologías de Team Foundation, como la creación de una generación y el seguimiento de elemento de trabajo (ver Ilustración 12), el control de código fuente también incluye una interfaz gráfica de usuario autónoma y una interfaz de línea de comandos.

De la herramienta se aprovecha su metodología de desarrollo basada en el Microsoft Solutions Framework, sin embargo también se puede elegir CMMI como metodología y así, ya sea con MSF o CMMI se definen tareas, roles, guías y plantillas; lo mismo que su capacidad para documentar cada modificación y asociarla a las tareas del plan general de trabajo, hecho que permite medir las transformaciones del código y las aportaciones de cada integrante del equipo (Hundhausen, 2006).

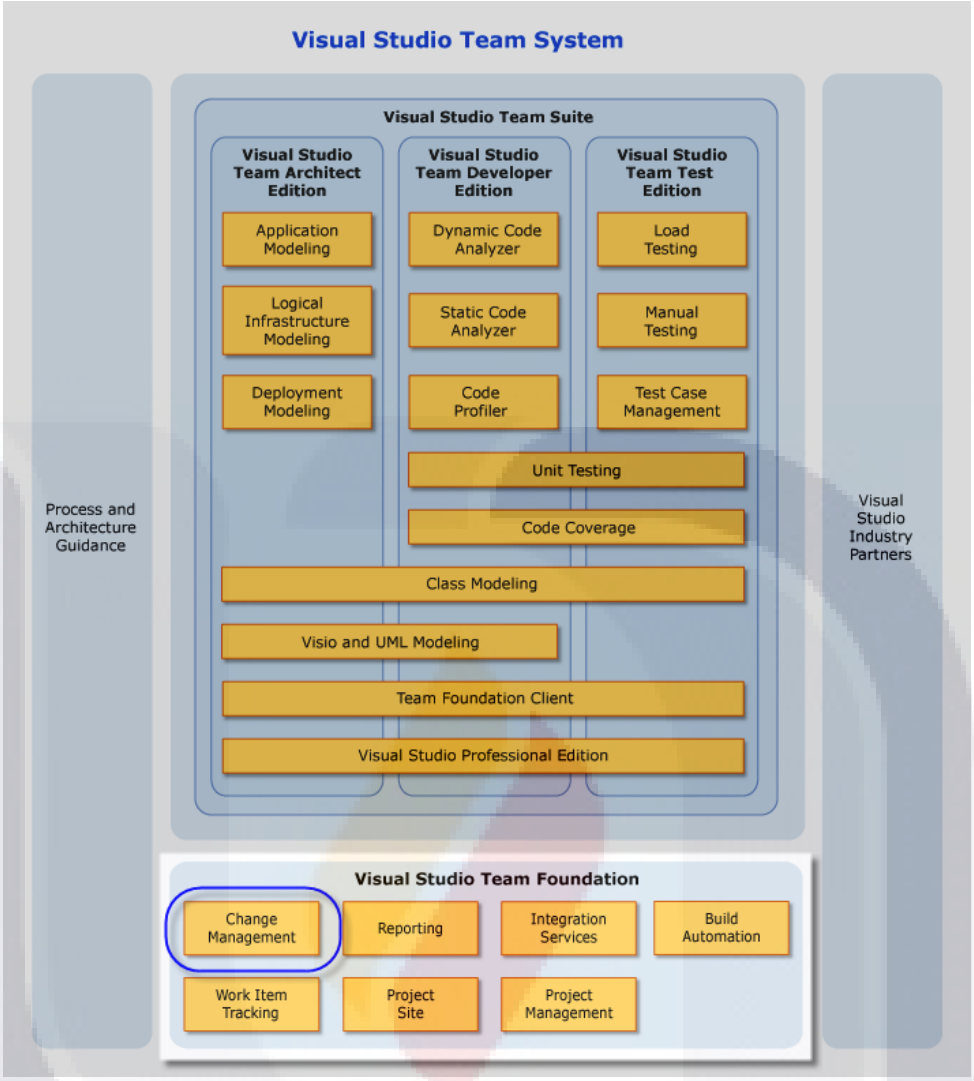


Ilustración 12 Visual Studio Team System (St.Jean, 2006)

Ventajas

Control de código fuente Team Foundatio n cuenta con las ventajas siguientes:

- Conjunto completo de características de control de versiones.
- Protecciones de un cambio al mismo tiempo.
- Bifurcación y combinación eficaces.
- Aplazamiento de cambios.
- Directivas de protección

Desventajas

- Si por espacio de algún tiempo no se ha actualizado la versión, a la primera vez que se actualice se puede llegar a presentar problemas con el auto merge puesto que por decirlo así, queda tan desactualizado que no puede resolver los problemas por sí solo, ocasionando inclusive que no se pueda actualizar la versión.
- El TFS amarra el nombre de la máquina (Workitem) del usuario, si por alguna razón, ya sea por una política aplicada al directorio activo, este nombre cambia, el proyecto se desliga al repositorio, teniendo por consecuencia la necesidad de reconfigurar la conexión con el servidor de TFS

Recomendaciones

- Tener siempre la última versión del código fuente, para esto se tiene obtener de manera seguía el código ejecutando un “Get Lastes Version (Recursive)” el cual nos trae la última versión de todo el proyecto.
- Al hacer un check-in de cualquier cambio, es recomendable describir el cambio que se realiza, y si es posible enlazarlo a un *bug* o *issue* reportado.
- Realizar check-in de archivos cuyo código compila al 100% y agregarle una descripción que documente de manera clara el cambio realizado.

II. 4 Estudio de casos similares

II.4.1 Caso 1

Titulo

"Towards virtual software configuration management a case study* (Rahikkala, 2000)*".

Publicado por: Universidad Oulu, facultad de ciencias

Reseña

La Universidad de Oulu es una comunidad científica internacionalmente conocida por investigaciones de alta calidad y la educación que proporciona a los expertos para tareas exigentes tanto a nivel nacional e internacional.

La Universidad promueve el bienestar y la educación en el norte de Finlandia y juega un papel importante en el campo de investigación Finlandés y Europeo, basándose en la innovación y la educación.

Descripción del Caso

El caso realiza una investigación sobre la necesidad e importancia de implementar un proceso que permita llevar la correcta gestión de la configuración del software en organizaciones dedicadas al desarrollo de aplicaciones, destaca los siguientes puntos

a) La tendencia global de las organizaciones dedicadas al desarrollo hacia convertirse en empresas transorganizacionales, con equipos de trabajo geográficamente distribuido, muchos de estos equipos de trabajo pueden estar conformado por trabajadores de outsourcing. Este tipo de empresas, son una clase de organizaciones a las cuales el autor les nombra

corporaciones de software virtuales (VSC) y las cuales representan un reto para la implementación de procesos que ayuden a la gestión e la configuración del software.

b) Es mediante el estudio de estas organizaciones la forma en que el autor destaca el surgimiento de nuevos retos para la implementación de un sistema de gestión de configuraciones

c) Las preguntas de investigación del caso de estudio demuestran 1) ¿Cómo los requerimientos de la gestión de la configuración de software deben ser definidos para las corporaciones de software virtuales (VSC)?, 2) ¿Cómo puede ser expandido el proceso de la gestión de la configuración del software en una corporación virtual de software? Y 3) ¿Cómo se puede desarrollar un proceso de gestión de la configuración de software para que cumpla con los requerimientos de las corporaciones virtuales de software (VSC)?

d) Para lo cual se identifica como principal motivo para el caso de estudio, la exanimación de cómo las practicas de la gestión de configuración del software puede ser analizadas y llevas a cabo a un proceso virtual.

Lecciones Aprendidas

- El caso de estudio destaca de manera clara los puntos de los cuales consiste las actividades generales que se deben de llevar a cabo para la gestión de la configuración del software, los cuales son la identificación de los elementos de configuración, el establecimiento de las líneas bases, el etiquetado de los elementos de configuración, el control de los cambios, la gestión de la configuración y la realización de auditorías a la configuración.
- Los retos de implementar un proceso de gestión de la configuración del software en una empresa y que estas actividades permitan una mayor rentabilidad y eficiencia en la empresa son: la infraestructura, su facilidad, la seguridad, cultura comunicación, el cambio y la gestión

II.4.2 Caso 2

Titulo

"Impacto de la aplicación de un modelo CMMI Nivel 2 en el Ciclo de Vida de un proyecto* (Peña García, 2009)*".

Publicado por: Universidad Politécnica de Madrid

Reseña

La Universidad Politécnica de Madrid cumplió 25 años en 1996 como tal Universidad, si bien la mayoría de sus centros son más que centenarios pues fueron fundados en los siglos XVIII y XIX y cada uno de ellos mantuvo su vida independiente hasta ser agrupados en la UPM.

Descripción del Caso

El caso de estudio, muestra el proceso de implementación para el desarrollo de software siguiendo el modelo CMMI nivel 2 en una organización la cual se encontraba en el nivel 1, explica las carencias y problemas con las que contaba dicha organización en sus proyectos informáticos, y se detallan cuales son las mejoras que se pretende obtener, con la implantación de nuevos procedimientos que cubran las áreas de proceso que marca el nivel 2 de CMMI incluida la gestión de la configuración.

a) En el caso práctico se realiza un estudio detallado del nivel 2 del CMMI , mostrando las experiencias particulares referentes a la implementación de procesos siguiendo el modelo CMMI nivel 2, en donde la organización donde se realizó el caso de estudio partía del nivel 1.

b) Analiza la situación actual por la que atraviesa el desarrollo de software que se encuentra en el nivel 1 de madurez, explicando las carencias y

problemas por las que se encuentra dicha organización en sus proyectos informáticos, detallando las mejoras que pretende obtener con la implementación de procesos que se encuentren alineados con el marco de referencia CMMI.

c) Describe una serie de procedimientos para cada actividad de proceso contenida en CMMI nivel 2.

Lecciones Aprendidas

- El estudio del caso, destaca la importancia de aplicar el nivel 2 de CMMI dentro de las empresas de desarrollo de software.
- A su vez, en caso estudiado se define un proceso para el manejo de la gestión de la configuración alineado al marco de referencia CMMI dicho proceso comprende las siguientes actividades:
    - Una planificación de la gestión de la configuración: comprende las actividades de planificación inicial de la Gestión de la Configuración (realizada en la etapa de Definición) y las re planificaciones posteriores o modificaciones del plan si es necesario (realizadas en el resto de etapas).
    - Identificar los elementos de configuración del proyecto: comprende las actividades de registro de los elementos de configuración a generar por el proyecto.
    - Controlar los elementos de configuración en el proyecto: comprende las actividades de alta/modificación de un elemento de configuración, incorporación y extracción de un elemento de configuración en la línea base
    - Gestión del Cambio en el proyecto: comprende las actividades de registrar el cambio, evaluarlo y seguirlo hasta su cierre.
    - Auditoria de la Configuración: comprende las actividades de revisión y gestión de no conformidades detectadas.

II.4.3 Caso 3

Titulo

"Evaluando herramientas de software para la gestión de la configuración* (Tosun, 2004)*".

Publicado por: La Universidad de Ámsterdam

Reseña:

La Universidad de Ámsterdam es una de las más importantes instituciones de educación superior de los Países Bajos. La institución hunde sus raíces en el siglo XVII, con la creación en 1632 del Athenaeum Illustre, que formaba a sus estudiantes en las artes del comercio y en filosofía. En 1877, el Ateneo pasó a ser la Universidad de Ámsterdam.

Actualmente su prestigio la ha hecho pertenecer a la League of European Research Universities (Liga de Universidades de Investigación Europeas). Además, coopera activamente con la Hogeschool van Amsterdam (HvA), institución dedicada a la formación profesional y técnica.

Descripción del Caso

a) El estudio lo realiza en Opticon una empresa Japonesa dedicada a la manufacturación, principalmente de lectores de códigos de barras y RFID.
b) Su problemática surge al no poder dar un adecuado seguimiento a los desarrollos de software que ahí se generan, principalmente derivado de la falta de comunicación entre los equipos de desarrollo distribuidos geográficamente entre Holanda y Japón, esto más que nada debido a las barreras de idioma así como de husos horarios.

Lecciones Aprendidas

- Define lo que es un Software Configuration Management (SCM) así como los aspectos principales que se deben tener en cuenta para su manejo, los cuales identifica como:
    a. Identificación: un esquema de identificación refleja la estructura del producto, identifica los componentes y su tipo, que los hace únicos y accesibles de alguna forma.
    b. Control: controlar la liberación de un producto y los cambios a lo largo de todo el ciclo de vida
    c. Estado: Es el registro y notificación del estado de los componentes y las solicitudes de cambio.
    d. Auditoría y revisión: consiste en la validación de la integridad de un producto y mantener la coherencia entre los componentes, garantizando que el producto es un conjunto bien definido de componentes.
- Menciona los beneficios que la empresa Opticon obtuvo al implementar un sistema para el control de versiones, entre los que se destacan:
    a. La posibilidad y deseo de utilizar una nueva y mejor tecnología
    b. Optimización de equipos de trabajo.
    c. La realización de auditorías más fácilmente.
    d. La eliminación de errores evitables en el desarrollo de software.
    e. Una reducción en el número de errores en los productos comercializados.
    f. Recuperación de fallas.
    g. Respetabilidad de todas las etapas de desarrollo.
    h. Todos los elementos de configuración se encuentran versionados.
    i. Rápidos ciclos de cambios.

- También, dentro del caso de estudio, enumera las características con las que debe contar un sistema administrador de la configuración del software SCM, tras analizar 22 herramientas automatizadas de SCM, tanto de código libre, como de código propietario, algunas de esas características son:
    a. Su plataforma: analiza las herramientas dependiendo de su plataforma (Linux/Unix/Windows o Mac).
    b. Si es que cuentan con una Interfaz Gráfica
    c. Si está basadas en Web
    d. Si es que las herramientas pueden realizar el merge, branch, Tag y el label de los elementos de configuración que se encuentren bajo el control de versiones.
    e. Si es que las herramientas cuentan con Logs de cambios, ya que de ahí se pueden sacar varios reportes que sirven para realizar las auditorias de la línea base
    f. Analiza la documentación de las herramientas evaluadas, dependiendo de parámetros de (excelente, buena, media, o pobre).
- Para finalizar, el autor del caso de estudio, recomienda el uso de Subversion para la implementación del SCM ya que soluciona el acceso simultáneo a su grupo de desarrolladores distribuidos geográficamente en Holanda y Japón, al repositorio central de software, rompiendo con las barreras de la distancia y de husos horarios, fomentando la comunicación y el trabajo en equipo.

II.4.4 Caso 4

Titulo

"Análisis descriptivo del proceso de implementación del nivel 2 del modelo CMMI en una empresa regional de desarrollo de software* (Picazzo M., 2008)*".

Publicado por: Universidad ICESI.

Reseña:

La Universidad ICESI es una universidad privada, sin ánimo de lucro, que desde hace 30 años participa de manera efectiva en la formación y especialización profesional, de los jóvenes y ejecutivos de la región. Para lograrlo, invierte una buena parte de los recursos que su operación genera en la formación y consolidación de su planta de profesores de tiempo completo capacitados, en la actualización tecnológica y en la modernización del soporte académico.

Descripción del Caso

La empresa de desarrollo de software en donde se realizó el caso de estudio, se encuentra en el sector de servicios y está clasificada como de tamaño mediano, dicha empresa contaba con la certificación ISO 9001:2000, y actualmente cuenta con la certificación CMMI nivel 2 en sus áreas de proceso.

a) El caso de estudio se realzó en una empresa de servicios calcificada de tamaño mediano en Colombia. A principios del año 2001 decidió inicial el proyecto de estandarización de procesos para obtener la certificación ISO 9001:2000 y posteriormente a comienzos del año 2005 inicio con el proceso de valoración del nivel 2 de CMMI, este caso de estudio realiza un análisis descriptivo del proceso que siguió la empresa para poder implementar el nivel 2 del modelo CMMI en su concepción escalonada con la guía y asesoría de una empresa asociada del SEI.

b) La empresa carecía de una metodología estándar que le permitiera documentar y controlar sus procesos para la prestación de servicios.

c) La necesidad de la empresa de trabajar de forma ordenada en sus proyectos y poder hacer frente a los múltiples cambios de reglamentación de ley que presentó el sector en el que se desenvuelve dicha empresa y al determinar que se tenía poco control sobre los cambios y que se incurría en un alto costo en modificaciones de los mismos, la organización tomó la decisión de que el departamento de TI desarrollara desarrollar su propia metodología de “Gestión de Proyectos”

Lecciones Aprendidas

- La implementación del Modelo CMMI nivel 2 en la empresa que sirvió de caso de estudio, requirió de arduas sesiones de entrenamiento y orientación.
- En la empresa, se presentaron problemas de con el cambio cultural de la organización.
- Para la gestión de la configuración se establecieron una lista estándar de ítems de configuración y atributos base que conforman la línea base de cada producto
- En cuanto a la gestión de la configuración, como política de la organización se determino el uso de CVS como herramienta de control de código fuente.
- La implementación del nivel 2 del Modelo CMMI en la empresa desarrolladora de software tomó alrededor de dos años y medio. La empresa certificadora tomó como piloto de evaluación, cuatro proyectos, siendo en total diez los proyectos nuevos cumplidos con el nuevo modelo.
- Las inversiones para la implementación del modelo CMMI son considerables y pueden afectar presupuestalmente. No obstante, en el caso de la entidad en cuestión, la percepción del Comité Directivo de la empresa desarrolladora de software, es que esta es una inversión que vale la pena por los resultados conseguidos y el mejoramiento de sus procesos alcanzado.

CAPÍTULO III METODOLOGÍA PARA EL DESARROLLO DEL CASO

El desarrollo de sistemas en la actualidad es una tarea altamente compleja y multidisciplinaria, ya que desde su inicio genera grandes cantidades de elementos, que por su naturaleza evolucionan y están cambiando constantemente, estos elementos denominados ítems de configuración y dada su importancia, sobre todos aquellos que son parte del código, y sus versiones, deben de estar debidamente controladas, al igual que las versiones globales del producto que ellos componen (Picazzo M., 2008).

Es por todo esto que el desarrollo del caso de investigación permitirá ir cubriendo uno a uno cada objetivo establecido en el presente trabajo de tesis (ver Ilustración 13).

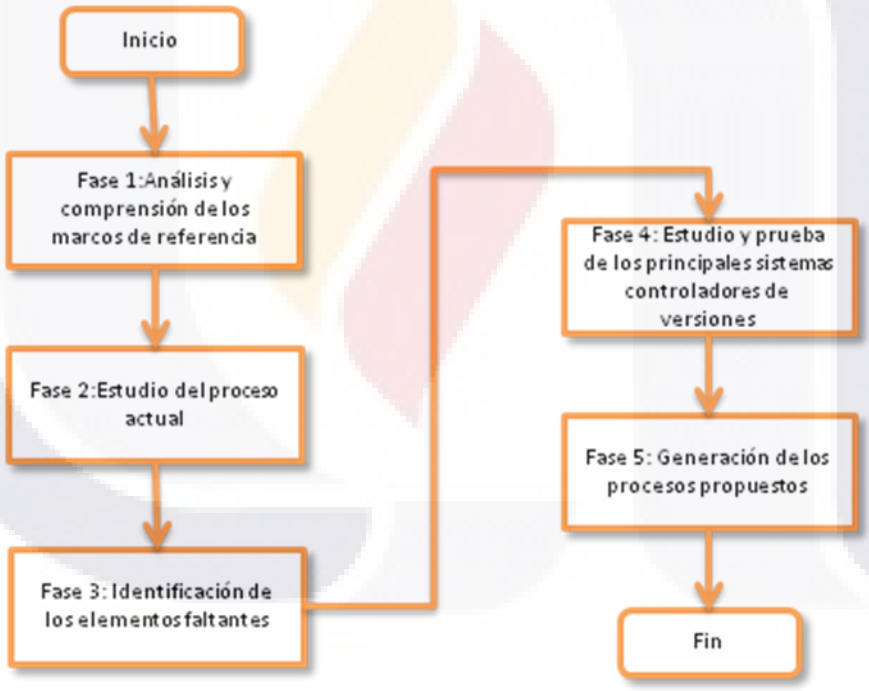


Ilustración 13 Metodología para el desarrollo del caso

El primer objetivo específico: "*desarrollar un procedimiento detallado que permita la implementación de un sistema controlador de versiones"* y segundo objetivo específico: *"desarrollar un procedimiento detallado que permita el manejo de la gestión de la configuración",* con base al modelo continuo del maco de referencia

CMMI nivel 2, serán cubiertos en la fase 5 de esta metodología *"Generación de los procesos propuestos".*

El tercer objetivo *"Determinar en qué grado cumple tanto el proceso actual como el proceso propuesto en relación a lo que se define en el marco de referencia CMMI nivel 2, referente a la gestión de la configuración"*, se cumple con el desarrollo de las fase 1, fase2 y fase 3.

Y por último, el cuarto objetivo *"Investigar que sistemas comerciales, tanto en su modalidad de software libre o propietario, se recomiendan para su uso como herramienta de control de versiones en el INEGI."*, se cubre con las actividades de la fase 4 *"Estudio y prueba de los principales controladores de versiones".*

III.1 Descripción de fases, actividades y productos generados por cada fase

Fase 1: Análisis y comprensión de los marcos de referencia

Objetivo por fase

Se pretende en esta etapa contar con la comprensión adecuada de lo que implica la gestión del cambio o gestión de la configuración de acuerdo a como lo definen los distintos marcos de referencia estudiados.

Desarrollo de la fase

Como parte del desarrollo de esta fase se analizaron mediante la revisión de la literatura, los marcos de referencia mencionados dentro del apartado de marco teórico, los cuales fueron CMM, CMMI, SPICE, ITIL y COBiT, poniendo especial interés en el tema de la gestión de la configuración objeto de estudio del actual trabajo de tesis, así como de las actividades que se desarrollan en cada modelo.

Resultado del análisis de dichos marcos de referencia, se pudo constatar que se guarda una gran similitud de acuerdo con las actividades que se llevan a cabo en cada uno de ellos para la gestión de la configuración en cada modelo.

Estas actividades generales de la gestión de la configuración se muestran en la Tabla 3 que a continuación se muestra.

CMM	CMMI	SPICE	ITIL	COBIT
Planear las actividades de la administración Identificar, controlar y poner a disposición los productos de trabajo de software seleccionados Controlar los cambios a los productos de trabajo de software identificados Informar el estado y contenido de las líneas base del software a los grupos y a las personas afectadas	Identificación y control de elementos Establecer un sistema de gestión de la configuración Crear o lanzar líneas bases Rastreo de solicitudes de cambio Control de elementos de configuración Establecer registros de administración de la configu-ración Realizar auditorías de configuración	Desarrollo de la documentación Realización de la gestión de la configuración Realización del aseguramiento de la calidad Verificación Validación Llevar a cabo auditorias Realización de la Resolución de Problemas	Surgimiento y registro del cambio Revisión del cambio Asignar cambios Autorización del cambio Coordina la implementa-ción Revisión y cierre	Establecer y mantener un repositorio de configura-ciones Recolección de informa-ción, de la configuración inicial Estableci-miento de normas Verificación y auditoria de la informa-ción de la configura-ción

Tabla 3 Gestión de la configuración en distintos marcos de referencia

Como resultado de la Tabla 1 CMM áreas clave de proceso, se muestra que en los marcos de referencia las actividades de: identificar y controlar los productos de trabajo, establecer o mantener la integridad en los elementos de configuración, el realizar una gestión de la configuración y revisión de los cambios, así como del de realizar revisiones y auditorias a los elementos de configuración, son actividades que se encuentran presentes en los cinco marcos de referencia estudiados.

Fase 2: Estudio análisis del proceso actual

Objetivo por fase

El objetivo de esta etapa es el de estudiar el proceso actual, la normativa con la que cuenta el Instituto para el desarrollo de software, e identificar los elementos de gestión del cambio que en ella se manejen.

Proceso actual

Para cubrir con el objetivo específico número 3 en el cual se requiere determinar en qué grado cumple el proceso actual con en el marco de referencia CMMI nivel 2, referente a la gestión de la configuración, se realizó una investigación del proceso que se lleva actualmente para la gestión de los cambios o las configuraciones, para lo cual se consultó en la Intranet Institucional en el apartado de Normatividad Informática la norma vigente para el desarrollo de sistemas.

La actual normativa para el desarrollo de sistemas es una versión adaptada/recortada de la metodología Proceso de Desarrollo Unificado de Software o RUP, la cual se encuentra avalada por un grupo multidisciplinario de expertos del propio Instituto y tiene la finalidad de estandarizar el desarrollo y la documentación que se genere en el ciclo de vida de desarrollo sistemas de información dentro del Instituto.

La documentación que se debe llevar por norma en el Instituto es la siguiente:

- Análisis del Negocio o Visión
- Arquitectura de Software
- Casos de Uso
- Conformación del Equipo de Trabajo

- **Control de Cambios**[2]
- Diagrama de Clases
- Guía para el Desarrollo y Documentación de Software_Plantillas
- Identificación de Clases
- Liberación y Aceptación del Sistema
- Lista de Riesgos
- Modelo de Datos
- Modelo de Despliegue
- Plan de Iteración
- Plan de Proyecto
- Pruebas
- Requerimientos
- Seguimiento de Pruebas
- Solicitud de Servicio para Centro de Pruebas

Dentro de esta metodología se identificó que no se cuenta con una herramienta que realice la administración del control de versiones, ya que tan solo cuenta con un formato destinado al control de los cambios, y en la cual no se lleva un adecuado registro a los cambios hechos al código, y que a su vez identifique el estado de la configuración que guardaba al momento de realizar el cambio.

A su vez se identificó que la actual problemática es que cada desarrollador toma el código, le hace cambios y lo regresa, integra o publica sin antes probar y sin tener en cuenta si sus cambios afectan alguna etapa anterior o posterior del desarrollo; pero entre este tomar, cambiar y regresar, se cruzan y modifican versiones finales. Además no se cuentan con respaldos adecuados que nos ayuden a restablecer cualquier versión que se haya liberado o se encuentre en desarrollo en determinado tiempo.

[2] El formato de control de cambios propuesto en la metodología INEGI se muestra en el **Apéndice E**

La Ilustración 14 muestra de manera gráfica el proceso actual que se sigue para la documentación y registro de los cambios que se generen durante todo el ciclo de vida de desarrollo de los sistemas de acuerdo a la norma vigente.

Ilustración 14 El proceso actual de gestión de los cambios en el INEGI

Como se muestra en la Ilustración 14 el proceso se activa mediante la recepción o identificación de un cambio, esta se registra mediante el llenado de la forma correspondiente “Control de Cambios” (apéndice E) se realiza el cambio y se libera el cambio.

Proceso informal

Típicamente el proceso informal para el control de versiones o control de la configuración que se lleva a cabo en el Instituto es mediante la generación previa de carpetas compartidas o áreas que sirvan de repositorio de archivos (código fuente), por lo regular estos repositorios consisten en sitios FTP habilitados principalmente en la maquina del integrador o líder del proyecto, para “facilitarle” el trabajo de integración y liberación del producto de software que se esté desarrollando, cabe mencionar que por lo regular, este repositorio no cuenta con mecanismos de respaldo y recuperación del código (carpeta compartida o sitio FTP), y teniendo en cuenta que por lo general todos los miembros del equipo tienen permiso para escritura, lectura y eliminación de archivos sobre la carpeta compartida o repositorio FTP, cualquier cambio o movimientos que se efectué en este repositorio resulta muy arriesgado a la vez de que no se puede identificar al autor de los cambios realizados en este.

En la Ilustración 15 se muestra de manera gráfica los pasos que por lo general se siguen en el proceso informal, de control de versiones o control de la configuración.

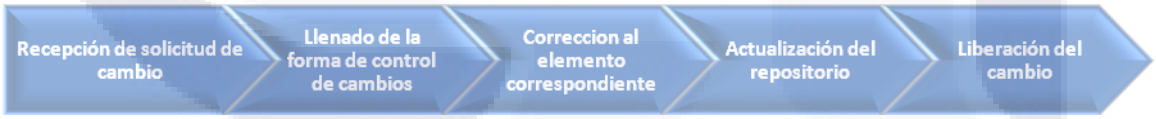


Ilustración 15 El proceso informal de gestión de los cambios en el INEGI

Como se puede ver en la Ilustración 15, el proceso informal adolece del uso de herramientas que permitan una correcta integración de los cambios realizados por el equipo de desarrollo, ya que no se cuenta con un chequeo y resolución de conflictos, y el historial de las versiones se lleva principalmente mediante el

copiado y renombrado de los archivos o elementos de configuración contenidos los dichas carpetas compartidas o sitios FTP que sirven repositorio.

Fase 3: Identificación de los elementos faltantes del actual proceso

Objetivo por fase

El objetivo de esta etapa es el de identificar los elementos faltantes en la actual metodología con la que cuenta Instituto, de acuerdo principalmente a lo que dicte el marco de referencia CMMI nivel 2 en cuanto a gestión de la configuración se refiere.

Elementos faltantes

Como ya se ha mencionado anteriormente, en la actual metodología con la que cuenta el Instituto, solamente se sugiere el llenado de un formato para llevar a cabo la administración de los cambios que se den en el proceso de desarrollo de software, dicha metodología no cuenta con una herramienta(s) o formato(s) adecuado para el control de versiones o gestión del cambio que nos permita:

- Establecer y mantener la integridad de la línea base,
- contar con una identificación de la configuración que se tenía en determinado tiempo,
- llevar un adecuado control de configuración,
- registrar el estado de configuración y
- realizar auditorías de configuración

La Tabla 4 muestra una comparativa de las actividades generales para llevar a cabo la gestión de la configuración, de acuerdo a la definición de gestión de configuración del modelo de SCM de CMMI nivel 2 contra la actual metodología de desarrollo INEGI y el proceso informal que se sigue en el Instituto.

Elementos de gestión de la configuración (Chrissis et al., 2006)	CMMI	Proceso formal	Proceso informal
Identificación de elementos de configuración	√		√
Control de los elementos de configuración	√	√	√

Registro del estado de la configuración	√		
Auditorias de configuración	√		

Tabla 4 Comparativa de las actividades generales de la gestión de la configuración contra CMMI, metodología INEGI y proceso informal

Como se puede ver en la Tabla 4, a diferencia de lo descrito por el marco de referencia CMMI, el proceso formal e informal que se siguen en el Instituto no cumple correctamente con lo descrito en la literatura, referente a los elementos de gestión de la configuración, ya que solamente se realiza el registro del cambio mediante el llenado de un solo formato (proceso actual e informal) y la identificación de los elementos de configuración (proceso informal) dejando de lado el registro del estado de la configuración y las auditorias de configuración.

Fase 4: Estudio y prueba de los principales sistemas controladores de versiones existentes en el mercado

Objetivo por fase

En esta etapa se realizará el estudio de los principales sistemas controladores de versiones, listando de una manera objetiva sus ventajas y desventajas, en este estudio se realizará la investigación de herramientas de software propietario, como de código abierto, dichas herramientas deberán enfocarse en gran medida a las dos grandes plataformas de desarrollo que se utilizan actualmente dentro del Instituto, J2EE y .Net

Comparativas entre las herramientas evaluadas

La Tabla 5 muestra una comparativa de las actividades generales para llevar a cabo la gestión de la configuración, de acuerdo a la definición de gestión de configuración del modelo de SCM de CMMI nivel 2 y las herramientas automatizadas evaluadas en el actual trabajo de tesis.

Elementos de gestión de la configuración (Chrissis et al., 2006)	CVS	Subversion	Visual Source Safe	Team Foundation Server
Identificación de elementos de configuración				√
Control de los elementos de configuración	√	√	√	√
Registro del estado de la configuración	√	√	√	√
Auditorias de configuración	√	√		√

Tabla 5 Comparativa elementos de gestión de la configuración vs herramienta evaluada

Como se muestra en la Tabla 5, la herramienta Team Foundation Server cumple de forma nativa con lo descrito por la literatura en cuando los elementos de la configuración se refiere, sin embargo con las otras herramientas el cumplimiento

de estos elementos de básicos de gestión de configuración son complementados mediante el uso y/o configuración de herramientas “complementarias”.

La Tabla 6 muestra una comparativa entre las herramientas automatizadas evaluadas dentro del Instituto para el control de versiones, y sus características básicas.

Sistema Controlador de Versiones	Modelo de Desarrollo	Tipo de repositorio	Commits Automáticos
CVS	Simultaneo o exclusivo	Central	Si
Subversion	Simultaneo o exclusivo	Central	Si
Visual Source Safe	Exclusivo	Central	No
Team Foundation Server	Simultaneo o exclusivo	Central	Si

Tabla 6 Comparativa herramientas evaluadas y sus características básicas de acuerdo al resultado de su análisis

Como se puede ver las herramientas evaluadas son muy similares en cuanto a sus características básicas de modelo de desarrollo y tipo de repositorio, siendo la única diferencia entre las herramientas el soporte de *commits automáticos* ya que Visual Souce Safe no da soporte a esta característica.

Comparativa de las características más representativas desempeñadas por un sistema de gestión de configuración

La Tabla 7 muestra la comparativa entre las características más representativas (Mahotkin, 2008) con las que deben contar los sistemas de controladores de versiones, y las herramientas evaluadas en el actual trabajo de tesis.

Característica SCM:	CVS	Subversion	VSS	Team Foundation Server
Commits atómicos	Si.	Si	No.	Si.
Capacidad para mover los archivos y directorios o renombrarlos	No.	Si	No.	Si
Fusión inteligente después de movimientos o cambios de nombre	No.	No	No.	Se desconoce.
Copias de directorios y archivos	No	Si.	Si	Si
Repositorio de replica remota	Indirectamente, mediante el uso de CVSup	Indirecta-mente	No directamente posible con la interfaz gráfica proporcionada	TFS Proxy es habilitado pero las replicas no tienen equivalente.
Permisos de repositorio	Limitada por el scripts "pre-commit" (hook)	Si, con el servicio basado en WebDAV	Si, permisos específicos por proyecto.	Si.
Soporte de cambios	No. los cambios son archivos específicos.	Parcialmente soportado	No. Los cambios son archivos específicos.	Si.
Seguimiento en línea del Historial	Si	Si	No directamente	Si.
Habilidad de trabajar en un directorio del repositorio	Si	Si	Si	Si

Tabla 7 (parte 1). Características más representativas de los sistemas controladores de versiones (Mahotkin, 2008)

Característica SCM:	CVS	Subversion	VSS	Team Foundation Server
Seguimiento de los cambios	Si, utilizando cvs diff	Si, utilizando svn diff	Si, utilizando herramientas diff	Si, utilizando diff o cambios pendientes.
Documentación	Excelente	Muy buena	Regular	Buena
Facilidad de instalación	Buena	Buena, aunque requiere la instalación de un servidor Apache	Muy buena	Compleja, require de IIS, MS-SQL Server and Reporting Services, cuentas especiales y SharePoint-Services
Conjunto de líneas de comandos	Simples.	Simples.	Básicas, se basa más en su GUI	Simples, aunque se basa más en su GUI
Soporte a redes	Buena	Muy buena.	Regular, utiliza carpetas compartidas de Windows	Buena
Portabilidad	Buena	Excelente	Regular	Regular
Interface Web	Si	Si	No	Si
Interfaz gráfica del usuario	Muy buena	Muy buena.	Standalone	Integración con Visual Studio.

Tabla 7 (parte 2). Características más representativas de los sistemas controladores de versiones (Mahotkin, 2008)

Como se muestra en la Tabla 7, parte uno y dos, la herramientas mejor evaluadas de acuerdo a las características más representativas con las que debe contra una herramienta de estas, fueron Team Foundation Sever y Subversion, siendo CVS y Visual Souce Safe, las herramientas con mayores limitaciones.

Algunos proyectos que utilizan el control de versiones dentro del Instituto

El desarrollo de aplicaciones dentro del Instituto se divide en dos grandes ramas de acuerdo a la tecnología que se utilizan para su desarrollo, ya que estas principalmente son desarrolladas en Java J2EE o en Microsoft .NET.

A su vez la diversidad de proyectos es tal, que principalmente se encuentran enfocados hacia el desarrollo de aplicaciones de escritorio, dispositivos móviles, sistemas Unix, bases de datos y en mayor medida al desarrollo de aplicaciones Web, ya sean para ser publicadas en Internet o en portal de Intranet Institucional.

Por tal motivo es dentro de de la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones, donde se realizan la mayor parte de aplicaciones Web enfocadas principalmente hacia usuarios de Internet e Intranet. A su vez como parte de la actividades destinadas al área de Investigación fue la Dirección en la cual se probaron las distintas herramientas de control de versiones, siendo Visual Source Safe, Subversion y Team Foundation Server, las herramientas que pasaron de una fase de investigación, a proyectos piloto por la gran mayoría de los proyectos de desarrollo con los que al momento se cuentan.

Proyectos de Intranet

Dentro de los proyectos de desarrollo que son dirigidos a el portal de la Intranet y que se llevan a cabo en la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones que se encuentran bajo el esquema de control de versiones, se tiene un total de 54 proyectos (ver Tabla 8) estos proyectos principalmente son desarrollados bajo la tecnología de .NET y/o ASP.

Nombre del proyecto	Tipo de proyecto	CVS	VSS	Subversion	TFS
Consola de administración para COESME	NET		√		
Tableros de control de II CPV 2005	NET		√		
Repositorio de Metadatos	NET		√		
Matriz Insumo Producto de la Encuesta complementaria del Económico 2004	Java		√		
Sistema de replicas	NET		√		
Aplicación de Monitoreo del Sitio INEGI	NET		√		
Sitio Miércoles de tecnología	NET		√		
Sitio Boletín de informática	NET		√		
Errores de Intranet	ASP		√		
Pantallas de plasma	Documentación		√		
Foros Genéricos	ASP		√		
Capital Humano - Consulta	ASP		√		
Capital Humano - Actualización	Java		√		
Asistente de consulta	Java		√		
Asistente de consulta - NET	NET		√		
Administración de Sitios en Internet	ASP		√		
Estadísticas del Sitio INEGI en Internet	ASP		√		
Seguimiento de contratos de Hildebrando Software Factory	Documentación		√		
Intercambio de indicadores económicos BIE - Web Service	NET		√		
Forma para la solicitud de documentos Gartner - SharePoint	NET		√		

Tabla 8 (parte 1). Proyectos de Intranet bajo control de versiones

Nombre del proyecto	Tipo de proyecto	CVS	VSS	Subversion	TFS
Servicios Informáticos	HTML		√		
Consulta de Indicadores	NET		√		
Aseguramiento de la calidad de Software - Investigación	Documentación		√		
Desarrollo de código seguro - Investigación	Documentación		√		
Inventario de Software - Requerimiento	NET		√		
Investigaciones - Application Tester, Web Query, Optimización y verificación de código, etc.	Documentación		√		
COESME de MetaDatos.	ASP		√		
REN - Análisis de pantallas para su estimación de transformación a NET	Documentación		√		
Componente de consulta multidimensional de datos - Cubos	ASP		√		
Componente de consulta de tabulado de censos - Consulta dinámica	ASP		√		
Componente para gráficas	ASP		√		
Componente de exportación de datos	ASP		√		
Reservación de Salas	NET				√
Directorio Telefónico	ASP				√
Directorio Único	NET				√
Mapa del sitio de Intranet	NET		√		

Tabla 8 (parte 2). Proyectos de Intranet bajo control de versiones

Nombre del proyecto	Tipo de proyecto	CVS	VSS	Subversion	TFS
Encabezado dinámico para Intranet	NET		√		
Nuestro acontecer	HTML		√		
Monitoreo de Llamadas	NET		√		
Telefonía	NET				√
Navegación	NET		√		
SAC	NET		√		
SIPAM	NET		√		
Evaluación	NET		√		
Doc. Procesos	ASP		√		
Intranet - Capacitación	HTML		√		
Intranet - Capacitación - Cine Club	HTML		√		
Intranet - Calidad	HTML		√		
Intranet - Semanas de CCT	NET		√		
Ven a verte	HTML		√		
Biblioteca Virtual	ASP		√		
Ventanilla de servicios	NET		√		
Material didáctico	NET		√		
Encuestas	NET		√		

Tabla 8 (parte 3). Proyectos de Intranet bajo control de versiones

Esta relación de proyectos y herramienta de control de versiones fue tomada a finales del mes noviembre del 2009 tiempo en el cual dentro de la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones apenas se comenzaba a realizar la migración de los proyectos que se tenían bajo Visual Source Safe a Team Foundation Server, siempre y cuando el proyecto y equipo de desarrollo así lo requirieran.

En la Ilustración 16 se muestra el porcentaje de uso de las herramientas evaluadas en el actual trabajo de tesis en los proyectos de Intranet desarrollados en la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones.

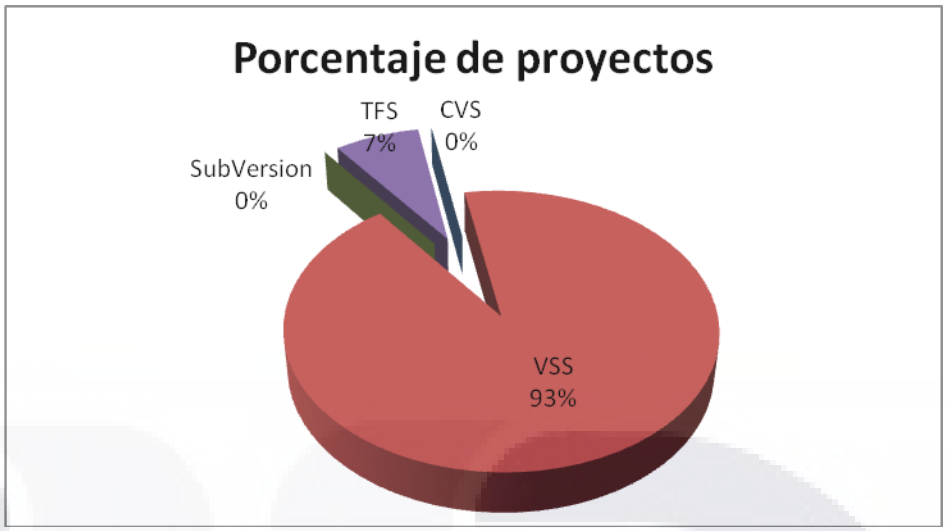


Ilustración 16 Porcentaje de uso de las herramientas evaluadas en proyectos Intranet

Como se puede ver, al momento de la obtención de dicha relación de proyectos y herramientas utilizadas, los sistemas controladores de versiones que se repartían la totalidad de los proyectos eran VSS con un 93% de los proyectos de Intranet y TFS con un 7%, como ya se mencionó, esta información fue recabada en noviembre del 2009, fecha en la cual, apenas se comenzaba con la utilización de TFS como una cuarta herramienta de control de versiones, quedando al final CVS y Subversion con un 0% cada uno, ya que en la Dirección no se tienen hasta el momento, proyectos de Intranet que se adapten o requieran de a estas herramientas.

De los resultados también se destaca que la tecnología mayormente utilizada para el desarrollo de los proyectos Intranet en la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones es la tecnología .NET por lo cual la herramienta de control de versiones seleccionada fue VSS, con una tendencia lógica a migrar hacia TFS.

Proyectos de Internet

Dentro de los proyectos que se encuentran bajo el esquema de control de versiones y que son dirigidos a la Internet, (ver Tabla 9) se tiene un total de 29 proyectos.

Nombre del proyecto	Tipo de proyecto	CVS	VSS	Subversion	TFS
Cuéntame	NET		√		
Librerías del Sitio INEGI	ASP		√		
Pruebas de stress de aplicaciones en producción	Documentación		√		
Web Service de presentación de información económica de coyuntura	NET		√		
Chat Institucional	ASP				√
Tienda Virtual INEGI	ASP		√		
Glosario de términos institucionales	NET			√	
CEPAFOP	ASP		√		
SAIC	ASP		√		
SINEVI	NET			√	
Registro único de usuarios	ASP		√		
Componente de consulta multidimensional de datos - Cubos	ASP		√		
Componente de consulta de tabulado de censos - Consulta dinámica de cuadros	ASP		√		
Componente para gráficas	ASP		√		
Componente de exportación de datos	ASP		√		
Indicadores	NET		√		
Mapa del sitio INEGI - Mapa Genérico	NET		√		

Tabla 9 (parte 1). Proyectos de Internet bajo control de versiones

Nombre del proyecto	Tipo de proyecto	CVS	VSS	Subversion	TFS
Sitio RNI	NET		√		
Foros SNEIG	ASP		√		
Sitio de Clasificadores	NET		√		
Sitio INEGI - Buscador del Sitio	NET		√		
Sitio SNEIG (Buscador incluido)	NET		√		
Buscador de Cuéntame	NET				√
Buscador del sitio TRI	NET				√
Buscador	NET				√
Mapa del Sitio - Mapa Genérico	ASP		√		
Intercambio de indicadores económicos BIE - Web Service	NET		√		
Buscador Siabuc	NET				√
Sitio INEGI	NET				√

Tabla 9 (parte 2). Proyectos de Internet bajo control de versiones

Al igual que lo desarrollos enfocados a la Intranet, la relación de proyectos y herramienta de control de versiones para los desarrollos enfocados hacia Internet, fue tomada a finales del mes noviembre del 2009 tiempo en el cual dentro de la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones apenas se comenzaba a realizar la migración de los proyectos que se tenían bajo Visual Source Safe a Team Foundation Server, siempre y cuando el proyecto y equipo de desarrollo así lo requirieran.

La Ilustración 17 nos muestra los porcentajes de uso de las herramientas de control de versiones para los proyectos Internet desarrollados dentro de la Dirección de Investigación y Desarrollo de Tecnologías de Información y Comunicaciones.

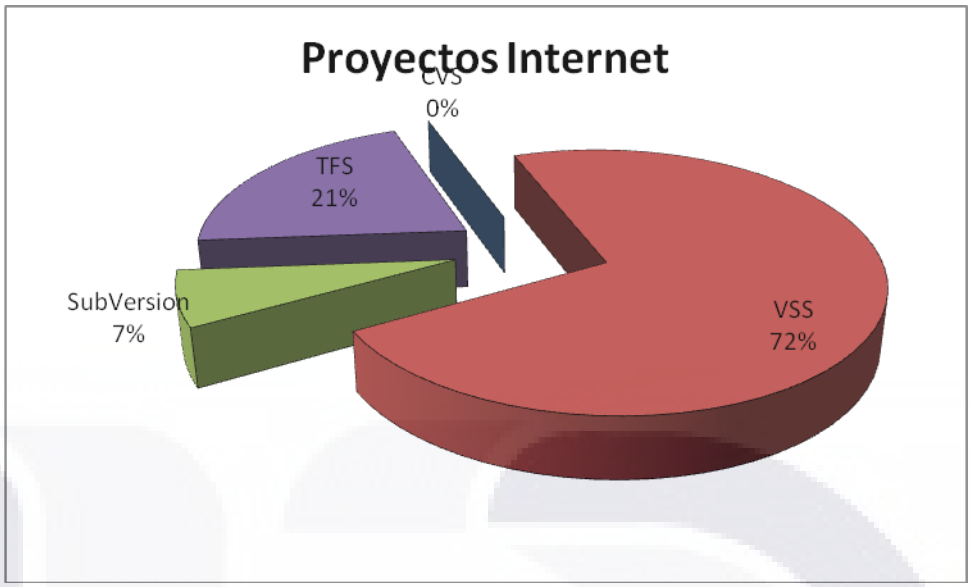


Ilustración 17 Porcentaje de uso de las herramientas evaluadas en proyectos Internet

Como resultado, se muestra una disminución de utilizar Visual Source Safe como herramienta de control de versiones, con el 72% de utilización y se nota un aumento en el uso de Team Foundation Server en relación a los proyectos dirigidos hacia la Intranet, donde Visual Source Safe era el sistema controlador de versiones que mayor porcentaje tenia. El aumento al uso de Team Foundation Server como herramienta de control de versiones se da en base a una migración "lógica" de los proyectos que se encontraban en VSS, hacia esta herramienta.

Cabe mencionar que en los proyectos destinados a Internet se comienza a utilizar Subversion, representado con un 7% de uso, esto debido a su gran facilidad de uso y características, que lo hacen un contendiente a Team Foundation server.

Para finalizar queda CVS con un 0% ya que en la dirección no se tienen hasta el momento de proyectos que se adapten a esta herramienta.

Fase 5: Generación de los procesos propuestos

Objetivo por fase

El objetivo de esta etapa es la generación de los procesos propuestos que sirvan como base para la implantación y manejo de un sistema controlador de versiones, así como de formatos (si se consideran necesarios) para la adecuada administración de la configuración, enfocándose en logra el nivel 2 de acuerdo al marco de referencia CMMI.

Desarrollo de la fase

Para dar cumplimiento a los objetivos uno y dos de presente trabajo de tesis, Desarrollar un proceso de implementación y manejo de un sistema controlador de versiones con base al modelo CMMI nivel 2 dentro del Instituto

Proceso propuesto

¿Porqué utilizar el CMMI continuo como modelo?

Se determino seguir el modelo CMMI Continuo ya que ofrece una mayor flexibilidad cuando se utiliza en la mejora de procesos. Una organización puede elegir mejorar el rendimiento de un punto problemático relacionado con un solo proceso, o puede trabajar en varios dominios que están fuertemente alineados con los objetivos estratégicos. La representación continua también permite que una organización mejore diferentes procesos a diferentes niveles (Chrissis et al., 2006), además de ser un modelo internacionalmente reconocido y uno de los de mayor uso y referencia en el mercado.

Cada una de las áreas de procesos se encuentran conformadas por diferentes prácticas específicas y generales. Estas prácticas permiten conocer el nivel de madurez de cada uno de los procesos (Álvarez Rodríguez et al., 2008).

Comparativa Proceso propuesto vs. Proceso actual vs. Proceso Informal

En la Tabla 10 se muestra una comparativa del proceso propuesto contra el proceso actual y el proceso informal, alineándolos a las metas específicas y prácticas específicas del marco de referencia CMMI nivel 2 para la gestión de la configuración para contar con un panorama de la situación que actualmente se tiene en materia de gestión de la configuración del software en el instituto.

Metas especificas y prácticas especificas CMMI	Proceso propuesto	Proceso actual	Proceso Informal
Establecer líneas base.	√		
• Identificar elementos de configuración.	√		√
• Establecer un sistema de gestión de configuración.	√		
• Crear o liberar líneas base.	√		
Seguir y controlar los cambios.	√		√
• Seguir las peticiones de cambio.	√	√	√
• Controlar los elementos de configuración.	√		√
Establecer la integridad.	√		
• Establecer registros de gestión de configuración.	√		
• Realizar auditorías de configuración.	√		

Tabla 10 Comparativa en base a las practicas especificas del modelo CMMI entre el proceso propuesto, el proceso actual y el proceso informal

Como se puede apreciar, el proceso propuesto cumple en mayor o menor medida con la totalidad de las prácticas específicas del modelo CMMI nivel 2 en cuanto a la gestión de configuración se refiere, a diferencia del proceso actual e informal las cuales dejan de lado el establecer un sistema de gestión de la configuración y el de realizar auditorías de la configuración, por lo cual se aprecia que el proceso actual se queda corto, en cuanto a lo dictado por el marco de referencia CMMI referente a la gestión de la configuración.

Proceso de implementación y manejo de control de versiones

La implementación y manejo de un sistema controlador de versiones es complejo, ya que impacta directamente a los datos, los procesos y a las personas, un mal entendimiento de este punto impacta en la implementación de esta tecnología, y es por eso la razón principal del porqué no se llega a un manejo adecuado (Kingsbury, 1996).

Es por esto antes que de seleccionar un sistema controlador de versiones para implementar para determinado proyecto, se deberá considerar lo siguiente:

- El tamaño y la complejidad que tendrá el sistema.
- El cambio hacia una plataforma Cliente/servidor.
- Software y hardware heterogéneo.
- Aspectos legales (licencias).

A todo esto se tiene que sumar el rechazo de los desarrolladores al manejo de un sistema controlador de versiones, varios desarrolladores de software tienen la perspectiva que un sistema controlador de versiones es intrusivo, y tiene poco conocimiento de lo que realmente es y de las ventajas a largo plazo que se tendrían con su implementación y manejo (Kingsbury, 1996).

5.1 Generación de los procesos

5.1.1 Proceso propuesto para la Implementación

Es por todo esto que en esta sección se desarrollara un proceso para la implementación y manejo de un sistema controlador de versiones que sirva de guía o checklist dentro del Instituto (ver Figura 12). Dicho proceso se encuentra basado en el proceso de “adopción de tecnologías SCM” (Kingsbury, 1996) en el cual destaca las fases que se necesitan seguir como guía para la adopción de sistemas de gestión de configuraciones.

1) Preparación y planeación.
2) Definición del proceso.
3) Evaluación de herramientas.
4) Implementación en un proyecto piloto.
5) Puesta en marcha.
6) Proceso de mejora.

Figura 12 Proceso propuesto para la implementación

Preparación y planeación

Los objetivos de esta fase son planificar las actividades de implementación de sistemas controladores de versiones, para establecer apoyo a la gestión, y para evaluar el estado de las actividades actuales de gestión del cambio.

En primer lugar, se propone la creación de un equipo de implementación del sistema controlador de versiones. El equipo de implementación será el responsable de la aplicación de la estrategia de implementación y desempeña el papel más importante en el esfuerzo de la implementación. Los miembros del equipo de implementación pueden estar conformados por:

- Un líder que es responsable de los esfuerzos de la implementación.
- Expertos técnicos que entienden la tecnología.
- Representantes de la comunidad de usuarios.

El equipo de implementación comienza por el desarrollo del plan de implementación y manejo del sistema controlador de versiones. El plan detallara

los beneficios del sistema controlador de versiones, desarrolla el calendario de implementación, define las políticas y procedimientos involucrados en el esfuerzo de la implementación y manejo, establece los criterios de éxito, y establece las funciones del equipo de implementación (Kingsbury, 1996).

Definición del proceso

Los objetivos de esta fase son evaluar el proceso actual de control de versiones y definir un nuevo proceso mejorado, si procede.

El proceso se analiza para identificar qué áreas se beneficiarán de la automatización. Un proceso definido es pertinente para la aplicación exitosa de un sistema controlador de versiones. Sin un proceso definido, la organización hará poco progreso en el esfuerzo de implementación (Kulpa & Johnson, 2003). Los pasos en esta fase son:

- Generación de planes de acción.
- Revisión, modificación y aprobación de plan.
- Asignación de trabajo de acuerdo al plan.
- Realizar el trabajo acordado al plan de acción.
- Desarrollo de métricas de soporte.
- Revisión y recomendación de herramientas.
- Actualización del plan de acción.

Evaluación de herramienta.

El objetivo de esta fase es seleccionar una herramienta que satisfaga los requisitos de la organización.

Antes de la iniciar con la evaluación de la herramienta, los requisitos de la herramienta son refinados y se asignan prioridades, se selecciona el método de evaluación, y se establecen escenarios de prueba necesarios para probar las capacidades de las herramientas. Es importante incluir a representantes de los

grupos de todos los usuarios en la evaluación para obtener una mejor comprensión de los diferentes grupos a utilizar la herramienta. En general, se evalúan por lo menos tres herramientas que satisfagan los requisitos de alto nivel seleccionados, y se realiza una evaluación a fondo. El equipo participa en la implementación y supervisa las evaluaciones de la herramienta (Kingsbury, 1996).

Implementación en un proyecto piloto.

El objetivo de esta fase es determinar cómo las herramientas de control de versiones seleccionadas, el proceso y procedimientos satisfacen los requerimientos de la organización. Un proyecto piloto permite probar la funcionalidad de las herramientas seleccionadas con datos reales en proyectos reales. En adición el proyecto piloto prevé de retroalimentación de los usuarios en respuesta al manejo y experiencia que estos vayan adquiriendo al probar las distintas herramientas implementadas (Kingsbury, 1996).

Es importante tener en cuenta que al seleccionar el proyecto piloto este se adecue a las necesidades de la prueba y que no se vea afectado de forma critica el quehacer de la organización.

En esta fase, la labor del equipo de adopción consistirá en desarrollar estándares, políticas y procedimientos que faciliten la capacitación de los usuarios, para la correcta implementación del sistema controlador de versiones que se encuentre a prueba (Kingsbury, 1996).

Algunas de las actividades que se realizaran en esta fase comprenden (Kulpa & Johnson, 2003):

- Selección del proyecto piloto.
- Orientación y entrenamiento a los miembros del proyecto piloto.
- Desarrollo del piloto.
- Monitoreo del proyecto piloto.
- Análisis de resultados.
- Actualizar el proceso y repetirlo en otro piloto si es necesario.

Puesta en marcha.

Una vez que se ha decidido que el piloto se ha completado correctamente, comienza la puesta en marcha. Esta fase se enfoca en la implementación del sistema controlador de versiones en otros proyectos (Kulpa & Johnson, 2003).

El equipo de implementación elaborará un plan de migración de los proyectos actuales hacia un sistema controlador de versiones, en esta fase el equipo de implementación tendrá que luchar contra la resistencia al cambio de los propios desarrolladores, completándose esta actividad cuando la herramienta seleccionada, el proceso y procedimientos se utilicen rutinariamente.

Actividades que se realizaran en esta fase (Kulpa & Johnson, 2003):

- Selección de uno o más proyectos actuales.
- Orientación y entrenamiento a los miembros del proyecto piloto.
- Monitoreo y medición del éxito.
- Análisis de resultados.
- Actualización del proyecto si es necesario.
- Implementación en más proyectos.

Proceso de mejora.

El objetivo de esta fase es el de capturar, recabar información de las estrategias implementadas en la implementación y manejo de un sistema controlador de versiones, con el propósito de evaluar y proponer mejoras el proceso aquí propuesto.

5.1.2 Proceso propuesto para el manejo de sistemas controladores de versiones

Para dar cumplimiento al objetivo número dos del actual trabajo de tesis, en esta sección se desarrollara un proceso para la manejo de la administración de la configuración que sirva de guía dentro del Instituto y que se encuentre alineado a los objetivos específicos y practicas específicas descritas en el marco de referencia CMMI.

Las actividades de la administración de los cambios o administración de la configuración es esencial para satisfacer los objetivos de los proyectos, los cabios surgen durante toda la evolución del ciclo de vida del proyecto y cada cambio da como resultado una versión diferente del producto.

Es por tal motivo que a continuación se muestra el proceso propuesto para el manejo de la administración de la configuración para el Instituto (ver Figura 13).

1. Desarrollo del plan de gestión de configuración del software.
2. Identificación de los elementos de configuración.
3. Establecimiento de la línea base.
4. Etiquetado los elementos de configuración.
5. Administración de los cambios.
6. Auditar el estado y contenido de las líneas base.
7. Liberación de la línea base.
8. Proceso de mejora.

Figura 13 Proceso propuesto para el manejo de la gestión de configuración

Desarrollo del plan gestión de la configuración del software.

El propósito de esta etapa es la de planear, monitorear y controlar las practicas del sistema controlador de versiones, para asegurarse que el desarrollo del software es llevado a cabo de acuerdo a lo definido en el modelo CMMI. El plan de la administración de configuración comienza cuando el proyecto puede ser inicializado, se cuenta con el medio ambiente necesario para su operación/desarrollo adecuado y los requerimientos se encuentran claramente especificados y documentados.

Las actividades llevadas a cabo en esta etapa serian:

- Configuración del control de cambios.
- Identificación de los roles y responsabilidades que desempeñaran los involucrados en el proyecto.
- Establecer la herramienta de control de versiones a utilizar.
- Establecer las políticas y restricciones de acceso.

El control de cambios envuelve el control y administración de los cambios del ciclo de vida del proyecto, el objetivo del control de cambios, como ya se ha visto es el de establecer los mecanismos que puedan ayudar a asegurar la integridad de los elementos de configuración.

En el plan de la administración de configuración, se deberán de plasmar, la fecha de elaboración, nombre del proyecto, integrantes del equipo de desarrollo y sus roles, usuarios, compromisos, metas a las que se pretende llegar y las actividades a desarrollar (calendario), para tal efecto se propone el uso de la *forma* ***INEGI-SCM1*** que se encuentra en el *Apéndice G* y se basa en la propuesta "Interpretación del Modelo de Madurez de Capacidades (CMM) para pequeñas industrias de software" (Álvarez Rodríguez et al., 2008).

Identificación de los elementos de configuración.

Un elemento de configuración consiste en múltiples productos de trabajo relacionados que forman la línea base, estos son documentos o artefactos explícitamente puestos bajo el control de la configuración. Como ya se ha visto estos pueden incluir: los requerimientos, especificaciones, documentación de diseño, código fuente planes de prueba y los manuales de usuario y mantenimiento, el diseño de interfaces etc.
Las actividades llevadas en esta etapa serán:

- Identificación de los elementos de configuración y productos de trabajo
- Identificar a los responsables de cada elemento de configuración

Establecimiento de la línea base.

El objetivo de esta fase es la de establecer una línea base para el proyecto, la gestión de configuración ayuda al establecimiento de la línea base poniendo los elementos de configuración bajo el control de la configuración, y la línea base solo podrá ser modificada a través de procedimientos de control de cambios.
Actividades a realizar

- Documentación de los elementos de configuración que son contenidos en la línea base.

Para identificar los elementos requeridos para cada línea base, se deberá especificar el nombre del proyecto, componente y una breve descripción que ilustre al componente que conforme la línea base, para tal efecto se propone el uso de la *forma* ***INEGI-SCM2*** que se encuentra en el *Apéndice G* y se basa en la propuesta "Interpretación del Modelo de Madurez de Capacidades (CMM) para pequeñas industrias de software" (Álvarez Rodríguez et al., 2008).

Etiquetado de los elementos de configuración.

Un aspecto importante de la administración de la configuración y control de cambios es el etiquetado de los elementos de configuración, Una vez identificados los elementos de configuración, estos deberán ser etiquetados de manera única. Un aspecto de la administración de configuración es el etiquetado de elementos de configuración. Los productos de software deben tener un nombre que no sea ambiguo y/o un identificador numérico.

El identificador único o etiqueta es usado para poder dar seguimiento a cada elemento de configuración y así poder generar reportes en los que se muestre el estado que guarden, un ejemplo de estos estado pueden ser: en análisis, desarrollo, pruebas o implementación. Para tal efecto se propone el uso del formato ***INEGI-SCM3*** que se encuentra en el *Apéndice G* y se basa en la propuesta "Interpretación del Modelo de Madurez de Capacidades (CMM) para pequeñas industrias de software" (Álvarez Rodríguez et al., 2008).

Administración de los cambios.

Los cambios son alteraciones permanentes a la línea base establecida, y estas pueden surgir cuando se recibe un nuevo requerimiento, o se desea una mejora, o cuando es necesario dar solución a un problema, entre otras fuentes de los cambios.

Las actividades que se llevan en esta etapa son:

- Iniciar, registrar la petición de cambio.
- Analizar el impacto de los cambios.
- Asignar el cambio a un integrante del equipo de desarrollo, si es que es aprobado.
- Realizar el cambio.
- Comprometer el cambio.
- Dar seguimiento al estado de las solicitudes de cambio.

Un ejemplo de un reporte propuesto para esta actividad pudiera ser la forma ***INEGI-SCM4***, la cual se encuentra en el *Apéndice G*

Auditar el estado y contenido de la línea base.

El principal objetivo de las auditorias es asegurase que el proceso de administración de configuración realmente se sigue de forma adecuada. La línea base del sistema puede ser auditada para asegurarnos que se cuenta con la integridad en la línea base

Las actividades que se realizan en esta etapa son:

- Auditorias funcional de la línea base
- Auditorias físicas

La auditoria funcional a la línea base consiste en una exanimación formal de registros de prueba que nos ayuden a verificar que las caristias funcionales del sistema cumplen con los requerimientos. Para tal efecto se propone el uso del formato ***INEGI-SCM5*** que se encuentra en el *Apéndice G* y se basa en la

propuesta “Interpretación del Modelo de Madurez de Capacidades (CMM) para pequeñas industrias de software” (Álvarez Rodríguez et al., 2008).

Las auditorias físicas es la exanimación formal a cada versión de código de una configuración. Consiste en una evaluación de la documentación técnica del sistema, comparándolo con las pruebas operacionales del sistema para asegurarnos que se cumple con los estándares acordados en los requerimientos.

Liberación de la línea base.

La liberación de la línea base se lleva a cabo cuando se alcanzan los milestone fijados en el proyecto, en esta liberación, se deberá de identifica el número de revisión o versión el cual va a ser liberado, características generales de la liberación así como su fecha de liberación.

Un ejemplo de un reporte propuesto para esta actividad pudiera ser la forma ***INEGI-SCM6***, la cual se encuentra en el *Apéndice G*

Proceso de mejora.

El proceso de mejora consiste en una revisión completa del proceso de administración de configuraciones propuesto, es una retroalimentación del proceso en donde se identifican debilidades, fortalezas y oportunidades de mejora.

CAPÍTULO IV Conclusiones

IV.1 Del objetivo general y objetivos específicos

Objetivo General

Desarrollar un procedimiento que sirva como base para la implementación y manejo de un sistema controlador de versiones dentro del Instituto, recomendando mediante la investigación y su respectivo análisis, cual sistema controlador de versiones es el más adecuado para su uso dentro del Instituto, aunado a esto, dicho proceso deberá estar debidamente sustentado bajo el marco de referencia CMMI.

El objetivo general se cumple de manera adecuada puesto que se propone un proceso para la implementación de un sistema controlador de versiones y otro proceso para el manejo del control de versiones, los cuales se encuentran debidamente sustentados conforme a lo que dicta al marco de referencia CMMI nivel 2 en cuanto a la administración de la configuración, a su vez mediante la investigación y el análisis de distintas herramientas de control de versiones, se determinó principalmente que la elección de un sistema controlador de versiones depende principalmente de la tecnología a utilizar como plataforma de desarrollo, siendo las herramientas de Microsoft Team Foundation Server y Subversion las que mejor desempeño y que se encientan más apegadas al marco de referencia CMMI.

Objetivo Específico 1

Desarrollar un procedimiento detallado que permita la implementación de un sistema controlador de versiones con base al marco de referencia CMMI nivel 2 dentro del Instituto.

Del objetivo especifico 1 se puede concluir que se cumplió de manera adecuada, puesto que se elaboró el proceso propuesto para la implementación de un sistema controlador de versiones el cual nos permite la adecuada selección e implementación de una herramienta de control de versiones.

Objetivo Específico 2

Desarrollar un procedimiento detallado que permita el manejo de un sistema controlador de versiones con base al marco de referencia CMMI nivel 2 dentro del Instituto.

Del objetivo especifico 2 se puede concluir que se cumplió de manera adecuada, puesto que se desarrolló el proceso enfocado al manejo del control de versiones, proceso que complementario al de implementación puesto que es en este proceso donde se realiza el llenado de formatos de documentación, los cuales no todas las herramientas de control de versiones contemplan y que son de vital importancia para el adecuado control y seguimiento de los cambios que surjan durante el ciclo de vida de desarrollo de los sistemas.

Objetivo Específico 3

Determinar en qué grado cumple el proceso de control de versiones actual con relación a lo que se define en el marco de referencia CMMI nivel 2.

Fue mediante análisis e investigación, la forma en cómo se logró dar cumplimiento al objetivo específico 3 logrando identificar el grado en que el proceso actual cumple a lo referente de la gestión de configuración, pero principalmente las carencias que se tienen en cuanto lo dictado en el marco de referencia CMMI.

Como se puede apreciar en la Tabla 10 “Comparativa en base a las prácticas específicas del modelo CMMI entre el proceso propuesto, el proceso actual y el proceso informal”, el proceso actual solo cumple con el *seguimiento de las peticiones de cambio* mediante el uso y llenado de un único formato que compone la actual metodología de desarrollo INEGI, faltando las practicas especificas de “establecimiento de líneas base” y el “establecimiento de la integridad”.

Objetivo Específico 4

Investigar que sistemas comerciales, tanto en su modalidad de software libre o propietario, se recomiendan para su uso como herramienta de control de versiones en el INEGI.

Mediante la investigación de distintas herramientas de control de versiones, fue como se dio cumplimiento al objetivo especifico 4, dándonos como resultado que Subversion es la mejor herramienta de código libre y Team Foundation Server es la mejor recomendación en cuanto a herramientas de código cerrado o propietario se refiere, puesto que dichas herramientas fueron las que más se apegaron en cuanto a las actividades y características que debe contener una herramienta de control de versiones y como quedo demostrado en la Tabla 7 (parte 1). Características más representativas de los sistemas controladores de versiones (Mahotkin, 2008) .

IV.2 De las preguntas y proposiciones

Pregunta 1 *¿Cómo se lleva a cabo el control de versiones en el Instituto y en qué grado cumple el proceso actual con lo que dicta el marco de referencia CMMI nivel2?*

Como se demostró en la fase 2 de la metodología para el desarrollo del caso, (Estudio del proceso actual) el control de versiones en la metodología de desarrollo con la que cuenta el Instituto no especifica el uso de un sistema de control de versiones, ya que formalmente solamente cuenta con un formato para dar seguimiento a las solicitudes de cambios que surjan en las líneas base de los desarrollos de software, si hablamos de compararlo con los niveles de madurez del marco de referencia CMMI en su modelo continuo se puede decir que el actual proceso que se lleva a cabo en el INEGI se encuentra en un nivel 1 de madurez, puesto que su desempeño no es estable, se sobrepasan los calendarios fijados para el desarrollo de software, pero por lo regular el trabajo es realizado.

Pregunta 2 *¿Qué actividades se deben considerar en el proceso propuesto para la implementación de un sistema controlador de versiones, que de cumplimiento al marco de referencia CMMI nivel 2 en cuanto a gestión del cambio se refiere?*

Como se menciona en el análisis del marco de referencia CMMI nivel 2 en cuanto a la administración de la configuración se refiere, el objetivo de la administración de la configuración es establecer y mantener la integridad de los productos de trabajo (CMMI Product Team, 2002), dicha integridad se logra mediante los procesos de evaluación, selección e implementación de herramientas de control de versiones que nos faciliten el trabajo de:

- Identificar los elementos de configuración.
- El control de cambios en la configuración de elementos
- Mantener la integridad de las líneas de base.
- Proporcionar situación exacta y datos de configuración actuales para los desarrolladores, usuarios finales y clientes.

Pregunta 3: *¿Qué actividades se deben considerar en el proceso propuesto para el manejo de un sistema controlador de versiones, que se encuentre debidamente alineado al marco de referencia CMMI nivel 2 en cuanto a la gestión de cambios se refiere?*

En el apartado del proceso propuesto para el manejo sistemas controladores de versiones, se mencionan las actividades con las que cumple el proceso propuesto y las cuales se encuentran debidamente fundamentadas con el modelo continuo del marco de referencia CMMI nivel 2, en el cual se llevan a cabo las prácticas específicas que dicho modelo y las cuales son:

- Establecimiento de la línea base
    - Identificación y control de elementos
    - Establecer un sistema de gestión de la configuración
    - Crear o lanzar líneas bases.

- Seguimiento y control de los cambios
    - Rastreo de solicitudes de cambio
    - Control de elementos de configuración
- Establecimiento de la integridad.
    - Establecer registros de administración de la configuración
    - Realizar auditorías de configuración

Pregunta 4: *¿Cuál de los sistemas de control de versiones analizado es el más apegado al CMMI?*

Para dar respuesta a la pregunta 4 se puede concluir que la mejor herramienta y la que más cumple con el marco de referencia CMMI es Team Foundatin Server. En la Tabla 5 se hace un análisis comparando las actividades generales que se tienen que llevar a cabo para un correcto control de versiones y gestión de la configuración, de acuerdo a lo definido en CMMI nivel 2 contra las actividades que desempeñan las herramientas automatizadas evaluadas en el actual trabajo de tesis.

Cabe mencionar que Team Foundatin Server se alinea más una metodología, puesto que cuenta con plantillas de procesos predefinidas, donde cada plantilla de procesos de Team Foundation Server proporciona un conjunto diferente de elementos de trabajo predeterminados, consultas de elementos de trabajo, plantillas de producto, informes, grupos de seguridad e instrucciones y que se encuentran alineados al maco de referencia CMMI, así como a las metodologías MSF, RUP, SCRUM, etc. entre otras. No así las otras herramientas evaluadas, las cuales, algunas solo desempeñan las tareas de control de versiones, y otras, es con la utilización de herramientas adicionales la forma en cómo se puede explotar y/o robustecer la funcionalidad de estos sistemas, sin embargo, y como resultado del análisis de la Tabla 5 se puede apreciar que Subversion es la herramienta de código abierto que cumplió con la mayoría de las actividades recomendadas por el marco de referencia CMMI, con la gran ventaja de no tener que pagar costosos licenciamientos como ocurre en el caso de Team Foundation Server.

Proposición 1: *El proceso que se lleva actualmente en el INEGI para el control de los cambios no contempla el uso de un sistema de control de versiones.*

La proposición 1 es cierta, ya que fue mediante el análisis de la actual metodología la forma en cómo se identificó que el proceso que se lleva actualmente no contempla el uso de un sistema de control de versiones como herramienta que ayude a una adecuada gestión de la configuración ya que establece el uso de un único formato para llevar el control de los cambios.

Proposición 2: *El proceso que se lleva actualmente en el INEGI no cumple con lo estipulado en CMMI.*

Como se demuestra en la Tabla 10, la proposición 2 se dice que es cierta, puesto que el proceso actual no cumple con lo estipulado en el marco de referencia CMMI ya que solo cuenta con un formato que sirve para el seguimiento de las peticiones de cambio, pero que deja de lado actividades tales como: la identificación de los elementos de configuración, el control de los elementos de configuración, el establecimiento de registro de gestión de configuración y la realización de auditorías de configuración.

Proposición 3: *El proceso propuesto para la implementación de un sistema controlador de versiones complementa al proceso actual.*

Cierto, el proceso propuesto para la implementación de un sistema controlador de versiones complementa al proceso actual, puesto que como ya se mencionó el actual proceso queda corto con relación a las actividades de control de versiones que se deben desempeñar en el marco de referencia CMMI (ver Tabla 10 Comparativa en base a las practicas especificas del modelo CMMI entre el proceso propuesto, el proceso actual y el proceso informal) por lo cual se tuvo que desarrollar un procedimiento que permita de primera instancia una pronta implementación de un sistema controlador de versiones para ampliar las actividades de control de cambios que se llevan en la actual metodología de desarrollo con la que cuenta el INEGI.

Proposición 4: *El proceso propuesto para el manejo del control de versiones complementa al proceso actual.*

Al igual que la proposición 3, la proposición 4 es cierta puesto que el proceso propuesto para el manejo de un del control de versiones, además de estar apegado a lo que dicta el marco de referencia CMMI, complementa las actividades que actualmente se llevan a cabo en la actual metodología de desarrollo con la que cuenta el INEGI, ya que dicho proceso nos indica cómo llevar un adecuado control de los cambios y de las versiones apoyándonos en las herramientas (sistemas controladores de versiones) y en el adecuado llenado de los formatos propuestos para subsanar las carencias identificadas en el proceso actual.

Proposición 5: *Siguiendo los procedimientos propuestos se tendrá un mayor control de versiones sobre los cambios que surjan en los desarrollos de software dentro del Instituto.*

La proposición 5 es cierta puesto que con la correcta implementación de un sistema controlador de versiones y mediante el manejo de la administración de la configuración propuesto en los procesos, los desarrollos de software que se realicen en el Instituto contaran con las bases necesarias para llevar un correcto control de cambios debidamente sustentado con el modelo CMMI nivel 2.

IV.3 Relación con las áreas de conocimiento vistas en la maestría que se utilizaron para la realización del caso.

El actual trabajo de tesis se encuentra muy relacionado con el área de Ingeniería de Sistemas, por su estrecha relación que guarda la administración de la configuración, el control de los cambios y el control de versiones con el desarrollo de sistemas de información, a su vez, se encuentra íntimamente ligado a las aéreas de conocimiento de Business Process Management, para el modelado de negocios y modelado de procesos, puesto que se desarrollaron los procesos para necesarios para la implementación y manejo de un sistema controlador de versiones en el INEGI.

En la Tabla 11 se muestra una relación de las materias cursadas en la maestría y los temas vistos, que se relacionan con el actual trabajo de tesis.

Materias recibidas	Temas relacionados
Ingeniería de Sistemas	Proceso de desarrollo de software Modelos de Madurez • CMM • CMMI Administración de proyectos • Monitoreo y control Herramientas CASE
Sistemas de Información en Manufactura	Proceso de negocio Modelado de procesos Automatización de flujos de trabajo
BPM - Business Process Management	Integridad y calidad en los procesos Automatización en los procesos

Tabla 11 Relación con las áreas de conocimiento vistas en la maestría que se utilizaron para la realización del caso.

Como se puede ver la materias vistas durante el estudio de esta maestría y con las que se guarda una mayor relación en la realización de esta tesis es Ingeniería de Sistemas, puesto que ahí fue donde se ve el proceso de desarrollo de sistemas así como se abordaron temas Madurez de los proceso de desarrollo, calidad y Administración de proyectos, etc., y las materias de Sistemas de Información en Manufactura y BPM ya que nos abrieron el panorama en cuanto a la visión que se deben de tener al querer mejorar e implementar procesos en nuestras áreas de

trabajo.

IV.4 Lecciones aprendidas

Lo que se aprendió durante la realización del actual trabajo de tesis fue lo siguiente:

1. Resultado del análisis y estudio de los marcos de referencia, se aprendió la forma en cómo en cada uno de ellos aborda la problemática del correcto control de versiones, la correcta administración de los cambios, destacándose que si bien, siendo algunos de esto marcos de referencia distintos en cuanto a su enfoque, puesto que algunos se enfocan plenamente al desarrollo de software y otros a la gestión de servicios e infraestructura, hacen uso de procedimientos muy similares para controlar los cambios y gestionar de manera adecuada las configuraciones de los sistemas que se están controlando.
2. También se aprendió la necesidad de contar con una metodología en la cual nos lleve a un adecuado control de versiones y una adecuada gestión de la configuración, realizando las actividades de:
    - La identificación adecuada de los elementos de configuración.
    - El establecimiento de las líneas base.
    - Un adecuado control de cambios.
    - Gestión y seguimiento de los cambios.
    - Y que se realicen auditorias a la configuración.
3. Algo que si hay que tener en cuenta al pensar en implementar un proceso para el manejo de un control de versiones en un área de desarrollo de software o en organización en al cual no se cuente con experiencia previa en cuanto al uso de herramientas automatizadas que nos sirvan para llevar el control e versiones, así como el llenado adecuado de formatos, es el que esto implica un cambio total en la cultura organizacional, y por lo tanto se

requerirá de tiempo, capacitación y mucha paciencia para que esta implementación pueda llegar a dar frutos.

CAPÍTULO V Recomendaciones

V.1 Recomendaciones

Como se ha demostrado en el presente trabajo de tesis, el contar con un sistema de control de versiones que permita una adecuada gestión de los cambios, es un factor fundamental para lograr el éxito dentro del las actividades de desarrollo software, ya que sin un adecuado control de los cambios aún en etapas tempranas en el proceso de desarrollo, pueden ocasionar grandes dolores de cabeza para los miembros del equipo de desarrollo, por lo cual se recomienda ampliamente el uso de herramientas de control de versiones así como el seguimiento de un proceso para su manejo adecuado, ya que con esto nos dará un mayor control de los cambios que surjan durante el ciclo de vida de desarrollo de sistemas en el Instituto.

Por lo tanto se recomienda el involucramiento de los directivos del Instituto, para que los proceso aquí propuestos tanto para la implementación y manejo de sistemas de control de versiones formen parte de una actualización a la actual metodología del desarrollo de sistemas, ya que con la inclusión de estos proceso, se subsanarán las fallas y debilidades con las que está actualmente cuenta y las cuales fueron puestas de manifiesto en la *fase 3* de la metodología utilizada para el desarrollo del caso.

A su vez se recomienda prestar mayor atención al marco de referencia CMMI, en cuanto a lo que dicta a la gestión de la configuración, más que a las herramientas destinadas para el control de versiones como tal, puesto que es mediante su estudio el cómo nos podemos dar cuenta lo que realmente requiere para un adecuado control y seguimiento de los cambios, aparte de ser un referente internacional y aporta un plus a los procesos que desempeñemos actualmente en el Instituto.

V.2 Futuros Trabajos

Queda para trabajos futuros, aplicar los procesos propuestos, primero a nivel dirección general, después a nivel Institucional, evaluando el impacto que estos tendrían en la práctica, poniendo mucha atención en cuanto a si se logra un mejoramiento en cuanto al control de los cambios y a la administración de la configuración, siendo bajo retroalimentación de los involucrados, la forma en cómo estos procesos pudieran ser mejorados y adaptados para que un futuro estos lleguen a ser una actualización de la actual metodología de desarrollo de software en cuanto al tema de control de cambios y gestión de la configuración se refiere, ¿y por qué no? pensar en una futura certificación en CMMI para el Instituto.

V.3 Futuras Investigaciones

Queda para futuros trabajos de investigación, analizar las herramientas más representativas de control de versiones o gestión del cambio que faltaron de investigar, como puede ser el caso y solo por mencionar algunas de estas herramientas son: GIT, Mecurial, IBM Rational ClearCase, etc. Así como realizar una investigación más a fondo sobre herramientas o conjunto de herramientas de código libre y/o propietario, que nos faciliten una adecuada gestión de proyectos, seguimiento de errores y casos, y que en funcionamiento, vinieran siendo algo muy parecido al realizado por Microsoft Team Foundation Server.

Apéndices

Apéndice A

Manual de CVS

Instalación

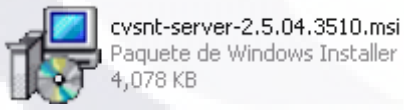
En el presente documento se verá la instalación y la creación de un repositorio con CVSNT

Pre-requisitos:

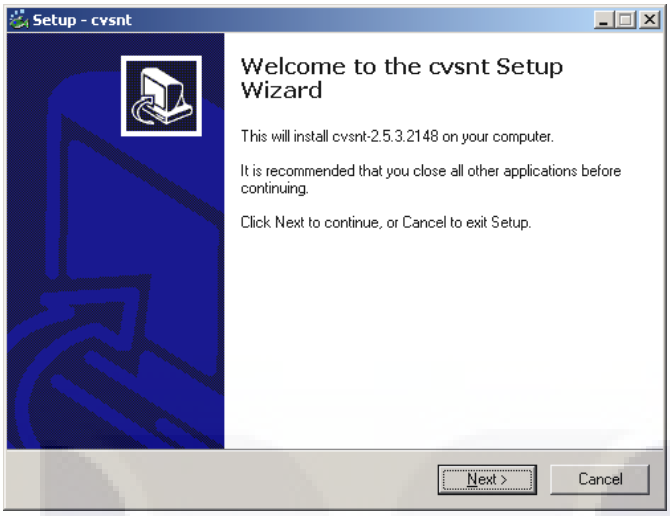
- Tener privilegios de Administrador en el servidor donde se encuentra el repositorio de proyectos.
- Descargar el servidor CVS de la página http://www.cvsnt.org, se recomienda descargar la versión estable
- Antes de iniciar con la instalación es recomendable la creación de dos directorios, para este ejemplo fueron:

  C:/Repositorios/Prueba

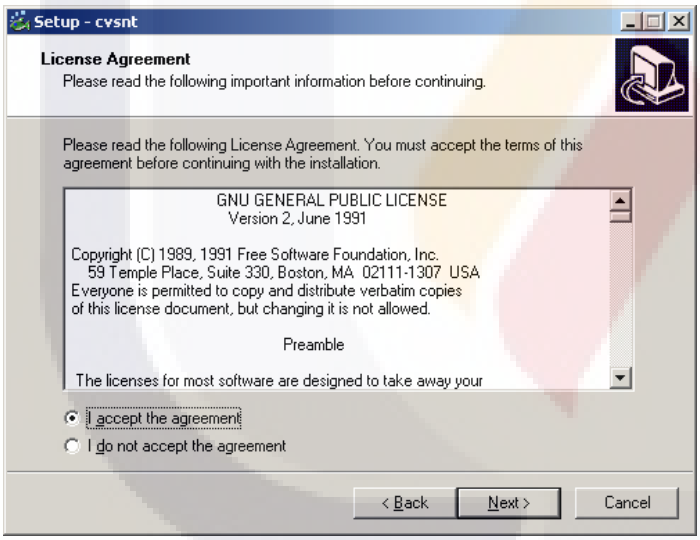
  C:/Repositorios/Prueba/cvstemp
- Ejecutar el instalador:



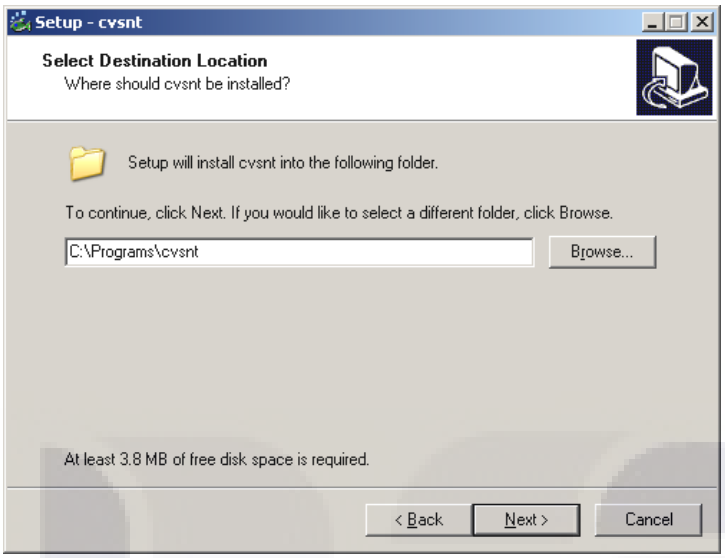
Pantalla de bienvenida



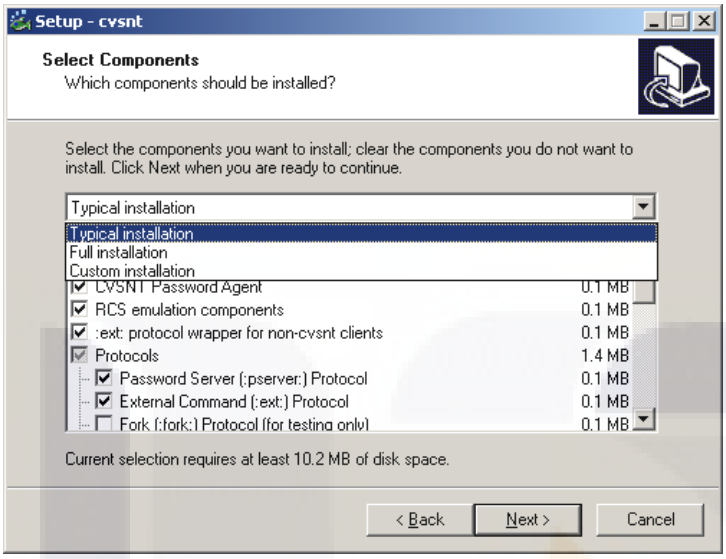
Pantalla de licenciamiento



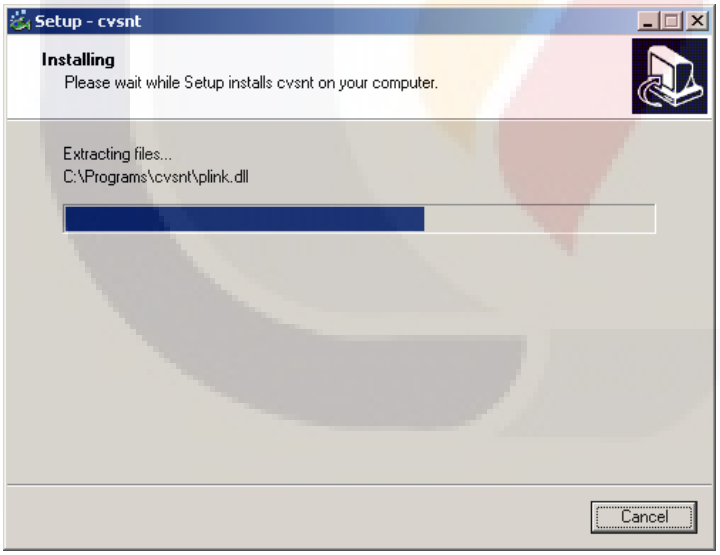
Selección del directorio donde CVSNT será instalado



Selección de componentes



Pantalla de progreso de instalación



Creando un proyecto nuevo

Una vez que se ha instalado el servidor, el siguiente paso es empezar a trabajar con el cliente.

1. Abrimos una consola (o línea de comando, ya sabes: Inicio->ejecutar->cmd).
2. Indicamos al cliente dónde está el repositorio:

   set cvsroot=C:/Repositorios/Prueba
3. Ahora tenemos que indicar al CVS que queremos crear un nuevo proyecto. Vamos al directorio donde está nuestro proyecto, por ejemplo c:\misprogramas\miproyecto y tecleamos en la consola:

   cvs import -m "Creación del proyecto" miproyecto usuario start

-m “creación del proyecto”: cada ver que enviamos una actualización al CVS o creamos un nuevo proyecto podemos añadir un mensaje, como por ejemplo “Creación del proyecto”. Es muy útil para tener una idea general de qué cambios se han hecho. Si no ponemos este parámetro se abrirá un editor de texto (probablemente el notepad) y nos pedirá que escribamos un mensaje.

miproyecto: es el nombre del proyecto. Este nombre nos permitirá acceder a los ficheros que hemos enviado al repositorio más adelante.

“usuario” y “start” no se usan habitualmente, así que por ahora no se trataran en el presente documento.

Con esto ya tenemos nuestra primera versión del proyecto guardada en el repositorio. Puedes teclear:

cvs ls

Para listar el contenido del repositorio, verás que hay dos directorios, uno llamado “CVSROOT” y otro “miproyecto”.

Obteniendo el código del repositorio

Antes de hacer nada más tenemos que actualizar el proyecto con el contenido del CVS, para eso teclearemos desde c:\misprogramas:

cvs checkout miproyecto

Y nos actualizará el contenido del directorio “miproyecto” con el contenido del repositorio.

Enviar modificaciones al repositorio

Una vez estés satisfecho con tu trabajo puedes enviar los cambios al repositorio. Entra en la carpeta c:\misprogramas\miproyecto y teclea:

cvs commit -m "Modificaciones al proceso de carga de archivos"

Este comando comprobará todos los ficheros del proyecto y, si alguno ha sido modificado, lo enviará al repositorio sin sobrescribir las versiones anteriores.

Si sólo quieres enviar un fichero determinado puedes hacer por ejemplo:

cvs commit -m "Sólo he cambiado un par líneas " fichero.cs

Al enviar un fichero al repositorio se crea una nueva revisión del mismo. La primera vez que enviamos un fichero (con el comando import) se crea la revisión 1.0. Si hacemos cambios en el fichero y lo volvemos a enviar se creará la revisión 1.1, la segunda vez que lo enviemos será la 1.2.

Clientes de CVS

Tortoise CVS – Un cliente de CVS muy cómodo y sencillo de utilizar que se integra en el explorador de Windows y se usa desde los menús contextuales del botón derecho del ratón.

WinCVS – Un cliente más completo que TortoiseCVS (y en el que se basó originalmente).

Apéndice B

Manual de Microsoft Visual SourceSafe

Creación de un Proyecto en el Repositorio de VSS.

En el presente documento veremos cómo dar crear un proyecto en el repositorio de proyectos, con el controlador de versiones Microsoft Visual SourceSafe.

Pre-requisitos:

- Tener privilegios de Administrador en el servidor donde se encuentra el repositorio de proyectos.
- Instalar el Administrador y cliente de Visual SourceSafe.
- Identificar el área del proyecto (Internet, Intranet o Investigación), la tecnología a utilizar, la Dirección a la que pertenece, el nombre del proyecto, así como los usuarios y privilegios que tendrán los mismos para acceder al proyecto.

Crear el directorio en el servidor

1. Creamos el directorio en el servidor. Por ejemplo en D:\VSS_Prueba.
2. Compartir el directorio.

Ilustración 18 VSS Compartir una carpeta

3. A continuación administraremos los permisos para el directorio compartido. Eliminar el usuario **Todos** (Everyone) y agregar los usuarios que accederá al proyecto, así como los permisos de lectura y cambios.

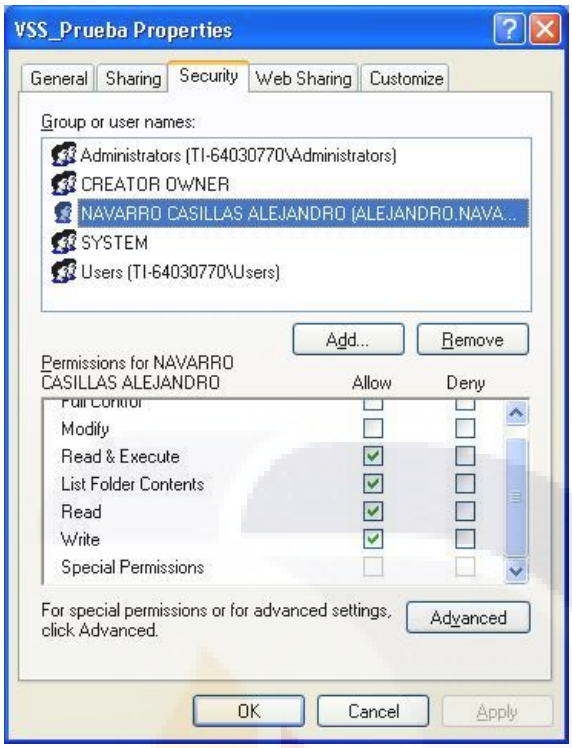


Ilustración 19 VSS Seguridad

4. Ahora damos clic en la pestaña de Seguridad y agregamos los mismos usuarios que dimos de alta en los permisos para el directorio compartido (Paso anterior). Seleccionamos cada uno de los usuarios que agregamos y les damos el permiso de Modificar y damos clic en el botón Aceptar.

Crear y Configurar el Proyecto en VSS

En la presente sección vamos a Crear y configurar la base de datos para nuestro proyecto.

1. Abrimos el Administrador de Visual Source.

Ilustración 20 VSS Administrador

2. Dar clic en Abrir base de datos de SourceSafe del menú Archivo

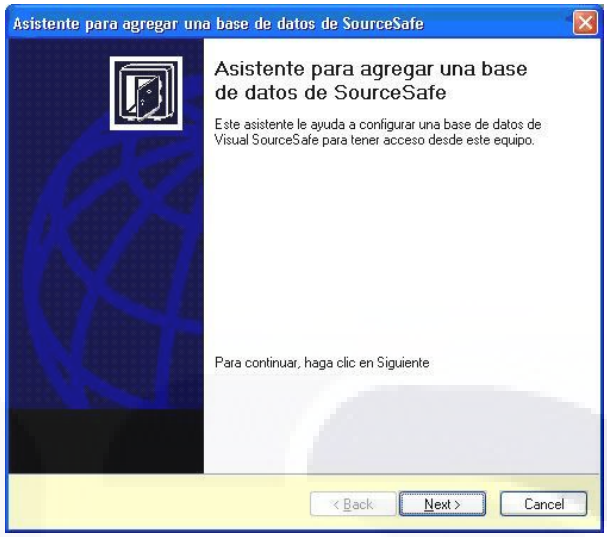


Ilustración 21 VSS Asistente para agregar una base de datos

3. Clic en Siguiente [Next].

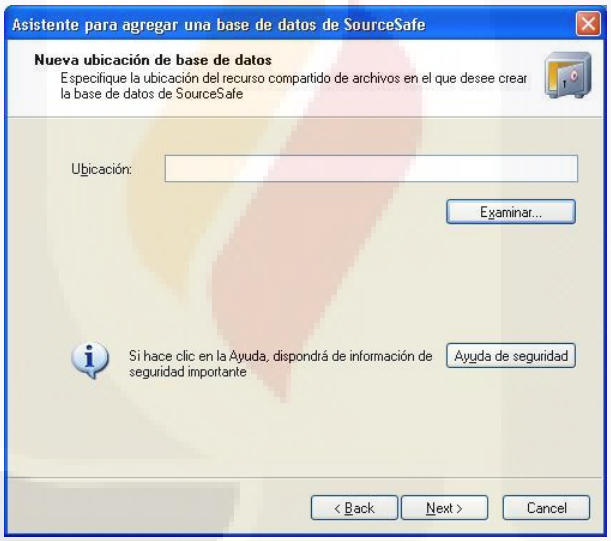


Ilustración 22 VSS Ubicación de la base de datos

4. En la presente pantalla indicamos la ruta donde crearemos la base de datos de SourceSafe para nuestro proyecto y damos clic en Siguiente [Next].

Para el ejemplo que estamos siguiendo, estableceríamos la ruta D:\VSS_Prueba

Ilustración 23 VSS Ruta de la base de datos

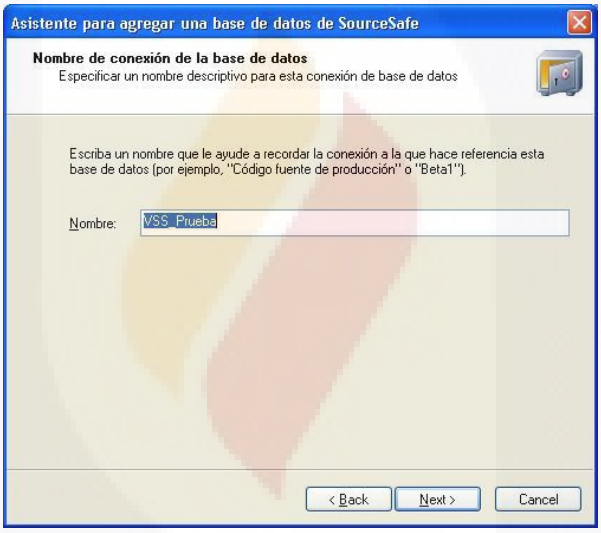


Ilustración 24 VSS Nombre de la base de datos

5. A continuación establecemos el nombre de nuestra conexión a la base de datos del proyecto.

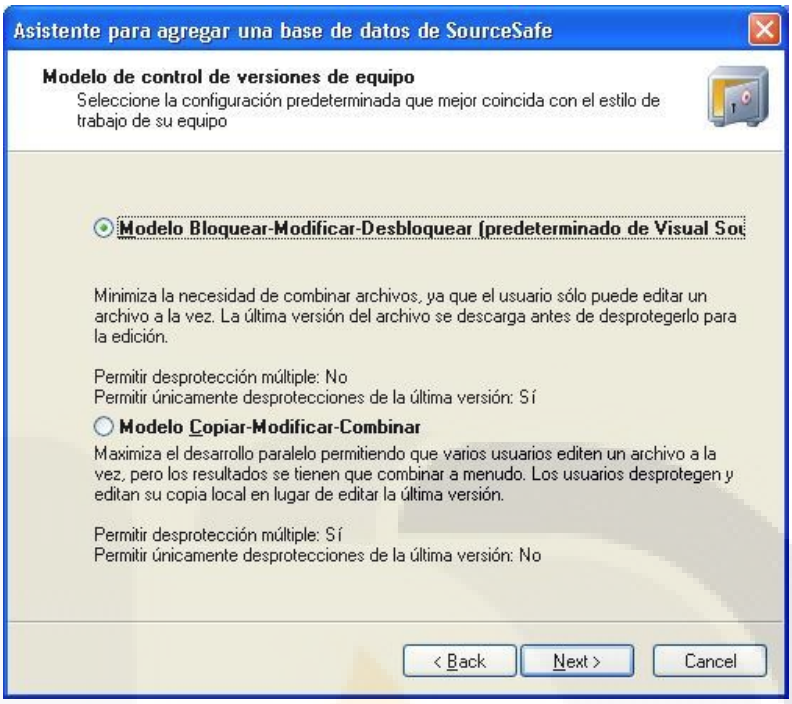


Ilustración 25 VSS Modo de control de versiones

6. Aquí seleccionamos el tipo de control de versiones que deseamos para nuestro proyecto. El más común y para nuestro ejemplo seleccionamos Modelo Bloquear-Modificar-Desbloquear y damos clic en el botón Siguiente [Next]. Si deseamos, más adelante podemos cambiar el tipo de modelo.

Nota:

Modelo Bloquear-Modificar-Desbloquear.- Este tipo de modelo nos permite que solo una persona a la vez pueda actualizar un archivo, por ejemplo, si el Desarrollador Juan está actualizando la clase CLSM_Ejemplo1 y el Desarrollador Pedro necesita agregar un método, no lo podrá hacer hasta que Juan termine de hacer uso de ella y la libere, en ese momento Pedro podrá descargar la última versión de CLSM_Ejemplo1 (con los cambios de Juan) y ahora ya podrá proceder a actualizar dicha clase. **Recomendado**.

Modelo Copiar-Modificar-Combinar.- Este modelo permite que más de una persona trabaje sobre un mismo archivo y al subir el archivo permite realizar una combinación de cambios entre el documento que se encuentra

en el servidor y cada una de las versiones que se vayan a subir al servidor. Se recomienda para usuarios avanzados y casos especiales.

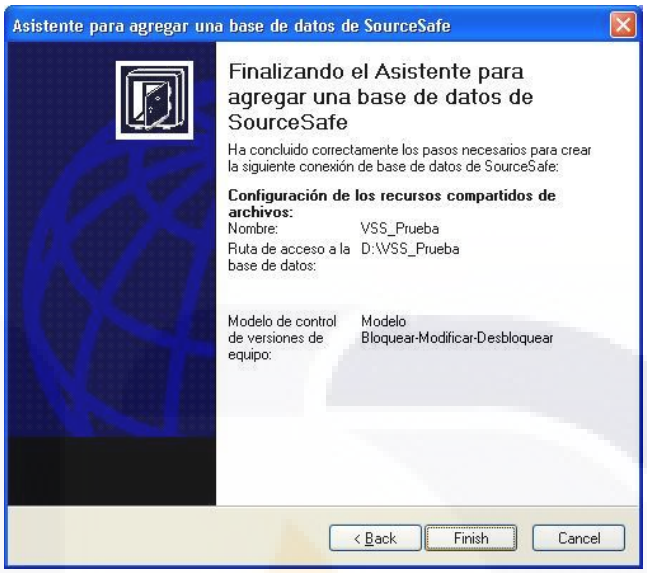


Ilustración 26 VSS Finalización del Asistente

7. Clic en el botón Finalizar [Finish].

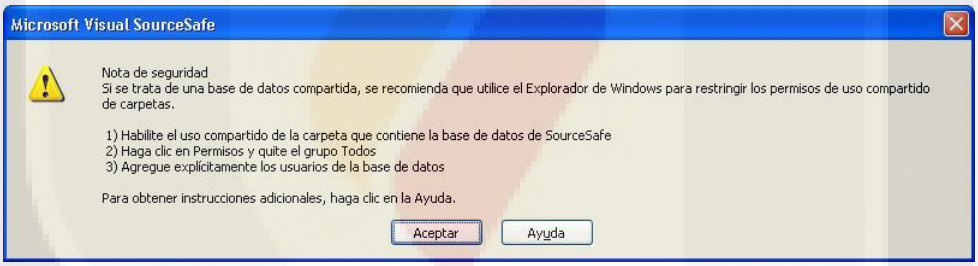


Ilustración 27 VSS Resumen

8. Una vez que Visual SourceSafe finalice de crear la base de datos para nuestro proyecto presentará el siguiente mensaje, en el cual se nos orienta a que como el Directorio donde se encuentra la base de datos del proyecto es una carpeta compartida seamos cuidadosos de que usuarios podrán tener permiso de acceder a este directorio. Damos clic en el botón Aceptar y listo ya hemos creado la base de datos para nuestro proyecto.

Agregar un usuario

A continuación vamos a ver los pasos a seguir para agregar un usuario a nuestro repositorio del proyecto.

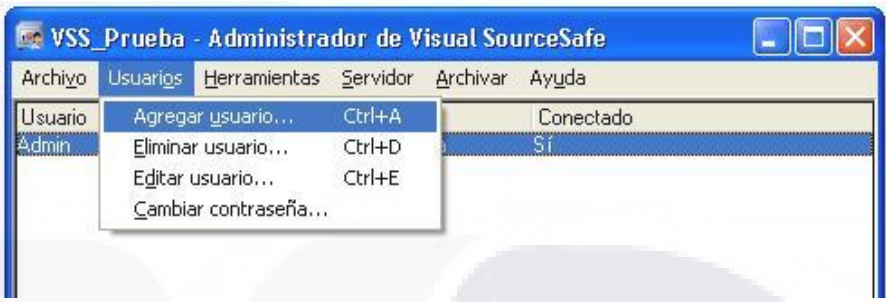


Ilustración 28 VSS Agregando usuario

1. Clic en la opción Agregar Usuario del menú Usuarios.

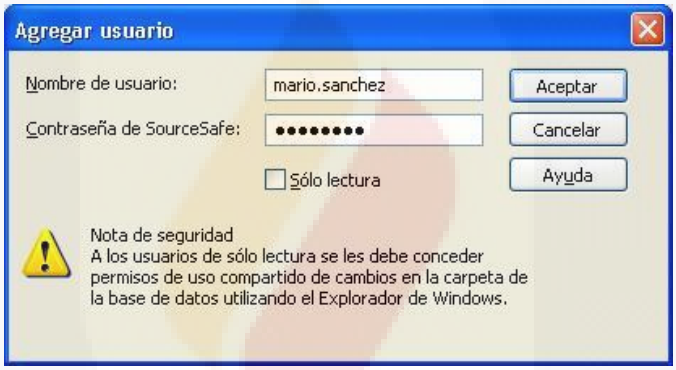


Ilustración 29 VSS Selección del usuario

2. A continuación escribimos el Usuario que deseamos agregar a nuestro repositorio, así mismo podemos establecer una contraseña predeterminada. La contraseña podrá ser cambiada en cualquier momento por el usuario.

Ilustración 30 VSS Administración de permisos por usuario

3. Listo, ya se encuentra dado de alta el usuario en el repositorio de nuestro proyecto, y ya puede acceder al repositorio desde cualquier computadora que cuente con el cliente de SourceSafe y con acceso al servidor.

Eliminar un usuario

Para eliminar un usuario del repositorio seguimos los dos siguientes pasos:

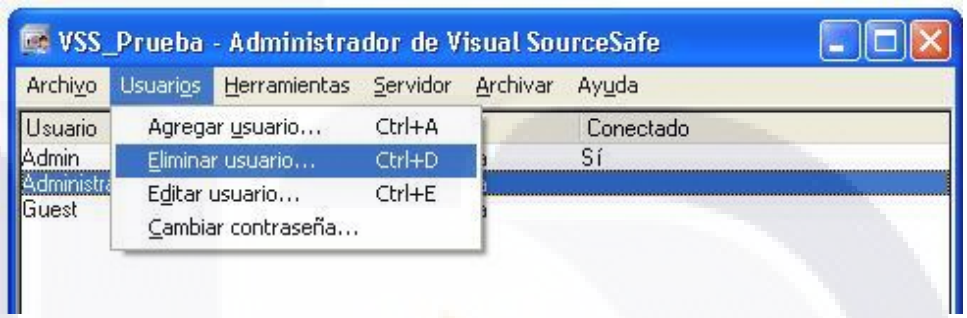


Ilustración 31 VSS Eliminar usuarios

1. Seleccionamos el usuario que deseamos eliminar y damos clic en la opción Eliminar usuario del menú Usuarios.

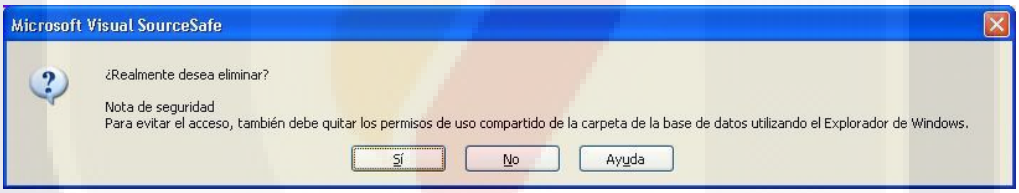


Ilustración 32 VSS Eliminar el usuario seleccionado

2. Nos aparece el siguiente mensaje de confirmación, aquí confirmamos que deseamos eliminar el usuario dando clic el botón Si.

Nota importante: También se recomienda eliminar el usuario de la carpeta compartida. Para ello es necesario dirigirnos a la carpeta de nuestro proyecto en el servidor, D:\VSS_Prueba y damos clic con el botón secundario del Mouse sobre dicho directorio y seleccionamos la opción Propiedades. Seleccionamos la pestaña Compartir y damos clic sobre el botón Permisos, seleccionamos el usuario a dar de baja y damos clic en Eliminar y luego en Aceptar. A continuación nos dirigimos a la pestaña de Seguridad, seleccionamos al usuario a dar de baja y damos clic en el botón Eliminar y posteriormente en Aceptar.

Habilitar comandos de Derechos por proyecto

Esta opción nos permite asignar derechos al contenido del repositorio para cada uno de los usuarios.

Para habilitar los comandos de derecho por proyecto seguimos lo siguiente:

1. Abrir el Administrador de Visual SourceSafe.

Ilustración 33 VSS Derechos por proyecto

2. En el menú **Herramientas**, haga clic en **Opciones** y, a continuación, en la ficha **Derechos del proyecto**.

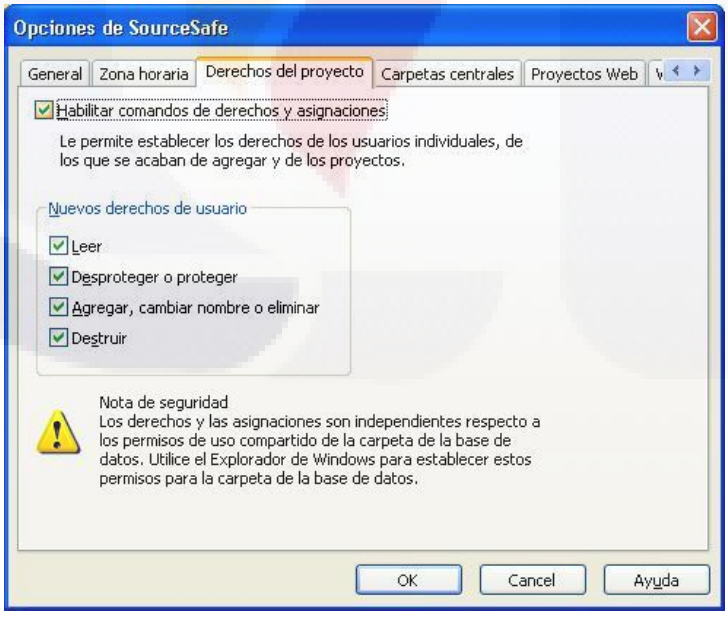


Ilustración 34 VSS permisos por proyecto

3. En el cuadro de diálogo **Opciones de SourceSaf**e, active la casilla de verificación **Habilitar comandos de derechos y asignaciones**. En el área

Nuevos derechos de usuario del cuadro de diálogo, puede anular la selección de los derechos de proyecto que no se apliquen a ningún usuario de la base de datos de su equipo.

4. Clic en **Aceptar** cuando termines de configurar las opciones de derechos de proyecto.
5. Consulta el menú **Herramientas** para comprobar que los comandos **Asignaciones de derechos para usuarios**, **Derechos por proyecto** y **Copiar derechos de usuario** están activos.

Asignar Derechos por proyecto

A continuación veremos cómo asignar derechos por proyecto a una lista de usuarios.

Ilustración 35 Derechos por proyecto

1. En el Administrador de Visual SourceSafe, haz clic en **Herramientas** y, a continuación, en **Derechos por proyecto**.

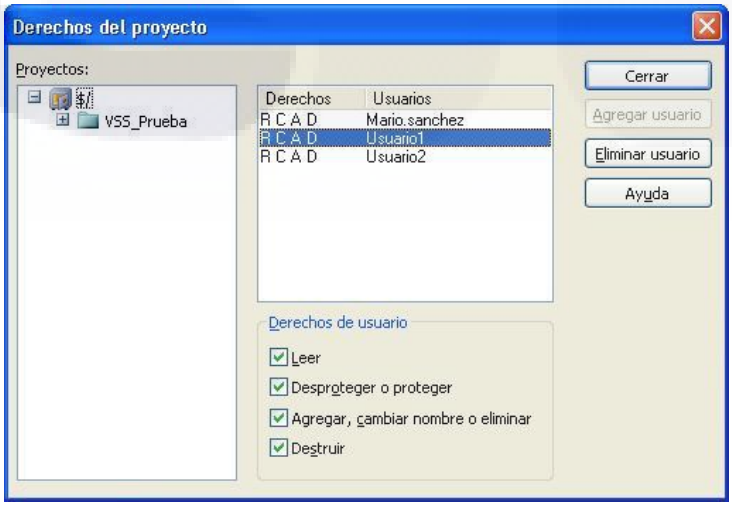


Ilustración 36 VSS Selección de usuarios

2. En el cuadro de diálogo **Derechos del proyecto** puedes seleccionar cada uno de los usuarios e iniciar a la asignación de permisos que tendrá para cada carpeta.

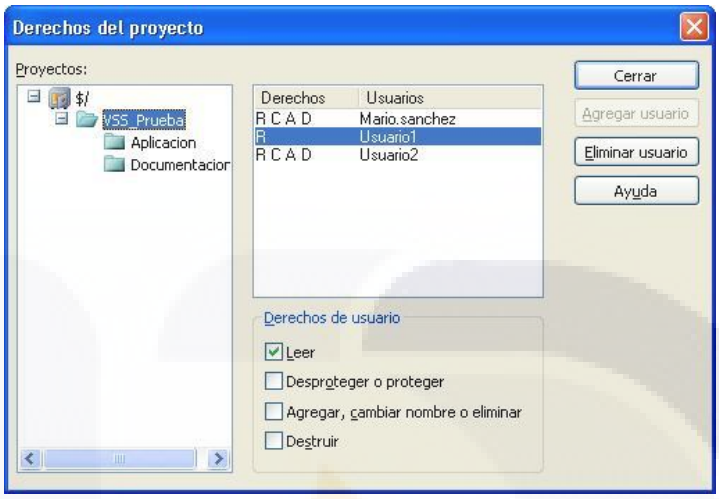


Ilustración 37 VSS Asignación de permisos por proyecto

3. También puedes seleccionar un proyecto (carpeta) y haz clic en **Agregar usuario** para adjuntar el usuario al que se asignarán los derechos de proyecto. Ten en cuenta que puedes seleccionar el nodo raíz de la base de datos ($/) si deseas asignar los derechos a toda la base de datos del repositorio.
4. En **Derechos de usuario**, especifica los tipos de comandos que el usuario puede ejecutar para el proyecto seleccionado y, a continuación, haz clic en **Aceptar** para aceptar las opciones.
5. Una vez que hayamos brindado los permisos para cada uno de los usuarios damos clic en el botón **Cerrar**.

Apéndice C

Manual de Subversion

Preparando un servidor.

En el siguiente apartado veremos cómo instalar, crear y administrar un repositorio de Subversion.

Pre-requisitos:

1. Tener privilegios de Administrador en el servidor donde se encuentra el repositorio de proyectos.
2. Servidor Apache.
3. Subversion.
4. Cliente para Subversion (TortoiseSVN).

Instalando Apache

1. Descargue la versión 2.0.54 del servidor web Apache desde http://httpd.apache.org/download.cgi
2. Una vez que tenga el instalador de Apache2 puede hacer doble clic en él y le guiará a través del proceso de instalación. Asegúrese de que ha introducido la URL del servidor correctamente (si no tiene un nombre DNS para su servidor introduzca la dirección IP). Por default Apache se instala en directorio de C:\Archivos de Programa\Apache Group\Apache2
3. Configure el puerto, por default el puerto que utiliza el servidor apache es el puerto 80, el cual puede causar conflictos si es que se tiene instalado previamente algún otro servidor web, como lo puede ser el IIS, para tal efecto, para cambiar el puerto edite localice el fichero httpd.conf, el cual se encuentra en C:\Archivos de programa\Apache Group\Apache2\conf\. Edite dicho fichero para cambiar Listen 80 por un puerto libre, por ejemplo, Listen 81. Luego reinicie el servidor.

4. Ahora compruebe si el servidor web Apache funciona correctamente apuntando desde su navegador web a la dirección http://localhost:81/ debería aparecer un sitio web pre configurado.

Instalando Subversion

1. Descargue la última versión de Subversion desde http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91.
    - Ejecute el instalador de Subversion y siga las instrucciones. Si el instalador de Subversion reconoce que se ha instalado Apache, por lo regular, Subversion se instala en el directorio "C:\Archivos de programa\Subversion".
    - Copie de la carpeta "C:\Archivos de programa\Subversion\bin" a la carpeta "C:\Archivos de programa\Apache Group\Apache2\modules" los archivos:
        i. mod_authz_svn.so
        ii. mod_dav_svn.so

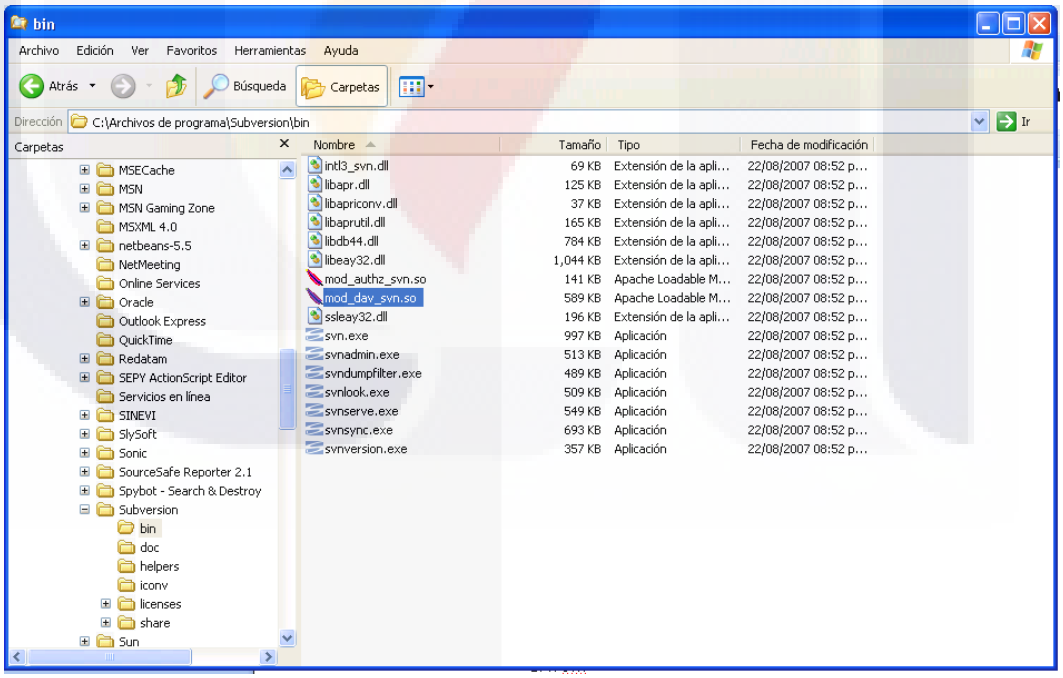


Ilustración 38 SVN Instalación Apache

2. Edite el fichero de configuración de Apache para habilitar los módulos en el apache:

```
# Modulos del suvbersion
LoadModule  dav_module          modules/mod_dav.so
LoadModule  dav_svn_module      modules/mod_dav_svn.so
LoadModule  authz_svn_module    modules/mod_authz_svn.so
```

Configuración

Ahora ya ha preparado Apache y Subversion, pero Apache aún no sabe cómo manejar los clientes de Subversion como TortoiseSVN. Para que Apache sepa qué URL debe utilizarse para los repositorios de Subversion debe editar el fichero de configuración de Apache (normalmente está en C:\Archivos de programa\Apache Group\Apache2\conf\httpd.conf) con cualquier editor de texto que desee (por ejemplo, el Bloc de Notas):

1. Al final del fichero Config añada las siguientes líneas:

```
<Location /svn>
DAV svn
SVNListParentPath on
SVNParentPath D:\SVN
AuthType Basic
AuthName "Repositorios de Subversion"
AuthUserFile passwd
#AuthzSVNAccessFile svnaccessfile
Require valid-user
</Location>
```

   Esto configura el Apache de forma que todos sus repositorios de Subversion están físicamente localizados bajo D:\SVN. Los repositorios se sirven al mundo exterior desde la URL: http://MiServidor/svn/.
2. Reinicie el servicio de Apache de nuevo.

Creación de repositorios

Puede crear un repositorio bajo el sistema FSFS o bien con el viejo pero estable formato Berkeley Data base (BDB). El formato FSFS es más rápido y ahora funciona en carpetas compartidas y en Windows 98 sin problemas.

Creando el repositorio con TortoiseSVN

1. Abra el explorador de Windows
2. Cree una nueva carpeta y llámela por ejemplo SVNRepositorio
3. Haga clic con el botón derecho sobre la carpeta recién creada y seleccione TortoiseSVN ®

   Crear Repositorio aquí....

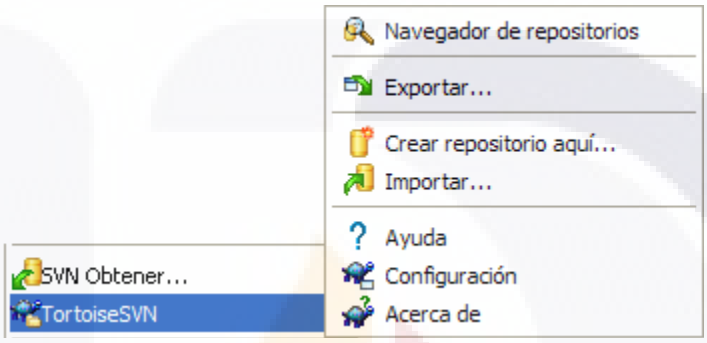


Ilustración 39 SVN Creación de un repositorio

Entonces se creará un repositorio dentro de la nueva carpeta. Si obtiene algún error asegúrese de que la carpeta esté vacía y que no esté protegida contra escritura.

Apéndice D

Manual de TFS

Preparando un servidor.

A continuación se describe una guía rápida para la instalación de *Microsoft Team Foundation Server 2005* dentro de una misma caja.

Pre-requisitos:

- Instalar Microsoft Windows Server 2003 o 2003 R2 con SP2
    - Instalar IIS, en esta parte se debe habilitar ASP.NET y tener cuidado de no habilitar las extensiones para FrontPage
- Instalar MS SQL 2005 con todas sus características.
    - Se debe asegurar de que todos los servicios en MS SQL arranquen automáticamente.
- Instalar Windows SharePoint services 2.0 con SP2.
    - No configure WSS y seleccione instalación de tipo Farm
- Instalar MS SQL server 2005 Service Pack 2
- Instalar .NET Framework 2.0 Service Pack 2
- Cree dos cuantas en el servidor TFSSERVICE y TFSREPORTS (si se tiene un dominio y se utiliza directorio activo entonces se deben crear estas cuentas en el directorio activo).

Ahora es tiempo de iniciar con la instalación de TFS. Si se está utilizando un controlador de dominio entonces acceda con una cuenta de administrador de dominio y no con una cuenta de administrador local.

- Inserte el CD/DVD de TFS e inicie la instalación

El instalador en primer lugar checará el estado del servidor y si cuenta con los componentes necesarios para realizar dicha instalación (instalaciones anteriores), si todo está bien, entonces le pedirá acceso a los servicios, inicie sesión con la

cuenta de TFSSERVICE, a continuación informe de servicio de acceso a datos, introduzca a continuación TFSREPORTS después SMTP (puede saltarse esta configuración)

- Inicie la instalación
- Finalice la instalación
- Instale Team Explorer.

Si se desea acceder a Team Foundation Server a través de Internet Explorer entonces descargue Team System Web Access Tool del sitio Microsoft, es libre de costo (seleccione un sitio web diferente, normalmente utilice el puerto 8090 o cualquier otro puerto distinto al 8080).

- Por último, instale Team Foundation Server 2005 Service Pack 1

Configuración

Para configurar Team Foundation Server utilizando la implementación de servidor único en un equipo de un dominio de Active Directory, se necesitan tres cuentas de usuario de dominio de Active Directory que se describen en la siguiente tabla.

Ejemplo de nombre de usuario de inicio de sesión	Finalidad
TFSSETUP	• Se utiliza para ejecutar el programa de instalación de Team Foundation Server. • Esta cuenta debe ser de administrador en equipos de Team Foundation Server. • Esta cuenta debe ser miembro del mismo dominio que las dos cuentas de servicio siguientes. Por ejemplo, no se pueden tener las dos cuentas de servicio en un dominio y utilizar después una cuenta local para ejecutar la instalación.
TFSSERVICE	• Se utiliza como cuenta de servicio por los servicios de Windows de Team Foundation Server (Servicio de análisis de cobertura de código y TFSSchedulerService), y SharePoint Timer Service. • Se utiliza como identidad del grupo de aplicaciones por el grupo de aplicaciones de Team Foundation Server (VSTF AppPool) y los grupos de aplicaciones

	para Windows SharePoint Services (TFWSS y WSS_AppPool).
	• Debe tener el permiso **Inicio de sesión local** en los equipos de Team Foundation Server.
	• Para que la seguridad sea óptima, esta cuenta de servicio:
	  ○ No debe ser de administrador en equipos de Team Foundation Server.
	  ○ Debe tener la opción **La cuenta es importante y no se puede delegar** seleccionada para Active Directory en el dominio.

TFSREPORTS	• Se utiliza como cuenta de servicio por los orígenes de datos de SQL Server Reporting Services. • Esta cuenta no debe ser de administrador en equipos de Team Foundation Server. • Esta cuenta debe tener el permiso **Inicio de sesión local** en los equipos de Team Foundation Server.

Para habilitar la comunicación entre Team Foundation Server, los requisitos previos y los clientes, debe asegurarse de que los firewalls ubicados entre estos componentes permiten la comunicación a través de un número de puertos TCP.

Puertos requeridos para SQL Server 2005

Microsoft SQL Server 2005 (Developer, Standard o Enterprise) utiliza los puertos TCP siguientes:

Contexto de aplicación o servicio	Puerto TCP
SQL Server Reporting Service	80
SQL Service	1433
Servicio explorador SQL	1434
Supervisión de SQL	1444
Redirector SQL Server Analysis Service	2382
SQL Server Analysis Service	2383

Puertos necesarios para Windows SharePoint Services

Windows SharePoint Services utiliza los puertos TCP siguientes:

Contexto de aplicación o servicio	Puerto TCP
Windows SharePoint Services	80
Administración central de SharePoint	17012

Puertos necesarios para Team Foundation Server

Team Foundation Server utiliza los puertos TCP siguientes:

Contexto de aplicación o servicio	Puerto TCP
Team Foundation Server	8080
Proxy de Team Foundation Server	8081
Team Foundation Build Remoting1	9191

Apéndice E

Formato control de cambios actual dentro del Instituto

<Nombre del Proyecto>

17. CONTROL DE CAMBIOS

Versión <1.0>

Fecha de Solicitud del Cambio <dd/mm/aa>

Control de Cambios

CUADRO 12: CONTROL DE CAMBIOS

Módulo:	[Nombre del módulo involucrado en el cambio.]
Caso de uso:	[Caso de uso involucrado en el cambio.]
Justificación:	[Descripción de la justificación del cambio.]
Descripción:	[Descripción del cambio.]
Responsable:	[Nombre del responsable de realizar el cambio.]
Observaciones:	[Observaciones que puedan servir como apoyo para la realización del cambio que se realizar.]

Solicitante del cambio	Recibe solicitud

Nota: Para el llenado de este formato se deberá sustituir el texto que se encuentra entre corchetes con la información que en cada sección se especifica

Apéndice F

Formatos propuestos para SCM en CMM (Álvarez Rodríguez et al., 2008)

Forma SCM1

DATOS DEL PROYECTO

Nombre del proyecto:
Integrantes del equipo:
Cliente:
Fecha:
Periodo:
No. Semana: 00/00

DESARROLLO DEL PLAN DE SCM1

Objetivo

Definir el plan de actividades a seguir durante la realización del proyecto, correspondiente a SCM

Instrucciones:

Realice una descripción clara y precisa de los puntos que se mencionan

Compromisos	Por parte de quién

Metas

Actividades	Calendarización

Recursos

Herramientas

Líder de proyecto

Ingeniero de Software

Realizó

Supervisó

Forma SCM2

DATOS DEL PROYECTO

Nombre del proyecto:

Integrantes del equipo:

Cliente:

Fecha:

Periodo:

No. Semana: 00/00

BIBLIOTECA DE LÍNEAS BASE DE SOFTWARE

Objetivo

Definir un sistema de biblioteca de líneas base de software para que sirva como repositorio del mismo

Instrucciones:

Realice una descripción clara y precisa de los puntos que se mencionan

1. Definir el tipo de arquitectura a utilizar para la biblioteca

Compromisos	**Por parte de quién**

2. Definir los componentes de la biblioteca

Componentes	**Descripción**

Líder de proyecto

Ingeniero de Software

Realizó

Supervisó

Forma SCM3

DATOS DEL PROYECTO

Nombre del proyecto:

Integrantes del equipo:

Cliente:

Fecha:

Periodo:

No. Semana: 00/00

REPORTE DE LOS PRODUCTOS DE TRABAJO DE LAS LÍNEAS BASE

Objetivo

Lograr identificar los productos de trabajo que están bajo la administración del SCM.

Instrucciones:

Realice una descripción clara y precisa de los productos de trabajo efectuando hasta el momento, según la categoría correspondiente

1. Procesos

Identificador	Proceso	Descripción y función	Realizada por	Fecha de inicio	Fecha de Término	Versión

2. Requerimiento de software

Identificador	Requerimiento	Descripción y función	Responsable	Fecha de inicio	Fecha de Término	Versión

3. Código de las unidades de software

Identificador	Módulos	Descripción y función	Tamaño	Responsable	Fecha de inicio	Fecha de Término	Versión

4. Pruebas de software

Identificador	Módulo	Prueba aplicada	Responsable	Fecha de inicio	Fecha de Término	Versión

Líder de proyecto

Ingeniero de Software

Realizó

Supervisó

Forma SCM4

DATOS DEL PROYECTO

Nombre del proyecto:

Integrantes del equipo:

Cliente:

Fecha:

Periodo:

No. Semana: 00/00

REPORTE DE DE CAMBIOS EN LÍNEA BASE DE SOFTWARE

Objetivo

Llevar el control de los cambios que se realizan a las diferentes versiones de productos

Instrucciones:

Realice una descripción clara y precisa de los cambios realizados a los productos de trabajo, según la categoría correspondiente

1. Procesos

Identificador	Línea base	Identificador cambio	Copiado de cambios(id)	Actualización de biblioteca de línea base	Registro de archivar línea base reemplazada	Responsable	Fecha

Líder de proyecto

Ingeniero de Software

Realizó

Supervisó

Forma SCM5

DATOS DEL PROYECTO

Nombre del proyecto:

Integrantes del equipo:

Cliente:

Fecha:

Periodo:

No. Semana: 00/00

REPORTE DE DE LAS ACTIVIDADES DE LA ADMINISTRACIÓN DE CONFIGURACIÓN DE SOFTWARE

Objetivo

Reportar cada una de las actividades correspondientes a la KPA de administración de la configuración del software, según se haya realizado durante la identificación y definición de las líneas base de software

Instrucciones:

Realice una descripción clara y precisa de las actividades de trabajo realizado hasta el momento, según la categoría correspondiente

1. Actividades para procesos

Identificador	Proceso	Descripción	Realizada por	Fecha de inicio	Fecha de Término

2. Actividades para requerimientos de software

Identificador	Proceso	Descripción	Realizada por	Fecha de inicio	Fecha de Término

3. Actividades para código de las unidades de software

Identificador	Proceso	Descripción	Realizada por	Fecha de inicio	Fecha de Término

4. Actividades para pruebas de software

Identificador	Proceso	Descripción	Realizada por	Fecha de inicio	Fecha de Término

Líder de proyecto

Ingeniero de Software

Realizó

Supervisó

Forma SCM6

DATOS DEL PROYECTO

Nombre del proyecto:

Integrantes del equipo:

Cliente:

Fecha:

Periodo:

No. Semana: 00/00

ESTADO DE LOS OBJETIVOS/UNIDADES DE CONFIGURACIÓN

Objetivo

Llevar un registro del estado que tiene cada uno de los objetos y unidades de configuración

Instrucciones:

1. Responda cada uno de los cuestionamientos planteados, realizando una descripción clara y precisa de los problemas que se desea solucionar.

Objetivo/unidad	Descripción	Estado, porcentaje (terminado/ en proceso inicial)	Realizado por

2. Si el estado no es el esperado, ¿Qué propone para solucionarlo?

Objetivo/unidad	**Propuesta**

Líder de proyecto

Ingeniero de Software

Realizó

Supervisó

Forma SCM7

DATOS DEL PROYECTO

Nombre del proyecto:

Integrantes del equipo:

Cliente:

Fecha:

Periodo:

No. Semana: 00/00

CONTROL DE AUDITORÍAS

Objetivo

Llevar el control de las auditorias que se realizan a las líneas base de software

Instrucciones:

Realice una descripción clara y precisa de los puntos que se menciona.

Módulos/unidades	Responsable módulo	Versión	No. auditoría	Responsable auditoría	Observaciones	Fecha de inicio auditoria	Fecha de Término auditoria

Líder de proyecto

Ingeniero de Software

Realizó

Supervisó

Apéndice G

Formatos propuestos para el proceso de manejo de la gestión de configuraciones en el INEGI.

Forma INEGI-SCM1

DATOS DEL PROYECTO

Fecha:

Nombre del proyecto:

Integrantes del equipo:

Usuario(s):

DESARROLLO DEL PLAN DE LA ADMINISTRACIÓN DE CONFIGURACION

Compromisos	Por parte de quién

Metas

Actividades	Calendarización

Líder de proyecto

Usuario

Forma INEGI-SCM2

DATOS DEL PROYECTO

Fecha:

Nombre del proyecto:

Integrantes del equipo:

Usuario(s):

BIBLIOTECA DE LÍNEAS BASE DE SOFTWARE

1. Definir los componentes de la biblioteca

Componentes	**Descripción**

Líder de proyecto

Usuario

Forma INEGI-SCM3

DATOS DEL PROYECTO

Fecha:

Nombre del proyecto:

Integrantes del equipo:

Usuario(s):

REPORTE DE LOS PRODUCTOS DE TRABAJO DE LAS LÍNEAS BASE

Objetivo

Lograr identificar los productos de trabajo que están bajo la administración del SCM.

Identificador	Descripción y función	Realizada por	Estado	Fecha de	Versión O Revisión

Líder de proyecto

Usuario

Forma INEGI-SCM4

DATOS DEL PROYECTO

Fecha:

Nombre del proyecto:

Integrantes del equipo:

Usuario(s):

REPORTE DE LAS ADMINISTRACION DE CAMBIOS

Objetivo

Lograr llevar una adecuada administración de los cambios que se realicen a la línea base dl desarrollo

1) Registro y análisis de la petición del cambio

Fecha	Descripción del cambio	Análisis del cambio	Analizado por	Solicitado por:	Asignado a:	Versión

2) Seguimiento del cambio

Fecha de recepción	**Descripción del cambio**	**Realizado por:**	**Fecha de atención**	**Versión**

Líder de proyecto

Usuario

Forma INEGI-SCM5

DATOS DEL PROYECTO

Fecha:

Nombre del proyecto:

Integrantes del equipo:

Usuario(s):

CONTROL DE AUDITORÍAS

Objetivo

Llevar el control de las auditorias que se realizan a las líneas base de software

Módulos/unidades	Responsable módulo	Versión	No. auditoría	Responsable auditoría	Observaciones	Fecha de inicio auditoria	Fecha de Término auditoria

Líder de proyecto

Usuario

Forma INEGI-SCM6

DATOS DEL PROYECTO

Fecha:

Nombre del proyecto:

Integrantes del equipo:

Usuario(s):

LIBERACIÓN DE LA LÍNEA BASE

Objetivo

Llevar el control de las liberaciones de la línea base realizadas al proyecto

Fecha de liberación	Responsable de la liberación	Versión	Observaciones

Líder de proyecto

Usuario

Glosario de términos

Source Code Control System (SCCS)

Fue el primer sistema controlador de versiones, desarrollado originalmente en los laboratorios Bell en 1972 por Marc J. Rochkind para una IBM System/370, SCCS dominó los sistemas controladores de versiones hasta el surgimiento de Revision Control System

Revision Control System (RCS)

Es una aplicación de software de control de revisión, que automatiza el almacenamiento, recuperación, registro, identificación, y la fusión de las revisiones. RCS es útil para el texto que se revisa con frecuencia, para los programas de ejemplo, documentación, gráficos de procedimientos, documentos y cartas.

Check-out

El comando *Check-out* se utiliza para obtener una copia del código que se encuentra en el repositorio para su edición de forma local

Check-in

El comando check-in es utilizado para actualizar el repositorio, con las porciones de código que se hayan editado de manera local

Commit

Un *commit* sube los cambios realizados al repositorio.

Merge

Es mesclar los cambios y/o diferencias que se crean durante el desarrollo del sistema.

Branch o rama:

Un branch o rama representa una línea paralela de desarrollo, son utilizados para hacer revisiones de una versión que se encuentra en producción.

Repositorio

Es el lugar donde se concentran las copias maestras del proyecto. El repositorio puede estar en la misma máquina o en un servidor remoto y el cual se accede a través de la red.

Workspace

El workspace representa una copia local de todas las cosas necesitamos para trabajar en un proyecto y que se encuentran almacenadas en el repositorio

Stakeholder

La definición más correcta de stakeholder es parte interesada (del inglés stake, apuesta, y holder, poseedor). Se puede definir como cualquier persona o entidad que es afectada por las actividades de una organización; por ejemplo, los trabajadores de esa organización, sus accionistas, las asociaciones de vecinos, sindicatos, organizaciones civiles y gubernamentales, etc.

Proceso

Un proceso es una serie de pasos que ayudan a dar solución a un problema. Estos pasos deben ser definidos como forma de terminar con la ambigüedad que estos pudieran tener, y a su vez sean rápidamente entendidos y capaces de ser llevados a cabo de una manera consistente por cualquier persona que utilice el proceso. Enfocar nuestro esfuerzo a utilizar un proceso dentro de la organización tiene la finalidad de reducir el trabajo redundante, ¿por qué se tiene que recrear el ciclo cada vez que se inicia un proyecto? (Kulpa & Johnson, 2003)

Referencias bibliográfica:

Álvarez Rodríguez, F. J., Muñoz Arteaga, J., Cardona Salas, J. P., Brizuela Sandoval, M., Quezada Aguilera, F. S., & Ponce Gallegos, J. C. (2008). *Interpretación del Modelo de Madurez de Capacidades (CMM) para pequeñas industrias de software* (Primera., Vols. 1-200). Aguascalientes, Aguascalientes: Universidad Autónoma de Aguascalientes.

Azad, K. (2009). Intro to Distributed Version Control. Recuperado a partir de http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated.html

Bach, J. (1998). The Highs and Lows of Change Control. *IEEE TRANSACTIONS ON EDUCATION*, *31*(8), 113-115.

Beck, J. (2005). Using The Cvs Version Manag Ment System In A Software Engineering Course. *Division of Mathematics and Computer Science, Truman State University*, 57 - 65.

Beth-Anne Sullivan. (2008, Noviembre 12). *Controlling a World of Chaos: Change Management in Higher Education*. Microsoft Power Point, Northeastern University. Recuperado a partir de http://64.3.162.168/media/Change Management for NERCOMP.ppt

Chrissis, M. B., Konrad, M., & Shrum, S. (2006). CMMI Guía para la integración de procesos y la mejora de productos. Carnegie Mellon University. Recuperado a partir de http://www.sei.cmu.edu/library/assets/cmmi-dev-v12-spanish2.pdf

Clifton, C., Kaczmarczyk, L. C., & Mrozek, M. (2007, Marzo 7). Subverting the Fundamentals Sequence: Using Version Control to Enhance Course Management.

Department of Computer Science and Software Engineering , RoseHulman Institute of Technology.

CMMI Product Team. (2002). *Capability Maturity Model Integration (CMMISM), Version 1.1*. Carnegie Mellon University.

Collins-Sussman, B., W. Fitzpatrick, B., & Pilato, M. (2004). *Version Control with Subversion, Next Generation Open Source Version Control*. O'Reilly.

Dunaway, D. K., & Masters, S. (1996). CMM Based Appraisal for Internal Process Improvement (CBA IPI): Method Description. Software Engineering Institute, Carnegie Mellon University.

Garvin, J. (2007, Junio 22). Source Code Management Systems: Trends, Analysis and Best Features. *CIO*. Recuperado a partir de www.cio.com

Hundhausen, R. (2006). *Working with Microsoft Visual Studio 2005 Team System*. Redmond, Washington: Microsoft Press, A Division of Microsoft Corporation. Recuperado a partir de www.microsoft.com/learning/

INEGI. (2000). Políticas de Calidad INEGI. Instituto Nacional de Estadística, Geografía e Informática. Recuperado a partir de http://intranet.inegi.gob.mx

INEGI. (2006, Diciembre 20). Guía para el desarrollo y documentación de software.

IT Governance Institute. (2007). COBIT 4.1. IT Governance Institute. Recuperado a partir de www.itgi.org

J Murphy, D. (2007). Understanding the Problem. En *Managing Software Development with Trac and Subversion* (pág. 116). Packt Publishing.

Jacobs, R. (2004). Software Configuration Management for LabVIEW Programming Environments. *LabVIEW Technical Resource, 11*(2), 20-24.

Jones, M. K. (2000). Software Configuration Management for the Web. *Advertise with*

Methods & Tools. Recuperado a partir de http://www.methodsandtools.com/archive/archive.php?id=3

Kingsbury, J. (1996). Adopting SCM Technology. *Software Technology Support Center*.

Kulpa, M. K., & Johnson, K. A. (2003). *Interpreting the CMMI A process Improvement Approach*. Auerbach.

Mahotkin, A. (2008). Version control system comparison. *Version control blog*. Recuperado a partir de http://www.versioncontrolblog.com/

Mallette, D. (2005). IT Performance Improvement With COBIT and the SEI CMM. Information Systems Audit and Control Association. Recuperado a partir de http://www.isaca.org/Content/ContentGroups/Journal1/20058/IT_Performance_Improvement_With_COBIT_and_the_SEI_CMM1.htm

Meli, M. A. (2005, Abril). *Técnicas y herramientas de desarrollo ágil para Microsoft .Net Framework*. Universidad Nacional del Sur, Buenos aires, Argentina.

Oktaba, H. (2003, Septiembre). Moprosoft: el nuevo modelo que impondrá una norma mexicana para la calidad en la industria del software.

Ortiz Núñez, P. A., & Hoyos Franco, A. M. (2005). ITIL: Una nueva alternativa en el aprovechamiento de los recursos informáticos para las empresas colombianas. *Revista de Ingenierías Universidad de Medellín, 4*(6), 25-39.

Paulk, M. C., Konrad, M. D., & Garcia, S. M. (1995). CMM Versus SPICE Architectures. *IEEE Software Process Newsletter*, (3), 7-11.

Peña García, M. (2009, Mayo). *Impacto de la aplicación de un modelo CMMI Nivel 2 en el Ciclo de Vida de un proyecto* (Tesis). Universidad Politécnica de Madrid, Madrid España.

Picazzo M., C. (2008, Octubre 24). Análisis descriptivo del proceso de implementación del

nivel 2 del modelo CMMI en una empresa regional de desarrollo de software, *6*(12).

Pressman, R. S. (2001). Software configuration management. En *Software Engineering, A practitioner's approach* (Fifth Edition., págs. 193 - 241). McGraw-Hill.

Rahikkala, T. (2000, Mayo 30). *Towards virtual software configuration management a case study* (Tesis). Oulu, Finlandia.

Software Technology Support Center. (2005). Configuration Management Fundamentals. Presented at the CROSSTALK The Journal of Defense Software Engineering, July 2005, U.S. Air Force's Software Technology Support Center. Recuperado a partir de http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA489512&Location=U2&doc=GetTRDoc.pdf

St.Jean, S. (2006). From VSS to TFS, An Introduction to Team Foundation Server Version Control from a Visual SourceSafe User's Perspective.

Thomas, D., & Hunt, A. (2003). *Pragmatic Version Control Using CVS* (1º ed.). McGraw-Hill.

Tosun, F. (2004, Junio). *Evaluating Software Configuration Management Tools For Opticon Sensors Europe B.V.* Universiteit van Amsterdam.

Vesperman, J. (2006). What Is CVS?, Chapter 1. En *Essential CVS, 2nd Edition* (2º ed., pág. 428). United States of America: O'Reilly.

Wein, M., Cowan, W., & Gentleman, W. (1992). Visual Support for Version Management. *National Research Council of Canada, University of Waterloo*, 1217-1223.